Волинський національний університет імені Лесі Українки

Кафедра теоретичної та комп'ютерної фізики імені А.В. Свідзинського

Дмитро Шваліковський

Моделювання процесів та систем у середовищі CAS Maxima

ЛУЦЬК
2024

УДК 004.94, 517.9

Вченою радою
Волинського національного університету імені Лесі Українки
надано гриф «Рекомендовано»
(протокол № 7 від 31 травня 2024 р.)

Рецензенти:
Головацький В. А., доктор фізико-математичних наук, Чернівецький національний університет імені Юрія Федьковича;
Федосов С. А., доктор фізико-математичних наук, Луцький національний технічний університет;
Щепанський П. А., кандидат фізико-математичних наук, Львівський національний університет імені Івана Франка.

Шваліковський Д. М. Моделювання процесів та систем у середовищі **CAS Maxima**. — Луцьк: ВНУ імені Лесі Українки, 2024. — 252 с.

Навчальний посібник містить опис багатьох команд для розв'язання низки задач математичного аналізу та лінійної алгебри за допомогою комп'ютера в програмному пакеті **CAS Maxima**, а також набір завдань для проходження здобувачами освіти обчислювальної практики. Розроблена реалізація методів для числового розрахунку фундаментальних рівнянь математичної фізики, розв'язки супроводжуються візуалізацією різними засобами. У посібнику міститься 14 розділів, які охоплюють найбільш важливі області вищої математики та математичної фізики для допомоги студентам фізичних та технічних спеціальностей.

Матеріали розроблені відповідно до навчальних планів підготовки здобувачами вищої освіти математичних компетентностей у галузі знань 10 Природничі науки, освітніх програм за спеціальностями 104 Фізика та астрономія, 105 Прикладна фізики та матеріали, 111 Математика, 113 Прикладна математика.

УДК 004.94, 517.9

Зміст

Вступні зауваги

Походження Maxima

Проєкт **Maxima** є наступником DOE Macsyma — системи комп'ютерної алгебри, розробка якої велася з 1968 року в MIT (Массачусетський технологічний інститут) у лабораторії Project MAC, що була заснована та профінансована Міністерством енергетики США (Departament of Energy, DOE). Це була перша всеосяжна система символьної математики та одна із найбільш ранніх систем для аналітичних обчислень. Багато ідей, що з'явилися в Macsyma, згодом були запозичені такими застосунками як **Mathematica**, **Maple** та іншими.

Професор Вільям Шелтер (William Schelter) з Техаського університету в Остіні продовжував підтримку та покращення версії DOE Macsyma з 1982 року, зокрема він виконав перенесення системи на Common Lisp. 1998 року Шелтер з дозволу DOE опублікував вихідний код DOE Macsyma під ліцензією GNU General Public License, а 2000 року створив проект **Maxima** на SourceForge.net для підтримки та подальшого розвитку цієї системи як вільного програмного забезпечення. Після смерті Шелтера в 2001 році проєкт продовжує розвиватися силами спільноти розробників, що склалася. Зараз над проєктом працює велика кількість математиків і програмістів на чолі з Джеймсом Емундсоном (James Amundson).

Maxima знаходиться в активній розробці, вона може бути скомпільована під кілька різних реалізацій Common Lisp, у відкритому доступі заходяться готові збірки для GNU/Linux, Microsoft Windows, Mac OS X та інших систем. Характерною особливістю **Maxima** є те, що існує кілька варіантів графічних інтерфейсів користувача (можлива також робота і без графічного інтерфейсу, у консольному режимі); також її ядро може підключатись як зовнішній обчислювальний пакет до таких систем для наукової роботи як Sage, TeXmacs, Cantor, Jupiter тощо. Ця обставина є нормою для середовища OpenSource — вільного відкритого програмного забезпечення. Аналітичні можливості **Maxima** багато в чому не поступаються таким комерційним продуктам як **Maple** чи **Mathematica**.

Можливості Maxima

Maxima дозволяє вирішувати досить широке коло завдань, що відносяться до різних розділів математики:

- обчислення з елементарними функціями;
- операції з поліномами;
- обчислення зі спеціальними функціями;
- диференціальне числення різних типів;
- аналітичне обчислення означених та неозначених інтегралів;
- розв'язання алгебричних, диференціальних, інтегральних рівнянь;
- операції зі степеневими рядами, рядами Тейлора, рядами Фур'є;
- операції з матрицями, списками, масивами;

∘ операції з тензорами, теорія груп, абстрактна алгебра.

Дослідниками розроблений ряд додаткових пакетів для **Maxima**, котрі можна завантажувати перед використанням, та які істотно розширюють її можливості і коло розв'язуваних завдань. Значна кількість їх розміщена на репозиторіях сайтів GitHub та SourceForge. В даній роботі однак будуть використовуватись лише ті пакети, які вбудовані у офіційний дистрибутив застосунку.

Maxima має великі можливості для побудови різного роду графіків, об'єктів та візуалізації процесів. Вона виступає бекендом для проектів **gnuplot** та **VTK**, що є добре знаними у світі науки відкритими процесорами графічних зображень та 3d-моделей.

Для студентів, інженерів чи науковців системи комп'ютерної математики є потужним інструментом для аналізу різних типів задач моделювання. Метод математичного моделювання полягає в перенесенні реальних властивостей об'єктів або процесів на певні математичні структури та співвідношення між ними. Фізичні, біологічні чи соціальні процеси часто вивчаються з використанням диференціальних рівнянь різних видів, і у класичних університетських курсах диференціальних рівнянь чи математичної фізики їх розв'язанню приділяється багато уваги. У цьому випадку CAS (Computer Algebra System) є незамінним помічником для вирішення складних випадків рівнянь чи їх числового розрахунку.

Дуже важливим завданням є візуалізація кінцевого результату обчислень. Навіть для спеціалістів розв'язок рівняння у вигляді формули може бути занадто громіздким та важко читаним, тим більше що нерідко він містить спеціальні функції (Бесселя, Якобі та ін.) чи їх похідні. Побудова фазового портрету чи поверхні інтегрування відразу показує якісну поведінку та прогнозує стан динамічної системи в майбутньому, і **Maxima** надає для цього всі засоби.

В основі даного посібника лежить навчально-методичне видання 2022 року «**CAS Maxima**: основи роботи». У процесі проведення дослідницької та навчальної діяльності стало необхідним його виправлення та суттєве доповнення новим матеріалом.

Система комп'ютерної алгебри **Maxima** буде корисною для всіх дослідників, пов'язаних з комп'ютерним моделюванням: не лише фізиків чи математиків, а й економістів, екологів, хіміків, широкого загалу науковців.

Розділ 1

Основи роботи з Maxima

1.1 Інтерфейс wxMaxima

Після запуску **wxMaxima** маємо вікно введення команд з панелями швидкого доступу до найбільш вживаних операцій **Maxima**.

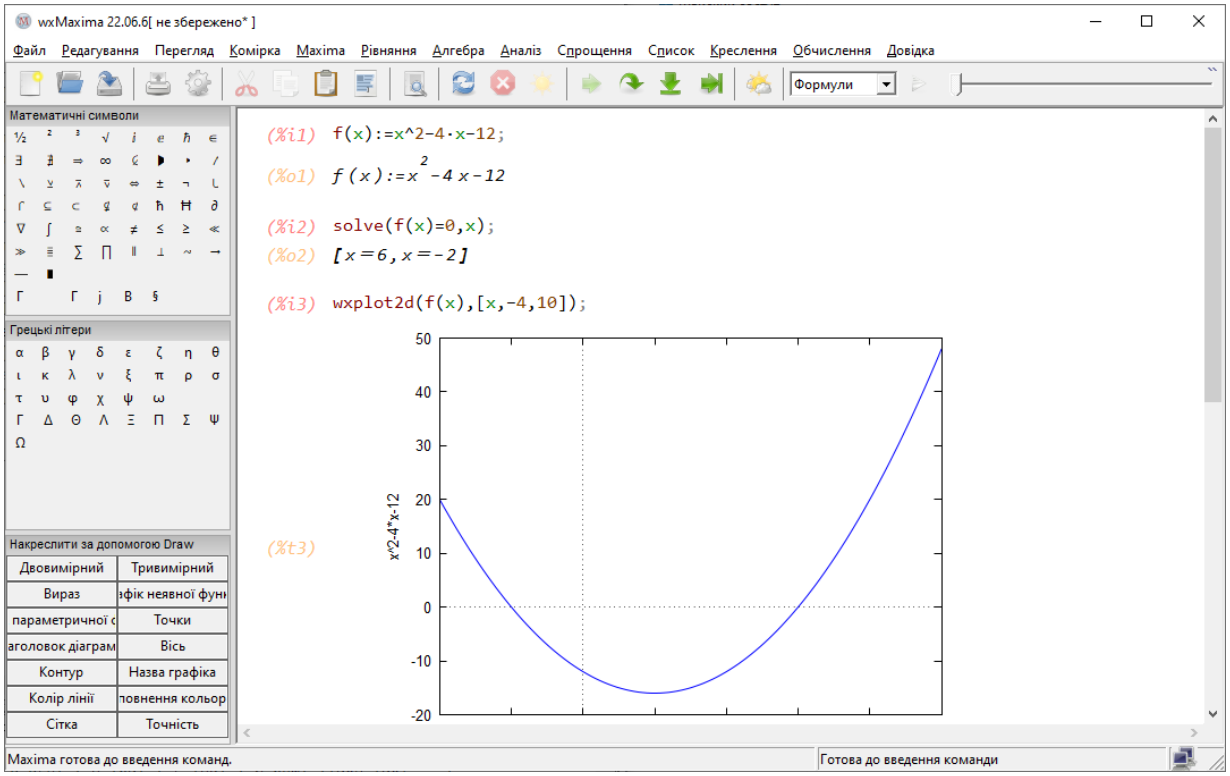


Рис. 1.1: Інтерфейс **wxMaxima**

Здійснимо найпростішу операцію — перемножимо два числа: вводимо команду з клавіатури 111*123 та натиснемо на `Shift+Enter`.

```
(%i1)  111*123;

(%o1)     13653
```

Таким чином, у документі були сформовані дві області: `(%i1)` — вхідна команда та `(%o1)` — відповідний їй вихідний результат. Кожна область має власну назву, поміщену в дужки. Це дуже зручно, оскільки замість повторного введення довгої команди можна просто вказати дужку `(%oN)`, де `N` — присвоєний цьому виразу номер.

Система **wxMaxima** передбачає можливість введення декількох команд одразу в одному рядку. Для цього треба ввести команди, розділені символами «;». У цьому випадку формується одна область введення та кілька областей виведення:

```
(%i4)  a:10; b:5; a*b;
```

(%o2) 10

(%o3) 5

(%o4) 50

Запуск команди на виконання за замовчуванням здійснюється комбінацією `Shift+Enter`, тоді як `Enter` переносить курсор на інший рядок. Це необхідно для форматування тексту та розбиття довгих команд на менші, більш зручні для читання та впорядкування. Втім, оптимальним видається запускати виконанням натиском `Enter`, тому при першому запуску рекомендується поставити в Налаштуваннях галочку на пункті «Enter для обчислення в комірках», тоді ці дві опції поміняються місцями. Надалі будемо вважати, що такий вибір зроблено.

Будь-яка команда в **Maxima** повинна закінчуватись знаком «;». Оболонка **wxMaxima** автоматично додає його в кінці при натисненні `Enter`, але в інших інтерфейсах (консольний чи **XMaxima**) необхідно записувати його явно. Щоб вказати на кінець командного рядка, можна замість крапки з комою використовувати знак «$». Це зручно, якщо виведення проміжних розрахунків на екран не потрібні, тоді їх можна приховати. Прихований результат все одно буде обчислений. Наприклад,

```
(%i5)  R:10$ L:5$ S=%pi*R*(R+L),numer;
```

(%o5) $S = 471.2388980384689$

Як бачимо, для виведення числового значення величини S була додана команда `numer`. Справа в тому, що **Maxima** завжди намагається перебувати у точних математичних рамках, де ірраціональні числа (π, e, $\sqrt{2}$,...) залишаються необчисленими. Без цього уточнення вивід дав би вираз $S = 150\pi$. Схожу функцію має команда `float`, а команда `bfloat` видає результат у нормальній формі числа, тобто після першої значущої цифри йде кома з десятковими значеннями, в кінці bN, що означає $\cdot 10^N$.

За замовчуванням **wxMaxima** відображає 16 цифр після коми. Це значення можна змінити двома способами: задавши команду `fpprec:N`, де `N` — бажана кількість цифр, або використавши меню «Обчислення — Встановити підвищену точність обчислень».

```
(%i6)  bfloat(150*%pi);
```

(%o6) $4.7123889803846 9b2$

Назви функцій та змінних у системі **Maxima** чутливі до регістру, тобто великі та малі літери відрізняються.

```
(%i7)  sin(%pi/2);
```

(%o7) 1

```
(%i8)  Sin(%pi/2);
```

(%o8) $\mathrm{Sin}\left(\frac{\pi}{2}\right)$

Значення імен змінних зберігаються протягом всієї роботи з документом. Якщо необхідно зняти означення з якоїсь змінної `var`, це можна зробити за допомогою команди `kill(var)`, причому це може бути як ім'я змінної, так і комірка введення чи виведення.

Також можна очистити всю пам'ять сеансу і звільнити всі імена, використавши команду `kill(all)`, або вибрати пункт меню «Maxima — Спорожнити пам'ять». Тоді нумерація комірок знову почнеться з одиниці.

Якщо навести курсор миші на верхній лівий кут квадратної дужки та клацнути лівою кнопкою, то ця частина документу згорнеться, повторне клацання миші поверне клітинку до початкового стану. При потребі вставити додаткову комірку між двома, необхідно клікнути мишею на проміжку між клітинами, щоб з'явилась горизонтальна лінія; після цього можна вводити нову команду. Щоб видалити комірку, необхідно виділити її зліва мишкою, та натиснути `Del`. Надзвичайно зручним інструментом є контекстне меню виділеної комірки, при натисненні правої кнопки миші: можна скопіювати цю комірку як команду **Maxima**, код LaTex, код MathML або як малюнок. **wxMaxima** має і певні можливості для форматування звичайного тексту. При наборі можна користатись стилями, заголовками різних рівнів, можна також вставляти малюнки просто у тіло документу.

Особливістю графічного інтерфейсу системи **wxMaxima** є те, що коли в робочому вікні відкривається раніше збережений документ, будуть відображатися лише команди введення, тоді як клітини з результатами не відображатимуться. Для їх виконання потрібно натиснути пункт меню «Комірка — Обчислити всі комірки».

1.2 Довідка

При роботі з виразами буває необхідним дізнатись синтаксис команди або пригадати написання додаткових опцій при введенні команд. Для цього існує обширна довідкова система, яка викликається в меню «Довідка — Довідка з Maxima». Ця дія виводить на екран документ з розбитими на розділи можливостями **Maxima** та повним списком всіх команд **Maxima** з їх прикладами.

Однак зручним є також використання довідки у самій системі **wxMaxima** , що здійснюється двома основними способами: командами `example` та `??`. Перша команда дає приклад синтаксису команди з результатами виконання:

```
(%i8) example(ratsimp);
```

(%i9) $\mathrm{b}*(\mathrm{a}/\mathrm{b}-\mathrm{x})+\mathrm{b}*\mathrm{x}+\mathrm{a}$

(%o9) $b\cdot x+b\cdot\left(\frac{a}{b}-x\right)+a$

```
(%i10) ratsimp(%)
```

(%o10) $2\cdot a$

Друга команда дублює відповідну інформацію з Довідки:

```
(%i11) ?? ratsimp;
0: fullratsimp  (Functions and Variables for Polynomials)
1: ratsimp  (Functions and Variables for Polynomials)
2: ratsimp <1>  (Functions and Variables for Polynomials)
3: ratsimpexpons  (Functions and Variables for Polynomials)};
Enter space-separated numbers, 'all' or 'none': 1
```

1.3 Базові операції, константи, функції

Maxima працює з так званими *атомами* — це число, символ або список (множина чисел чи символів). Основними в роботі з символьними виразами є два оператори: присвоєння «:» та функціональної залежності «:=».

Оператор присвоєння є операцією заміни одного типу атому на інший, наприклад символ на число, або масив чисел.

```
(%i2)  a:2; b:[3,4,5];
```
(%o1) 2

(%o2) $[3, 4, 5]$

```
(%i3)  b^a*c;
```
(%o3) $[9c, 16c, 25c]$

```
(%i4)  [d,e,f]:[-1,-2,-3];
```
(%o4) $[-1, -2, -3]$

```
(%i5)  e;
```
(%o5) -2

Функціональна залежність вказує явно на незалежні змінні функції, по яких її можна диференціювати, інтегрувати, підставляти конкретні значення змінних та ін.

```
(%i6)  g(x,y):=x^2*sin(y);
```
(%o6) $\mathrm{g}(x, y) := x^2 \cdot \sin(y)$

```
(%i7)  diff(g(x,y),x);
```
(%o7) $2 \cdot x \cdot \sin(y)$

```
(%i8)  g(2,3*%pi/2);
```
(%o8) -4

Математичні операції є цілком звичними: додавання (+), віднімання (−), множення (*), ділення (/), піднесення до степеня (^), факторіал (!).

Вбудовані константи: `%e` — число e; `%pi` — число π; `%i` — уявна одиниця $\sqrt{-1}$; `%phi` — золоте відношення $(1+\sqrt{5})/2$; `inf` — додатня дійсна нескінченність $+\infty$; `minf` — від'ємна дійсна нескінченність $-\infty$; `infinity` — комплексна нескінченність.

Зарезервовані слова (їх не можна використовувати як назви функцій чи виразів): `af; else; ic2; plog; and; elseif; if; psi; av; erf; ift; product; args; ev; ilt; put; array; exp; in; rat; at; inf; rk; fix; integrate; step; for; is; sum; col; from; li; then; limit; thru; min; del; get; next; unless; diag; go; not; diff; while; do; ic1; or; zeta`.

Команди вбудованих функцій:

`sin(x)`	$\sin x$	`sinh(x)`	$\mathrm{sh}\ x$
`cos(x)`	$\cos x$	`cosh(x)`	$\mathrm{ch}\ x$
`tan(x)`	$\mathrm{tg}\ x$	`tanh(x)`	$\mathrm{th}\ x$
`cot(x)`	$\mathrm{ctg}\ x$	`coth(x)`	$\mathrm{cth}\ x$
`asin(x)`	$\arcsin x$	`log(x)`	$\ln x$
`acos(x)`	$\arccos x$	`sqrt(x)`	$\sqrt{x}$
`atan(x)`	$\mathrm{arctg}\ x$	`abs(x)`	$\lvert x\rvert$
`acot(x)`	$\mathrm{arcctg}\ x$	`exp(x)`	e^x

Розділ 2

Поліноми, вирази, рівняння

Maxima має широкий набір інструментів для роботи з виразами, зокрема спрощення та інші перетворення: автоматичне розкриття дужок та винесення за дужки; арифметичні дії над елементами поліномів, а також над виразами із вмістом степеневих, показникових та логарифмічних функцій; обробка тригонометричних виразів тощо. Всі ці дії покликані полегшувати легкочитність математичних формул і підвищувати простоту їх сприйняття, при правильному використання даних маніпуляцій вони дозволять заощадити в процесі роботи значну кількість часу.

2.1 Перетворення раціональних виразів

Математики раціональним називають вираз, що складається тільки з арифметичних операторів та піднесення до натурального степеня; природно, елементи таких виразів можуть містити і неарифметичні та нестепеневі функції, тоді такі елементи з точки зору раціональних виразів вважаються атомарними, тобто неподільним і неперетворюваними.

Нехай маємо поліномний вираз $P(x) = a+bx+cx^2+dx^3$. Maxima дозволяє працювати з кожним елементом цього виразу окремо.

```
(%i1)  P:a+b*x+c*x^2+d*x^3;
```

$$(\%o1) \quad d \cdot x^3 + c \cdot x^2 + b \cdot x + a$$

`first(P)`— повертає перший елемент `P`.

```
(%i2)  first(P);
```

$$(\%o2) \quad d \cdot x^3$$

`last(P)` — повертає останній елемент `P`.

```
(%i3)  last(P);
```

$$(\%o3) \quad a$$

`rest(P)` — повертає `P` з вилученим першим елементом.

```
(%i4)  rest(P);
```

$$(\%o4) \quad c \cdot x^2 + b \cdot x + a$$

`part(P,3)` — виділяє третю частину виразу `P`.

```
(%i5)  part(P,3);
```

$$(\%o5) \quad b \cdot x$$

Необхідно звернути увагу, що **Maxima** нумерує частини виразів саме у виведеному блоці `(%o1)`, а не введеному `(%i1)`, потрібно уважно простежити за цим щоб не було помилок.

`ev(expr,arg1,arg2,...)` — основна функція, що обробляє вирази. Вона обчислює вираз `expr` в оточенні, що визначаються аргументами `argN`. Це можуть бути ключі, булеві вирази, присвоєння, рівняння, функції. На функцію `expr` за замовчуваням діє спрощення виразів. Аргументами `argN` можуть бути і вбудовані команди: `expand, factor, trigexpand, trigreduce, diff`.

```
(%i1) ev((a+b)^2,a=2*x,b=[1,2,3]);
```

$$(\%o1)\quad [(2\cdot x+1)^2,(2\cdot x+2)^2,(2\cdot x+3)^2]$$

```
(%i2) ev((a+b)^2,expand);
```

$$(\%o2)\quad b^2+2\cdot a\cdot b+a^2$$

`factor(expr)` — розбиває вираз на множники, виконуючи одночасно дії спрощення, тобто зведення подібних множників чи членів суми, скорочення, розкладання і пониження степеня.

```
(%i3) factor(2^63+1);
```

$$(\%o3)\quad 3^3\cdot 19\cdot 43\cdot 5419\cdot 77158673929$$

```
(%i4) factor(1+exp(3*x));
```

$$(\%o4)\quad (e^x+1)\cdot\left(e^{2\cdot x}-e^x+1\right)$$

```
(%i5) factor((x^2-2*x*y+y^2)/(x^2-y^2));
```

$$(\%o5)\quad -\frac{y-x}{y+x}$$

`gfactor(expr)` — функція, аналогічна до `factor`, лише на полі комплексних чисел.

```
(%i6) gfactor(x^4-y^4);
```

$$(\%o6)\quad -(y-x)\cdot(y+x)\cdot(y-i\cdot x)\cdot(y+i\cdot x)$$

`factorsum(expr)` — факторизує окремі доданки у виразі (а `gfactorsum` — на полі комплексних чисел).

```
(%i7) factorsum(a*x*z^2+a*z^2+2*a*w*x*z+2*a*w*z+
      a*w^2*x+v^2*x+2*u*v*x+u^2*x+a*w^2+v^2+2*u*v+u^2);
```

$$(\%o7)\quad (x+1)\cdot\left(a\cdot(z+w)^2+(v+u)^2\right)$$

```
(%i8) gfactorsum(a^3+3*a^2*b+3*a*b^2+b^3+x^2+2*%i*x*y-y^2);
```

$$(\%o8)\quad (b+a)^3-(y-i\cdot x)^2$$

`expand(expr)` — функція, протилежна до `factor`, вона розкриває дужки, перемножуючи та підносячи до степеня.

```
(%i9) expand((a-b)^4);
```

$$(\%o9)\quad b^4-4\cdot a\cdot b^3+6\cdot a^2\cdot b^2-4\cdot a^3\cdot b+a^4$$

```
(%i10) (sqrt(2)+1)^5;
```

$$(\%o10)\quad \left(\sqrt{2}+1\right)^5$$

```
(%i11) expand(%o10);
```

(%o11) $29 \cdot \sqrt{2} + 41$

```
(%i12) (%o10),numer;
```

(%o12) 82.01219330881976

`expandwrt(expr,x,y,...)` — розкриває дужки не скрізь, а лише відносно тих символів, які задані в списку аргументів.

`combine(expr)` — об'єднує доданки з однаковими знаменниками.

```
(%i13) combine(a/(1+a^2)+b/(1+a^2)+b/(1+a));
```

(%o13) $\dfrac{b+a}{a^2+1} + \dfrac{b}{a+1}$

`xthru(expr)` — зводить вираз до спільного знаменника, не розкриваючи дужок і не намагаючись факторизувати доданки.

```
(%i14) xthru( 1/(a+b)^9+1/(a+b)^11);
```

(%o14) $\dfrac{(b+a)^2+1}{(b+a)^{11}}$

`multthru(mult,expr)` — домножує кожен доданок в сумі `expr` на множник `mult`, причому при множенні дужки не розкриваються.

```
(%i15) multthru ((a-b)^3,1/(a-b)^2+b/(a-b)+1);
```

(%o15) $(a-b)^2 \cdot b - b + (a-b)^3 + a$

`divide(expr1,expr2)` — обчислює частку і остачу від ділення многочлену `expr1` на многочлен `expr2`.

```
(%i16) divide(a^3-3,a-1);
```

(%o16) $[a^2+a+1, -2]$

Останній вираз означає, що

$$\frac{a^3-3}{a-1} = a^2 + a + 1 - \frac{2}{a-1}.$$

`gcd(expr1,expr2,...)` — знаходить найбільший спільний знаменник виразів.

```
(%i17) gcd(a/(a+b),b/(a+b)^2);
```

(%o17) $\dfrac{1}{b^2 + 2 \cdot a \cdot b + a^2}$

`num(expr)` та `denom(expr)` — видає чисельник та знаменник виразу expr.

```
(%i18) G:(a-b)/(a^2+b^2);
```

(%o18) $\dfrac{a-b}{b^2+a^2}$

```
(%i19) num(G);
```

(%o19) $a - b$

```
(%i20) denom(G);
```

(%o20) $b^2 + a^2$

`rat(expr)` — перетворює раціональний вираз на вираз канонічної форми, тобто розкриває всі дужки, потім все зводить до спільного знаменника, підсумовує і скорочує.

```
(%i21) ((x - 2*y)^4/(x^2 - 4*y^2)^2 + 1)*(y + a)*(2*y + x) /
       (4*y^2 + x^2);
```

$$(\%o21)\quad \frac{(y+a)\cdot(2\cdot y+x)\cdot\left(\frac{(x-2\cdot y)^4}{(x^2-4\cdot y^2)^2}+1\right)}{4\cdot y^2+x^2}$$

```
(%i22) rat(%o21);
```

$$(\%o22)/R/\quad \frac{2\cdot y+2\cdot a}{2\cdot y+x}$$

`ratsimp(expr)` — приводить всі частини (в тому числі аргументи функцій) виразів, які не є дробово-раціональною функцією, до канонічного представлення, здійснюючи спрощення, що не виконує команда `rat`. Повторний виклик команди в загальному випадку може змінити результат, тобто не обов'язково спрощення відбувається до кінця.

```
(%i23) ratsimp((x-1)^(3/2)-(x+1)*sqrt(x-1))/sqrt((x-1)*(x+1));
```

$$(\%o23)\quad -\frac{2\cdot\sqrt{x-1}}{\sqrt{(x-1)\cdot(x+1)}}$$

```
(%i24) ratsimp(%o23);
```

$$(\%o24)\quad -\frac{2\cdot\sqrt{x-1}}{\sqrt{x^2-1}}$$

`fullratsimp(expr)` — викликає команду `ratsimp` до тих пір, поки вираз не перестане змінюватися.

```
(%i25) H:(x^(a/2)+1)^2*(x^(a/2)-1)^2/(x^a-1);
```

$$(\%o25)\quad \frac{\left(x^{\frac{a}{2}}-1\right)^2\cdot\left(x^{\frac{a}{2}}+1\right)^2}{x^a-1}$$

```
(%i26) ratsimp(H);
```

$$(\%o26)\quad \frac{x^{2\cdot a}-2\cdot x^a+1}{x^a-1}$$

```
(%i27) fullratsimp(H);
```

$$(\%o27)\quad x^a-1$$

`ratexpand(expr)` — разкриває дужки в виразі `expr`. Відрізняється від `expand` тим, що зводить вираз до канонічної форми.

```
(%i28) J:(x-1)/(x+1)^2+1/(x-1);
```

$$(\%o28)\quad \frac{x-1}{(x+1)^2}+\frac{1}{x-1}$$

```
(%i29) expand(J);
```

$$(\%o29)\quad \frac{x}{x^2+2\cdot x+1}-\frac{1}{x^2+2\cdot x+1}+\frac{1}{x-1}$$

```
(%i30) ratexpand(J);
```

$$(\%o30)\quad \frac{2\cdot x^2}{x^3+x^2-x-1}+\frac{2}{x^3+x^2-x-1}$$

```
(%i31) combine(%o30);
```

$$(\%o31)\quad \frac{2\cdot x^2+2}{x^3+x^2-x-1}$$

`ratsubst(expr1,expr2,expr3)` — команда підстановки виразу `expr1` замість виразу `expr2` у многочлені `expr3`.

```
(%i31) ratsubst (a,x*y^2,x^4*y^3+x^4*y^8);
```

$$(\%o31)\quad a\cdot x^3\cdot y+a^4$$

`partfrac(expr,var)` — перетворює дробово-раціональний вираз `expr` в прості дроби по заданій змінній `var` (це важливо зокрема при інтегруванні складних дробів).

```
(%i32) K:-x/(x^3+4*x^2+5*x+2);
```

$$(\%o32)\quad -\frac{x}{x^3+4\cdot x^2+5\cdot x+2}$$

```
(%i33) partfrac(K,x);
```

$$(\%o33)\quad \frac{2}{x+2}-\frac{2}{x+1}+\frac{1}{(x+1)^2}$$

`at(expr,[x=a,y=b,...])` — підставляє у вираз `expr` замість змінних `x,y,...` вирази `a,b,...`, причому змінні `x,y,...` також можуть бути виразами.

```
(%i34) at(x^2+y^3+z^4,[x^2=a,y^3=b,z^4=c]);
```

$$(\%o34)\quad c+b+a$$

`rhs(expr)`, `lhs(expr)` — відображає відповідно праву та ліву частину виразу `expr`.

2.2 Показникові, логарифмічні вирази

Функція `radcan(expr)` — спрощує вирази, що містять експоненти, логарифми та радикали, шляхом перетворення до форми, що є канонічною для широкого класу виразів. Змінні в виразах впорядковуються. Еквівалентні вирази в цьому класі не обов'язково однакові, але їх різниця застосуванням `radcan` зводиться до нуля.

```
(%i1)  K:(log(x+x^2)-log(x))^a/log(1+x)^(a/2);
```

$$(\%o1)\quad \frac{(\log(x^2+x)-\log(x))^a}{\log(x+1)^{\frac{a}{2}}}$$

```
(%i2)  radcan(K);
```

$$(\%o2)\quad \log(x+1)^{\frac{a}{2}}$$

```
(%i3)  L:(%e^x-1)/(1+%e^(x/2));
```

$$(\%o3)\quad \frac{e^x-1}{e^{\frac{x}{2}}+1}$$

```
(%i4)  radcan(L);
```

$$(\%o4)\quad e^{\frac{x}{2}}-1$$

Функція `logcontract(expr)` — рекурсиво сканує вираз `expr`, перетворюючи вирази виду $a\log b+c\log d+C$ до форми $\log(\texttt{ratsimp}(b^a\cdot d^c))+C$ (числа a та c попередньо необхідно оголосити цілими).

```
(%i5)  declare(n,integer);
```
(%o5) *done*

```
(%i6)  logcontract(3*(2*n*log(x)+3*n*log(y)));
```
(%o6) $\log\left(x^{6\cdot n}\cdot y^{9\cdot n}\right)$

2.3 Тригонометричні вирази

Перетворення тригонометричних виразів здійснюється за допомогою схожих команд, як і у випадку раціональних виразів, лише на початку ставиться префікс `trig`. Для найкращого результату їх слід комбінувати з командами `ratsimp`, `fullratsimp`, `radcan` та ін.

`trigexpand(expr)` — розкладає всі тригонометричні та гіперболічні функції від сум і добутків в комбінації відповідних функцій одиничних кутів та аргументів.

```
(%i1)  F:x+sin(3*x)/sin(x);
```
(%o1) $\dfrac{\sin(3\cdot x)}{\sin(x)}+x$

```
(%i2)  trigexpand(F);
```
(%o2) $\dfrac{3\cdot\cos(x)^2\cdot\sin(x)-\sin(x)^3}{\sin(x)}+x$

```
(%i3)  ratsimp(%o2);
```
(%o3) $-\sin(x)^2+3\cdot\cos(x)^2+x$

```
(%i4)  trigexpand(sin(2*%alpha+3*%beta));
```
(%o4) $\cos(2\cdot\alpha)\cdot\sin(3\cdot\beta)+\sin(2\cdot\alpha)\cdot\cos(3\cdot\beta)$

```
(%i5)  trigexpand(sin(3*%alpha)+cos(4*%beta));
```
(%o5) $\sin(\beta)^4-6\cdot\cos(\beta)^2\cdot\sin(\beta)^2+\cos(\beta)^4-\sin(\alpha)^3+3\cdot\cos(\alpha)^2\cdot\sin(\alpha)$

Зауважимо, що в записі **Maxima** $\sin(\alpha)^4$ позначає $\sin^4\alpha$ тощо.

`trigreduce(expr)` — понижує степінь виразів, згортає всі добутки тригонометричних і гіперболічних функцій в комбінації відповідних функцій від сум. Функція спрацьовує не до кінця, тому повторний виклик може змінити вираз. При записі команди в форматі `trigreduce(expr,var)` — перетворення здійснюються відносно змінних `var`.

```
(%i6)  trigreduce(cos(x)^4+cos(x)^3+cos(x)^2+cos(x)+1);
```
(%o6) $\dfrac{\cos(4\cdot x)+4\cdot\cos(2\cdot x)+3}{8}+\dfrac{\cos(3\cdot x)+3\cdot\cos(x)}{4}+\dfrac{\cos(2\cdot x)+1}{2}+\cos(x)+1$

```
(%i7)  trigreduce(-sin(x)^2+3*cos(x)^2+x);
```
(%o7) $\dfrac{\cos(2\cdot x)}{2}+3\cdot\left(\dfrac{\cos(2\cdot x)}{2}+\dfrac{1}{2}\right)+x-\dfrac{1}{2}$

`trigsimp(expr)` — спрощує вирази, що містять тригонометричні та гіперболічні функції, застосовуючи до них тотожності $\sin^2 x+\cos^2 x=1$ та $\mathrm{ch}^2x-\mathrm{sh}^2x=1$.

```
(%i8)  trigsimp(sin(x)^2+3*cos(x)^2);
```

$$(\%o8) \quad 2\cdot\cos(x)^2+1$$

```
(%i9)  trigreduce(cos(x)^4+sin(x)^4);
```

$$(\%o9) \quad \frac{\cos(4\cdot x)+4\cdot\cos(2\cdot x)+3}{8}+\frac{\cos(4\cdot x)-4\cdot\cos(2\cdot x)+3}{8}$$

```
(%i10) trigsimp(%o9);
```

$$(\%o10) \quad \frac{\cos(4\cdot x)+3}{4}$$

Для гіперболічних функцій:

```
(%i11) trigsimp(sinh(x)^2+3*cosh(x)^2);
```

$$(\%o11) \quad 4\cdot\cosh(x)^2-1$$

```
(%i12) trigsimp(sinh(x)^4-cosh(x)^4);
```

$$(\%o12) \quad 1-2\cdot\cosh(x)^2$$

`trigrat(expr)` — зводить заданий тригонометричний вираз `expr` до канонічної спрощеної квазілінійної форми, причому він розглядається як раціональний, що містить sin, cos, tan, аргументи яких є лінійними формами змінних і π/n (n — ціле). Завжди, коли можливо, заданий вираз лінеаризується.

```
(%i13) trigrat((1+sin(2*%beta)-cos(2*%beta))/sin(%beta));
```

$$(\%o13) \quad 2\cdot\sin(\beta)+2\cdot\cos(\beta)$$

```
(%i14) trigrat(sin(x)^3*(1+cot(x))+cos(x)^3*(1+tan(x)));
```

$$(\%o14) \quad \sin(x)+\cos(x)$$

2.3.1 Пакет `trigtools`

Цей пакет розширює стандартні засоби роботи з тригонометричними функціями, надаючи нові можливості оперування виразами. Його необхідно завантажити додатковою командою `load(trigtools)`.

`c2sin(expr)`, `c2cos(expr)` — конвертують вирази `expr` $= a\cos x + b\sin x$ відповідно до $r\sin(x+\varphi)$ та $r\cos(x-\varphi)$.

```
(%i1)  load(trigtools);
(%i2)  expr:3*sin(x)+4*cos(x);
```

$$(\%o2) \quad 3\cdot\sin(x)+4\cdot\cos(x)$$

```
(%i3)  c2sin(expr);
```

$$(\%o3) \quad 5\cdot\sin\left(x+\operatorname{atan}\left(\frac{4}{3}\right)\right)$$

```
(%i4)  c2cos(expr);
```

$$(\%o4) \quad 5\cdot\cos\left(x-\operatorname{atan}\left(\frac{3}{4}\right)\right)$$

```
(%i5)  c2cos(cos(x)+sin(x));
```

$$(\%o5) \quad \sqrt{2}\cdot\cos\left(x-\frac{\pi}{4}\right)$$

`c2trig(expr)` — конвертує гіперболічні функції до тригонометричних.

```
(%i9)  sinh(x)=c2trig(sinh(x));
       cosh(x)=c2trig(cosh(x));
       tanh(x)=c2trig(tanh(x));
       coth(x)=c2trig(coth(x));
```

$$(\%o6) \quad \sinh(x) = -i\cdot\sin(i\cdot x)$$

$$(\%o7) \quad \cosh(x) = \cos(i\cdot x)$$

$$(\%o8) \quad \tanh(x) = -i\cdot\tan(i\cdot x)$$

$$(\%o9) \quad \coth(x) = i\cdot\cot(i\cdot x)$$

```
(%i10) trigexpand(tan(a+%i*b));
```

$$(\%o10) \quad \frac{i\cdot\tanh(b)+\tan(a)}{1-i\cdot\tan(a)\cdot\tanh(b)}$$

```
(%i11) c2trig(%);
```

$$(\%o11) \quad \tan(i\cdot b+a)$$

`c2hyp(expr)` — конвертує вирази, які містять експоненти, до гіперболічних функцій.

```
(%i12) c2hyp(exp(x));
```

$$(\%o12) \quad \sinh(x)+\cosh(x)$$

```
(%i13) c2hyp(exp(x)+exp(x^2)+1);
```

$$(\%o13) \quad \sinh\left(x^2\right)+\cosh\left(x^2\right)+\sinh(x)+\cosh(x)+1$$

```
(%i14) c2hyp(exp(x)/(2*exp(y)-3*exp(z)));
```

$$(\%o14) \quad \frac{\sinh(x)+\cosh(x)}{2\cdot(\sinh(y)+\cosh(y))-3\cdot(\sinh(z)+\cosh(z))}$$

`trigfactor(expr)` — перетворює усі суми виду $A\sin ax + B\cos bx$ на добутки з відповідними змінами множників та аргументів.

```
(%i15) trigfactor(sin(x)+cos(x));
```

$$(\%o15) \quad \sqrt{2}\cdot\cos\left(x-\frac{\pi}{4}\right)$$

```
(%i16) trigfactor(sin(x)+cos(y));
```

$$(\%o16) \quad 2\cdot\cos\left(\frac{y}{2}-\frac{x}{2}+\frac{\pi}{4}\right)\cdot\cos\left(\frac{y}{2}+\frac{x}{2}-\frac{\pi}{4}\right)$$

```
(%i17) trigfactor(-sin(5*x)-cos(3*y));
```

$$(\%o17) \quad -2\cdot\cos\left(\frac{3\cdot y}{2}-\frac{5\cdot x}{2}+\frac{\pi}{4}\right)\cdot\cos\left(\frac{3\cdot y}{2}+\frac{5\cdot x}{2}-\frac{\pi}{4}\right)$$

```
(%i18) sin(%alpha)+sin(%beta)=trigfactor(sin(%alpha)+sin(%beta));
```

$$(\%o18) \quad \sin(\beta)+\sin(\alpha) = 2\cdot\cos\left(\frac{\beta}{2}-\frac{\alpha}{2}\right)\cdot\sin\left(\frac{\beta}{2}+\frac{\alpha}{2}\right)$$

`trigsolve(eqn,a,b)` — розв'язує рівняння `eqn` на проміжку $x \in [a;b]$.

```
(%i19) trigsolve(sin(3*x-%pi/6)=sqrt(3)/2,-%pi,%pi);
```

$$(\%o19) \quad \{-\frac{\pi}{2}, -\frac{7\cdot\pi}{18}, \frac{\pi}{6}, \frac{5\cdot\pi}{18}, \frac{5\cdot\pi}{6}, \frac{17\cdot\pi}{18}\}$$

```
(%i20) trigsolve(cos(2*x)+sin(x)=1,-2*%pi,2*%pi);
```

$$(\%o20) \quad \{0, -2\cdot\pi, -\frac{11\cdot\pi}{6}, -\frac{7\cdot\pi}{6}, -\pi, \frac{\pi}{6}, \frac{5\cdot\pi}{6}, \pi\}$$

`trigvalue(expr)`, `trigeval(expr)` — відповідно обчислює значення виразів $\sin\frac{k\pi}{n}$, $\cos\frac{k\pi}{n}$, $\mathrm{tg}\,\frac{k\pi}{n}$, $\mathrm{ctg}\,\frac{k\pi}{n}$ у радикалах, та комбінацій цих виразів у радикалах.

```
(%i21) trigvalue(sin(%pi/10));
```

$$(\%o21) \quad \frac{\sqrt{5}-1}{4}$$

```
(%i22) trigvalue(cos(%pi/10));
```

$$(\%o22) \quad \frac{\sqrt{\sqrt{5}+5}}{2^{\frac{3}{2}}}$$

```
(%i23) trigvalue(tan(%pi/10));
```

$$(\%o23) \quad \frac{\sqrt{5-2\cdot\sqrt{5}}}{\sqrt{5}}$$

```
(%i24) trigvalue(cot(%pi/10));
```

$$(\%o24) \quad \sqrt{2\cdot\sqrt{5}+5}$$

```
(%i25) trigeval(cos(%pi/60)+sin(%pi/20));
```

$$(\%o25) \quad \frac{\sqrt{\sqrt{\sqrt{2}\cdot\sqrt{3}\cdot\sqrt{\sqrt{5}+5}+\sqrt{5}+7}+4}}{2^{\frac{3}{2}}} + \frac{\sqrt{4-\sqrt{2}\cdot\sqrt{\sqrt{5}+5}}}{2^{\frac{3}{2}}}$$

2.4 Розв'язування рівнянь

Розв'язання алгебричних рівнянь та їх систем здійснюється за допомогою команди `solve`. Синтаксис наступний:

`solve(eqn=0,x)` — знаходить розв'язок рівняння відносно змінної `x`.

`solve([eqn1=0,eqn2=0,...eqnN=0],[x1,x2,...xN])` — знаходить розв'язок системи рівнянь відносно змінних `x1,x2,...xN`.

```
(%i1)  solve(x^2-2*x-8=0,x);
```

$$(\%o1) \quad [x=-2, x=4]$$

```
(%i2)  solve(x-a/x+b=0,x);
```

$$(\%o2) \quad [x=-\frac{\sqrt{b^2+4\cdot a}+b}{2}, x=\frac{\sqrt{b^2+4\cdot a}-b}{2}]$$

```
(%i3)  solve([x^2+y^2=10,x-y=4],[x,y]);
```

$$(\%o3) \quad [[x=1,y=-3],[x=3,y=-1]]$$

В останньому прикладі **Maxima** видала розв'язок у вигляді списку, оскільки є дві точки, що задовільняють систему рівнянь.

Команда `solve` застосовується і до розв'язання тригонометричних рівнянь. При цьому виникає попередження, що видається лише перший додатний розв'язок, без періодичного продовження, яке користувачу треба буде знайти самостійно.

```
(%i4)  solve(2*sin(x-%pi/4)=sqrt(3));
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

$$(\%o4) \quad [x = \frac{7 \cdot \pi}{12}]$$

```
(%i5)  solve(sin(x)^2+2*sin(x)+1=0,x);
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

$$(\%o5) \quad [x = -\frac{\pi}{2}]$$

Команда `solve` у **Maxima** шукає корені і на полі комплексних чисел.

```
(%i6)  solve(x^2+x+1=0,x);
```

$$(\%o6) \quad [x = -\frac{\sqrt{3} \cdot i + 1}{2}, x = \frac{\sqrt{3} \cdot i - 1}{2}]$$

Якщо є потреба знайти корені в числовому вигляді, в кінці необхідно додати команду `numer`.

```
(%i7)  solve(x^2+3*x+1=0,x),numer;
rat: replaced 2.2360 by 16692641/7465176 = 2.2360
rat: replaced 2.2360 by 16692641/7465176 = 2.2360
```

$$(\%o7) \quad [x = -2.618033988749896, x = -0.3819660112501031]$$

`allroots(eqn=0,x)` обчислює всі дійсні і комплексні корені алгебричних рівнянь.

```
(%i8)  allroots((1+2*x)^3=10*(1+x^5),x);
```

$$(\%o8) \quad [x = 0.6057379117337299, x = 0.9658759245461883 \cdot i - 0.4382771080081766,$$

$$x = -0.9658759245461883 \cdot i - 0.4382771080081766, x = -1.021758604074856,$$

$$x = 1.292574908357479]$$

`realroots(eqn=0,num)` шукає всі дійсні корені (комплексні не беруться до уваги) поліноміального виразу з точністю до числа `num`.

```
(%i10) realroots(-1-x+x^5=0,6.0e-6);
```

$$(\%o10) \quad [x = \frac{612003}{524288}]$$

Хоч `solve` і розв'язує широкий клас рівнянь, ця команда все ж пасує перед трансцендентними рівняннями, тобто такими, де присутні як поліномні вирази, так і експоненціальні чи тригонометричні. Для прикладу, рівняння $\sin x = x/2$ та $e^x = x^2$ дадуть такі результати:

```
(%i11) solve(sin(x)=x/2,x);
```

(%o11) $[x = 2 \cdot \sin(x)]$

```
(%i12) solve(exp(x)=x^2,x);
```

(%o12) $[x = -e^{\frac{x}{2}}, x = e^{\frac{x}{2}}]$

На такий випадок існує дуже корисна команда, яка числово шукає корені в певних межах змінної, використовуючи метод Ньютона наближеного обчислення. Для використання цього методу потрібно, щоб функція, яка описує даний вираз (`eqn1-eqn2`), на краях проміжка `[a,b]` приймала протилежні за знаком значення. Якщо ця умова не буде дотримуватись, система видасть попередження. Щоб дізнатись, в яких межах приблизно знаходиться корінь, корисно спочатку побудувати графіки обох виразів `eqn1` та `eqn2` та подивитись на точку їх перетину (про побудову графіків докладніше у Розділі 4).

`find_root(eqn1=eqn2,var,a,b)` — шукає наближені корені рівняння `eqn1=eqn2` за змінною `var` на інтервалі `[a,b]`.

Накреслимо графіки виразів на різних сторонах наших рівнянь. Бачимо, що корінь

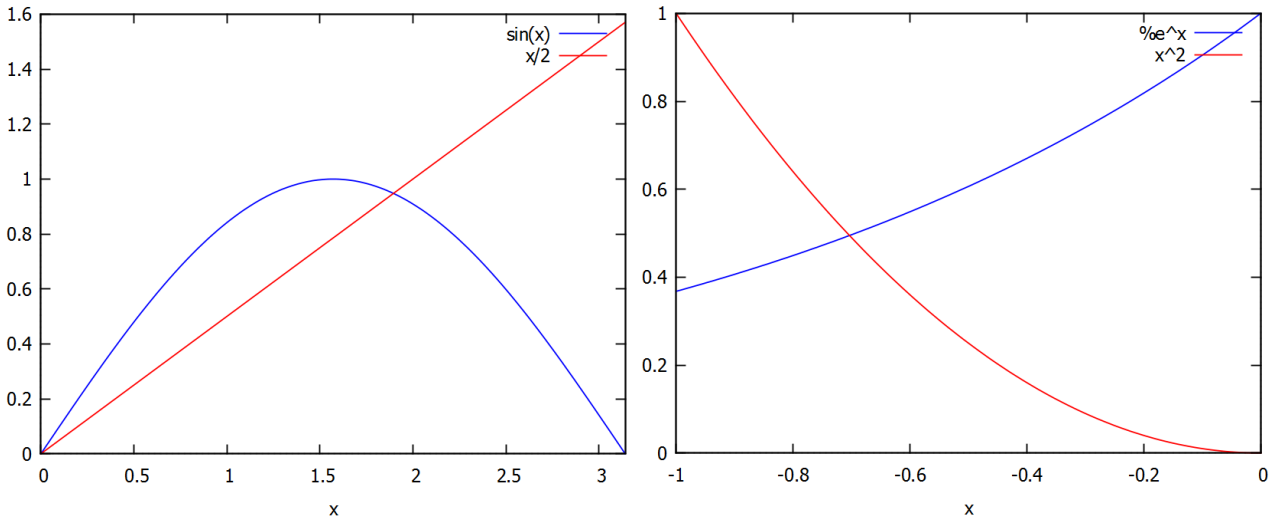


Рис. 2.1: Графіки лівих та правих частин трансцендентних рівнянь.

першого рівняння лежить у проміжку $[0, \pi]$, другого у проміжку $[-1, 0]$. Але якщо ми впишемо обидва ці проміжки в команду `find_root`, вона обидва рази дасть тривіальний корінь 0, тому будемо обмежувати проміжки з 0.1.

```
(%i13) find_root(sin(x)=x/2,x,0.1,%pi);
```

(%o13) 1.895494267033981

```
(%i14) find_root(exp(x)=x^2,x,-1,-0.1);
```

(%o14) −0.7034674224983917

2.5 Нерівності

Для розв'язання нерівностей у **Maxima** існує досить добре розроблений пакет `solve_rat_ineq`, який працює з раціональними та дробово-раціональними виразами. Системи нерівностей він не розглядає, лише окремі вирази. Пакет відразу включений до дистрибутиву **Maxima** і його потрібно завантажити командою `load`.

```
(%i1)  load(solve_rat_ineq);
(%i2)  solve_rat_ineq(x^2-3*x-4<0);
```

(%o2) $[[x > -1, x < 4]]$

```
(%i3)  solve_rat_ineq((x^2-1)/(x^2+x-6)>0);
```

(%o3) $[[x < -3], [x > -1, x < 1], [x > 2]]$

Для розв'язання нерівностей корисною також є команда `factor`, що розбиває вираз на множники, які вже можна аналізувати на зміну знаку функції при застосуванні методу інтервалів.

Якщо у нерівності не існує раціональних коренів, `solve_rat_ineq` шукає їх наближеними числовими методами.

```
(%i4)  solve_rat_ineq(x^2>x^3+1);
```

(%o4) $[[x < -0.754877616175949]]$

```
(%i5)  solve_rat_ineq(x^5+x^3<x^2+x);
```

(%o5) $[[x < -0.5698403072569234], [x > 0, x < 1]]$

2.6 Завдання до Розділу 2

Числові задачі

2.1. Що більше: π^e чи e^π? Знайти на скільки процентів одне з цих чисел більше за інше з точністю до 19-го знаку після коми.

2.2. П'єр Ферма вважав, що всі числа виду $2^{2^n}+1$, де n натуральне, є простими. Знайти n, починаючи з якого це твердження перестає справджуватись.

2.3. Записати поліном $G(x) = 8x^5 - 45x^3 + 56x^2 - 7x + 11$. Вивести на екран: перший доданок, останній доданок, третій доданок, всі доданки крім першого.

2.4. Знайти найменше n, при якому число $10^{10}+n$ буде простим.

2.5. Встановити, що вираз $4^n+15n-1$ кратний до 9 для перших семи натуральних чисел.

2.6. Встановити, що вираз $6^{2n}+3^{n+2}+3^n$ кратний до 11 для перших перших п'яти натуральних чисел.

2.7. Знайти остачу від ділення $6x^4 - 7x^3 + 2x + 11$ на $3x^2 - 1$. Піднести цю остачу до 4-го степеня та розкрити дужки.

2.8. Знайти точне та наближене значення виразу $(\sqrt{11}+5)^2$ з точністю до 23 знаку після коми.

2.9. Обчислити довжину дуги кола радіуса 6 см та розхилом 23° з точністю до 45-ої цифри після коми.

2.10. Показати, що вираз $11^{n+2} + 12^{2n+1}$ ділиться на 133 без остачі для перших п'яти натуральних чисел.

Спростити вирази (алгебра)

2.11. $\dfrac{x^4-x^3-11x^2+9x+18}{x^4-3x^3-7x^2+27x-18} \div \dfrac{x^3-9x^2+26x-24}{x^3-8x^2+19x-12}$;

2.12. $\dfrac{3x^4-24x^3-3x^2+204x-252}{-x^5+10x^4-15x^3-70x^2+220x-168} \cdot \dfrac{2-x}{x+1}$;

2.13. $\dfrac{x^3+2x^2+4x+8}{x^5+5x^4-16x-80} \cdot \dfrac{2x^4+10x^3-16x-80}{x^2+2x+4}$;

2.14. $\dfrac{2x^4+10x^3-2x-10}{x^2+x+1} \cdot \dfrac{x^3+x^2+x+1}{x^5+5x^4-x-5}$;

2.15. $\dfrac{x^5+4x^4-81x-324}{3x^4+10x^3-81x-270} \cdot \dfrac{3x^3+19x^2+57x+90}{x^4+7x^3+21x^2+63x+108}$;

2.16. $\dfrac{x^3+3x^2-9x-27}{x^3-5x^2-15x-72} \cdot \dfrac{x^4-8x^3-27x+216}{49x^4-882x^2+3969}$;

2.17. $\dfrac{7x^4-126x^2+567}{x^5-8x^4-27x^2+216x} \cdot \dfrac{x^3-5x^2-15x-72}{x^3+3x^2-9x-27}$;

2.18. $\dfrac{4x^4+4x^3-48x^2-112x-64}{2x^3+4x^2-32x-64} \div \dfrac{x^2+3x+2}{x+4}$;

2.19. $\dfrac{x^4+x^3-7x^2-x+6}{5x^4+10x^3-100x^2-330x-225} \cdot \dfrac{x^3-2x^2-15x}{x^2-3x+2}$;

2.20. $\dfrac{x^3+2x^2+4x+8}{x^5+5x^4-16x-80} \cdot \dfrac{3x^4+10x^3-16x-80}{x^2+2x+4}$.

Спростити вирази (тригонометрія)

2.21. $\operatorname{ctg}^2 x - \dfrac{1+\operatorname{ctg} x+\operatorname{ctg}^2 x}{1+\operatorname{tg} x+\operatorname{tg}^2 x}$;

2.22. $\dfrac{1-2\sin 3x\cos(-3x)}{\cos 3x+\sin(-3x)}$;

2.23. $\cos^4 x-\cos^2 x+\sin^2 x$;

2.24. $\dfrac{\cos x}{1-\sin x}+\dfrac{1-\sin x}{\cos x}$;

2.25. $\dfrac{\sin^2 x\operatorname{tg}^2 x}{\operatorname{tg}^2 x+\cos^2 x-1}$;

2.26. $\operatorname{tg} x+\dfrac{\cos x}{1+\sin x}$;

2.27. $\dfrac{\cos^3 x-\sin^3 x}{1+\sin x\cos x}$;

2.28. $\sin^6 x+\cos^6 x+3\sin^2 x\cos^2 x$;

2.29. $\dfrac{\sin x+2\sin 2x+\sin 3x}{\cos x+2\cos 2x+\cos 3x}$;

2.30. $\dfrac{\sin x+\operatorname{tg} x}{1+\cos x}$.

Розв'язати рівняння (тригонометрія)

2.31. $\cos(2x+30^\circ)=-1$;

2.32. $\dfrac{\sqrt{3}}{\operatorname{tg}(4x+\pi/6)}=3$;

2.33. $\cos^2(x+\pi/6)=1/2$;

2.34. $\sin(\pi/4-x)=-\sqrt{3}/2$;

2.35. $\cos(\pi/9-4x)=1$;

2.36. $\operatorname{tg}(x/4-15^\circ)=\sqrt{3}$;

2.37. $2\cos(x/2+\pi/6)-\sqrt{3}=0$;

2.38. $\operatorname{ctg}(x/4-\pi/4)-1=0$;

2.39. $\sin(x/2+\pi/6)=1/2$;

2.40. $\cos(4x-\pi/6)=-\sqrt{3}/2$.

Розв'язати рівняння числово (скористатись відомостями з Розділу 4)

2.41. $\ln^2(x-1) = 3\cos 2x + 1;$

2.42. $\dfrac{10}{1+x^2} = 2\sin 2x + x;$

2.43. $\dfrac{10x-2}{3+x^2} = \sqrt[4]{x};$

2.44. $\dfrac{10}{1+x^2} = 2\cos 2x + x;$

2.45. $\sin x \cdot \sqrt{(81-x^2)} = 5\operatorname{arctg} x;$

2.46. $\sqrt{(25-x^2)} = \operatorname{arctg} 2x;$

2.47. $\sin^2 x \cdot \sqrt{(81-x^2)} = 5e^{-x^2};$

2.48. $\dfrac{x^2-9}{x^2+4} = \sqrt{(x^2+1)} \cdot e^{-x};$

2.49. $x + \sqrt{x} = e^{-x^2};$

2.50. $\sqrt{x} \cdot \cos x = \ln\left(x^2+1\right);$

Розділ 3

Робота з числовими масивами

3.1 Списки

Списки — основні об'єкти оперування для **Maxima** та **Lisp**. Щоб задати список, достатньо записати його елементи через кому і обрамити квадратними дужками. Список може бути порожнім або складатись із одного елементу.

```
(%i1)  list1:[1,2,3,x,a+b];
```

$$(\%o1) \quad [1, 2, 3, x, b + a]$$

```
(%i2)  list2:[];
```

$$(\%o2) \quad []$$

Елементом списку може бути й інший список.

```
(%i3)  list3:[1,2,[3,4],[5,6,7]];
```

$$(\%o3) \quad [1, 2, [3, 4], [5, 6, 7]]$$

Посилання на елемент списку здійснюється за номером цього елементу у списку.

```
(%i4)  list3[3];
```

$$(\%o4) \quad [3, 4]$$

```
(%i5)  list3[4][2];
```

$$(\%o5) \quad 6$$

`length(list)` — повертає кількість елементів списку (при цьому самі елементи можуть бути достатньо складними конструкціями).

```
(%i6)  length(list3);
```

$$(\%o6) \quad 4$$

`copylist(list)` — дає копію списку `list`.

```
(%i7)  list4:copylist(list3);
```

$$(\%o7) \quad [1, 2, [3, 4], [5, 6, 7]]$$

Команда `makelist` створює список, кожний елемент якого генеруеться з деякого виразу. Можливі два варіанти виклику команди:

`makelist(expr,i,i1,iN)` — повертає список, `j`-й елемент якого рівний `ev(expr,i=j)`, при цьому індекс `j` міняється від `i1` до `iN`.

```
(%i8)  makelist(concat(x,i),i,1,6);
```

(%o8) $[x1, x2, x3, x4, x5, x6]$

```
(%i9)  makelist(a*i^2,i,1,5);
```

(%o9) $[a, 4 \cdot a, 9 \cdot a, 16 \cdot a, 25 \cdot a]$

`makelist(expr,x,list)` — повертає список, `j`-й елемент якого рівний `ev(expr,x=list[j])`, при цьому індекс `j` міняється від `1` до `length(list)`.

```
(%i10) list5:[1,2,3,4,5,6,7];
```

(%o10) $[1, 2, 3, 4, 5, 6, 7]$

```
(%i11) makelist(exp(i),i,list5);
```

(%o11) $[e, e^2, e^3, e^4, e^5, e^6, e^7]$

Аналогічні дії виконує команда `create_list(expr,x1,list1,...,xn,listN)` — вона будує список шляхом обчислення виразу `expr`, що залежить від `x1`, до кожного елементу списку `list1` (аналогічно `expr`, що залежить від `x2`, застосовується до `list2` і т.д.)

```
(%i12) create_list(x^i,i,[1,3,7]);
```

(%o12) $[x, x^3, x^7]$

`append(list1,list2,...listN)` — послідовно об'єднує списки.

```
(%i13) append([1],[2,3],[4,5,6,7]);
```

(%o13) $[1, 2, 3, 4, 5, 6, 7]$

`join(list1,list2)` — створює новий список з двох, компонуючи їх елементи почергово в порядку слідування. Новий список містить `list1_1`, потім `list2_1`, потім `list1_2`, `list2_2` і т.д. Якщо кількість елементів у списках не рівна, створений список обривається на останньому спільному номері елементу.

```
(%i14) join([1,2,3],[10,20,30,40]);
```

(%o14) $[1, 10, 2, 20, 3, 30]$

`reverse(list)` — міняє порядок елементів списку на зворотній.

```
(%i15) list6:[11,12,x-y,a+b,[15,16]];
```

(%o15) $[11, 12, x - y, b + a, [15, 16]]$

```
(%i16) reverse(list6);
```

(%o16) $[[15, 16], b + a, x - y, 12, 11]$

`member(list1,list2)` — повертає `«true»` якщо `list1` є елементом списку `list2`, або `«false»` в протилежному випадку.

```
(%i17) member(b,[a,b,[b,c]]);
```

(%o17) true

```
(%i18) member(b,[[a,b],[b,c]]);
```

(%o18) false

`first(list)` — виводить перший елемент списку, `last(list)` — останній елемент, `rest(list)` — видає залишок списку після видалення першого елементу, `rest(list,n)` — залишок списку після видалення перших n елементів.

```
(%i19) list7:[1,2,3,4,a,b];
```

(%o19) $[1, 2, 3, 4, a, b]$

```
(%i20) first(list7);
```

(%o20) 1

```
(%i21) last(list7);
```

(%o21) b

```
(%i22) rest(list7);
```

(%o22) $[2, 3, 4, a, b]$

```
(%i23) rest(list7,2);
```

(%o23) $[3, 4, a, b]$

`sum(list,i,i1,iN)` — підсумовує значення списку `list` при зміні индексу `i` від `i1` до `iN`.

```
(%i24) sum(i^2,i,1,5);
```

(%o24) 55

```
(%i25) sum(x+i*(i+1)/2,i,1,4);
```

(%o25) $4 \cdot x + 20$

`product(list,i,i1,iN)` — перемножує значення списку `list` при зміні индексу `i` від `i1` до `iN`.

```
(%i26) product(i^2,i,1,5);
```

(%o26) 14400

```
(%i27) product(x+i*(i+1)/2,i,1,4);
```

(%o27) $(x + 1) \cdot (x + 3) \cdot (x + 6) \cdot (x + 10)$

Існує корисна команда `map(F,expr1,...,exprN)`, яка дозволяє застосувати функцію (оператор, символ операції) `F` до виразів `expr1,...,exprN`. При її застосуванні до списків F діє на кожен елемент списку. Необхідно звернути увагу, що F — це саме ім'я функції (без вказування змінних, від яких вона залежить). Функція F може бути і заданою користувачем.

```
(%i28) map(ratsimp,x/(x^2+x)+(y^2+y)/y);
```

(%o28) $y + \dfrac{1}{x+1} + 1$

```
(%i29) map("=",[a,b],[-2,3]);
```
(%o29) $[a = -2, b = 3]$

```
(%i30) map(exp,[0,1,2,3,4,5]);
```
(%o30) $[1, e, e^2, e^3, e^4, e^5]$

```
(%i31) f(x):=x^3;
```
(%o21) $\mathrm{f}\,(x) := x^3$

```
(%i32) map(f,[1,2,3,4,5]);
```
(%o32) $[1, 8, 27, 64, 125]$

3.2 Масиви

Масиви в **Maxima** — сукупності однотипних об'єктів з індексами. Кількість індексів не повинно перевищувати п'яти. У **Maxima** існують і функції з індексами (функції масиву). Можливе створення та використання змінних з індексами до оголошення відповідного масиву, тоді такі змінні розглядаються як елементи масивів з невизначеними розмірами (так звані хеш-масиви). Розміри неозначених масивів зростають динамічно в міру надання значень елементам. Індекси масивів з невизначеними межами не обов'язково мають бути числами. Для підвищення ефективності обчислень рекомендується перетворювати масиви з невизначеними межами на звичайні масиви (для цього використовується команда `array`).

Створення масиву виконується функцією `array`. Синтаксис звернення до команди:

```
array(name,dim_1,...,dim_N)
array(name, type, dim_1,...,dim_N)
array([name_1,...,name_M],dim_1,...,dim_N)
```

```
(%i1)  array(a,1,1);
```
(%o1) a

```
(%i5)  a[0,0]:0; a[0,1]:1; a[1,0]:2; a[1,1]:3;
```
(%o2) 0

(%o3) 1

(%o4) 2

(%o5) 3

```
(%i6)  listarray(a);
```
(%o6) $[0, 1, 2, 3]$

Команда `listarray`, використана в прикладі, перетворює масив у список. Синтаксис виклику: `listarray (A)`. Аргумент `A` може бути означеним або неозначеним масивом, функцією масиву або функцією з індексами. Порядок включення елементів масиву до списку — рядками.

`arrayinfo (A)` — виводить інформацію про масив `A`. Аргумент `A`, як і у випадку `listarray`, може бути означеним або неозначеним масивом, функцією масиву чи функцією з індексами.

```
(%i8)  array(AR,2,3);
```
(%o8) AR

```
(%i9)  AR[2,3]:%pi;
```
(%o9) π

```
(%i10) AR[1,2]:%e;
```
(%o10) e

```
(%i11) arrayinfo(AR);
```
(%o11) $[declared, 2, [2, 3]]$

```
(%i12) ARR[foo]:(a+b)^2;
```
(%o12) $(b + a)^2$

```
(%i13) ARR[bar]:(a-b)^3;
```
(%o13) $(a - b)^3$

```
(%i14) arrayinfo(ARR);
```
(%o14) $[hashed, 1, [bar], [foo]]$

```
(%i15) listarray(ARR);
```
(%o15) $[(a - b)^3, (b + a)^2]$

Команда `make_array(type,dim_1,...,dim_N)` створює та повертає масив Lisp. Тип масиву може бути `any, flonum, fixnum, hashed, functional`. Індекс `i` може змінюватися в межах від `0` до `dim_i-1`. Переваги `make_array` в порівнянні з `array` — можливість динамічно керувати розподілом пам'яті для масивів. Присвоювання `y:make_array(...)` створює посилання на масив. Коли масив більше не потрібний, посилання знищується присвоюванням `y:false`, пам'ять звільняється потім очищувачем сміття.

```
(%i16) A1:make_array(fixnum,10);
```
(%o16) $Lisparray[10]$

```
(%i17) A1[1]:8;
```
(%o17) 8

```
(%i18) A2:make_array(any,10);
```
(%o18) $Lisparray[10]$

```
(%i19) arrayinfo(A2);
```
(%o19) $[declared, 1, [9]]$

Команда `fillarray(array1,array2)` дозволяє заповнювати масиви значеннями з іншого масиву чи списку. Заповнення провадиться по рядках.

```
(%i20) array(a,1,1);
```
(%o20) a

```
(%i21) fillarray(a,[1,2,3,4]);
```
(%o21) a

```
(%i22) a[1,1];
```
(%o22) 4

```
(%i23) a2:make_array(fixnum,8);
```
(%o23) $Lisparray[8]$

```
(%i24) fillarray(a2,[1,2,3,4,5]);
```
(%o24) $Lisparray[8]$

Як видно з розглянутих прикладів, довжина списку може і не збігатися з розмірністю масиву. Якщо вказано тип масиву, він повинен заповнюватись елементами того самого типу. Вилучення масивів з пам'яті здійснюється функцією `remarray(array)`.

3.3 Матриці

У **Maxima** означені прямокутні матриці. Основний спосіб створення матриць — використання функції `matrix`. Синтаксис виклику:

`matrix(row1,...,rowN)`.

Кожен рядок — список виразів, усі рядки однакової довжини. На множині матриць означені операції додавання, віднімання, множення та ділення. Ці операції виконуються поелементно, якщо операнди — дві матриці, скаляр та матриця або матриця та скаляр. Піднесення до степеня можливе, якщо один із операндів — скаляр. Перемноження матриць (загалом некомутативна операція) позначається символом «`.`». Операція множення матриці на себе може розглядатися як зведення в степінь. Піднесення до степеня «-1» означає отримати обернену матриці (якщо це можливо).

```
(%i1)  M:matrix([4,3],[-2,1]);
```
(%o1) $\begin{pmatrix} 4 & 3 \\ -2 & 1 \end{pmatrix}$

```
(%i2)  N:matrix([1,-2],[2,-3]);
```
(%o2) $\begin{pmatrix} 1 & -2 \\ 2 & -3 \end{pmatrix}$

Виконання арифметичних операцій з матрицями.

```
(%i3)  M+N;
```
(%o3) $\begin{pmatrix} 5 & 1 \\ 0 & -2 \end{pmatrix}$

```
(%i4)  M-N;
```

$$(\%o4)\quad \begin{pmatrix} 3 & 5 \\ -4 & 4 \end{pmatrix}$$

```
(%i5)  M*N;
```

$$(\%o5)\quad \begin{pmatrix} 4 & -6 \\ -4 & -3 \end{pmatrix}$$

```
(%i6)  M/N;
```

$$(\%o6)\quad \begin{pmatrix} 4 & -\frac{3}{2} \\ -1 & -\frac{1}{3} \end{pmatrix}$$

Необхідно звернути увагу — ці операції здійснюються поелементно. При спробі виконати ці арифметичні операції над матрицями різних розмірів видається помилка.

Приклад операцій з матрицями і скалярами:

```
(%i7)  M^2;
```

$$(\%o7)\quad \begin{pmatrix} 16 & 9 \\ 4 & 1 \end{pmatrix}$$

```
(%i8)  2^M;
```

$$(\%o8)\quad \begin{pmatrix} 16 & 8 \\ \frac{1}{4} & 2 \end{pmatrix}$$

Знову ж таки, при роботі з матрицями в **Maxima** необхідно бути уважними: вираз `M^2` зовсім не означає матричне помноження $M \cdot M$, а лише піднесення кожного елементу матриці до квадрата.

Для здійснення матричного множення використовується інший оператор — нижня крапка «`.`»:

```
(%i9)  M.N;
```

$$(\%o9)\quad \begin{pmatrix} 10 & -17 \\ 0 & 1 \end{pmatrix}$$

```
(%i10) N.M;
```

$$(\%o10)\quad \begin{pmatrix} 8 & 1 \\ 14 & 3 \end{pmatrix}$$

Очевидно, що для успішного перемноження матриці повинні бути узгоджені за розмірами (кількість стовпців першої рівна кількості рядків другої).

Обернену матрицю дозволяє знайти оператор «`^^`». Його не слід плутати з оператором «`^`», який видає матрицю з оберненими відповідними елементами.

```
(%i11) M^^(-1);
```

$$(\%o11)\quad \begin{pmatrix} \frac{1}{10} & -\frac{3}{10} \\ \frac{1}{5} & \frac{2}{5} \end{pmatrix}$$

```
(%i12) M^(-1);
```

$$(\%o12)\quad \begin{pmatrix} \frac{1}{4} & \frac{1}{3} \\ -\frac{1}{2} & 1 \end{pmatrix}$$

```
(%i13) M.(%o11);
```

$$(\%o13)\quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
(%i14) M*(%o12);
```

$$(\%o14)\quad \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Команда `genmatrix` повертає матрицю заданої розмірності, що складена з елементів двоіндексного масиву. Синтаксис виклику:

```
genmatrix (a, i2, j2, i1, j1)
genmatrix (a, i2, j2, i1)
genmatrix (a, i2, j2)
```

Індекси `i1`, `j1` и `i2`, `j2` вказують на лівий та правий нижній елементи матриці в вихідному масиві.

```
(%i15) h[i,j]:=1/(i+j-1);
```

$$(\%o15)\quad h_{i,j} := \frac{1}{i+j-1}$$

```
(%i16) genmatrix(h,3,3);
```

$$(\%o16)\quad \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

```
(%i17) array(A,fixnum,2,2);
```

$$(\%o17)\quad A$$

```
(%i18) A[1,1]:%e;
```

$$(\%o18)\quad e$$

```
(%i19) A[2,2]:%pi;
```

$$(\%o19)\quad \pi$$

```
(%i20) genmatrix(A,2,2);
```

$$(\%o20)\quad \begin{pmatrix} e & 0 \\ 0 & \pi \end{pmatrix}$$

`zeromatrix(M,N)` — повертає матрицу заданої размірності, складену з нулів.

```
(%i21) zeromatrix(2,2);
```

$$(\%o21)\quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

`ident(N)` — повертає одиничну матрицю заданоє розмірності $\mathrm{N} \times \mathrm{N}$.

```
(%i22) ident(2);
```

$$(\%o22)\quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

`copymatrix(M)` — створює копію матриці `M`.

Функції `row` і `col` дозволяють отримати відповідно рядок і стовпець заданої матриці, отримуючи список. Синтаксис виклику:

`row(M,i)` — повертає `i`-й рядок;

`col(M,i)` — повертає `i`-й стовпець.

Функції `addrow` та `addcol` додають до матриці рядок або стовпець відповідно. Синтаксис виклику:

`addrow(M,list1,...,listN)` — додає рядки;
`addcol(M,list1,...,listN)` — додає стовпці.

```
(%i24) K:matrix([1,2],[3,4]);
```

$$(\%o24)\quad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
(%i25) L:addrow(K,[10,20]);
```

$$(\%o25)\quad \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 10 & 20 \end{pmatrix}$$

```
(%i26) addcol(L,[x,y,z]);
```

$$(\%o26)\quad \begin{pmatrix} 1 & 2 & x \\ 3 & 4 & y \\ 10 & 20 & z \end{pmatrix}$$

Функція `submatrix` повертає нову матрицю, що складається з заданої підматриці. Синтаксис виклику:

`submatrix(i1,...,iN, M,j1,...,jM);`
`submatrix(i1,...,iN,M);`
`submatrix(M,j1,...,jM).`

Підматриця будується так: з матриці `M` видаляються рядки `i1,...,iN` і `j1,...,jM`. Приклад (використовуємо останній результат з попереднього прикладу, видаляємо перший рядок та другий стовпець):

```
(%i27) submatrix(1,(%o26),2);
```

$$(\%o27)\quad \begin{pmatrix} 3 & y \\ 10 & z \end{pmatrix}$$

Для заповнення матриці значеннями деякої функції використовується команда `matrixmap` (аналог `map, apply, fullmap`). Синтаксис виклику:

`matrixmap(f,M)` — повертає матрицю з елементами `i,j`, рівними `f(M[i,j])`.

```
(%i28) D:matrix([1,2],[3,4]);
```

$$(\%o28)\quad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
(%i29) f(x):=x^3;
```

$$(\%o29)\quad \mathrm{f}\,(x) := x^3$$

```
(%i30) matrixmap(f,D);
```

$$(\%o30)\quad \begin{pmatrix} 1 & 8 \\ 27 & 64 \end{pmatrix}$$

Існує ще багато команд для роботи з матрицями, їх детальний розгляд здійснено в Розділі 6, при розв'язанні задач лінійної алгебри.

3.4 Завдання до Розділу 3

Протабулювати функцію для будь-яких 30 значень x, результати вивести у вигляді списку числових значень y

3.1. $f(x) = \dfrac{10}{x}$;

3.2. $f(x) = \cos x^3$;

3.3. $f(x) = e^{\frac{1}{x}}$;

3.4. $f(x) = x^2 \ln x$;

3.5. $f(x) = \dfrac{1}{\sin x}$;

3.6. $f(x) = \operatorname{ctg} x + \cos x$;

3.7. $f(x) = (10 - x^2)^2$;

3.8. $f(x) = \operatorname{tg} \dfrac{1}{\sqrt{x}}$;

3.9. $f(x) = x^2 \sin x \cos x$;

3.10. $f(x) = \dfrac{5}{x} + \dfrac{x}{5}$.

Зґенерувати матриці 4×4 за поданими формулами, до них додати рядки та стовпці будь-яких чисел, щоб утворились матриці 6×6

3.11. $h_{i,j} = \dfrac{1}{i^2 + j^2}$;

3.12. $h_{i,j} = i + \dfrac{1}{j}$;

3.13. $h_{i,j} = \cos i + \sin j$;

3.14. $h_{i,j} = e^{i-j}$;

3.15. $h_{i,j} = \operatorname{arctg} i + \operatorname{arcctg} j$;

3.16. $h_{i,j} = \dfrac{1}{i^2 + j^2}$;

3.17. $h_{i,j} = \dfrac{1}{2 - i + j}$;

3.18. $h_{i,j} = (i + j)^2$;

3.19. $h_{i,j} = \ln(i \cdot j)$;

3.20. $h_{i,j} = 0.18i^3 - 1.16j$.

Розділ 4

Графічні можливості Maxima

Для побудови графіків **Maxima** використовує **gnuplot**, який викликається автоматично при заданні відповідних команд. Взагалі в **Maxima** можливі два методи для побудови графіків:

1. Задання стандартних графічних команд `plot2d, plot3d` викликає спливаюче вікно **gnuplot**, що містить необхідний графік. Продовжувати роботу з **Maxima** в цей час неможливо, поки вікно не закриється. Вихідне вікно **gnuplot** та відповідний графік можна масштабувати, використовуючи мишку. Також за допомогою мишки у 2d-графіках можна робити вимірювання, а 3d-графіки можливо обертати у всіх напрямках.

2. Добавляючи вираз `«wx»` до імен графічних команд (`plot2d, plot3d`), створюється малюнок формату .png, який вбудований прямо у вікно **wxMaxima**. Оскільки графік залишається видимим протягом усього сеансу **wxMaxima**, цей метод корисний для інтерактивної роботи. Права кнопка мишки на графіку дозволяє копіювати в буфер обміну або зберігати як файл. Тим не менш, з причини своєї низької роздільної здатності, подальше використання графіків, створених таким чином, не є доцільним.

4.1 Двовимірні графіки

Команда `plot` — це стандартний інтерфейс **gnuplot** від **Maxima**, зручний у застосуванні, але не дуже гнучкий щодо зміни вигляду графіків. При використанні цього інтерфейсу неможливо змінити зовнішній вигляд, формат виводу або вихідний графік на консолі **gnuplot** після того, як графік було побудовано.

Основні команди графічного інтерфейсу `Plot`:

`plot2d(f(x),[x,a,b],opts)` — будує графік функції $f(x)$ на проміжку $x \in [a,b]$ за допомогою опцій `opts`.

`plot2d([discrete,[x_list],[y_list]],opts)` — будує точки вказаних значень `[x_list],[y_list]` двох окремих списків, що позначають відповідно x- та y-координати точок.

`plot2d([discrete,[list]],opts)` — інший спосіб побудови точок, коли вони задані у вигляді послідовного списку `[x1,y1],[x2,y2]...`

`plot2d([parametric,x(t),y(t),[t,t1,t2]],opts)` — будує параметричну криву з параметром $t \in [t_1, t_2]$.

Побудуємо простий графік функції з заданими межами по x; зміна по осі y обчислюється автоматично (Рис. 4.1).

```
(%i1)  plot2d(sin(x),[x,0,2*%pi]);
(%o1)     [C : /Users/orreg/maxout.gnuplot]
```

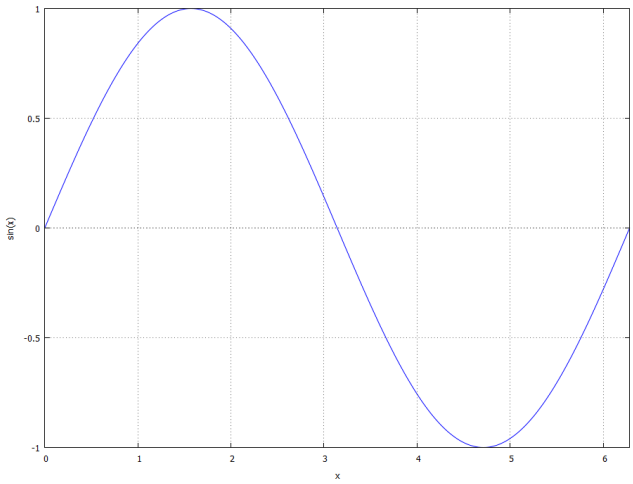


Рис. 4.1: Простий графік функції.

Команда `(wx)plot2d` малює криву або відрізки лінії між парами точок у двовимірній декартовій системі координат. Для цього є три можливості, які можуть бути об'єднані в одній діаграмі разом:

1. Функція $y = f(x)$; відображення діапазону вздовж осі x оголошується як `[x,a,b]`, оголошення y-діапазону необов'язкове.

2. Крива в параметричній формі $x(t)$, $y(t)$, що залежить від довільно обраного параметра t. Якщо цей параметр насправді має назву «t», оголошення діапазону можна опустити. У цьому випадку t отримує значення за замовчуванням, як це зазначено за допомогою `set_plot_option`.

3. Окремі точки, які можна з'єднати (застосовуючи відповідні параметри) за допомогою лінійних сегментів. Існує дві можливості для позначення точок: або в двох окремих списках, що містять значення x і y відповідно, або в одному вкладеному списку, який містить точки, де кожна точка представляє список, що містить його x-значення та y-значення.

Якщо необхідно побудувати два графіки на одній системі координат, із відповідних функцій необхідно сформувати список в квадратних дужках. Проміжок змінних буде спільним.

```
(%i2)  wxplot2d([sin(x),cos(x)],[x,0,2*%pi]);
```

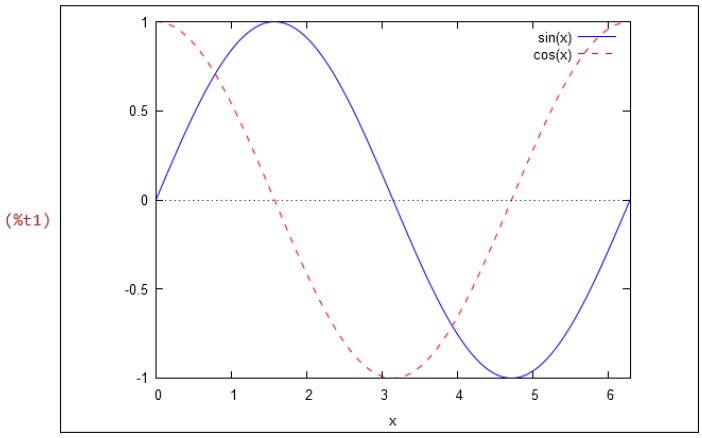


Рис. 4.2: Два графіки на одній системі координат.

Побудуємо графік множини точок, з'єднаних відрізками. Для цього необхідно задати

x- та y-значення списку (які є координатами точок):

```
(%i9)  x_list:[1,1,3,3,1];
```

(%o9) $[1, 1, 3, 3, 1]$

```
(%i10) y_list:[1,3,3,1,1];
```

(%o10) $[1, 3, 3, 1, 1]$

Зобразимо два точкових графіка в одній діаграмі з обома способами задання точок. Точки з'єднані відрізками лінії за замовчуванням.

```
(%i11) plot2d([[discrete,x_list,y_list],
               [discrete,[[3,1],[5,3],[3,5],[1,3],[3,1]]],
               [x,0,6],[y,0,6]);
```

(%o11) $[C : /Users/orreg/maxout.gnuplot]$

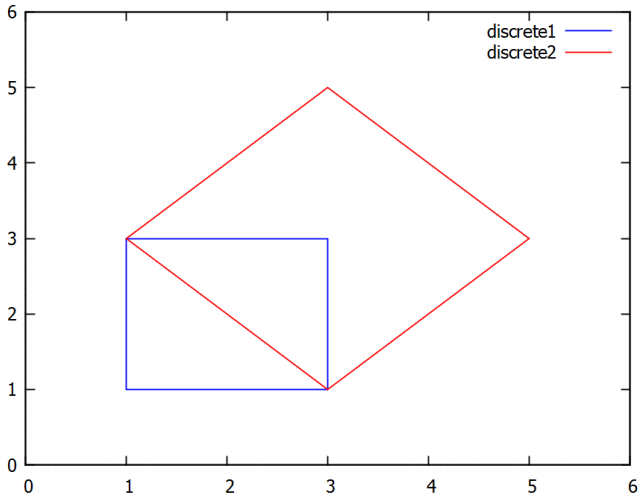


Рис. 4.3: Точкові графіки.

Розглянемо додаткові опції `opts` докладніше. Це необов'язкові параметри, що дозволяють адаптувати графіку до потрібних вимог стосовно кольорів, типів ліній, розмірів, міток, форматів виводу тощо. Опції — це списки, які в основному містять два елементи: перший елемент — це завжди ім'я параметра, другим елементом є пов'язані з цим параметром значення. Опції можуть виступати додатковими параметрами в кожній команді графіка; вони також можуть бути зазначні як значення за замовчуванням за допомогою команди `set_plot_option`. Команда `plot_options` показує всі значення за замовчуванням в опціях.

Якщо необхідно, щоб були побудовані просто точки, без з'єднання їх лініями, у команді `plot2d` потрібно додати опцію `[style,[points,m,n,k]]`, де `m` — її розмір, `n` — її колір, `k` — тип точки. Типів точок можна задати 6.

Команда `[style,[lines,m,n]]` керує виглядом ліній, тут `m` — її товщина, `n` — її колір. Кольорів можна задати 6 (0 = cyan, 1 = blue, 2 = red, 3 = green, 4 = majenta, 5 = black).

`[legend, "leg_1 "leg_2 ...]` — підписує лінії на графіку.

`[xlabel,"name_x"], [ylabel,"name_y"])` — підписує осі координат.

`[nticks,N]` — кількість точок, по яким будується графік.

`[axes,solid]` — виділяє осі виразнішими лініями.

`[same_xy]` — встановлює однаковий масштаб по осях.

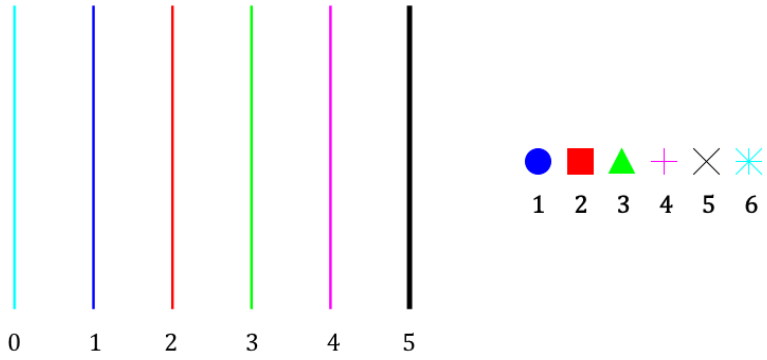


Рис. 4.4: Колір ліній та типи точок.

Значення `false` після коми відключає дану опцію, наприклад при заданні команди `[legend,false]` легенди кривих взагалі не будуть відображатись.

Нехай маємо наступну задачу: методом натурних досліджень проведено збір експериментальних даних, в яких деяка величина y залежить від змінної x. Отримані результати: $x = [1.389, 4.722, 5.000, 6.944, 8.611]$, $y = [30, 34, 38, 44, 46]$. Можна провести регресійний аналіз даних та встановити функціональну залежність:

$y = 25.713 + 2.379x$ — лінійна апроксимація;

$y = 27.463 + 1.443x + 0.0945x^2$ — квадратична апроксимація.

Необхідно зробити графіки трьох видів на одній системі координат. Код буде наступний:

```
(%i12) wxplot2d([25.713+2.379*x,27.463+1.443*x+0.0945*x^2,
       [discrete, [1.389,4.722,5.000,6.944,8.611], [30,34,38,44,46]]],
       [x,-3,13],[style,[lines,3,1],[lines,3,3],[points,4,2,14]],
       [box,false],[legend,"Лінійне наближення","Квадратичне наближення",
       "Вихідні дані"]);
```

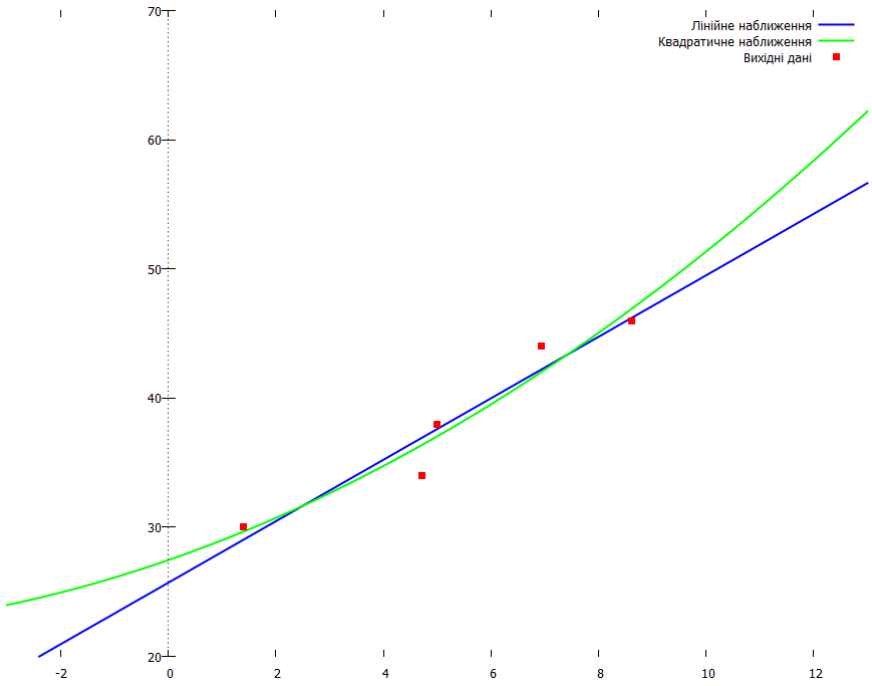


Рис. 4.5: Графіки результатів регресійного аналізу.

4.2 Параметричне та полярне задання функцій

При параметричному представленні функція задається у вигляді $x = \varphi(t)$; $y = \psi(t)$, $t \in [t_1, t_2]$. Команда **Maxima** побудови графіка в такому разі:

`plot2d([parametric,%phi(t),%psi(t),[t,t1,t2]],[nticks,N]).`

Додаткову опцію `nticks` в цій команді краще дописувати і приймати рівною 80-100, це збільшить гладкість кривої.

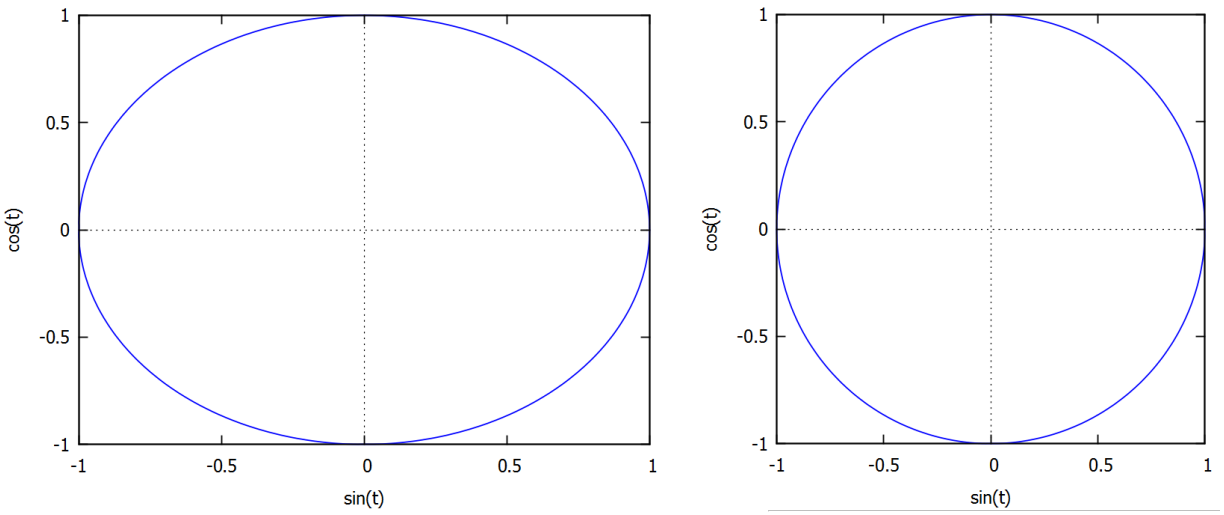


Рис. 4.6: «Коло» та коло з використанням `[same_xy]`.

Зобразимо, наприклад, коло введенням команди

```
(%i1)  plot2d([parametric,sin(t),cos(t),[t,0,2*%pi]],[nticks,100]);
(%o1)  [C : /Users/orreg/maxout.gnuplot]
```

Виведений на екран графік виявиться зовсім не колом, а еліпсом (Рис. 4.6). Причина в тому, що **Maxima** самовільно змінює масштаб осей графіка, для кращої легкочитності. Якщо потрібне са́ме намальоване коло, необхідно додати опцію `[same_xy]`.

```
(%i2)  plot2d([parametric,sin(t),cos(t),[t,0,2*%pi]],[nticks,100],
              [same_xy]);
(%o2)  [C : /Users/orreg/maxout.gnuplot]
```

Якщо обрамлення зовнішніми межами не потрібне, в опціях потрібно додати `[box,false]`. Щоб отриманий графік не дотикався меж бокса, необхідно дописати явно межі змін x та y, напиклад, `[x,-1.5,1.5], [y,-1.5,1.5]`.

Ще приклади: фігура Лісажу та кардіоїда Паскаля (Рис. 4.7).

```
(%i3)  wxplot2d([parametric,sin(3*t),sin(4*t),
       [t,0,2*%pi]],[nticks,100],[same_xy])$
(%i4)  wxplot2d([parametric,2*cos(t)*(1+cos(t)),2*sin(t)*(1+cos(t)),
       [t,0,2*%pi]],[nticks,100],[same_xy])$
```

Для побудови графіка в полярних координатах потрібно задати залежність полярного радіуса від полярного кута. Нехай $\rho = \rho(\theta)$ $(a \le \theta \le b)$ — така залежність. Тоді графік цієї функції в полярних координатах можна побудувати, додавши до команди `plot2d` опцію `[gnuplot_preamble,set polar;set zeroaxis]`. Ця опція буде діяти лише за умови, що вибраний формат графіка **gnuplot**.

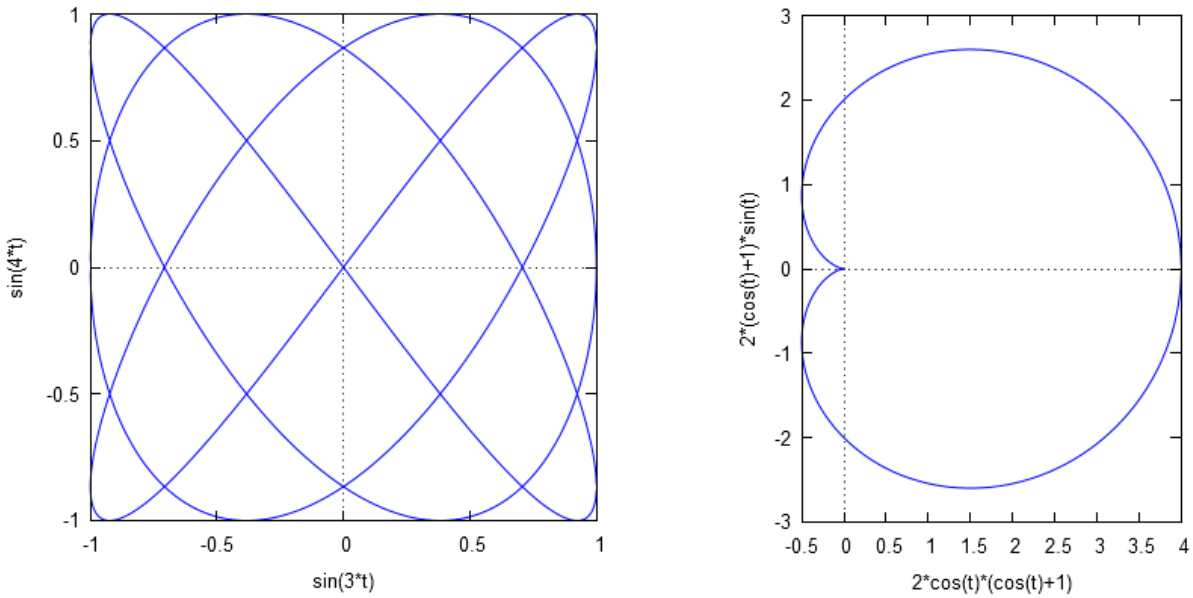


Рис. 4.7: Фігура Лісажу, кардіоїда Паскаля.

Є, однак, спосіб, який спрощує побудову графіка в полярних координатах, і зводить його до звичної форми написання. Це запровадження нової команди `plot_polar`:

```
plot_polar(expr,range):=block([theta_var: range[1]],plot2d(
['parametric,cos(theta_var)*expr,sin(theta_var)*expr,range],[nticks,100]));
```

Один раз ввівши таку функцію в **Maxima**, протягом сесії можна користуватись таким синтаксисом: `plot_polar(f(t),[t,t1,t2])`.

Накреслимо для прикладу графіки функцій $\rho = \sin\theta + 2\cos 2\theta$; $\rho = 3\theta^2$ (Рис. 4.8).

```
(%i5)  plot_polar(sin(t)+2*cos(2*t),[t,0,%pi]);
(%o5)  [C : /Users/orreg/maxout.gnuplot]

(%i6)  plot_polar(3*t^2,[t,0,5*%pi]);
(%o6)  [C : /Users/orreg/maxout.gnuplot]
```

Знову ж таки, товщину та стиль лінії можна регулювати, використовуючи опцію `style` (наприклад, `[style,[lines,3,3]]` видає товщину 3 і зелений колір).

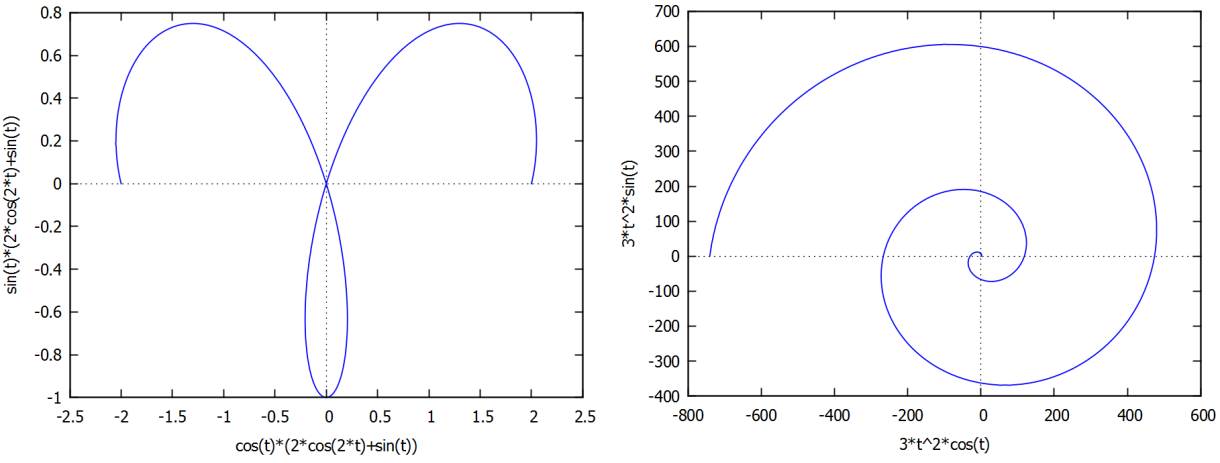


Рис. 4.8: Полярні координати.

4.3 Тривимірні графіки

3d-об'єкти у **Maxima** можуть бути утворені двома різними способами:

1. Задана функція $z = f(x, y)$; діапазони для x-значень та y-значень повинні бути оголошені, z-діапазон при цьому обчислюється автоматично.

2. Задана функція у параметричній формі $f_x(u, v)$, $f_y(u, v)$, $f_z(u, v)$, що залежать від двох (вибір позначень довільний) параметрів u і v. Необхідно оголосити діапазони параметрів u і v, діапазони координат f_x, f_y і f_z не оголошуються.

Синтаксис команди:

`(wx)plot3d(f(x,y),[x,a,b],[y,c,d])` — пряме задання функції $z = f(x, y)$;

`(wx)plot3d([fx,fy,fz],[u,a,b],[v,c,d])` — параметричне задання функції.

Накреслимо дві наступні функції для демонстрації обох випадків (Рис. 4.9):

$$z(x,y) = \frac{1}{1+x^2+y^2}; \qquad \begin{cases} f_x(u,v) = \cos u; \\ f_y(u,v) = v; \\ f_z(u,v) = u + \sin v. \end{cases}$$

```
(%i1)  plot3d(1/(1+x^2+y^2),[x,-2,2],[y,-2,2]);
```

(%o1) $[C : /Users/orreg/maxout.gnuplot]$

```
(%i2)  plot3d([cos(u),v,u+sin(v)],[u,0,2*%pi],[v,0,2*%pi],[same_xy])$
```

(%o2) $[C : /Users/orreg/maxout.gnuplot]$

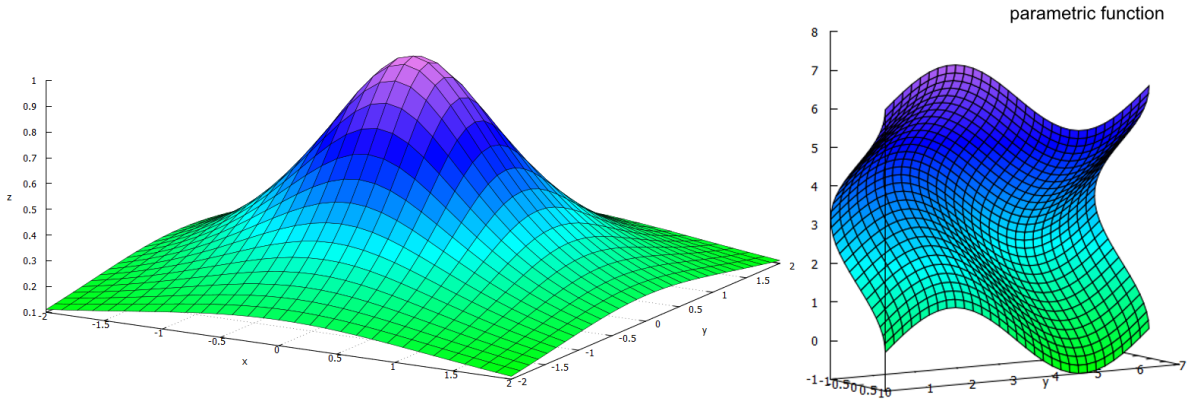


Рис. 4.9: Тривимірні графіки.

Якщо є потреба мати вбудовані в тіло документу **wxMaxima** графіки, необхідно вводити ті ж самі команди, але у вигляді `wxplot3d`. Але при цьому втрачається дуже корисна властивість побудованих у **gnuplot** графіків: їх можна обертати мишкою, міняючи огляд та кут зору.

Опція `[palette,false]` відключає кольорове забарвлення поверхні, залишаючи лінії характеристик.

Опція `[same_xyz]` — робить масштаби всіх осей однаковими.

Команда `contour_plot(f(x,y),[x,a,b],[y,c,d])` — зображає лінії рівнів функції двох змінних, їх число та колір можуть бути зазначені використавши опцію `gnuplot_preamble`.

```
(%i3)  plot3d(cos(x)*sin(y),[x,0,2*%pi],[y,0,2*%pi],[palette,false]);
```

(%o3) $[C : /Users/orreg/maxout.gnuplot]$

```
(%i4)  contour_plot(cos(x)*sin(y),[x,0,2*%pi],[y,0,2*%pi]);
```

(%o4) $[C : /Users/orreg/maxout.gnuplot]$

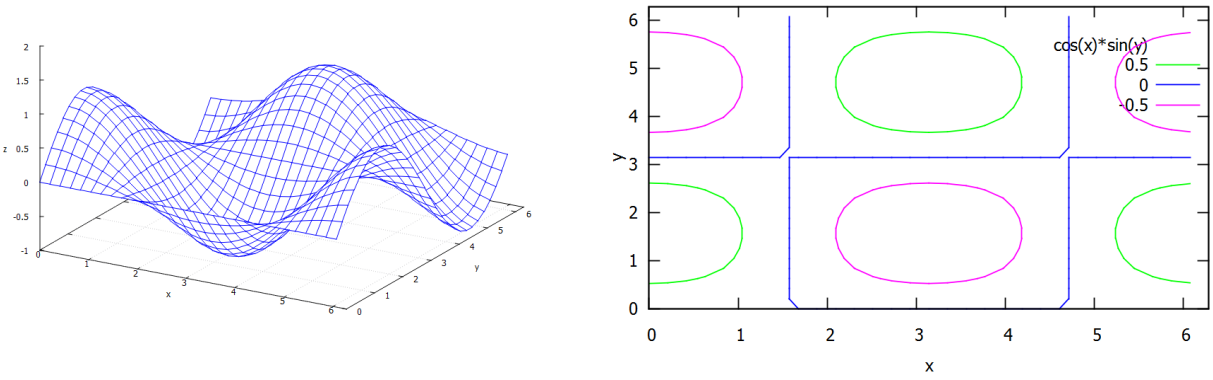


Рис. 4.10: Функція $z = \sin x \cdot \cos y$ та її лінії рівнів.

Опції `[nticks,N]` та `[adapt_depth,N]` задають кількість та максимальну множину початкових точок для побудови кривої. Вони потрібні, щоб зображена поверхня була достатньо «гладкою».

Опція `[grid,Nx,Ny]` — вказує, скількома точками полігону буде апроксимуватись крива характеристик. Ця команда з «плавної» кривої робить відрізки прямих. Наприклад, побудована вище поверхня $z(x,y) = \frac{1}{1+x^2+y^2}$ з додаванням `[grid,5,5]` буде виглядати так, як зображено на Рис 4.11. З допомогою `grid` можна зобразити просторову криву (ґвинтову лінію), без її застосування ця крива була б занадто «ламаною».

```
(%i5)  plot3d(1/(1+x^2+y^2),[x,-1,1],[y,-1,1],[grid,5,5]);
```

(%o5) $[C : /Users/orreg/maxout.gnuplot]$

```
(%i6)  plot3d([sin(u),cos(u),3*u],[u,0,6*%pi],[v,0,1],[grid,100,100]);
```

(%o6) $[C : /Users/orreg/maxout.gnuplot]$

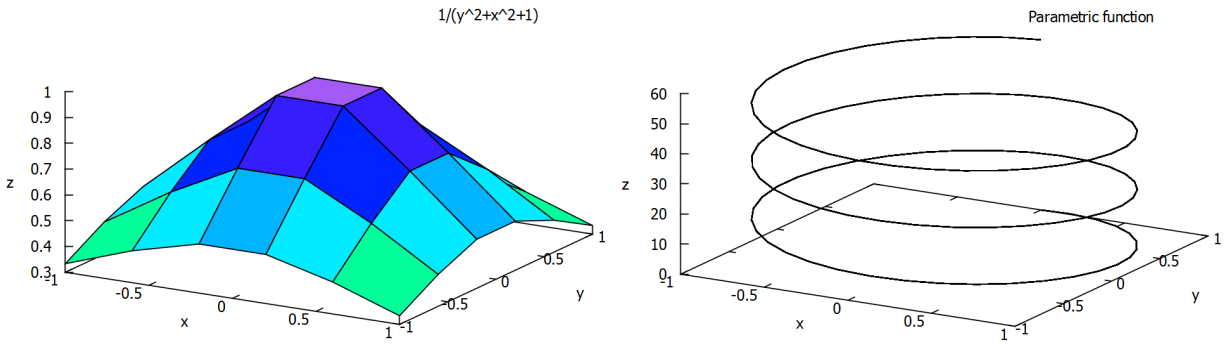


Рис. 4.11: Застосування опції `grid`.

Ще деякі приклади просторових поверхонь — одностороння смужка Мебіуса та одинична сфера (уважний читач одразу впізнає в командах вирази для x, y, z у сферичній системі координат).

```
(%i7)  plot3d([cos(u)*(3+v*cos(u/2)),sin(u)*(3+v*cos(u/2)),v*sin(u/2)],
              [u,-%pi,%pi],[v,-1,1]);
       plot3d([cos(u)*sin(v),sin(u)*sin(v),cos(v)],[u,-%pi/2,%pi/2],
              [v,0,2*%pi],[same_xy],[palette,false],[title, "Sphere"]);
```

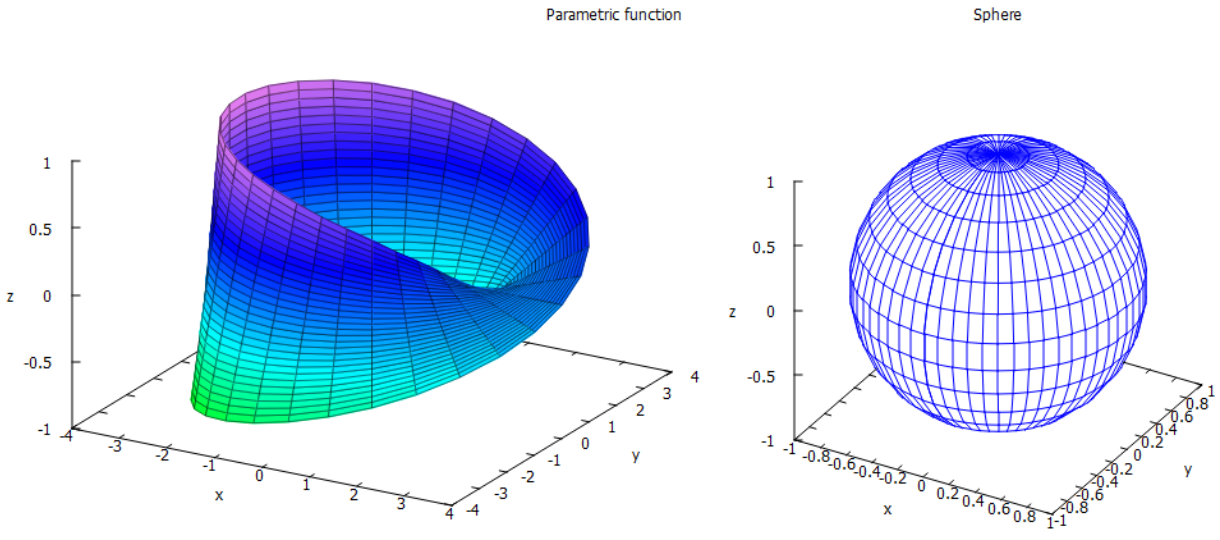


Рис. 4.12: Смужка Мебіуса та одинична сфера.

4.4 Функції, задані неявно

Для побудови графіків функцій, заданих неявним чином, використовується команда plot2d, у якій необхідно вказати вираз (або кілька виразів), та обов'язково інтервал змінних як по x, так і по y. Синтаксис в такому випадку:

`plot2d([eqn1,eqn2,...],[x,a,b],[y,c,d],opts).`

► Накреслити графіки неявних функцій: $xy^2 = x - y^3$, $xy = x^2 - y$ (Рис. 4.13(а)).

```
(%i1)  plot2d([x*y^2=x-y^3,x*y=x^2-y],[x,-6,6],[y,-4,4],
              [legend,"expr1","expr2"]);
```

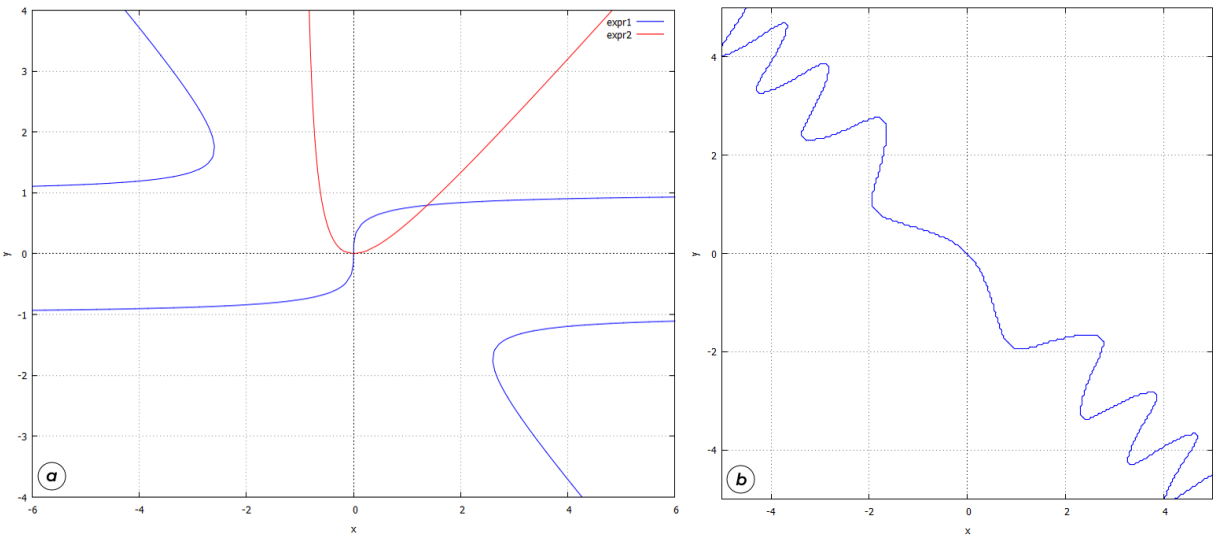


Рис. 4.13: Графіки неявно заданих функцій через `plot` (а) та `implicit_plot` (б).

Поряд з цим можна використовувати додатковий пакет `implicit_plot`, якого треба попередньо підключити командою `load`. Синтаксис команди:

`implicit_plot([eqn1(x,y),eqn2(x,y),...],[x,a,b],[y,c,d])`,

тут `eqni(x,y)` — одне співвідношення між x та y або список співвідношень, `[x,a,b]` — задання зміни по x, `[y,c,d]` — задання зміни по y.

► Зобразити графік кривої $x + y = \sin xy$.

```
(%i9)  implicit_plot(x+y=sin(x*y),[x,-5,5],[y,-5,5],[same_xy]);
(%o9)  done
```

Графік на Рис. 4.13(б). Видно, що якість та чіткість малюнка з командою `plot` є вищою, оскільки алгоритм побудови графіка є іншим.

4.5 Додаткові опції gnuplot

gnuplot — один з найбільш функціональних відкритих графічних процесорів з широкими можливостями. Попередньо ми бачили, що реалізація команд **gnuplot** у **Maxima** є простою та чіткою, однак водночас прагнення до простоти наклало певні обмеження на тип та вигляд побудованих зображень. Наприклад, часто постає питання про відображення кривої штрихованою лінією, задання спеціального кольору кривій (поза стандартним набором), або зміна місця розміщення легенди у графіку. Всі ці можливості (і набагато більше) реалізовані у **gnuplot**, але для їх здійснення необхідно звертатись до застосунку напряму через команду `gnuplot_preamble`. Загальний вигляд синтаксису наступний:

`[gnuplot_preamble, "set ..., set ..."]`

Розглянемо додаткові опції докладніше. Ми оберемо найбільш поширені та потрібні, взагалі їх кількість достатньо велика, тому для повного ознайомлення рекомендуємо звернутись за довідкою **gnuplot**.

`set output "filename"` — задання виводу об'єкта (назва файлу з відповідним розширенням);

`set size xscale, yscale` — масштабування діаграми відповідно до розміру всього графіка;

`set size ratio n` — задання множника масштабу діаграми;

`set|unset key` — включає/виключає легенду;

`set key left|right top|bottom` — задання місця розміщення легенди, `left`=зліва, `right`=справа, `top`=зверху, `bottom`=знизу;

`set polar` — побудова графіків у полярних координатах;

`set dashtype '.-'` — задання для списку функцій типів штрихованої лінії; вигляд штриховки керується виразом у лапках, він може складатись з будь-якої комбінації крапок та дефісів, і вони будуть повторені на графіку;

`set xtics ('π' 3.14, ...)` — задання окремих позначок на осі x; у лапках записується назва позначки, потім її числове значення.

Продемонструємо ці опції на Рис.4.14.

```
(%i1)  plot2d(
         [x, x*cos(x)], [x, 0, 2*%pi],
         [gnuplot_preamble, "
         set dashtype '..--',
         set dashtype '.-',
         set key left top,
         set xtics ('pi/2' 1.507, 'pi' 3.14, '3pi/2' 4.71)"],
         [style, [lines, 2, 4], [lines, 2, 7]]
       );
```

Для тривимірних графіків можливі наступні опції:

`set|unset hidden3d` — включає/виключає приховані лінії в 3d-графіках;

`set|unset grid` — включає/виключає координатну сітку;

`set cntrparam levels N` — кількість контурних ліній в команді `contour-plot`.

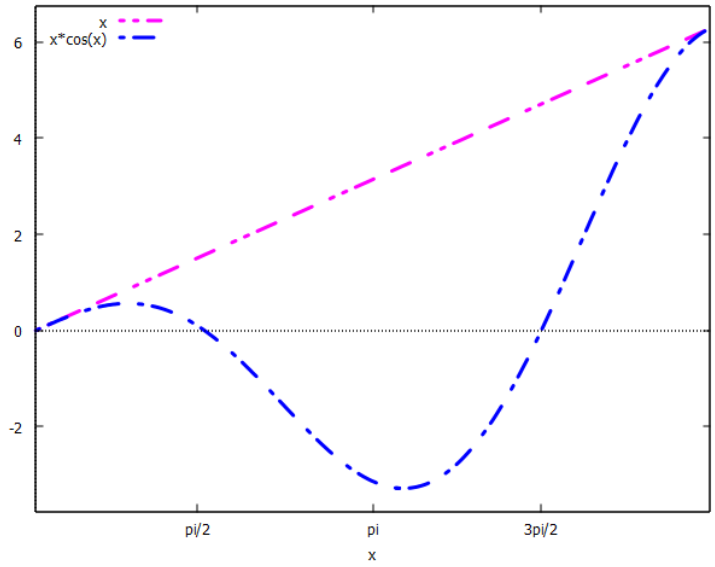


Рис. 4.14: Графіки функцій зі спеціальними опціями.

Кольорова палітра в 3d-об'єктах задається опцією
`[palette,[gradient|hue|saturation|value, N1, N2, N3, N4]]`,
у якій при обранні значення `gradient` величини `N1, N2, N3, N4` мають бути чотирма кольорами, за якими поверхня змінюється від низу до верху; а при обранні значень `hue|saturation|value` величини `N1, N2, N3, N4` мають бути чотирма числами від 0 до 1, у яких перше число позначає колірне місце на спектральній палітрі, а три інших є додатковими характеристиками цього кольору (насиченість та ін.)

```
(%i1)  plot3d(sin(x*y),[x,0,%pi],[y,0,%pi],[z,-1,2],
             [palette,[gradient,brown,red,orange,blue]]);
(%i1)  plot3d(sin(x*y),[x,0,%pi],[y,0,%pi],[z,-1,2],
             [palette,[saturation,0.09,0.2,0.8,0.7]],
             [gnuplot_preamble,"set grid"]);
```

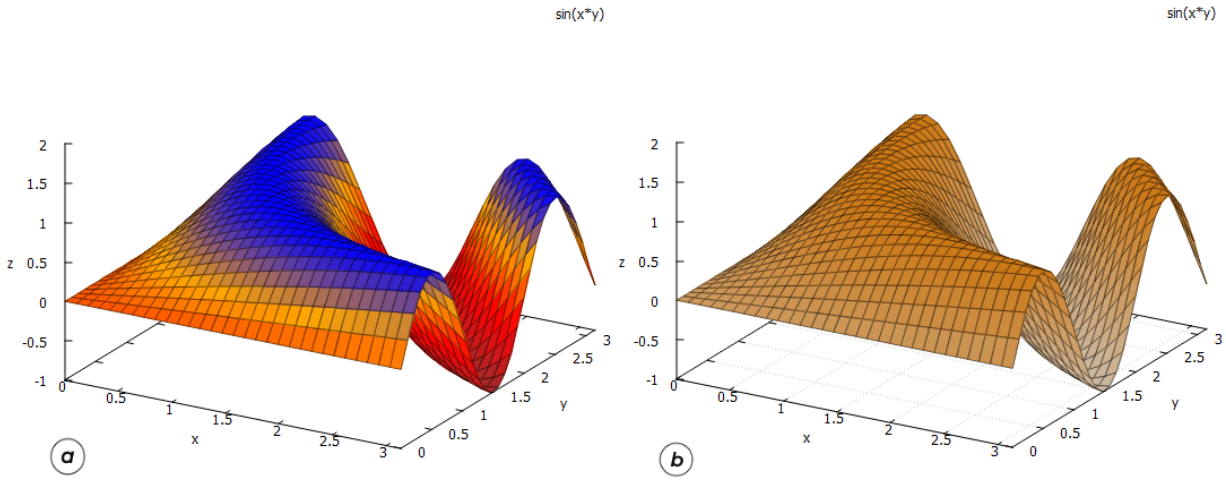


Рис. 4.15: Різні задання кольорового розфарбування командою `palette`.

4.6 Завдання до Розділу 4

Двовимірні графіки

4.1. Накреслити графіки функцій: $y = x$, $y = -x$, $y = x\sin x$, $x \in [-5\pi, 5\pi]$. Виділити осі, підписати криві «Меркурій», «Венера», «Марс».

4.2. Накреслити графіки функцій: $y = e^{-x^2}$, $y = -\sqrt{x}$, $y = \ln x$, $x \in [-6, 6]$, $y \in [-4, 4]$. Виділити осі, назвати їх «відстань», «час». Підписати криві «Гаусс», «Корінь», «Логарифм».

4.3. Накреслити графік функції: $y = \cos(t^3)$, $t \in [-3, 3]$. Графік не повинен дотикатись до меж бокса. Знайти будь-які три точки, що належать цій кривій, зобразити їх на графіку зірочками чорного кольору.

4.4. Накреслити графік неявної функції: $\operatorname{sh} x = x^2 + y^2$, $x \in [-1, 5], y \in [-5, 5]$. Виділити осі, назвати їх «амплітуда», «координата».

4.5. Накреслити дискретні графіки: точки з'єднані відрізками (0,0), (3,2), (6,0), (4,3), (6,6), (3,4), (0,6), (2,3), (0,0); окремі точки (трикутники) (3,1), (5,3), (3,5), (1,3). Масштаб осей зробити однаковим, підписати набори точок: «Зірка», «Ромб».

4.6. Накреслити дискретні графіки: точки з'єднані відрізками (1,1), (4,5), (9,5), (9,1), (1,1); окремі точки (хрестики) (4,1), (6.5,3), (9,5); $x \in [-2, 8], y \in [-2, 8]$. Масштаб осей зробити однаковим, підписати набори точок: «Трапеція», «Точки».

4.7. Накреслити графік параметричної функції: $x = 20(\cos t + 0.2\cos 5t)$; $y = 20(\sin t - 0.2\sin 5t)$, $t \in [0, 2\pi]$. Товщину ліній зробити 4, колір червоний, масштаб осей однаковий. Позначити на графіку такі точки: при $t = 1.5$, $t = 3.5$, $t = 5.5$.

4.8. Накреслити графік полярної функції: $\rho = (\theta + 1)^2$, $\theta \in [-2\pi, 2\pi]$. Товщину ліній зробити 3, колір чорний. Позначити на графіку дві точки: при $\theta = -5$ та $\theta = 5$.

4.9. Накреслити графік параметричної функції: $x = \sin 5t \cos t$; $y = \sin 5t \sin t$, $t \in [0, 2\pi]$. Товщину ліній зробити 3, колір чорний, масштаб осей однаковий. Позначити на графіку такі точки: при $t = 0.5\pi$, $t = 1.3\pi$, $t = 1.7\pi$.

4.10. Накреслити графік неявної функції: $x^4 - 96x^2 = y^4 - 100y^2$, $x \in [-22, 22], y \in [-22, 22]$. Виділити осі, назвати їх «час», «відстань».

Тривимірні графіки

4.11. Побудувати поверхню: $\sin(x^2 + y^2)$; $x \in [0, 3], y \in [0, 3]$. Підписати осі «Північ», «Південь», «Захід».

4.12. Побудувати тор: $f_x = (3 + \cos v)\cos u$; $f_y = (3 + \cos v)\sin u$; $f_z = \sin v$; $u \in [0, 2\pi], v \in [-\pi, \pi]$. Змінити межі так, щоб він став зрізаним.

4.13. Побудувати гіперболоїд: $f_x = (5 + \operatorname{ch} v)\cos u$; $f_y = (5 + \operatorname{ch} v)\sin u$; $f_z = \operatorname{sh} v$; $u \in [0, 2\pi], v \in [-\pi, \pi]$. Змінити межі так, щоб він став зрізаним.

4.14. Побудувати гелікоїд: $f_x = u\sin v$; $f_y = u\cos v$; $f_z = v/3$; $u \in [-1, 1], v \in [0, 10]$.

4.15. Побудувати мушлю равлика: $f_x = 1.16^v \cdot \cos v(1 + \cos u)$; $f_y = -1.16^v \cdot \sin v(1 + \cos u)$; $f_z = -2{\cdot}1.16^v{\cdot}(1{+}\sin u)$; $u \in [0, 2\pi], v \in [-10, 6]$. Встановити опцію згладжування [100,100].

4.16. Побудувати конус двома способами: параметричне задання $f_x = \sin u \cos v$; $f_y = \sin u \sin v$; $f_z = |\sin u|$; $u \in [0,5], v \in [0,5]$; пряме задання $z = 3\sqrt{x^2/4 + y^2/4}$; $x \in [-5,5], y \in [-5,5]$. Змінити межі так, щоб він став зрізаним.

4.17. Побудувати поверхню: $z = 20e^{-x^2-y^2} - 10$; $x \in [0,2]; y \in [-3,3]$. Зобразити контурні лінії цієї поверхні.

4.18. Побудувати просторову криву «торнадо»: $f_x = u \cos u$; $f_y = u \sin u$; $f_z = u$; $u \in [0, 8\pi], v \in [0,3]$. Встановити опцію згладжування [100,100].

4.19. Побудувати циліндр: $f_x = \sin u$; $f_y = \cos u$; $f_z = v$; $u \in [0, 2\pi], v \in [0,2]$. Масштаб осей зробити однаковим, розфарбування вимкнути.

4.20. Побудувати поверхню: $f_x = \cos u \cos v$; $f_y = \cos u \sin v$; $f_z = \operatorname{sh} v$; $u \in [0,\pi], v \in [0,\pi]$.

Розділ 5

Візуалізація процесів

При моделюванні фізичних процесів важливою складовою є візуалізація отриманих даних у тій чи іншій формі. Це дозволяє інтерпретувати інформацію у зручному для сприйняття людини вигляді, а зміна будь-яких параметрів системи миттєво набуває цілком наочного представлення.

Для створення графічних об'єктів **Maxima** використовує зовнішній застосунок **gnuplot**, який є повністю незалежним проєктом (такого роду симбіози є звичними для світу Open Source, відкритого вільного програмного забезпечення). Поряд з цим, у **Maxima** існує більш просунутий процесор візуалізації **VTK** (The Visualization Toolkit), для використання якого необхідно змінити рендерер відповідною командою. Основна відмінність між цими пакетами полягає у форматі вихідного об'єкту: якщо у **gnuplot** це растрові зображення або їх сукупності (анімовані .gif-файли), то у **VTK** це повноцінні 3d-моделі, які можливо зберігати у форматах .stl, .ply, .vrml зі всією повнотою векторних зображень. Ці формати підтримуються широким колом застосунків для подальшого редагування чи перегляду, для прикладу Blender, Autodesk Inventor CAD, SolidWorks, FreeCAD та ін.

5.1 Графічний інтерфейс `draw`

Для побудови графіків функцій потужним інстументом є команди `plot(2d|3d)`, синтаксис та можливості яких докладно розглянуті раніше. Ці команди вирізняються простотою та лаконічністю запису, та є зручними для зображення ліній та поверхонь при прямому заданні функціональних залежностей $y = f(x)$ або $z = f(x, y)$. Однак часто виникає потреба у побудові зображень, які взагалі не є графіками функцій (кругові чи стовпчикові діаграми, вектори, заповнена різними кольорами матрична сітка, або заповнені кольором різноманітні фігури). В такому випадку звертаються до інтерфейсу **gnuplot** `draw`, що надає додатковий набір графічних процедур **Maxima**. Ці команди пропонують набагато більшу гнучкість для налаштування та адаптації графіків до конкретних вимог. Ціною таких покращень стає дещо складніший запис синтаксису задання зображень.

Пакет `draw` необхідно завантажити перед його першим використанням.

```
(%i1)  load(draw)$
```

Команда `draw` зображає так звані *сцени*. Загальний синтаксис команди наступний:
`draw(scene1,scene2,...,opts)`.

Тут `scene1,scene2,...` — відповідні сцени, `opts` — глобальні опції. Задання сцени викликається командами

`gr2d(opts,graphic_object,...)` — створює 2d-сцену, скомпоновану з двовимірного графічного об'єкту з опціями `opts`;

`gr3d(opts,graphic_object,...)` — створює 3d-сцену, скомпоновану з тривимірного графічного об'єкту з опціями `opts`.

Поряд з таким записом використовуються ще дві можливості:

`draw2d(opts,graphic_object,...)`, `draw3d(opts,graphic_object,...)` — будує 2|3-вимірний графічний об'єкт з опціями `opts`. Найчастіше на практиці графік містить одну сцену, тому в такому випадку скорочений запис `draw2d|3d` є зручним. Загалом вирази `draw2d|3d` та `draw(gr2d|3d)` є еквівалентними. Запис у вигляді `draw(gr2d|3d)` доцільно використовувати тоді, коли на одному малюнку необхідно зобразити зображення різних типів (двовимірних та тривимірних об'єктів). У випадку використання функції `plot` така можливість не реалізовується.

Поряд з командами сімейства `draw` можна використовувати `wxdraw`, тоді відповідні графіки будуть виводитись не у окремому вікні, а вбудовані у тіло документу. Використавши меню **wxMaxima** «Редагування — Налаштувати — Робочий аркуш», можна вибрати у пункті «Типовий розмір креслень для нових сеансів **Maxima**» значення розширення вбудованих графіків.

Опції поділяються на глобальні та локальні. Локальні параметри, що стосуються до певного графічного об'єкту, повинні передувати цьому об'єкту. Положення глобальних параметрів (наприклад, для оголошення вихідного формату чи загального вигляду системи координат) є довільним. Опції оголошуються як рівняння `opt=flag` з назвою в лівій частині та значенням у правій; значення може бути числом, списком (наприклад, для призначення діапазонів) чи булевим виразом (`true|false`).

У складній графіці, що містить багато об'єктів, список параметрів може бути довгим і дещо заплутаним. В такому випадку корисними будуть наступні поради: структурувати вхідні дані за допомогою відповідних розривів рядків і відступів; не оголошувати графічні об'єкти безпосередньо в списку параметрів команд `draw`, а присвоювати їм імена змінних в окремих командах і використовувати їх пізніше у списку команд графіка.

Зобразимо для прикладу простий графік функції.

```
(%i2)  graph:gr2d(
           color =black,
           nticks=100,
       explicit(cos(t),t,0,2*%pi)
       )$

(%i3)  draw(graph);
```

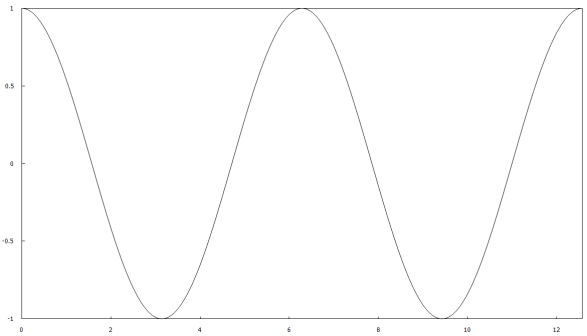


Рис. 5.1: Простий графік функції.

5.1.1 Графічні об'єкти у двовимірному просторі

Пакет `draw` використовує для створення зображень наступні команди:

`explicit(f(x),x,x1,x2)` — пряме задання функції $f(x)$ на інтервалі $x \in [x_1; x_2]$;

`parametric(x(t),y(t),t,t1,t2)` — параметричне задання функції $\{x(t), y(t)\}$ зі зміною параметра $t \in [t_1; t_2]$;

`implicit(eqn,x,x1,x2,y,y1,y2)` — непряме задання функції через рівняння `eqn` на інтервалі $x \in [x_1; x_2]$, $y \in [y_1; y_2]$;

`polar(r(%theta),%theta,%theta1,%theta2)` — задання функції в полярних координатах $r = r(\theta)$, при зміні полярного кута $\theta \in [\theta_1; \theta_2]$;

`points(x_list,y_list)` — множина точок, що задаються списками значень x та y відповідно;

`points([[x1,y1],[x2,y2],...])` — множина точок, кожна з яких задається парою $[x; y]$;

`polygon(x_list,y_list)` — многокутник, що задається списками координат точок x та y відповідно;

`polygon([[x1,y1],[x2,y2],...])` — многокутник, що задається значеннями координат точок $[x; y]$;

`rectangle([A1,A2],[B1,B2])` — прямокутник із протилежними точками $[A_1; A_2]$ та $[B_1; B_2]$;

`ellipse(x0,y0,a,b,%phi1,%phi2)` — еліпс з центром у точці $[x_0; y_0]$, півосями a та b, графік починається з кута ϕ_1, завершується кутом ϕ_2;

`bars([x1,h1,w1],[x2,h2,w2],...)` — стовпчикова діаграма, елементами якої є списки з трьома аргументами: положення на осі x, висота (може бути від'ємною), ширина;

`label([,,text",x,y],...)` — текст підпису на позиції $[x; y]$;

`vector([x,y],[A1,A2])` — вектор з початком в $[x; y]$ та компонентами $[A_1, A_2]$.

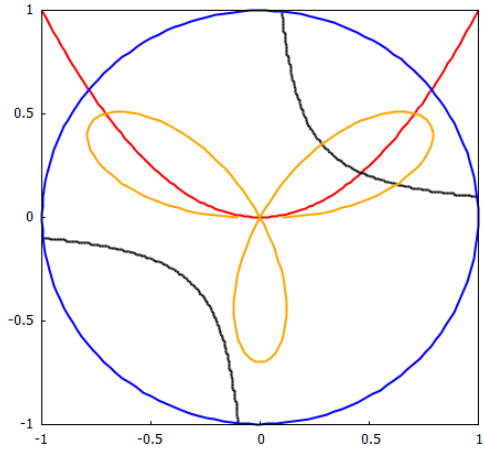


Рис. 5.2: Різні способи задання функцій на одному графіку.

Зобразимо приклади різних типів задання графічних елементів. Парабола $y = x^2$; коло в параметричному вигляді $\{x(t) = \cos t; y(t) = \sin t\}$; непряма функція $\sin(xy) = 0.1$; полярна функція $r(\theta) = 0.1 + 0.8 \sin 3\theta$ — зображені на Рис.5.2.

```
(%i16) draw2d(
       proportional_axes=xy,
       line_width=2,color=red,
       explicit(x^2,x,-1,1),
       color=black,nticks=160,
       implicit(sin(x*y)=0.1,x,-1,1,y,-1,1),
       color=blue,nticks=160,
       parametric(cos(t),sin(t),t,0,2*%pi),
       color=orange,nticks=160,
       polar(0.1+0.8*sin(3*t),t,0,%pi))$
```

Розглянемо тепер ті об'єкти, які властиві лише для пакету `draw`, і які складно було б реалізувати функцією `plot`, тоді як тут вони вже вбудовані.

Многокутник:

```
(%i1)  poly:polygon([[2,4],[2,6],[4,8],[6,8],[8,6],[8,4],[6,2],[4,2]]);
```

(%o1) polygon ([[2, 4], [2, 6], [4, 8], [6, 8], [8, 6], [8, 4], [6, 2], [4, 2]])

Окремі точки:

```
(%i2)  pnts:points([1,3,5,7,9,9,9,9,9,7,5,3,1,1,1,1],
       [1,1,1,1,1,3,5,7,9,9,9,9,9,7,5,3]);
```

(%o2) points([1, 3, 5, 7, 9, 9, 9, 9, 9, 7, 5, 3, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 3, 5, 7, 9, 9, 9, 9, 9, 7, 5, 3])

Прямокутник:

```
(%i3)  rct:rectangle([1,-2],[6,-7]);
```

(%o3) rectangle ([1, −2], [6, −7])

Еліпс:

```
(%i4)  ell:ellipse(6,-6,3,2,0,360);
```

(%o4) ellipse $(6, -6, 3, 2, 0, 360)$

Стовпчикова діаграма:

```
(%i5)  diagr:bars([-7,2,1],[-5,5,1],[-3,7,1],[-1,6,1]);
```

(%o5) bars $([-7, 2, 1], [-5, 5, 1], [-3, 7, 1], [-1, 6, 1])$

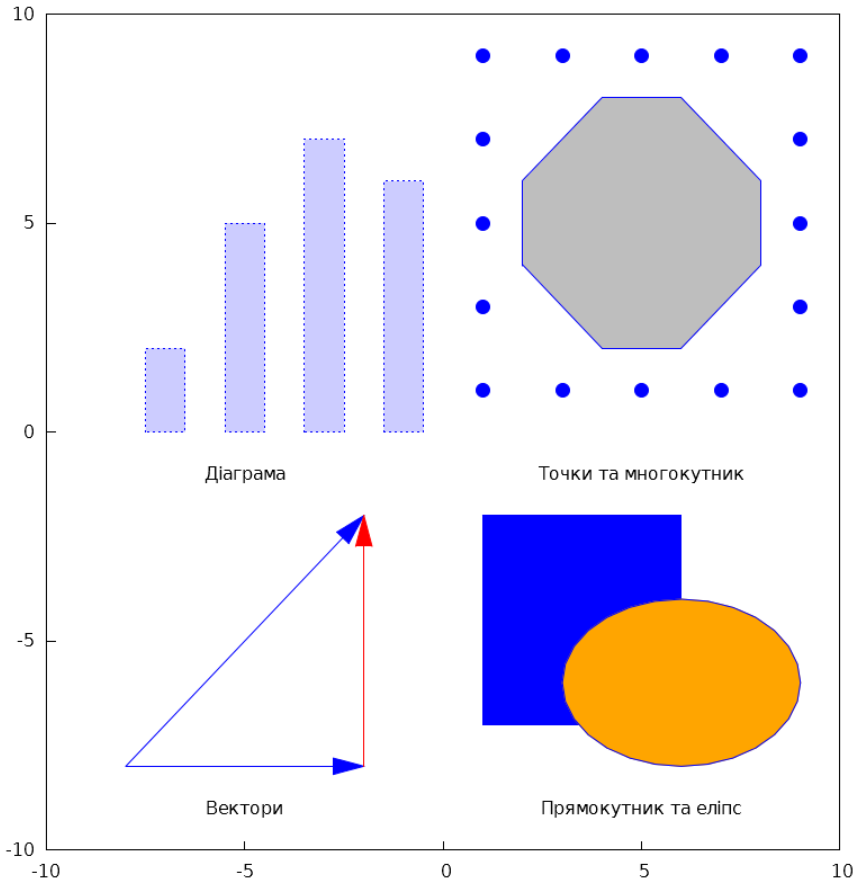


Рис. 5.3: Об'єкти, властиві пакету `draw`.

Вектори:

```
(%i6)  [v1,v2,v3]:[vector([-8,-8],[6,0]),
       vector([-8,-8],[6,6]),vector([-2,-8],[0,6])];
```

(%o6) $[\text{vector}([-8, -8], [6, 0]), \text{vector}([-8, -8], [6, 6]), \text{vector}([-2, -8], [0, 6])]$

Текст підписів:

```
(%i7)  text:label(["Діаграма",-5,-1],
       ["Точки та многокутник",5,-1],
       ["Вектори",-5,-9],
       ["Прямокутник та еліпс",5,-9]);
```

(%o7) label([Діаграма, $-5, -1$], [Точки та многокутник, $5, -1$], [Вектори, $-5, -9$], [Прямокутник та еліпс, $5, -9$])

Всі об'єкти, скомпоновані на одному (Рис. 5.3):

```
(%i8)  wxdraw2d(xrange=[-10,10],yrange=[-10,10],
       point_type = filled_circle, point_size = 2, pnts,
       fill_color = gray, fill_density = 0.4, poly,
       fill_color = blue, rct,
       fill_color = orange, ell,
       fill_color = blue, fill_density = 0.2, diagr,
       head_length = 0.8, head_angle = 15,v1,
       color = blue, v2, color = red, v3,
       color = black, text)$
```

Команда `image(matrix,x1,y1,x2,y2)` — ґенерує об'єкт зображення, що складається з квадратних областей–окремих пікселів. Вони зафарбовуються відповідно до кольорової палітри, визначеної за допомогою передвстановлених параметрів. Збоку при цьому виводиться шкала кольорів, крайні значення якої відповідать мнімальному та максимальному значенням чисел матриці.

```
(%i9)  M:matrix([2,3,5,4],
                [5,2.5,4,5],
                [3,1.75,2,3])$
(%i10) draw2d(image(M,0,0,5,10))$
```

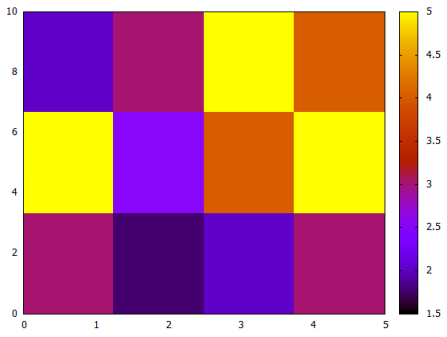


Рис. 5.4: Заповнення матриці кольорами командою `image`.

5.1.2 Графічні об'єкти у тривимірному просторі

`explicit(f(x,y),x,x1,x2,y,y1,y2)` — пряме задання функції $f(x,y)$ на інтервалах $x \in [x_1;x_2]$ та $y \in [y_1;y_2]$;

`implicit(eqn,x,x1,x2,y,y1,y2,z,z1,z2)` — непряме задання функції через рівняння `eqn` на інтервалах $x \in [x_1;x_2]$, $y \in [y_1;y_2]$, $z \in [z_1;z_2]$;

`parametric(x(t),y(t),z(t),t,t1,t2)` — параметричне задання просторової кривої $x(t), y(t), z(t)$ через параметр $t \in [t_1;t_2]$;

`parametric_surface(x(u,v),y(u,v),z(u,v),u,u1,u2,v,v1,v2)` — параметричне задання поверхні зі зміною параметрів $u \in [u_1;u_2]$, $v \in [v_1;v_2]$;

`cylindrical(r(z,%phi),z,z1,z2,%phi,%phi1,%phi2)` — задання поверхні в циліндричних координатах;

`spherical(r(%phi,%theta),%phi,%phi1,%phi2,%theta,%theta1,%theta2)` — задання поверхні в сферичних координатах;

`points(x_list,y_list,z_list)` — задання точок списками їх відповідних координат;

`label([,,text‘‘,x,y,z],...)` — текстові підписи на позиціях $[x,y,z]$;

`vector([x,y,z],[A1,A2,A3])` — задання вектору з початком $[x,y,z]$ та компонентами $[A_1,A_2,A_3]$.

`elevation_grid(matrix,x1,y1,x2,y2)` — будує поверхневу сітку із матриці M, використовуючи набір точок (i,j,M_{ij}).

`mesh([[x1,y1,z1],[x2,y2,z2],...])` — будує поверхневу сітку із набору точок, кожна з яких задана своїми координатами (x,y,z).

Еліпсоїд, заданий неявною функцією:

```
(%i1)  ell:implicit(x^2/2^2+y^2/3^2+z^2/4^2=1,x,-2,2,y,-3,3,z,-4,4);
```

(%o1) $$\text{implicit}\left(\frac{z^2}{16}+\frac{y^2}{9}+\frac{x^2}{4}=1, x, -2, 2, y, -3, 3, z, -4, 4\right)$$

Пряме задання функції:

```
(%i2) func:explicit(4*sin(x^2+y^2),x,-2,2,y,-2,2);
```

(%o2) $$\text{explicit}\left(4\cdot\sin\left(y^2+x^2\right), x, -2, 2, y, -2, 2\right)$$

Зображення двох функцій на одному графіку (Рис.5.5(а)):

```
(%i3) draw3d(surface_hide=true,
             color=navy,ell,
             color=orange,func)$
```

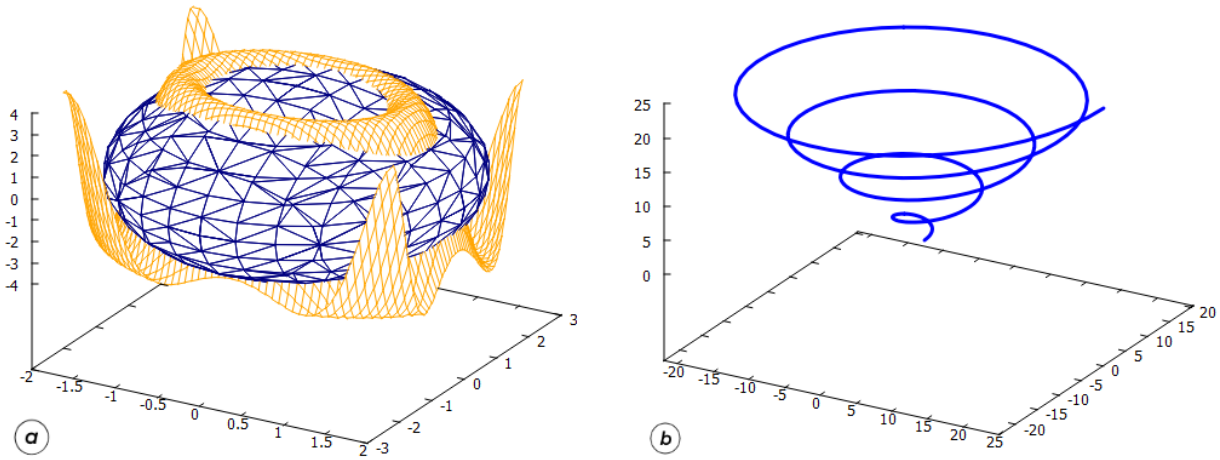


Рис. 5.5: Графіки функцій у тривимірному просторі.

Бачимо, що на початку команди `draw` була вказана глобальна опція `surface_hide=true`. Вона вказує на те, що потрібно ховати лінії тієї поверхні, яка потрапила всередину іншої. Якби стояв прапорець `false`, всі лінії були б зображені, але при цьому графік став би перевантаженим сумішшю ліній.

Просторова крива через параметричне задання (Рис.5.5(б)):

```
(%i4) draw3d(
         color      = blue,
         nticks     = 200,
         line_width = 3,
         parametric(u*cos(u),u*sin(u),u,u,0,8*%pi)) $
```

Конус у циліндричній системі координат:

```
(%i5) cone:cylindrical((z-15)*0.05,z,-15,15,phi,0,2*%pi);
```

(%o5) $$\text{cylindrical}\left(0.05\cdot(z-15), z, -15, 15, \phi, 0, 2\cdot\pi\right)$$

Сфера в сферичній системі координат:

```
(%i6) sphere:spherical(5,phi,-%pi,%pi,theta,0,%pi);
```

(%o6) $$\text{spherical}\left(5, \phi, -\pi, \pi, \theta, 0, \pi\right)$$

Побудова двох графіків (Рис.5.6(а)):

```
(%i7)  draw3d(surface_hide=true,
              proportional_axes = xyz,
              color=dark-orange,cone,
              color=dark-violet,sphere)$
```

Задання точок, що розміщені у площині xy:

```
(%i8)   pts:points(makelist([sin(t*%pi/10),cos(t*%pi/10),0],t,1,40))$
```

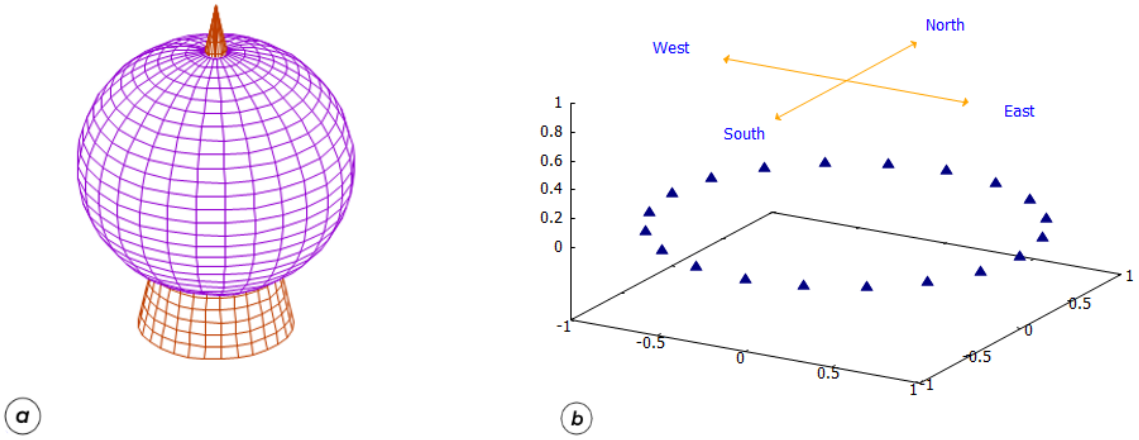


Рис. 5.6: Графічні об'єкти у 3d просторі.

Задання чотирьох векторів, що формують напрямки компаса:

```
(%i9)  [v1,v2,v3,v4]:[vector([0,0,1],[0.7,0,0]),
       vector([0,0,1],[0,0.7,0]),vector([0,0,1],
       [-0.7,0,0]),vector([0,0,1],[0,-0.7,0])]$
```

Підписи сторін світу (вони вирівнюються не по осях, а до напрямку зору, тому при обертанні малюнка текст завжди буде звернений до читача):

```
(%i10)  text:label(["North",0,1,1],["East",1,0,1],
                   ["South",0,-1,1],["West",-1,0,1])$
```

Всі об'єкти на одному зображенні (Рис.5.6(б)):

```
(%i11) draw3d(color=navy,point_type=filled_up_triangle,pts,
              color=orange,head_length=0.02,v1,v2,v3,v4,
              color=blue,text)$
```

5.1.3 Глобальні опції

Параметри, які стосуються усієї скомпонованої сцени, керуються глобальними опціями. Команди цих опцій можуть бути розміщені будь-де у заданні графіку.

`terminal=flag` — задає вихідний формат для графіки, він може бути як растровий (png, jpg, gif, pdf), так і векторний (eps, eps_color); якщо його не задавати явно, зображення виведеться на екран;

`file_name="file"` — ім'я утвореного файлу; за замовчуванням: maxima_out;

`dimensions=[width,height]` — розмір вихідного файлу: у пікселях для растрової графіки, у 1/10 мм для векторної;

`columns=N` — кількість колонок для кількох сцен на одному графіку;
`background_color=colorname` — колір фону на графіку;
`x(yz)range=[min,max]` — задання розмірів по осі $x(yz)$;
`logx(yz)=true/false` — логарифмічна шкала на відповідній осі;
`grid=true/false` — керує лініями поверхні просторових графіків;
`x(yz)tics=true/false` — керує позначками на відповідній осі;
`title="text"` — задає назву для всієї сцени;
`key="text"` — задає ім'я функції у легенді;
`x(yz)label="text"` — задає назву відповідної осі;
`x(yz)axis=true/false` — керує відображенням відповідної осі;
`x(yz)axis_width=N` — задає товщину лінії відповідної осі;
`x(yz)axis_color=colorname` — задає колір відповідної осі;
`x(yz)axis_type=solid/dots` — задає тип лінії відповідної осі;
`line_width=N` — товщина ліній графіків;
`line_type=solid/dots` — тип ліній графіків (за замовчуванням суцільна);
`point_size=N` — задає розмір точок;
`point_type=N` — задає тип точок, можливі значення: -1,0,1,2,...13.
`points_joined=true/false` — задає спосіб зображення точок (з'єднані вони чи ні), за замовчуванням: окремі;
`nticks=N` — задає кількість точок для побудови графіка, за замовчуванням: 30.

Набір кольорів для відображення графіків дуже різноманітний, і ця обставина вигідно відрізняє команду `draw` від команди `plot`, де їх було 7 базових. Повний список наступний: white, black, grey(0-100), gray(0-100), light-gray, light-grey, dark-gray, dark-grey, red, light-red, dark-red, yellow, light-yellow, dark-yellow, green, light-green, dark green, spring-green, forest-green, sea-green, blue, light blue, dark-blue, midnight-blue, navy, medium-blue, royalblue, skyblue, cyan, light-cyan, dark-cyan, magenta, light-magenta, dark-magenta, turquoise, light-turquoise, dark-turquoise, pink, light-pink, dark-pink, coral, light-coral, orange-red, salmon, light-salmon, dark-salmon, aquamarine, khaki, dark khaki, goldenrod, light-goldenrod, dark-goldenrod, gold, beige, brown, orange, dark-orange, violet, dark-violet.

Тип точок можна вказувати як за назвою, так і за його номером: none (-1), dot (0), plus (1), multiply (2), asterisk (3), square (4), filled_square (5), circle (6), filled_circle (7), up_triangle (8), filled_up_triangle (9), down_triangle (10), filled_down_triangle (11), diamant (12), filled_diamant (13).

5.1.4 Локальні опції для підписів та векторів

`label_alignment=flag` — вирівнювання підпису; можливі значення: `center, left, right`;
`label_orientation=flag` — орієнтація підпису; можливі значення: `horizontal, vertical`;
`head_length=N` — довжина верхівки вектора в одиницях осі x (за замовчуванням: 2);
`head_angle=N` — кут між верхівкою вектора та його основою (за замовчуванням: 45°);
`head_type=flag` — тип верхівки вектора; можливі значення: `filled, empty, nofilled`;
`head_both=true/false` — визначає чи вектор двонапрямлений; за замовчуванням: однонапрямлений.

```
(%i1)  [la,hs,ha,hl]:[label_alignment,
       head_size,head_angle,head_length]$
```

```
(%i2)  wxdraw2d(xrange=[-1,1],yrange=[-4,4],
       yaxis=true,line_width=3,grid=true,
       font="Century Gothic",font_size = 18,color=black,
       la=left,label(["Вирівнення тексту ліворуч",0,2]),
       la=center,label(["Вирівнення тексту по центру",0,1]),
       la=right,label(["Вирівнення тексту праворуч",0,0]),
       ha=15,hl=0.1,color=dark_orange,vector([-0.5,-1],[0.5,0]),
       ha=10,hl=0.2,color=navy,head_both=true,vector([-0.5,-2],[1,0]))$
```

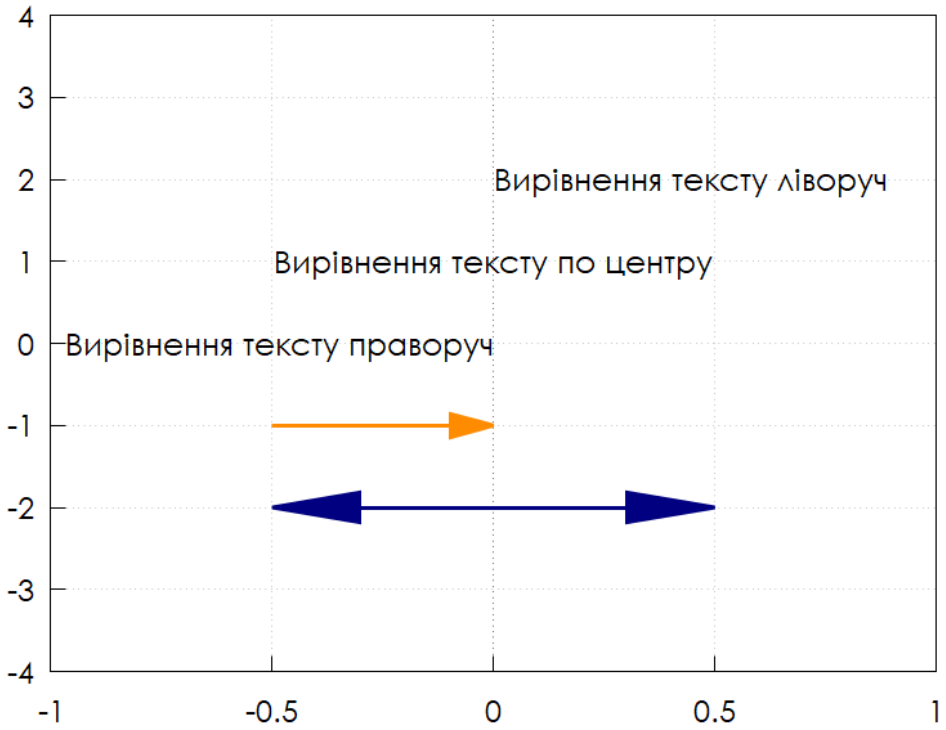


Рис. 5.7: Підписи та вектори.

5.1.5 Локальні опції для 2d-графіки

`axis_bottom=true/false`, `axis_top=true/false`, `axis_left=true/false`, `axis_right=true/false` — керує відображенням нижньої, верхньої, лівої, правої осі;

`color=colorname` — колір об'єктів;

`fill_color=colorname` — задає колір забарвлення фігур;

`fill_density` — задає густину кольору;

`filled_func=true/false` — заповнює відповідним кольором площу під функцією до осі x;

`filled_func=f` — заповнює площу між заданою функцією та функцією f;

`transparent=true/false` — многокутники, зафарбовані командою `fill_color`, будуть кольоровими у випадку `true`.

`border=true/false` — керує межами многокутників.

Задання прямокутника та еліпса з вирізаним сектором:

```
(%i3)  rect:rectangle([1,-2],[6,-7]);
```

(%o3) rectangle ([1, −2], [6, −7])

```
(%i4)  ell:ellipse(6,-6,3,2,0,330);
```

(%o4) $\text{ellipse}\,(6, -6, 3, 2, 0, 330)$

Зображення двох фігур, з чорною межею у прямокутника та без межі у еліпса, відображені ліва вісь без позначок та нижня з позначками, підпис шрифтом Liberation Serif 18, „Фігури“ (Рис.5.8(а)).

```
(%i5)  wxdraw2d(line_width=4,
       fill_color=dark-khaki,color=black,rect,
       border=false,fill_color=dark_orange,nticks=100,ell,
       font="Liberation Serif",font_size=18,title="Фігури")$
```

Зображення функцій $y = \cos x$ та $y = \sin x$, області між ними зафарбовані, самі функції мають межі різних кольорів (Рис.5.8(б)).

```
(%i6)  draw2d(
       filled_func=sin(x),fill_color=navy,
       explicit(cos(x),x,0,10),
       filled_func=false,color=red,line_width=3,nticks=100,
       explicit(sin(x),x,0,10),
       color=yellow,line_width=3,nticks=100,
       explicit(cos(x),x,0,10));
```

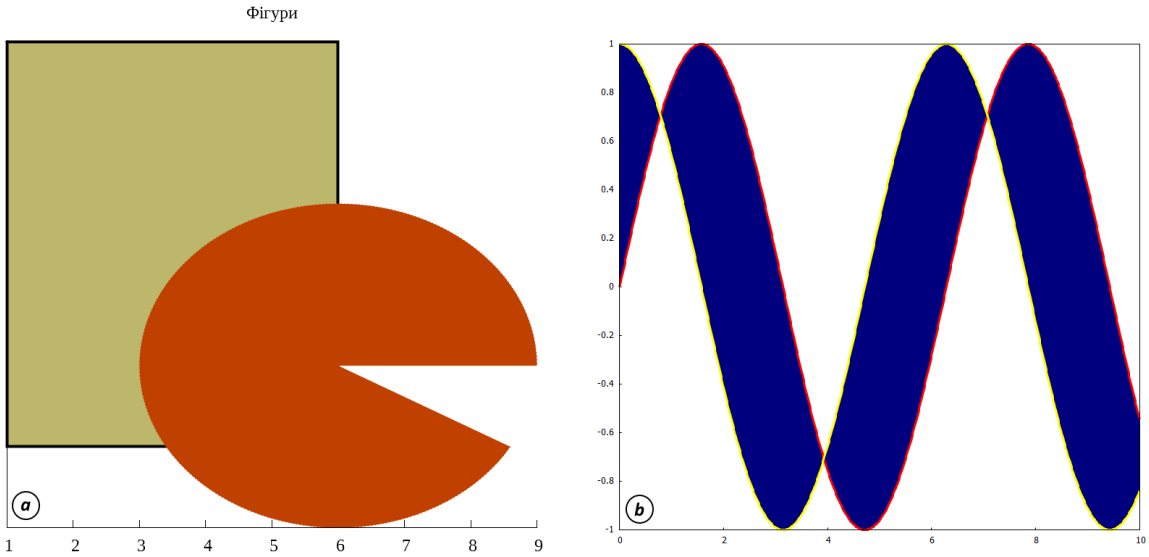


Рис. 5.8: Заповнення фігур кольорами та приклади виділення межами.

5.1.6 Локальні опції для 3d-графіки

`view=[%phi,%theta]` — визначає кут зору на зображення;
`axis_3d=true/false` — керує відображенням всіх трьох осей, за замовчуванням: true;
`xu_grid=N` — визначає кількість ліній поверхні вздовж осі x;
`yv_grid=N` — визначає кількість ліній поверхні вздовж осі y;
`surface_hide=true/false` — відповідає за видимість внутрішніх ліній поверхні у 3d-зображеннях;
`enhanced3d=true/false` — розфарбовує поверхні у 3d-зображеннях;

`palette=[r,g,b]` — числа кольорової палітри у форматі RGB, що використовуються для розфарбовування об'єктів;

`colorbox=true/false` — зображає кольорову шкалу збоку графіка, для відображення кольорів беруться дані із команди `palette` (якщо вона задана). За замовчуванням шкала відображається зі звичайною палітрою кольорів;

`cbrange=[x1,x2]` — задає крайні значення шкали кольорів;

`cbtics={[,,Name1“,x1],[,,Name2“,x2],[,,Name3“,x3]}` — задає підписи окремих значень на бічній шкалі;

`contour=flag` — керує лініями контурів поверхонь; можливі значення: `none, base, surface, both, map`;

`contour_levels=N` — задає кількість контурних ліній поверхні;

`contour_levels=[x1,dx,x2]` — контурні лінії відображаються від x_1 до x_2 із кроком dx;

`contour_levels=x1,x2,...` — контурні лінії відображаються лише на окремих рівнях x_1, x_2,..;

`ip_grid=[nx,ny]` — кількість точок побудови графіка непрямого задання функції; за замовчуванням: [50,50];

Просте задання поверхні (Рис.5.9(а)):

```
(%i1)  draw3d(surface_hide=true,
       explicit((x^2-y^2)*exp(1-x^2-y^2),x,-2,2,y,-2,2));
(%o1)     [gr3d (explicit)]
```

Задання поверхні, контурні лінії будуються як на нижній площині, так і на самій поверхні (Рис.5.9(б)):

```
(%i2)  draw3d(
         color          = gray80,
         explicit(20*exp(-x^2-y^2)-10,x,0,2,y,-3,3),
         contour_levels = 10,
         contour        = both,
         surface_hide   = true) $
```

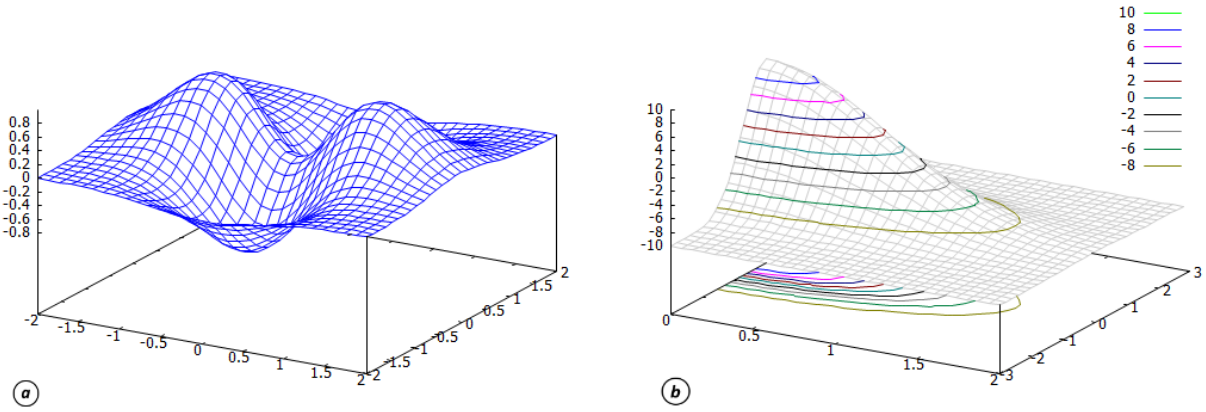


Рис. 5.9: Поверхні у 3d-просторі.

Зображення мушлі (Рис.5.10(а)):

```
(%i3)  draw3d(
              color=midnight-blue,
              surface_hide = true,
              axis_3d=false,
              xtics=none,
              ytics=none,
              ztics=none,
              spherical(a*z,a,0,2.5*%pi,z,0,%pi));
(%o3)     [gr3d (spherical)]
```

Просторова поверхня із заданою палітрою розфарбування, окремі значення на шкалі підписані (Рис.5.10(б)):

```
wxdraw3d( title       = "Задання значень на шкалі",
        enhanced3d    = true,
        dimensions    = [800,800],
        palette       = [-10,15,10],
        color         = gray80,
        cbtics        = {["Високий",10],["Середній",05],["Низький",0]},
        cbrange       = [0, 10],
        explicit(x^2+y^2, x,-2,2,y,-2,2));
```

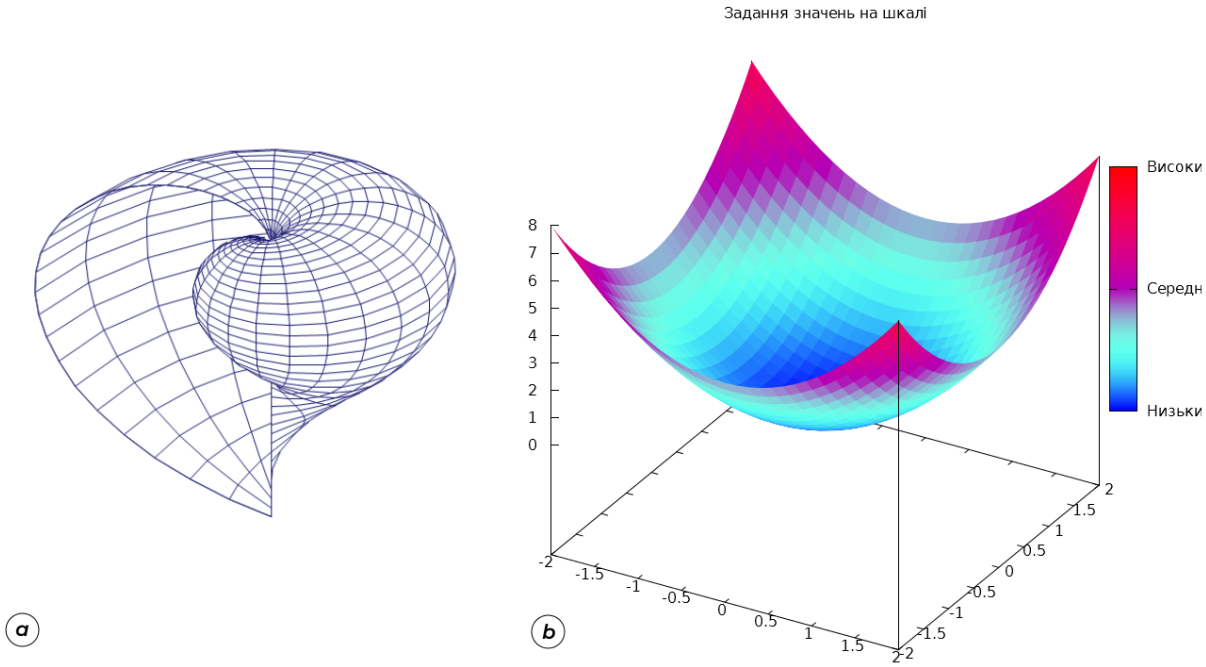


Рис. 5.10: Поверхні мушлі, функції зі шкалою кольору.

5.2 Анімація графіки

За допомогою **wxMaxima** дуже легко створювати графічні анімації. Наприклад, нехай маємо функцію $\sin(x+n)$, що залежить від параметра n. Ми можемо побудувати її графік, змінюючи значення n послідовно через певний інтервал, завдяки чому можна досягнути гарної візуалізації еволюції процесу (це буде хвиля, яка рухається).

Поданий приклад пояснює сам механізм створення анімації: спочатку задається список, який залежить якимось чином від параметра у певних межах, потім задається функція, у якій цей параметр міститься. Система будує послідовний ряд графіків функції, надаючи параметру його значення із списку, потім «склеює» ці зображення у один файл, найчастіше формату .gif. Вихідне зображення поміщається у тіло документу, його переглядають за допогою керуючого слайдера на панелі інструментів. Саме зображення можна зберегти, натиснувши на ньому правою кнопку миші, а на диску комп'ютера у службовій директорії знайти усі проміжні кадри. Трикутник на слайдері запускає та припиняє анімацію, а повзунок дозволяє переміщатися на окремі кадри.

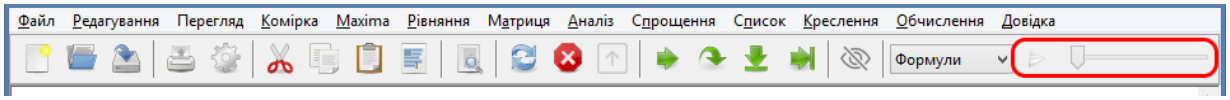


Рис. 5.11: Головна панель інструментів **wxMaxima**, червоним обведено слайдер.

Існує три можливості для створення анімації, які використовують різні засоби побудови зображень:

`with_slider` — анімація за допомогою `plot2d`;
`with_slider_draw` — анімація за допомогою `draw2d`;
`with_slider_draw3d` — анімація за допомогою `draw3d`.

Як бачимо, перші два способи дозволяють зробити анімацію у двовимірному просторі, а третій у тривимірному.

Наприклад, створимо анімацію за допомогою команди `with_slider` функції $\sin(x+n)$, де параметр n буде приймати значення від 1 до 20. Команда `makelist(i,i,1,20)` дає нам всі натуральні числа від 1 до 20.

```
(%i1)  with_slider(n,makelist(i,i,1,20),sin(x+n),
       [x,-2*%pi,2*%pi],[y,-1.1,1.1]);
```

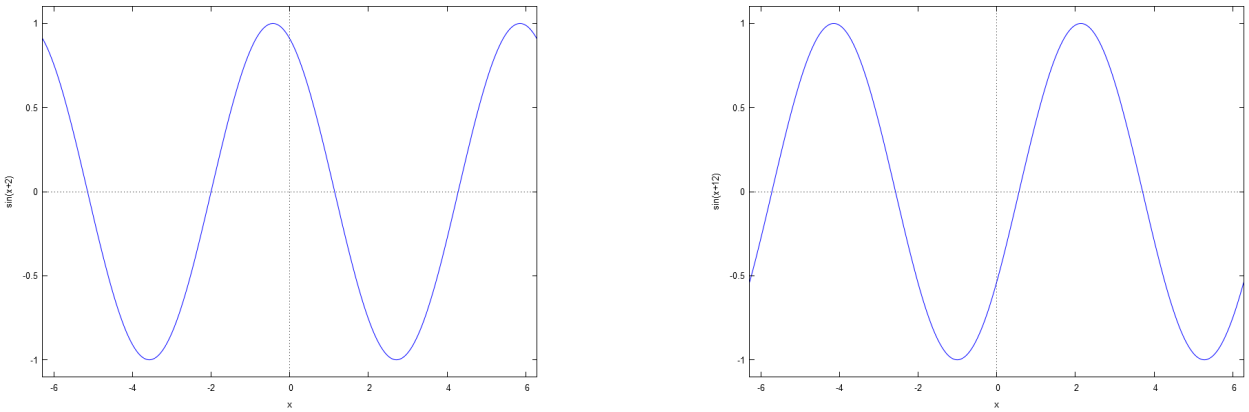


Рис. 5.12: Кілька кадрів анімації $\sin(x+n)$.

Якщо замість `plot2d` використовується пакет `draw` для створення графіків, застосовуються команди `with_slider_draw` або `with_slider_draw3d`. Знову ж таки, спочатку запроваджується параметр, потім список із зазначенням опцій, які прийматиме параметр, далі опції, що стосуються команд `draw` або `draw3d` відповідно. Важливою деталлю в цьому випадку є те, що параметр може впливати не тільки на функцію, але використовуватись

в будь-якій іншій частині виразу. Наприклад, можна скористатись цією обставиною для анімації побудови кола у параметричних координатах.

```
(%i2)  with_slider_draw(
       t,makelist(%pi*i/10,i,1,20),nticks=100,
       parametric(cos(x),sin(x),x,0,t),
       xrange=[-1,1],
       yrange=[-1,1],
       user_preamble="set size ratio 1")$
```

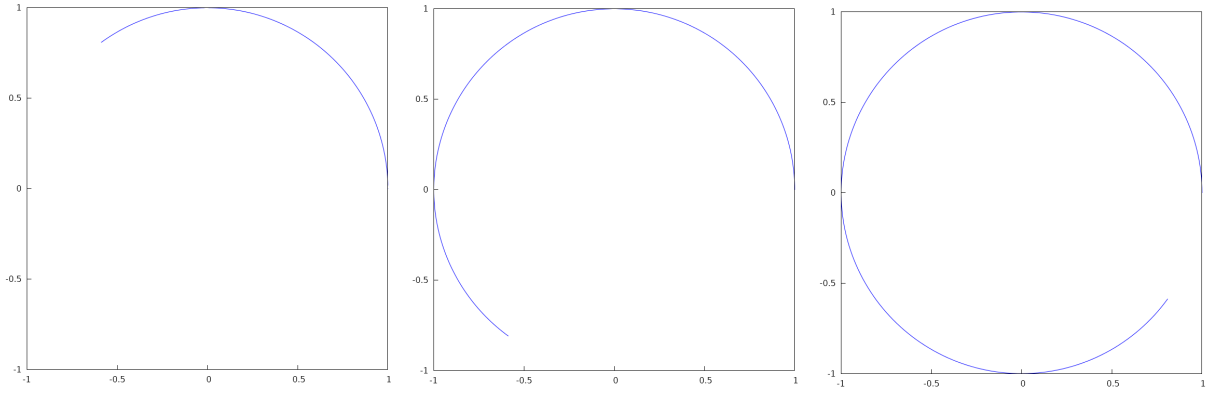


Рис. 5.13: Кадри анімації побудови кола.

Якщо є потреба вивести результат анімації у файл, необхідно скористатись командою `terminal` та вказати значення `animated_gif`. Додаткові параметри: `delay=N` — швидкість відтворення, `dimensions = [N1,N2]` — розмір вихідного файлу у пікселях.

```
(%i3)  draw(terminal   = animated_gif,
            delay      = 40,
            file_name  = "animation1",
            dimensions = [400, 400],
       makelist(gr2d( nticks = 200,explicit(x^(0.1*k),x,0,1)),k,30))$
```

Файл із назвою «animation1.gif» буде знаходитись у папці, де встановлена **Maxima**.

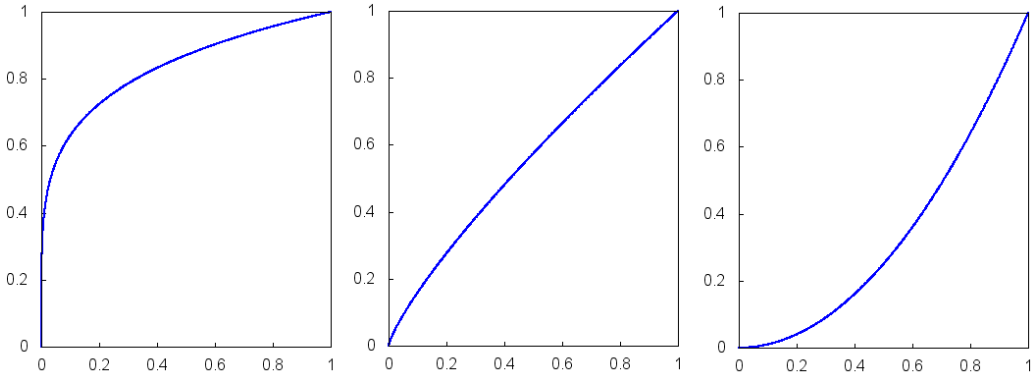


Рис. 5.14: Кадри з файлу «animation1.gif».

Наступний приклад: анімація колового руху радіус-вектора з наочним представленням виникнення функцій синуса та косинуса.

```
(%i4)  with_slider_draw(t,makelist(2*%pi*i/39,i,1,40),
       proportional_axes=xy,
       line_width=3, color=blue,
       parametric(cos(x),sin(x),x,0,t),
       color=light-red, key="sin a",
       explicit(sin(x),x,0,t),
       color=dark-red, key="cos a",
       explicit(cos(x),x,0,t),
       head_length=0.1,
       color=dark-red, key="",head_angle=15,
       vector([0,0],[cos(t),0]),
       color=light-red,
       head_length=0.1, key="",head_angle=15,
       vector([0,0],[0,sin(t)]),
       head_length=0.1,head_angle=15, head_type=empty,key="",
       vector([0,0],[cos(t),sin(t)]),
       xaxis=true,yaxis=true,
       title="Функції синуса та косинуса",
       xrange=[-1,2*%pi],yrange=[-1,1]);
```

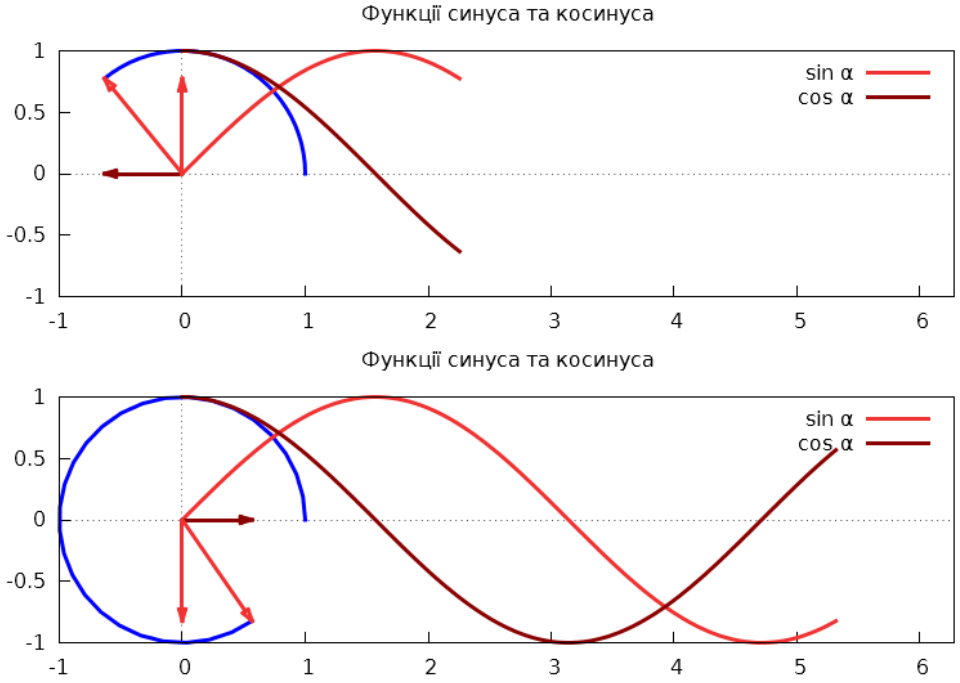


Рис. 5.15: Кадри анімації побудови функцій $\sin\alpha$, $\cos\alpha$.

Розглянемо тепер анімацію зображень у 3d-просторі. Створимо складну фігуру, яка буде обертатись навколо своєї осі. Обертання об'єкту можна розглядати як зміну кута зору на нього, тому тут в нагоді стане команда `view=[N1,N2]`, яка це і відповідає. Кут огляду по вертикалі оберемо незмінним, рівним 75°, тоді як кут огляду по горизонталі буде залежати від параметра, який міняється від 1 до певного кінцевого значення (його добуток на відповідний множник обов'язково повинен давати 360° для повного оберту).

```
(%i5)  with_slider_draw3d(
       k,makelist(i*18,i,1,20),
       parametric_surface(cos(u)+0.5*cos(u)*cos(v),
       sin(u)+0.5*sin(u)*cos(v),0.5*sin(v),
       u, -%pi, %pi, v, -%pi, %pi),
       parametric_surface(1+cos(u)+0.5*cos(u)*cos(v),
       0.5*sin(v),sin(u)+0.5*sin(u)*cos(v),
       u, -%pi, %pi, v, -%pi, %pi),
       surface_hide=true,view=[75,k],delay=10)$
```

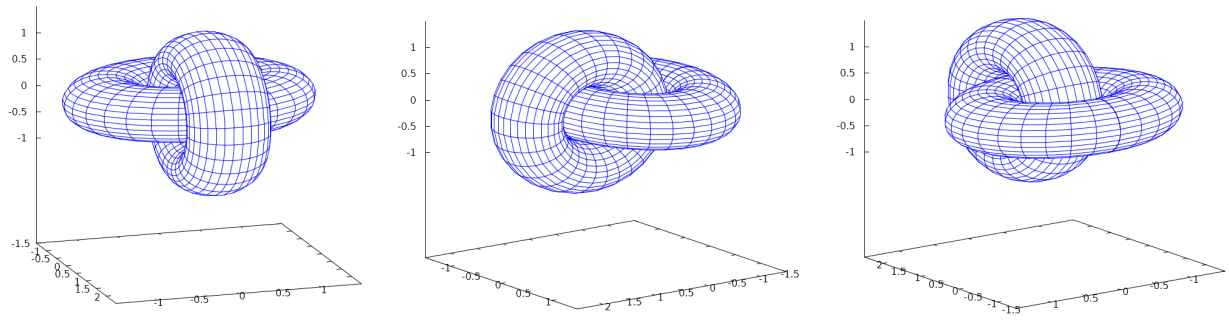


Рис. 5.16: Кадри обертання складної фігури.

Підкреслимо, що у команді `makelist(i*18,i,1,20)` множник біля i та кінцеве значення параметру мають бути пов'язані так, щоб давати добуток 360°. Гладкість обертання залежить від кількості побудованих фреймів, тому збільшення кінцевого значення від 20 до 100 посприяє плавності анімації. Але при цьому збільшується час побудови всіх графіків та зростає розмір кінцевого файлу. Помінявши місцями у команді `view=[75,k]` значення змінних, можна створити обертання вже по вертикалі. Сталий кут огляду можна зробити яким завгодно. Звісно, ніщо не заважає також запровадити два параметри і змінювати обертання по обох осях одночасно.

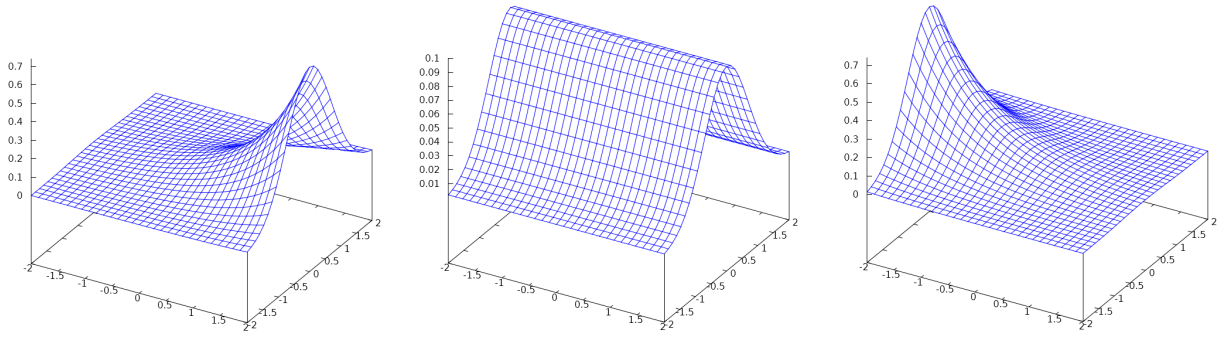


Рис. 5.17: Кадри зміни просторової функції.

Розглянемо ще трансформацію поверхні у 3d-просторі. Це вже буде не обертання, а просто зміна її форми (Рис.5.17).

```
(%i6)  with_slider_draw3d(
       k,makelist(i,i,-7,7),
       explicit(0.1*exp(-k*x-y^2),x,-2,2,y,-2,2),
       surface_hide=true,delay=10)$
```

Анімація побудови тора у параметричних координатах (Рис.5.18):

```
(%i7)  with_slider_draw3d(
       n,makelist(0.1*%pi*i,i,1,20),
       surface_hide=true,
       xrange=[-3,3],
       yrange=[-3,3],
       parametric_surface(cos(u)*(2+cos(v)),sin(u)*(2+cos(v)),sin(v),
       u,0,n,v,0,2*%pi)
       );
```

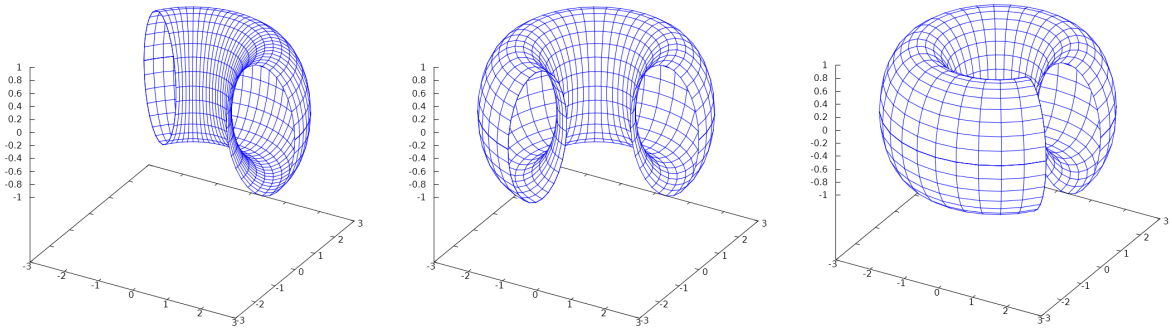


Рис. 5.18: Кадри анімації побудови тора.

Як бачимо, у команді задання поверхні параметр u міняється від 0 до n. Якщо такого ж значення надати параметру v, тор буде розгортатись по вертикалі.

5.3 Побудова полів напрямків диференціальних рівнянь командою `plotdf`

Команда `plotdf` дозволяє побудувати траєкторії та поле напрямків диференціального рівняння першого порядку або автономної системи двох звичайних диференціальних рівнянь першого порядку. Ця функція включена в додатковий пакет, тому для її використання треба спочатку завантажити його командою `load(plotdf)`.

Для побудови поля напрямків диференціального рівняння першого порядку воно має бути записане у вигляді $\frac{dy}{dx} = f(x,y)$. У випадку автономної системи воно має бути записане у вигляді $\begin{cases} \frac{dx}{dt} = G(x,y); \\ \frac{dy}{dt} = H(x,y). \end{cases}$

Синтаксис функції:

`plotdf(f(x,y),opts)` — для одиночного рівняння;

`plotdf([G(x,y),H(x,y)],opts)` — для автономної системи рівнянь.

Якщо у рівняннях використовуються не змінні (x,y), а наприклад (u,v), синтаксис команди буде подібним, лише необхідно явним вчином вказати на ті величини, відносно

яких ЗДР розв'язується:

`plotdf(f(u,v),[u,v],opts);`

`plotdf([F(u,v),G(u,v)],[u,v],opts).`

Крім цих змінних, можуть використовуватися опції `opts`, які обираються за допомогою пунктів меню, а також задаються за допомогою спеціальних команд. Опції записуються у квадратних дужках, їхні значення задаються через кому.

`nsteps` — задає кількість кроків, надану незалежній змінній для обчислення інтегральної кривої; значення за замовчуванням: 100;

`direction` — задає напрямок незалежної змінної, вздовж якої обчислюється інтегральна крива; можливі значення: `forward, backward, both`;

`tinitial` — задає початкове значення змінної t; значення за замовчуванням: 0;

`versus_t` — при значенні 1 створюється друге вікно з графіком інтегральної кривої як двох функцій $x(t)$, $y(t)$; значення за замовчуванням: 0;

`trajectory_at` — задає координати початкової точки інтегральної кривої;

`parameters` — задає список параметрів та їх числові значення, які використовуються в розв'язуванні диференціальних рівнянь;

`sliders` — визначає список параметрів, які будуть змінюватися інтерактивно за допомогою повзунка та діапазон зміни цих параметрів;

`[x,x1,x2]` — зміна значень змінної x; за замовчуванням $[-10; 10]$;

`[y,y1,y2]` — зміна значень змінної y; за замовчуванням $[-10; 10]$;

`number_of_arrows` — задає кількість стрілочок; значення за замовчуванням: 225.

Нехай маємо завдання:

► Побудувати поле напрямків диференціального рівняння $\dfrac{dy}{dx} = y + \dfrac{1}{x^2+1}$ та зобразити інтегральну криву, що проходить через точку $(2; 1)$.

```
(%i1)  load(plotdf)$
(%i2)  plotdf(y+1/(x^2+1),[trajectory_at,2,1],[direction,forward],
       [versus_t,1])$
```

Результатом є два зображення (Рис.5.19): поле напрямків та параметричний графік змінних x та y.

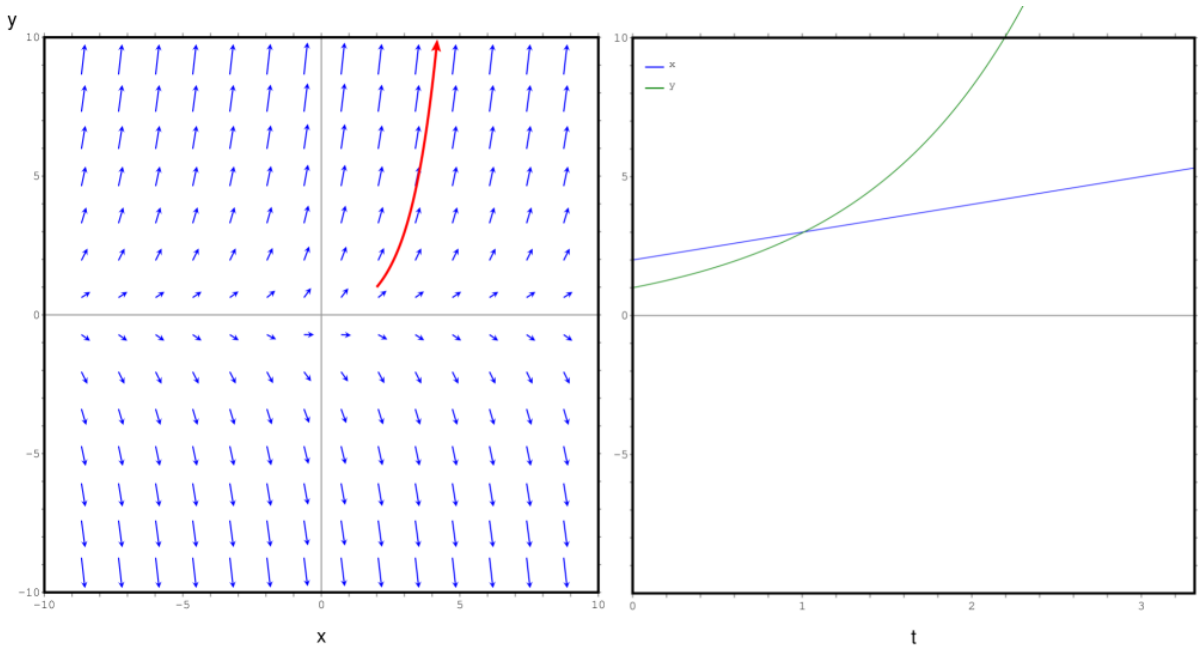
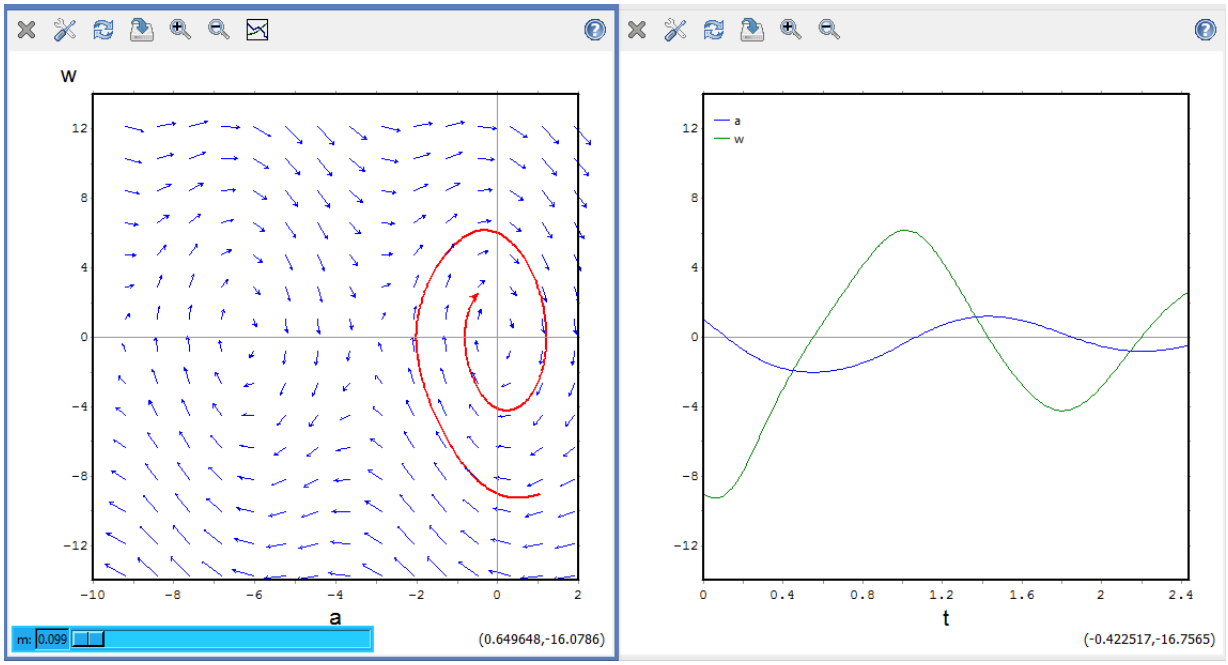
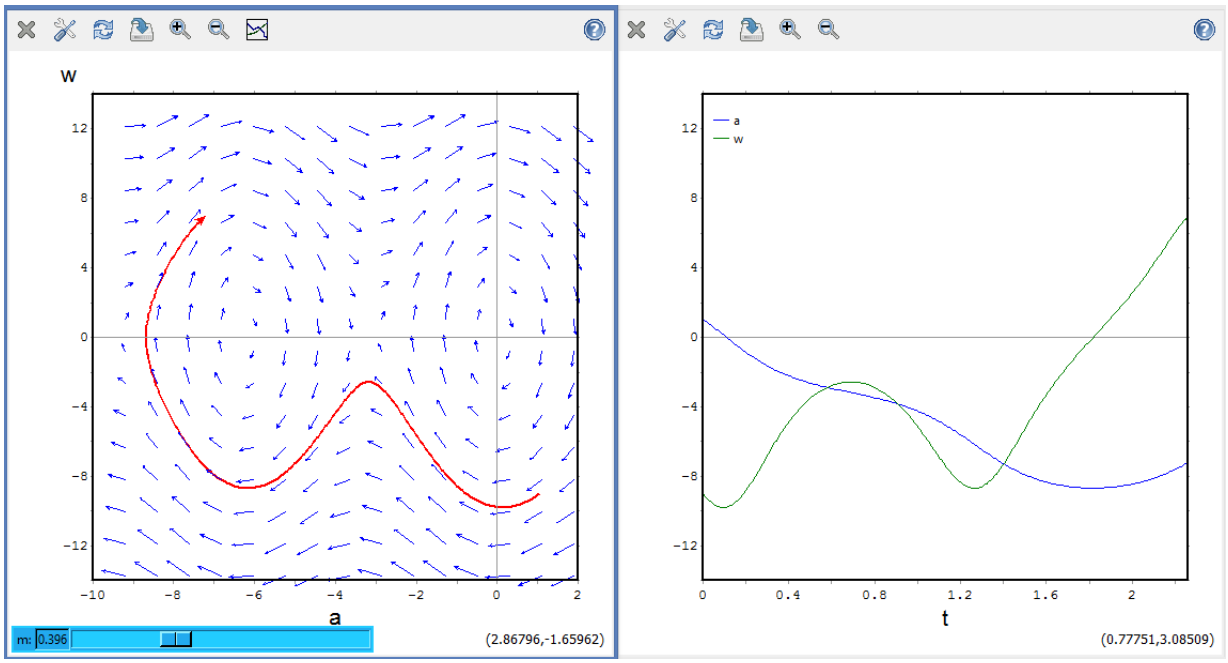


Рис. 5.19: Графіки поля напрямків та інтегральної кривої.

► Побудувати поле напрямків для згасаючого маятника $\begin{cases} \frac{da}{dt} = w; \\ \frac{dw}{dt} = -\frac{g}{l}\sin a - \frac{bw}{ml}, \end{cases}$ із початковими умовами $g = 9.81, l = 0.5, m = 0.3, b = 0.05$. Задати можливість зміни маси повзунком від 0.1 до 1 кг. Побудувати інтегральну криву для значень $a = 1.05$, $w = -9$. Побудувати графіки змінних стану a та w як функцій часу.

```
(%i3)  plotdf([w,-g*sin(a)/l-(b*w)/(m*l)], [a,w],
       [parameters,"g=9.8,l=0.5,m=0.3,b=0.05"],
       [trajectory_at,1.05,-9],[tstep,0.01],
       [a,-10,2], [w,-14,14], [direction,forward],
       [nsteps,300], [sliders,"m=0.1:1"], [versus_t,1])$
```





Побудова розв'язків даної динамічної системи дає нам два вікна: зліва зображене поле напрямків із виділеною інтегральною кривою, яка проходить через задану точку,

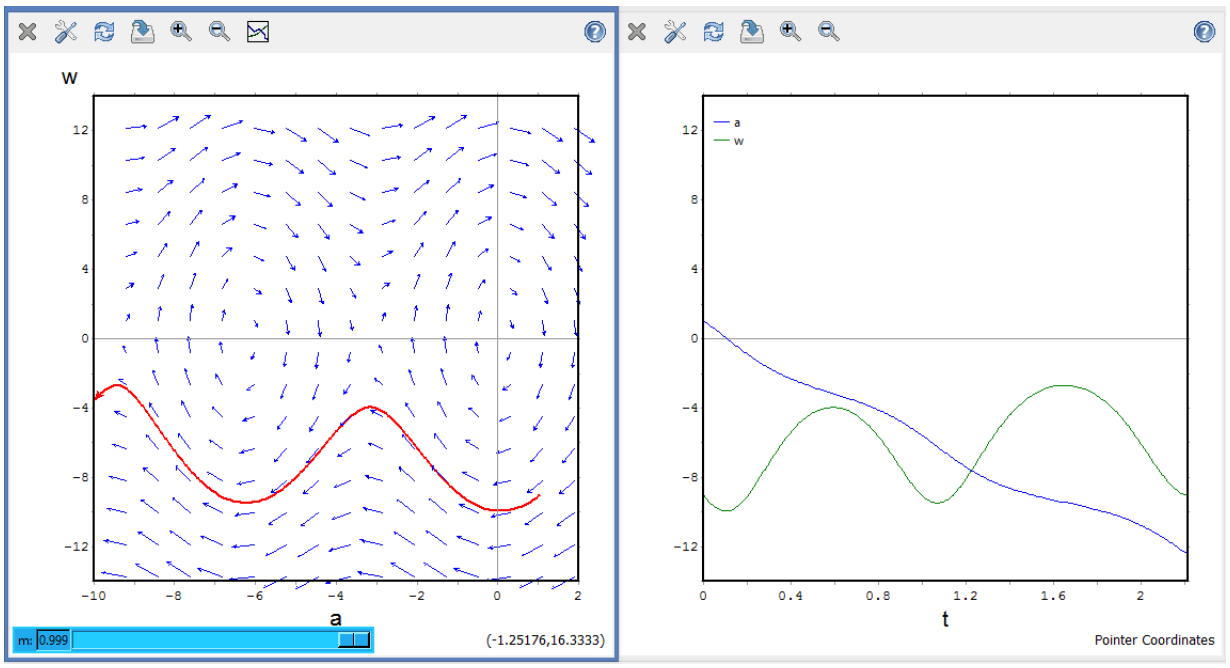


Рис. 5.20: Поле напрямків та інтегральні криві для різних значень маси маятника.

справа містяться графіки функцій $a(t)$ та $w(t)$. Знизу під графіком поля напрямків розміщений слайдер з повзунком, який може змінювати значення маси у заданих межах. При русі повзунка видно зміну поведінки системи, це чітко відображається як у вигляді інтегральної кривої, так і у вигляді ліній змінних a та w.

Розділ 6

Задачі лінійної алгебри

Застосунок **Maxima** містить значну кількість функцій для розв'язання різноманітних задач лінійної алгебри. Основні відомості про матриці вже були розглянуті раніше, в Розділі 3. Зараз зупинимось докладніше на застосуванні матриць та операціях з ними.

6.1 Основні операції з матрицями

Інтерфейс **wxMaxima** достатньо зручний, він не потребує вручну вводити дані для виклику команди `matrix` (в численних квадратних дужках можна легко загубитись). Потрібно заповнити допоміжні форми, використовуючи пункти меню «Алгебра — Створити матрицю» або «Створити матрицю за виразом». При цьому з'явиться вікно з комірками для введення чисел.

Нагадаємо основи. В **Maxima** на матрицях означені звичні операції домноження на число, додавання та матричного множення. Останнє реалізується за допомогою бінарної операції `«.»` (нижня крапка). Розмірності матриць-співмножників мають бути узгоджені.

```
(%i1)  M:matrix([1,2,3],[4,5,6]);
```

$$(\%o1)\quad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

```
(%i2)  N:matrix([1,-2],[2,-3],[3,-4]);
```

$$(\%o2)\quad \begin{pmatrix} 1 & -2 \\ 2 & -3 \\ 3 & -4 \end{pmatrix}$$

Якщо матриця — лівий множник, то правим співмножником може бути вектор-стовпець, вектор-рядок або список. **Maxima** дозволяє також підносити матриці до степеня, але фактично ця операція застосовується до кожного елемента.

```
(%i3)  M.N;
```

$$(\%o3)\quad \begin{pmatrix} 14 & -20 \\ 32 & -47 \end{pmatrix}$$

`transpose(M)` — транспонує матрицю.
`invert(M)` — шукає обернену матрицю.
`determinant(M)` — знаходить детермінант матриці.

```
(%i4)  M2:transpose(M);
```

$$(\%o4)\quad \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

```
(%i5)  K:(%o3);
```

$$(\%o5)\quad \begin{pmatrix} 14 & -20 \\ 32 & -47 \end{pmatrix}$$

```
(%i6)  invK:invert(K);
```

$$(\%o6)\quad \begin{pmatrix} \frac{47}{18} & -\frac{10}{9} \\ \frac{16}{9} & -\frac{7}{9} \end{pmatrix}$$

```
(%i7)  K.invK;
```

$$(\%o7)\quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
(%i8)  determinant(K);
```

$$(\%o8)\quad -18$$

```
(%i9)  determinant(invK);
```

$$(\%o9)\quad -\frac{1}{18}$$

`charpoly(M,%lambda)` — обчислює характеристичний поліном матриці (M — матриця, λ — змінна, відносно якої будується поліном). Корені характеристичного полінома є власними значеннями матриці.

```
(%i10) charpoly(K,%lambda);
```

$$(\%o10)\quad (-\lambda - 47)\cdot(14-\lambda)+640$$

```
(%i11) ratsimp(%o10);
```

$$(\%o11)\quad \lambda^2+33\cdot\lambda-18$$

```
(%i12) solve((%o11)=0,%lambda);
```

$$(\%o12)\quad [\lambda=-\frac{3\cdot\sqrt{129}+33}{2},\lambda=\frac{3\cdot\sqrt{129}-33}{2}]$$

Однак для обчислення власних значень та власних векторів матриці зазвичай використовують спеціальні команди: `eigenvalues` та `eigenvectors`.

`eigenvalues(M)` — обчислює власні значення матриці аналітично, якщо це можливо. Для знаходження коренів характеристичного полінома використовується функція `solve`. Результат повертається у вигляді списка, що містить два підсписки: перший містить власні значення, а другий — їх кратності.

```
(%i13) eigenvalues(K);
```

$$(\%o13)\quad [[-\frac{3\cdot\sqrt{129}+33}{2},\frac{3\cdot\sqrt{129}-33}{2}],[1,1]]$$

`eigenvectors(M)` — аналітично обчислює власні значення та власні вектори матриці, якщо це можливо. Вона повертає список, перший елемент якого — список власних значень (аналогічно до `eigenvalues`), а далі йдуть власні вектори, кожен з яких представлений як список своїх проєкцій.

```
(%i14) eigenvectors(K);
```

$$(\%o14)\quad [[[-\frac{3\cdot\sqrt{129}+33}{2},\frac{3\cdot\sqrt{129}-33}{2}],[1,1]],[[[1,\frac{3\cdot\sqrt{129}+61}{40}]],[[1,-\frac{3\cdot\sqrt{129}-61}{40}]]]]$$

Функція `uniteigenvectors` відрізняється від команди `eigenvectors` тим, що повертає нормовані до одиничної довжини власні вектори.

Maxima включає в себе спеціальну функцію для обчислення ортонормованого набору векторів із заданого. Використовується стандартний алгоритм Грама-Шмідта. Синтаксис виклику:

`gramschmidt(M)` або `gschmidt(M)`.

Аргумент функції — матриця або список. Як компоненти системи векторів, на базі якої будується ортонормована система, розглядаються рядки матриці `M` або підсписки списку `M`. Для використання цієї функції необхідно завантажити пакет `«eigen»`.

```
(%i15) load(eigen);
(%i16) L:gramschmidt(M);
```

$$(\%o16)\quad [[1,2,3],[\frac{2^2\cdot 3}{7},\frac{3}{7},-\frac{2\cdot 3}{7}]]$$

```
(%i17) L[1].L[2];
```

$$(\%o17)\quad \frac{2^2\cdot 3}{7}-\frac{12}{7}$$

```
(%i18) ratsimp(%o17);
```

$$(\%o18)\quad 0$$

Перетворення матриці до трикутної форми здійснюється методом виключення Гаусса за допомогою команди `echelon(M)`. Аналогічний результат дає команда `triangularize(M)`, відмінності полягають в тому, що `echelon` нормує діагональний елемент на 1, а `triangularize` — ні.

```
(%i19) triangularize(K);
```

$$(\%o19)\quad \begin{pmatrix}14 & -20\\ 0 & -18\end{pmatrix}$$

```
(%i20) echelon(K);
```

$$(\%o20)\quad \begin{pmatrix}1 & -\frac{10}{7}\\ 0 & 1\end{pmatrix}$$

Для розрахунку рангу матриці (порядку найбільшого невиродженого мінора матриці) використовується команда `rank(M)`.

```
(%i21) rank(M);
```

$$(\%o21)\quad 2$$

```
(%i22) rank(N);
```

$$(\%o22)\quad 2$$

```
(%i23) echelon(N);
```

$$(\%o23)\quad \begin{pmatrix}1 & -2\\ 0 & 1\\ 0 & 0\end{pmatrix}$$

Мінор матриці знаходиться командою `minor(M,i,j)`, де `M` — матриця, `i,j` — індекси елементу, для якого обчислюється мінор.

```
(%i24) minor(M,1,1);
```

$$(\%o24) \quad \begin{pmatrix} 5 & 6 \end{pmatrix}$$

```
(%i25) minor(N,2,2);
```

$$(\%o25) \quad \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

6.2 Розв'язок системи лінійних алгебричних рівнянь

Нехай маємо наступну систему рівнянь:

$$\begin{cases} x + 2y + 10z = -15; \\ 2x + 4y - z = 12; \\ x + y - 3z = 9. \end{cases}$$

Спробуємо розв'язати її, використовуючи для демонстрації різні підходи.

6.2.1 Стандартний метод `solve`

Нагадаємо синтаксис команди `solve` для систем: необхідно сформувати список із рівнянь, та в кінці виписати списком змінні, відносно яких шукати розв'язок.

```
(%i1) solve([x+2*y+10*z=-15,
             2*x+4*y-z=12,
             x+y-3*z=9],
            [x,y,z]);
```

$$(\%o1) \quad [[x = 1, y = 2, z = -2]]$$

6.2.2 Метод Крамера

Знайдемо основний визначник, який складається з коефіцієнтів при змінних.

```
(%i2) D:matrix([1,2,10],[2,4,-1],[1,1,-3]);
```

$$(\%o2) \quad \begin{pmatrix} 1 & 2 & 10 \\ 2 & 4 & -1 \\ 1 & 1 & -3 \end{pmatrix}$$

```
(%i3) %Delta:determinant(D);
```

$$(\%o3) \quad -21$$

Оскільки $\Delta \neq 0$, система має єдиний розв'язок. Знайдемо тепер додаткові визначники, які утворюються підстановкою стовпчика правих частин рівнянь замість відповідного стовпчика коефіцієнтів при кожній змінній.


```
(%i2) D:matrix([1,2,10],[2,4,-1],[1,1,-3]);
```

$$(\%o2) \quad \begin{pmatrix} 1 & 2 & 10 \\ 2 & 4 & -1 \\ 1 & 1 & -3 \end{pmatrix}$$

```
(%i3) %Delta:determinant(D);
```

$$(\%o3) \quad -21$$

```
(%i4)  D1:matrix([-15,2,10],[12,4,-1],[9,1,-3]);
```

$$(\%o4)\quad \begin{pmatrix} -15 & 2 & 10 \\ 12 & 4 & -1 \\ 9 & 1 & -3 \end{pmatrix}$$

```
(%i5)  %Delta1:determinant(D1);
```

$$(\%o5)\quad -21$$

```
(%i6)  D2:matrix([1,-15,10],[2,12,-1],[1,9,-3]);
```

$$(\%o6)\quad \begin{pmatrix} 1 & -15 & 10 \\ 2 & 12 & -1 \\ 1 & 9 & -3 \end{pmatrix}$$

```
(%i7)  %Delta2:determinant(D2);
```

$$(\%o7)\quad -42$$

```
(%i8)  D3:matrix([1,2,-15],[2,4,12],[1,1,9]);
```

$$(\%o8)\quad \begin{pmatrix} 1 & 2 & -15 \\ 2 & 4 & 12 \\ 1 & 1 & 9 \end{pmatrix}$$

```
(%i9)  %Delta3:determinant(D3);
```

$$(\%o9)\quad 42$$

Тоді, за правилом Крамера $x = \frac{\Delta_1}{\Delta}, y = \frac{\Delta_2}{\Delta}, z = \frac{\Delta_3}{\Delta}$, маємо

```
(%i13) x=%Delta1/%Delta; y=%Delta2/%Delta; z=%Delta3/%Delta;
```

$$(\%o11)\quad x = 1$$

$$(\%o12)\quad y = 2$$

$$(\%o13)\quad z = -2$$

6.2.3 Метод Гаусса

Цей метод полягає у послідовному виділенні кожної змінної, виразивши її через інші. Підставляючи отримані невідомі знизу вгору, можна утворювати рівняння зі зменшеною на одиницю кількістю змінних, аж поки не дійдемо до рівняння з однією змінною. В матричному представленні це означає зведення розширеної матриці Гаусса до трикутного вигляду.

Сформуємо розширену основну матрицю `G`, що утворена з усіх коефіцієнтів і правих сторін системи.

```
(%i1)  G:matrix([1,2,10,-15],[2,4,-1,12],[1,1,-3,9]);
```

$$(\%o1)\quad \begin{pmatrix} 1 & 2 & 10 & -15 \\ 2 & 4 & -1 & 12 \\ 1 & 1 & -3 & 9 \end{pmatrix}$$

Зведемо цю матрицю до трикутного виду через команду `echelon`.

```
(%i2)  echelon(G);
```

$$(\%o2)\quad \begin{pmatrix} 1 & 2 & 10 & -15 \\ 0 & 1 & 13 & -24 \\ 0 & 0 & 1 & -2 \end{pmatrix}$$

Отримали систему рівнянь, яку розв'язуємо знизу догори.

```
(%i3)  solve([x+2*y+10*z=-15,y+13*z=-24,z=-2],[x,y,z]);
```

(%o3) $[[x = 1, y = 2, z = -2]]$

6.2.4 Метод оберненої матриці

Виписану раніше систему можна представити в загальному вигляді як

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix},$$

або скорочено $A \cdot X = B$. Домноживши зліва цю рівність на матрицю A^{-1}, можна вивільнити X та знайти невідомі: $X = A^{-1} \cdot B$.

```
(%i1)  A:matrix([1,2,10],[2,4,-1],[1,1,-3]);
```

(%o1) $\begin{pmatrix} 1 & 2 & 10 \\ 2 & 4 & -1 \\ 1 & 1 & -3 \end{pmatrix}$

```
(%i2)  B:matrix([-15],[12],[9]);
```

(%o2) $\begin{pmatrix} -15 \\ 12 \\ 9 \end{pmatrix}$

```
(%i3)  invA:invert(A);
```

(%o3) $\begin{pmatrix} \frac{11}{21} & -\frac{16}{21} & 2 \\ -\frac{5}{21} & \frac{13}{21} & -1 \\ \frac{2}{21} & -\frac{1}{21} & 0 \end{pmatrix}$

```
(%i4)  X=invA.B;
```

(%o4) $X = \begin{pmatrix} 1 \\ 2 \\ -2 \end{pmatrix}$

Як бачимо, всі методи приводять до однакових результатів.

6.3 Спеціальні функції для розв'язання систем рівнянь

Команда `linsolve([expr1,expr2,...,exprN],[x1,x2,...,xN])` — розв'язує список системи лінійних рівнянь `[expr1,expr2,...,exprN]` відносно переліку змінних `[x1,x2,...,xN]`. Вирази `expr` можуть бути поліномами зазначених змінних і представлятися в вигляді рівнянь. Кількість змінних не обов'язково повинна бути рівна кількості рівнянь.

Нехай маємо систему з трьох рівнянь, яку розв'яжемо командою `linsolve`.

$$\begin{cases} x + y + z + t = 0; \\ 2x - 2y + z + 3t = 2; \\ 3x - y + 2z - t = 8. \end{cases}$$

```
(%i1)  linsolve([x+y+z+t=0,2*x-2*y+z+3*t=2,3*x-y+2*z-t=8],[x,y,z,t]);
```

(%o1) $[x = -\frac{3 \cdot \%r1 - 8}{4}, y = -\frac{5 \cdot \%r1 + 16}{20}, z = \%r1, t = -\frac{6}{5}]$

Таким чином, загальний розв'язок має вигляд:

$$x = -\frac{3C-8}{4}, \quad y = -\frac{5C+16}{20}, \quad z = C, \quad t = -\frac{6}{5},$$

де C — довільна стала. Їй можна задавати довільні дійсні значення, при кожному з яких виходить частинний розв'язок. Наприклад, при $C = 1$ отримаємо розв'язок

```
(%i2)  ev((%o1),%r1=1);
```

(%o2) $[x = \frac{5}{4}, y = -\frac{21}{20}, z = 1, t = -\frac{6}{5}]$

Спосіб подання розв'язків визначається опцією `linsolve_params` (за умовчанням `true`) Якщо вказаний прапор встановлений у `true`, розв'язання недовизначених систем включає параметри `%r1,%r2` і т.д. Якщо прапор `linsolve_params` встановлений у `false`, пов'язані змінні виражаються через вільні (у попередньому прикладі це було б z).

Багато в чому аналогічний результат дозволяє отримати функція `algsys` (фактично це надбудова над `solve`). Кількість рівнянь може перевищувати кількість невідомих, чи навпаки.

Для обчислення коренів одиничних поліноміальних рівнянь використовується функція `realroots`. Варіанти синтаксису:

`realroots(expr,bound);`
`realroots(eqn,bound);`
`realroots(expr);`
`realroots(eqn).`

Тут `bound` — це межа точності, з якою потрібно знайти корені. Команда знаходить всі корені виразу `expr=0` або рівняння `eqn`. Функція будує послідовність Штурма для ізоляції кожного кореня та використовує алгоритм розподілу навпіл для уточнення кореня з точністю `bound` або точністю, заданою за замовчуванням.

Наприклад, потрібно розв'язати рівняння $x^5 - x + 2 = 0$ з точністю до $5 \cdot 10^{-6}$.

```
(%i3)  realroots(2-x+x^5,5e-6);
```

(%o3) $[x = -\frac{664361}{524288}]$

```
(%i4)  (%o3),numer;
```

(%o4) $[x = -1.267168045043945]$

```
(%i5)  ev(2-x+x^5,x=(%o3)),numer;
```

(%o5) $[x^5 - x + 2 = 3.085850166506532 \cdot 10^{-6}]$

Усі корені полінома (дійсні та комплексні) можна знайти за допомогою команди `allroots`. Спосіб подання розв'язку визначається змінною `polyfactor` (за умовчанням `false`; якщо встановити в `true`, то команде поверне результат факторизації). Алгоритм пошуку коренів напівчисловий.

```
(%i6)  allroots(x^4+1);
```

(%o6) $[x = 0.7071067811865475 \cdot i + 0.7071067811865476,$
$x = 0.7071067811865476 - 0.7071067811865475 \cdot i,$
$x = 0.7071067811865478 \cdot i - 0.7071067811865476,$
$x = -0.7071067811865478 \cdot i - 0.7071067811865476]$

```
(%i7)  polyfactor:true$ allroots(x^4+1);
```

(%o7) $1.0 \cdot \left(x^2 - 1.414213562373095 \cdot x + 0.9999999999999999\right) \cdot \left(x^2 + 1.414213562373095 \cdot x + 1.0\right)$

Спрощення систем рівнянь досягається також функцією `eliminate`, що дозволяє виключити ті чи інші змінні. Виклик `eliminate([eqn1,...,eqnN], [x1,...,xK])` виключає змінні `[x1,...,xK]` із зазначених виразів.

6.4 Комплексні числа

Комплексні вирази означують в системі **Maxima** двома способами: в алгебричній формі $z = a + i \cdot b$ та експоненційній $z = r \cdot e^{i\phi}$, тригонометрична форма є додатковою та результати розв'язків у ній не відображаються.

```
(%i1)  solve(x^3-1=0,x);
```

(%o1) $[x = \frac{\sqrt{3} \cdot i - 1}{2}, x = -\frac{\sqrt{3} \cdot i + 1}{2}, x = 1]$

```
(%i2)  solve(x^5-1=0,x);
```

(%o2) $[x = e^{\frac{2 \cdot i \cdot \pi}{5}}, x = e^{\frac{4 \cdot i \cdot \pi}{5}}, x = e^{-\frac{4 \cdot i \cdot \pi}{5}}, x = e^{-\frac{2 \cdot i \cdot \pi}{5}}, x = 1]$

Кількість коренів, які повертаються **Maxima**, відповідає основній теоремі алгебри (рівняння третього степеня має три корені, п'ятого — п'ять і т.д.). Перетворення комплексних виразів може здійснюватися звичайними командами для спрощення (`radcan`, `expand` та ін), але передбачено і ряд специфічних функцій, що розраховані на операції саме з комплексними числами.

Команди `realpart(z)` та `imagpart(z)` обчислюють відповідно дійсну та уявну частину числа z.

```
(%i1)  z:sqrt(3)+%i;
```

(%o1) $i + \sqrt{3}$

```
(%i2)  realpart(z);
```

(%o2) $\sqrt{3}$

```
(%i3)  imagpart(z);
```

(%o3) 1

Команди `cabs(z)` та `carg(z)` — повертають відповідно модуль та аргумент числа z.

```
(%i4)  cabs(z);
```

(%o4) 2

```
(%i5)  cabs(exp((%pi*%i)/4));
```

(%o5) 1

```
(%i6)  carg(z);
```

(%o6) $\frac{\pi}{6}$

```
(%i7)  carg(exp((%pi*%i)/4));
```
(%o7) $\frac{\pi}{4}$

Комплексно спряжені числа обчислюються командою `conjugate(z)`.

```
(%i8)  conjugate(z);
```
(%o8) $\sqrt{3}-i$

```
(%i9)  conjugate(exp((%pi*%i)/4));
```
(%o9) $\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$

Команда `polarform(z)` перетворює алгебричну форму комплексного числа в експоненційну, `rectform(z)` — навпаки, із експоненційної в алгебричну.

```
(%i10) polarform(z);
```
(%o10) $2\cdot e^{\frac{i\cdot\pi}{6}}$

```
(%i11) rectform(exp((%pi*%i)/4));
```
(%o11) $\frac{i}{\sqrt{2}}+\frac{1}{\sqrt{2}}$

Команда `residue(expr,z,z0)` — обчислює рештку у комплексній площині для виразу `expr`, коли змінна z приймає значення z_0. Нагадаємо, що рештка — це коефіцієнт при $(z-z_0)^{-1}$ у розкладі виразу `expr` в ряд Лорана.

```
(%i12) residue(sin(a*z)/z^4,z,0);
```
(%o12) $-\frac{a^3}{6}$

6.5 Завдання до Розділу 6

Операції з матрицями

До поданих матриць: знайти обернену матрицю, детермінант, характеристичний поліном, власні значення і власні вектори, ранг, звести до трикутної.

6.1. $\begin{pmatrix}-2 & 3 & 5\\ 2 & 1 & 4\\ 1 & -3 & -1\end{pmatrix}$ **6.2.** $\begin{pmatrix}3 & -3 & 4\\ 5 & -1 & 2\\ -1 & 3 & -2\end{pmatrix}$ **6.3.** $\begin{pmatrix}-5 & -2 & -4\\ 2 & 2 & 3\\ 4 & -2 & -1\end{pmatrix}$ **6.4.** $\begin{pmatrix}4 & -5 & -4\\ -3 & -2 & 5\\ 2 & 3 & -3\end{pmatrix}$

6.5. $\begin{pmatrix}-3 & 5 & 1\\ -1 & -5 & -2\\ -4 & 3 & 1\end{pmatrix}$ **6.6.** $\begin{pmatrix}4 & -3 & -1\\ -3 & 2 & -2\\ -1 & 3 & 5\end{pmatrix}$ **6.7.** $\begin{pmatrix}-1 & 4 & 4\\ 3 & 1 & -5\\ 5 & -1 & -3\end{pmatrix}$ **6.8.** $\begin{pmatrix}1 & -4 & 4\\ -3 & -3 & 5\\ 2 & -5 & -2\end{pmatrix}$

6.9. $\begin{pmatrix}-3 & 2 & 5\\ 3 & -1 & -4\\ -1 & -2 & 1\end{pmatrix}$ **6.10.** $\begin{pmatrix}-2 & 4 & 2\\ 1 & -3 & 3\\ 5 & -4 & -1\end{pmatrix}$ **6.11.** $\begin{pmatrix}-5 & 0 & 1\\ 2 & 3 & -4\\ -1 & 4 & -1\end{pmatrix}$ **6.12.** $\begin{pmatrix}3 & -2 & 2\\ -4 & 1 & 0\\ -3 & 4 & 5\end{pmatrix}$

Системи лінійних рівнянь

Розв'язати подані системи: прямим методом, методом Крамера, методом Гаусса, матричним методом.

6.13. $\begin{cases} 2x + y - 3z = -12; \\ 3x - 2y - z = 3; \\ -x + 5y + 2z = -3. \end{cases}$

6.14. $\begin{cases} 6x - 5y + z = 7; \\ 5x + 3y + 2z = 0; \\ -2x + y - 3z = 11. \end{cases}$

6.15. $\begin{cases} -3x - 2y + 4z = -15; \\ 2x + 5y - 3z = 3; \\ 4x - y + 7z = 15. \end{cases}$

6.16. $\begin{cases} -3x + y - z = 8; \\ -4x + 2y + 3z = -3; \\ 2x + 3y - 2z = -1. \end{cases}$

6.17. $\begin{cases} x - 3y - 4z = -7; \\ 5x + 2y + 2z = -1; \\ 4x - y - 5z = -6. \end{cases}$

6.18. $\begin{cases} 2x + 3y + 3z = 15; \\ 3x - 6y - 6z = -23; \\ -9x - 3y + 6z = 8. \end{cases}$

6.19. $\begin{cases} x + y + z = 6; \\ -x + y - z = -2; \\ 2x + 3y + z = 11. \end{cases}$

6.20. $\begin{cases} x - y - z = -11; \\ x + y - z = 15; \\ 2x - y + z = -9. \end{cases}$

6.21. $\begin{cases} 2x + y - 3z = -7; \\ 3x - 2y + z = 11; \\ -2x - 3y - 2z = 3. \end{cases}$

6.22. $\begin{cases} x + 2y + z = 1; \\ 3x - y + 2z = 13; \\ 2x + 3y - z = -8. \end{cases}$

6.23. $\begin{cases} -7x + 2y + 4z = 13; \\ 3x - y + 2z = -2; \\ -x + 6y + 5z = 12. \end{cases}$

6.24. $\begin{cases} 2x - 3y + 5z = 10; \\ x - y + 5z = 4; \\ -3x - 4y + 3z = 2. \end{cases}$

Комплексні числа

Для заданих комплексних чисел: вивести дійсну та уявну частину, знайти спряжене число, знайти модуль та аргумент числа, перевести у експоненційну форму.

6.25. $z = 3 + 5i$

6.26. $z = 2 - 10i$

6.27. $z = 2 + 2i$

6.28. $z = 1 - 7i$

6.29. $z = 4 + i$

6.30. $z = 7 - 2i$

6.31. $z = 1 + 4i$

6.32. $z = 6 + 2i$

6.33. $z = 6 - 3i$

Розділ 7

Диференціальне числення

7.1 Границі

Границя виразу $f(x)$ при $x \to a$ обчислюється за допомогою команди
`limit(f(x),x,a,opts)`,
де опція `opts` може набувати два значення: `plus` та `minus`, вони позначають правосторонню та лівосторонню границю. Нагадаємо, що `inf` позначає $+\infty$, `minf` позначає $-\infty$.

```
(%i1)  limit((1+A/x)^x,x,inf);
```

$$(\%o1) \quad e^{A}$$

```
(%i3)  limit(1/x,x,0,plus); limit(1/x,x,0,minus);
```

$$(\%o2) \quad \infty$$

$$(\%o3) \quad -\infty$$

```
(%i4)  y(x):=(x^3-3*x-2)/(x^2-x-2)^2;
```

$$(\%o4) \quad \mathrm{y}\,(x) := \frac{x^3 - 3 \cdot x - 2}{\left(x^2 - x - 2\right)^2}$$

```
(%i5)  limit(y(x),x,-1);
```

$$(\%o5) \quad -\frac{1}{3}$$

```
(%i6)  limit((x/(2+x))^(3*x),x,inf);
```

$$(\%o6) \quad e^{-6}$$

При операціях з раціональними дробами часто буває випадок невизначеності $0/0$, тоді для виділення носіїв нуля доцільно спочатку використати факторизацію виразів, і наочність скорочення «нулів» стане явною. Нехай маємо границю саме такого типу:

► Знайти границю $\lim\limits_{x \to +\infty} \dfrac{x^2 - 4}{x^2 - 3x + 2}$.

```
(%i7)  f(x):=(x^2-4)/(x^2-3*x+2);
```

$$(\%o7) \quad \mathrm{f}\,(x) := \frac{x^2 - 4}{x^2 - 3 \cdot x + 2}$$

```
(%i8)  factor(num(f(x)));
```

$$(\%o8) \quad (x-2) \cdot (x+2)$$

```
(%i9)  factor(denom(f(x)));
```
(%o9) $(x-2)\cdot(x-1)$

```
(%i10) ev((%o8)/(%o9));
```
(%o10) $\dfrac{x+2}{x-1}$

```
(%i11) limit((%o10),x,2);
```
(%o11) 4

При дії з виразами, що містять ірраціональні операції, діють схожим чином, але при цьому використовують команду `radcan`.

► Знайти границю $\lim\limits_{x\to 1}\dfrac{\sqrt{x}-1}{x-1}$.

```
(%i12) radcan((sqrt(x)-1)/(x-1));
```
(%o12) $\dfrac{1}{\sqrt{x}+1}$

```
(%i13) limit((%o12),x,1);
```
(%o13) $\dfrac{1}{2}$

Звісно, пряме обчислення границь дало б ті ж самі результати.

При обчисленнях з використанням **Maxima** доцільно також використовувати для знаходження складних границь розклад чисельника та знаменника в ряд Тейлора (більш детально про степеневі ряди та ряд Тейлова див. нижче). Якщо використовувати меню **wxMaxima**, потрібно у вкладці «Аналіз — Знайти границю» встановити пункт «Ряд Тейлора». Для обчислень використовується функція `tlimit`, работа якої заснована на заміні функцій рядами Тейлора (де це можливо). За замовчуванням прапорець заміни поставлений у `false`, тому для використання `tlimit` його потрібно поставити в `true`:

```
(%i14) tlimswitch=true;
```
(%o16) $true = true$

```
(%i15) f(x):=(tan(x)-sin(x))/(x-sin(x));
```
(%o15) $\mathrm{f}\left(x\right) := \dfrac{\tan\left(x\right)-\sin\left(x\right)}{x-\sin\left(x\right)}$

```
(%i16) tlimit(f(x),x,0);
```
(%o16) 3

7.2 Диференціювання

Пакет **Maxima** надає потужні засоби для диференціювання функцій та обчислення диференціалів. Для обчислення похідної слід ввести команду

`diff(f(x),x,N)` — похідна порядку N функції $f(x)$ по одній змінній x;

`diff(f(x,y,...),xi,N)` — похідна порядку N функції багатьох змінних $f(x,y,...)$ по змінній x_i.

```
(%i1)  f(x):=sin(9*x^2);
```
(%o1) $\mathrm{f}\left(x\right) := \sin\left(9\cdot x^2\right)$

```
(%i2)  diff(f(x),x);
```

(%o2) $18 \cdot x \cdot \cos\left(9 \cdot x^2\right)$

```
(%i3)  diff(f(x),x,2);
```

(%o3) $18 \cdot \cos\left(9 \cdot x^2\right) - 324 \cdot x^2 \cdot \sin\left(9 \cdot x^2\right)$

```
(%i4)  g(x,y):=(x+y^2)/sqrt(x^2+y^4);
```

(%o4) $\mathrm{g}\left(x, y\right) := \dfrac{x + y^2}{\sqrt{x^2 + y^4}}$

```
(%i5)  diff(g(x,y),x);
```

(%o5) $\dfrac{1}{\sqrt{y^4 + x^2}} - \dfrac{x \cdot \left(y^2 + x\right)}{\left(y^4 + x^2\right)^{\frac{3}{2}}}$

Якщо подіяти командою `diff` на функцію, не вказуючи змінну диференціювання, **Maxima** видасть результат із використанням виразів типу `del(x)`, який аналогічний до диференціалу змінної dx. Тобто виведення дасть повний диференціал функції $f(x)$.

```
(%i11) diff(log(x));
```

(%o11) $\dfrac{\mathrm{del}\left(x\right)}{x}$

```
(%i12) diff(exp(x*y));
```

(%o12) $x \cdot e^{x \cdot y} \cdot \mathrm{del}\left(y\right) + y \cdot e^{x \cdot y} \cdot \mathrm{del}\left(x\right)$

Якщо вказати апостроф перед символом `diff`, то похідна не обчислюється і спрощення, яке зумовлене за замовчуванням, не здійснюється.

```
(%i13) f(x,z):=x^2*z+z^2*x;
```

(%o13) $\mathrm{f}\left(x, z\right) := x^2 \cdot z + z^2 \cdot x$

```
(%i14) diff(f(x,z),x,2)+diff(f(x,z),z,3)+diff(f(x,z),x)*x^2;
```

(%o14) $x^2 \cdot \left(z^2 + 2 \cdot x \cdot z\right) + 2 \cdot z$

```
(%i15) 'diff(f(x,z),x,2)+'diff(f(x,z),z,3)+'diff(f(x,z),x)*x^2;
```

(%o15) $\dfrac{d^3}{d\,z^3} \cdot \left(x \cdot z^2 + x^2 \cdot z\right) + \dfrac{d^2}{d\,x^2} \cdot \left(x \cdot z^2 + x^2 \cdot z\right) + x^2 \cdot \left(\dfrac{d}{d\,x} \cdot \left(x \cdot z^2 + x^2 \cdot z\right)\right)$

7.3 Рівняння дотичної та нормалі

З курсу аналізу відомо, що рівняння дотичної до функції $f(x)$ в точці x_0 має вигляд

$$g(x) = f'(x_0) \cdot (x - x_0) + f(x_0);$$

а рівняння нормалі

$$h(x) = -\frac{1}{f'(x_0)} \cdot (x - x_0) + f(x_0).$$

7.3.1 Пряма функціональна залежність

При явному заданні функції достатньо знайти значення функції в точці та значення похідної функції в точці. Тут будемо користуватись командами

`define(func2,oper(func1))` — означення нової функції `func2` через певні операції над функцією `func1`;

`at(expr,[x=a,y=b,...])` — оператор підстановки.

► Задано функцію $y = 1/x$ у точці $x_0 = 2$. Завдання: знайти рівняння дотичної та нормалі та накреслити їх графіки.

```
(%i1)  f(x):=1/x;
```

$$(\%o1) \quad \mathrm{f}(x) := \frac{1}{x}$$

```
(%i2)  x0:2;
```

$$(\%o2) \quad 2$$

```
(%i3)  y0:f(x0);
```

$$(\%o3) \quad \frac{1}{2}$$

```
(%i4)  define(df(x),diff(f(x),x));
```

$$(\%o4) \quad \mathrm{df}(x) := -\frac{1}{x^2}$$

```
(%i5)  k1:at(df(x),x=x0);
```

$$(\%o5) \quad -\frac{1}{4}$$

```
(%i6)  k2:at(-1/df(x),x=x0);
```

$$(\%o6) \quad 4$$

```
(%i7)  g:expand(k1*(x-x0)+f(x0));
```

$$(\%o7) \quad 1 - \frac{x}{4}$$

```
(%i8)  h:expand(k2*(x-x0)+f(x0));
```

$$(\%o8) \quad 4 \cdot x - \frac{15}{2}$$

```
(%i13) plot2d([f(x),g,h,[discrete,[[x0,y0]]]],[x,-4,4],[y,-4,4],
       [style,[lines,4,1],[lines,2,2],[lines,2,3],[points,3,4,1]],
       [nticks,100],[legend,false],[same_xy]);
```

Графік зображений на Рис. 7.1,(а).

7.3.2 Непряма залежність змінних

Перейдемо до складнішого випадку. Як знайти рівняння дотичної та нормалі до функції, якщо вона задана не функціональним виглядом, а у вигляді рівняння $f(x, y) = 0$? Тут

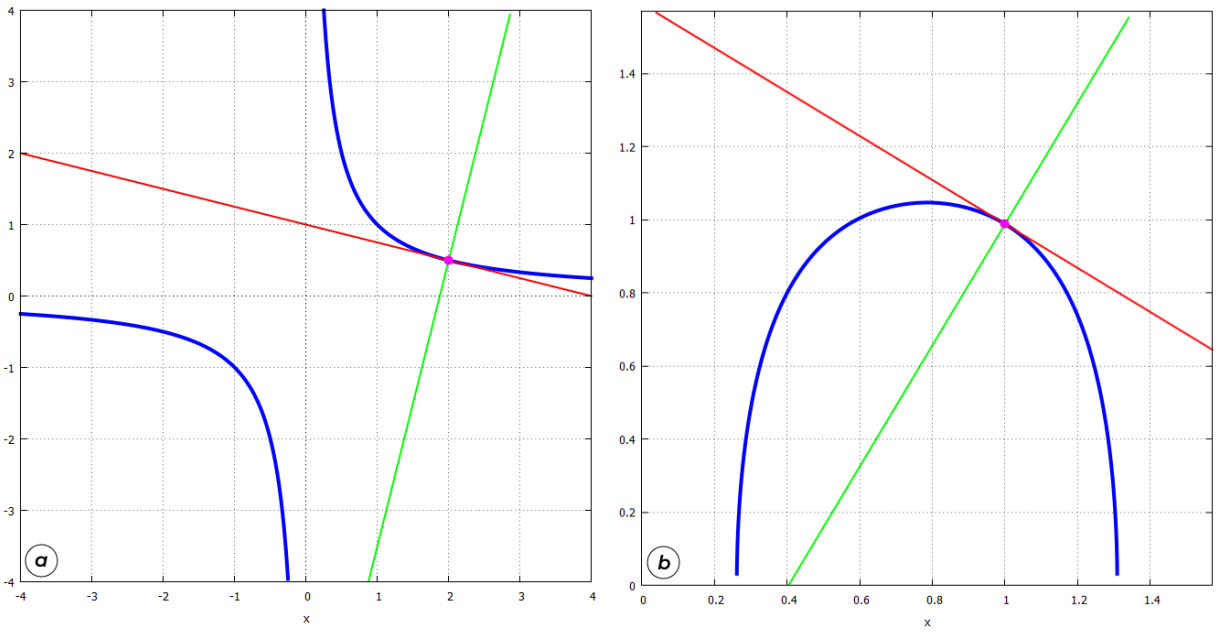


Рис. 7.1: Графіки функції, дотичної, нормалі: пряма (а) та непряма (б) залежності.

нам згодиться наступна рівність для похідної функції, що доводиться в математичному аналізі:

$$\frac{dy}{dx} = -\frac{\dfrac{\partial f(x,y)}{\partial x}}{\dfrac{\partial f(x,y)}{\partial y}}.$$

► Нехай маємо функціональну залежність $\sin 2x \cos y = 0,5$; $x_0 = 1$. Завдання: знайти рівняння дотичної та нормалі та накреслити їх графіки.

```
(%i1) f(x,y):=sin(2*x)*cos(y)-1/2;
```

$$(\%o1) \quad \mathrm{f}\,(x,y) := \sin\,(2\cdot x)\cdot\cos\,(y) - \frac{1}{2}$$

```
(%i2) define(df(x,y),-diff(f(x,y),x)/diff(f(x,y),y));
```

$$(\%o2) \quad \mathrm{df}\,(x,y) := \frac{2\cdot\cos\,(2\cdot x)\cdot\cos\,(y)}{\sin\,(2\cdot x)\cdot\sin\,(y)}$$

```
(%i3) x0:1;
```

$$(\%o3) \quad 1$$

```
(%i4) s1:solve(f(x,y)=0,y);
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

$$(\%o4) \quad [y = \mathrm{acos}\left(\frac{1}{2\cdot\sin\,(2\cdot x)}\right)]$$

```
(%i5) f:rhs(s1[1]);
```

$$(\%o5) \quad \mathrm{acos}\left(\frac{1}{2\cdot\sin\,(2\cdot x)}\right)$$

```
(%i6)  y0:at(f,x=x0),numer;
```

(%o6) 0.988581650693521

```
(%i7)  k1:at(df(x,y),[x=x0,y=y0]),numer;
```

(%o7) -0.6025870564746433

```
(%i8)  k2:at(-1/df(x,y),[x=x0,y=y0]),numer;
```

(%o8) 1.659511251121737

```
(%i9)  g:expand((k1*(x-x0)+y0));
```

(%o9) $1.591168707168164 - 0.6025870564746433 \cdot x$

```
(%i10) h:expand((k2*(x-x0)+y0));
```

(%o10) $1.659511251121737 \cdot x - 0.6709296004282159$

```
(%i11) plot2d([f,g,h,[discrete,[[x0,y0]]]],[x,0,%pi/2],[y,0,%pi/2],
       [style,[lines,4,1],[lines,2,2],[lines,2,3],[points,3,4,1]],
       [legend,false],[same_xy],[axes,solid]);
```

Графік зображений на Рис. 7.1,(б).

7.3.3 Параметричне представлення функції

При параметричному заданні функції маємо наступні співвідношення:

$$\begin{cases} x = \phi(t); \\ y = \psi(t); \end{cases} \qquad \frac{dy}{dx} = \frac{\psi'(t)}{\phi'(t)}.$$

► Нехай маємо функцію $x = \phi(t) = \sin t; \quad y = \psi(t) = \cos 2t; \quad x_0 = 0,5$. Завдання: знайти рівняння дотичної та нормалі та накреслити їх графіки.

```
(%i1)  x(t):=sin(t);
```

(%o1) $\mathrm{x}\,(t) := \sin(t)$

```
(%i2)  y(t):=cos(2*t);
```

(%o2) $\mathrm{y}\,(t) := \cos(2 \cdot t)$

```
(%i3)  define(df(t),diff(y(t),t)/diff(x(t),t));
```

(%o3) $\mathrm{df}\,(t) := -\dfrac{2 \cdot \sin(2 \cdot t)}{\cos(t)}$

```
(%i4)  x0:0.5;
```

(%o4) 0.5

```
(%i5)  s1:solve(x0=x(t),t),numer;
```

(%o5) $[t = 0.5235987755982994]$

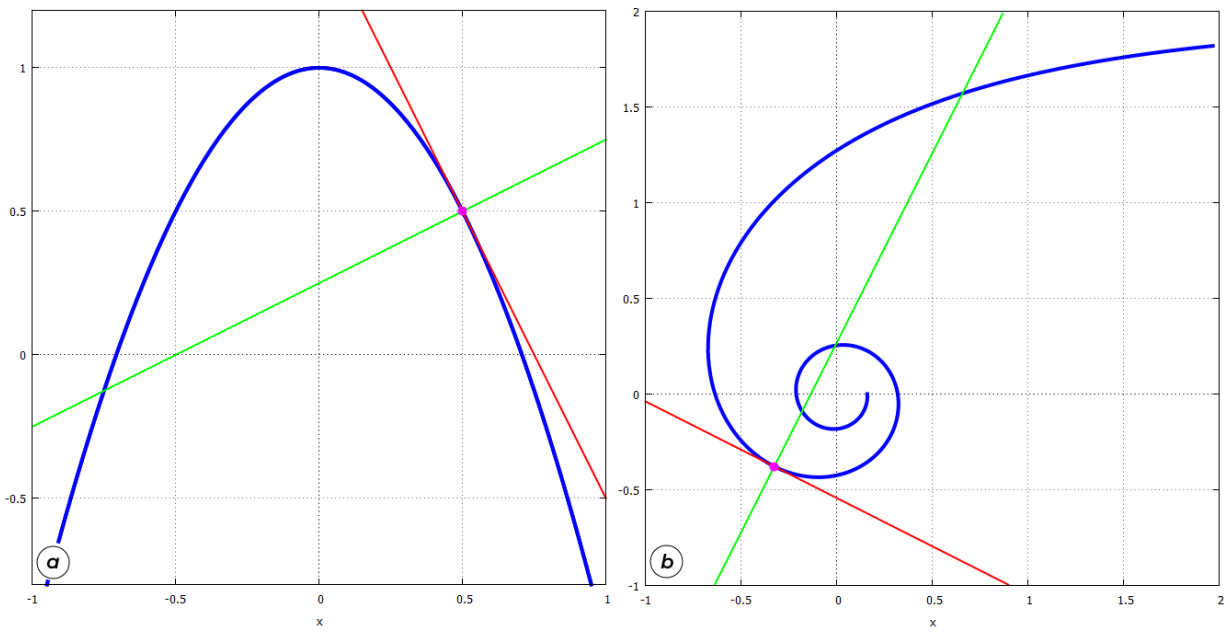


Рис. 7.2: Графіки функції, дотичної, нормалі: параметричне (а) та полярне (б) представлення.

```
(%i6)  t0:rhs(s1[1]);
```
(%o6) 0.5235987755982994

```
(%i7)  k1:at(df(t),t=t0);
```
(%o7) -2.000000000000001

```
(%i8)  k2:at(-1/df(t),t=t0);
```
(%o8) 0.4999999999999995

```
(%i9)  y0:y(t0);
```
(%o9) 0.4999999999999991

```
(%i10) g:expand(k1*(x-x0)+y0);
```
(%o10) $1.5 - 2.000000000000001 \cdot x$

```
(%i11) h:expand(k2*(x-x0)+y0);
```
(%o11) $0.4999999999999995 \cdot x + 0.2499999999999993$

```
(%i18) plot2d([[parametric,x(t),y(t),[t,-%pi,%pi]],g,h,
               [discrete,[[x0,y0]]]],[x,-1,1],[y,-0.8,1.2],
               [style,[lines,4,1],[lines,2,2],[lines,2,3],[points,3,4,1]],
               [legend,false],[nticks,100],[same_xy]);
```

Графік зображений на Рис. 7.2,(а).

7.3.4 Полярне представлення функції

При полярному заданні функції маємо наступні співвідношення:

$$\begin{cases} x(\theta) = \rho(\theta)\cos\theta; \\ y(\theta) = \rho(\theta)\sin\theta; \end{cases} \qquad \frac{dy}{dx} = \frac{y'(\theta)}{x'(\theta)}.$$

► Нехай маємо функціональну залежність $\rho(\theta) = \dfrac{2}{\theta}$, $\theta_0 = 4$ рад. Завдання: знайти рівняння дотичної та нормалі та накреслити їх графіки.

```
(%i1)  x(t):=(2/t)*cos(t);
```

$$(\%o1) \quad \mathrm{x}\,(t) := \frac{2\cdot\cos\,(t)}{t}$$

```
(%i2)  y(t):=(2/t)*sin(t);
```

$$(\%o2) \quad \mathrm{y}\,(t) := \frac{2\cdot\sin\,(t)}{t}$$

```
(%i3)  t0:4;
```

$$(\%o3) \quad 4$$

```
(%i4)  define(df(t),diff(y(t),t)/diff(x(t),t));
```

$$(\%o4) \quad \mathrm{df}\,(t) := \frac{\frac{2\cdot\cos(t)}{t} - \frac{2\cdot\sin(t)}{t^2}}{-\frac{2\cdot\sin(t)}{t} - \frac{2\cdot\cos(t)}{t^2}}$$

```
(%i5)  x0:at(x(t),t=t0),numer;
```

$$(\%o5) \quad -0.3268218104318059$$

```
(%i6)  y0:at(y(t),t=t0),numer;
```

$$(\%o6) \quad -0.3784012476539641$$

```
(%i7)  k1:at(df(t),t=t0),numer;
```

$$(\%o7) \quad -0.5047122730143312$$

```
(%i8)  k2:at(-1/df(t),t=t0),numer;
```

$$(\%o8) \quad 1.981326893494435$$

```
(%i9)  g:expand(k1*(x-x0)+y0);
```

$$(\%o9) \quad -0.5047122730143312\cdot x - 0.5433522264676598$$

```
(%i10) h:expand(k2*(x-x0)+y0);
```

$$(\%o10) \quad 1.981326893494435\cdot x + 0.2691395947351133$$

```
(%i11) plot2d([[parametric,x(t),y(t),[t,0.1,4*%pi]],g,h,
       [discrete,[[x0,y0]]]],[x,-1,2],[y,-1,2],
       [style,[lines,4,1],[lines,2,2],[lines,2,3],[points,3,4,1]],
       [legend,false],[nticks,100],[same_xy]);
```

Графік зображений на Рис. 7.2,(б).

7.4 Дослідження функції та побудова її графіка

Схема дослідження функції може включати в себе досить багато пунктів, але ми зупинимось на основних.

1. Точки перетину з осями:
   $f(x) = 0$ — перетин з віссю x;
   $f(0) = y_0$ — перетин з віссю y.
2. Асимптоти функції:
   $\lim_{x\to\infty} f(x) = a$; $\quad y = a$ — горизонтальна асимптота (шукаються окремо $x \to \pm\infty$);
   $\lim_{x\to b} f(x) = \infty$; $\quad x = b$ — вертикальна асимптота (шукаються окремо $x \to b \pm 0$).
   $k = \lim_{x\to\infty} \frac{f(x)}{x}$, $\quad b = \lim_{x\to\infty}(f(x) - kx)$; $\quad g(x) = kx + b$ — похила асимптота.
3. Критичні точки та точки, підозрілі на екстремум:
   $f'(x) = 0$ або не існує.
4. Області зростання та спадання:
   $f'(x) > 0$ — функція зростає;
   $f'(x) < 0$ — функція спадає;
5. Точки екстремумів:
   $f'(x) = 0$ та $f''(x) > 0$ — у цій точці функція досягає мінімума;
   $f'(x) = 0$ та $f''(x) < 0$ — у цій точці функція досягає максимума.
6. Точки перегину:
   $f''(x) = 0$.
7. Проміжки вгнутості та опуклості:
   $f''(x) > 0$ — функція вгнута;
   $f''(x) < 0$ — функція опукла.
8. Графік функції.

▶ Нехай задана функція $f(x) = \dfrac{x+1}{x^2+1}$. Завдання: дослідити цю функцію та побудувати її графік.

```
(%i1)  f(x):=(x+1)/(x^2+1);
```

(%o1) $\mathrm{f}(x) := \dfrac{x+1}{x^2+1}$

1. Точки перетину з осями:

```
(%i2)  solve(f(x)=0,x);
```

(%o2) $[x = -1]$

```
(%i3)  y0:at(f(x),x=0);
```

(%o3) 1

З віссю X: $(-1;0)$, з віссю Y: $(0;1)$.

2. Асимптоти:

```
(%i4)  limit(f(x),x,inf);
```

$$(\%o4) \quad 0$$

```
(%i5)  limit(f(x),x,minf);
```

$$(\%o5) \quad 0$$

```
(%i6)  realroots(denom(f(x)));
```

$$(\%o6) \quad []$$

Горизонтальна асимптота: $y = 0$. Вертикальних асимптот немає.

3. Критичні точки:

```
(%i7)  define(df(x),diff(f(x),x));
```

$$(\%o7) \quad \mathrm{df}\,(x) := \frac{1}{x^2+1} - \frac{2\cdot x\cdot (x+1)}{\left(x^2+1\right)^2}$$

```
(%i8)  s1:solve(df(x)=0,x);
```

$$(\%o8) \quad [x = -\sqrt{2}-1, x = \sqrt{2}-1]$$

```
(%i9)  x1:rhs(s1[1]);
```

$$(\%o9) \quad -\sqrt{2}-1$$

```
(%i10) x2:rhs(s1[2]);
```

$$(\%o10) \quad \sqrt{2}-1$$

```
(%i11) f(x1),numer;
```

$$(\%o11) \quad -0.2071067811865475$$

```
(%i12) f(x2),numer;
```

$$(\%o12) \quad 1.207106781186547$$

Дві критичні точки: $\left(-\sqrt{2}-1;-0.2071067811865475\right)$; $\left(\sqrt{2}-1;1.207106781186547\right)$.

4. Області зростання та спадання:

```
(%i13) load(solve_rat_ineq);
(%i14) solve_rat_ineq(df(x)>0);
```

$$(\%o14) \quad [[x > -\sqrt{2}-1, x < \sqrt{2}-1]]$$

```
(%i15) solve_rat_ineq(df(x)<0);
```

$$(\%o15) \quad [[x < -\sqrt{2}-1], [x > \sqrt{2}-1]]$$

Функція зростає при $x \in \left(-\sqrt{2}-1;\sqrt{2}-1\right)$, спадає при $x \in \left(-\infty;-\sqrt{2}-1\right) \cup \left(\sqrt{2}-1;+\infty\right)$.

5. Точки екстремумів:

```
(%i16)  define(ddf(x),diff(f(x),x,2));
```

$$(\%o16)\quad \mathrm{ddf}\,(x) := -\frac{2\cdot(x+1)}{\left(x^2+1\right)^2} - \frac{4\cdot x}{\left(x^2+1\right)^2} + \frac{8\cdot x^2\cdot(x+1)}{\left(x^2+1\right)^3}$$

```
(%i17) ddf(x1),numer;
```

$$(\%o17)\quad 0.06066017177982131$$

```
(%i18) ddf(x2),numer;
```

$$(\%o18)\quad -2.060660171779821$$

Друга похідна у першій критичній точці x_1 додатня, тому ця точка — мінімум; друга похідна у другій критичній точці x_2 від'ємна, тому ця точка — максимум.

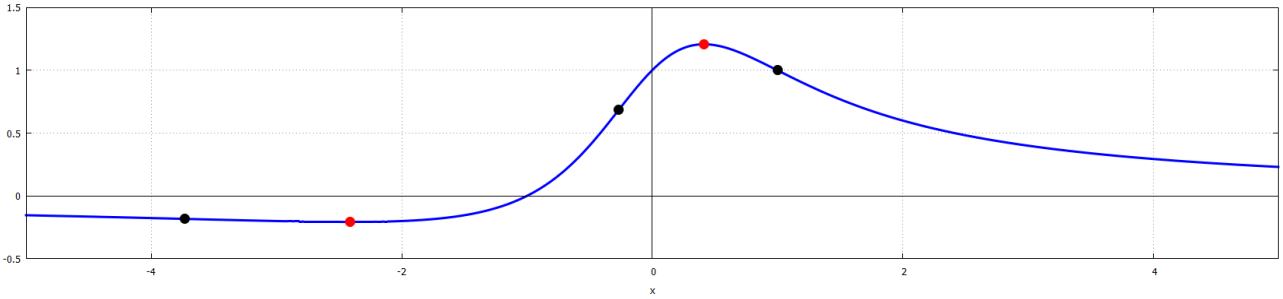


Рис. 7.3: Графік функції, точками позначені екстремуми (●) та точки перегину (●).

6. Точки перегину:

```
(%i19) s2:solve(ddf(x)=0,x);
```

$$(\%o19)\quad [x = -\sqrt{3} - 2, x = \sqrt{3} - 2, x = 1]$$

```
(%i20) xinf1:rhs(s2[1]);
```

$$(\%o20)\quad -\sqrt{3} - 2$$

```
(%i21) xinf2:rhs(s2[2]);
```

$$(\%o21)\quad \sqrt{3} - 2$$

```
(%i22) xinf3:rhs(s2[3]);
```

$$(\%o22)\quad 1$$

```
(%i23) f(xinf1),numer;
```

$$(\%o23)\quad -0.1830127018922193$$

```
(%i24) f(xinf2),numer;
```

$$(\%o24)\quad 0.6830127018922193$$

```
(%i25) f(xinf3),numer;
```

$$(\%o25)\quad 1$$

Є три точки перегину: $(-\sqrt{3}-2; -0.1830127018922193)$, $(\sqrt{3}-2; 0.6830127018922193)$, $(1; 1)$.

7. Проміжки вгнутості та опуклості:

```
(%i26) solve_rat_ineq(ddf(x)>0);
```

(%o26) $[[x > -\sqrt{3}-2, x < \sqrt{3}-2], [x > 1]]$

```
(%i27) solve_rat_ineq(ddf(x)<0);
```

(%o27) $[[x < -\sqrt{3}-2], [x > \sqrt{3}-2, x < 1]]$

Функція вгнута при $x \in \left(-\sqrt{3}-2; \sqrt{3}-2\right) \cup (1; +\infty)$, опукла при $x \in (-\infty; -\sqrt{3}-2) \cup (\sqrt{3}-2; 1)$.

8. Графік (Рис. 7.3):

```
(%i24) plot2d([f(x),[discrete,[[x1,f(x1)],[x2,f(x2)]]],
       [discrete,[[xinf1,f(xinf1)],[xinf2,f(xinf2)],[xinf3,f(xinf3)]]]],
       [x,-5,5],[y,-0.5,1.5],[style,[lines,3,1],[points,3,2,1],
       [points,3,5,1]],[legend,false],[axes,solid],[same_xy]);
```

► Дослідити функцію та побудувати її графік:

$$f(x) = \frac{x^3}{x^2 + x - 2}.$$

```
(%i1)  f(x):=(x^3)/(x^2+x-2);
```

(%o1) $\mathrm{f}(x) := \dfrac{x^3}{x^2 + x - 2}$

1. Точки перетину з осями:

```
(%i2)  sol1:solve(f(x)=0,x);
```

(%o2) $[x = 0]$

```
(%i3)  y0:at(f(x),x=0);
```

(%o3) 0

З віссю X та віссю Y: $(0; 0)$.

2. Асимптоти:

```
(%i4)  limit(f(x),x,inf);
```

(%o4) ∞

```
(%i5)  limit(f(x),x,minf);
```

(%o5) $-\infty$

Горизонтальних асимптот немає.

```
(%i6)  realroots(denom(f(x)));
```

(%o6) $[x = -2, x = 1]$

Дві вертикальні асимптоти: $x = -2$ та $x = 1$.

```
(%i7)  k:limit(f(x)/x,x,inf);
```

(%o7) 1

```
(%i8)  b:limit(f(x)-k*x,x,inf);
```

$$(\%o8) \quad -1$$

```
(%i9)  g(x):=k*x+b;
```

$$(\%o9) \quad \mathrm{g}\,(x) := k \cdot x + b$$

Похила асимптота: $g(x) = x - 1$.

3. Критичні точки:

```
(%i10) define(df(x),diff(f(x),x));
```

$$(\%o10) \quad \mathrm{df}\,(x) := \frac{3 \cdot x^2}{x^2 + x - 2} - \frac{x^3 \cdot (2 \cdot x + 1)}{(x^2 + x - 2)^2}$$

```
(%i11) sol2:solve(df(x)=0,x);
```

$$(\%o11) \quad [x = -\sqrt{7} - 1, x = \sqrt{7} - 1, x = 0]$$

```
(%i12) x1:rhs(sol2[1]);
```

$$(\%o12) \quad -\sqrt{7} - 1$$

```
(%i13) x2:rhs(sol2[2]);
```

$$(\%o13) \quad \sqrt{7} - 1$$

```
(%i14) x3:rhs(sol2[3]);
```

$$(\%o14) \quad 0$$

```
(%i15) [dy1,dy2,dy3]:map(f,[x1,x2,x3]),numer;
```

$$(\%o15) \quad [-6.337835372767143, 1.893390928322696, 0]$$

Три критичні точки: $(-\sqrt{7} - 1; -6.3378)$, $(\sqrt{7} - 1; 1.8933)$, $(0; 0)$.

4. Області зростання та спадання:

```
(%i16) load(solve_rat_ineq);
(%i17) solve_rat_ineq(df(x)<0);
```

$$(\%o17) \quad [[x > -\sqrt{7} - 1, x < -2], [x > -2, x < 0], [x > 0, x < 1], [x > 1, x < \sqrt{7} - 1]]$$

```
(%i18) solve_rat_ineq(df(x)>0);
```

$$(\%o18) \quad [[x < -\sqrt{7} - 1], [x > \sqrt{7} - 1]]$$

Функція зростає при $x \in (-\infty; -\sqrt{7} - 1) \cup (\sqrt{7} - 1; +\infty)$; спадає при $x \in (-\sqrt{7} - 1; -2) \cup (-2; 0) \cup (0; 1) \cup (1; \sqrt{7} - 1)$.

5. Точки екстремумів:

```
(%i19) define(ddf(x),diff(f(x),x,2));
```

$$(\%o19) \quad \mathrm{ddf}\,(x) := \frac{6 \cdot x}{x^2 + x - 2} - \frac{2 \cdot x^3}{(x^2 + x - 2)^2} - \frac{6 \cdot x^2 \cdot (2 \cdot x + 1)}{(x^2 + x - 2)^2} + \frac{2 \cdot x^3 \cdot (2 \cdot x + 1)^2}{(x^2 + x - 2)^3}$$

```
(%i20) [ddy1,ddy2,ddy3]:map(ddf,[x1,x2,x3]),numer;
```

$$(\%o20) \quad [-1.203130568416622, 2.585846617799338, 0]$$

Друга похідна в точці x_1 від'ємна, тому ця точка — максимум; друга похідна в точці x_2 додатня, тому ця точка — мінімум; третя точка x_3 не є ні мінімумом, ні максимумом.

6. Точки перегину:

```
(%i21) sol3:solve(ddf(x)=0,x);
```

(%o21) $[x = 1 - \sqrt{3}\cdot i, x = \sqrt{3}\cdot i + 1, x = 0]$

```
(%i22) xinf1:rhs(sol3[3]);
```

(%o22) 0

Є лише одна дійсна точка перегину: $(0; 0)$.

7. Проміжки вгнутості та опуклості:

```
(%i23) solve_rat_ineq(ddf(x)<0);
```

(%o23) $[[x < -2], [x > 0, x < 1]]$

```
(%i24) solve_rat_ineq(ddf(x)>0);
```

(%o24) $[[x > -2, x < 0], [x > 1]]$

Функція вгнута при $x \in (-2; 0) \cup (1; +\infty)$; опукла при $x \in (-\infty; -2) \cup (0; 1)$.

8. Графік зобразимо разом з асимптотами (Рис.7.4).

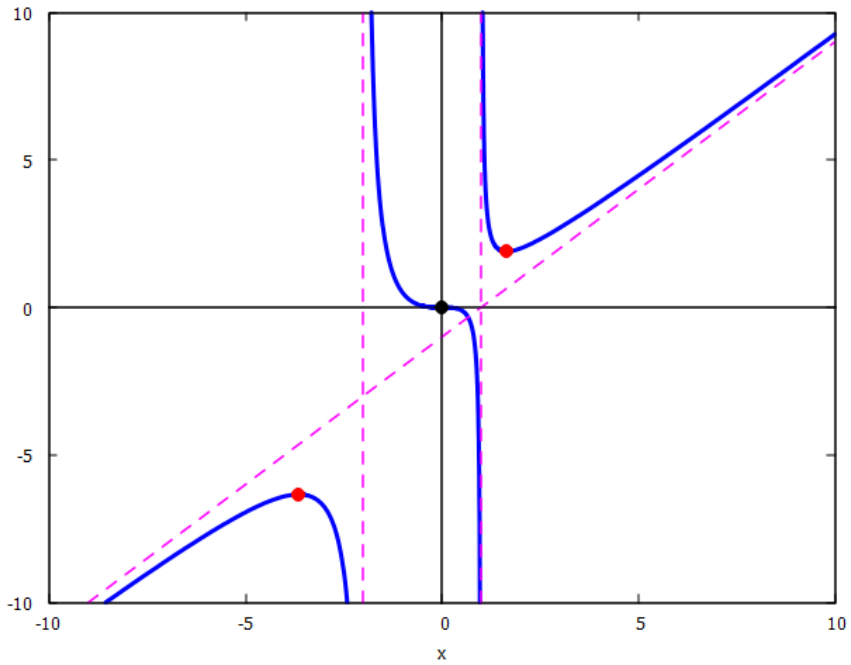


Рис. 7.4: Графік функції з асимптотами, точками познач. екстремуми (•) та перегини (•).

```
(%i25) plot2d([f(x),g(x),[discrete,[[-2,-10],[-2,10]]],
       [discrete,[[1,-10],[1,10]]],[discrete,[[x1,f(x1)],[x2,f(x2)]]],
       [discrete,[[xinf1,f(xinf1)]]]],
       [x,-10,10],[y,-10,10],[style,[lines,3,1],[lines,2,4],
       [lines,2,4], [lines,2,4] ,[points,3,2,1],
       [points,3,5,1]],[legend,false],[axes,solid]);
```

7.5 Екстремуми функцій двох змінних

Нехай маємо функцію двох змінних $f(x,y)$. Критичні точки такої функції шукаються з системи рівнянь $\{f'_x(x,y)=0; f'_y(x,y)=0\}$. Тип критичної точки визначається другими похідними функції за такими правилами:

— якщо $f''_{xx}\cdot f''_{yy}-(f''_{xy})^2>0$ та $f''_{xx}>0$ — мінімум;
— якщо $f''_{xx}\cdot f''_{yy}-(f''_{xy})^2>0$ та $f''_{xx}<0$ — максимум;
— якщо $f''_{xx}\cdot f''_{yy}-(f''_{xy})^2<0$ — екстремума немає.

Тут відповідні похідні другого порядку обчислені в отриманих парах (x_0,y_0) (їх може бути кілька).

► Дослідити функцію на екстремум: $z=x^3-\frac{9}{2}x^2+6x+y^2-4y-12$.

Задаємо функцію.

```
(%i1)  f(x,y):=x^3-9/2*x^2+6*x+y^2-4*y-12;
```

$$(\%o1)\quad \mathrm{f}(x,y):=x^3-\frac{9\cdot x^2}{2}+6\cdot x+y^2-4\cdot y-12$$

Знаходимо перші похідні та розв'язуємо систему рівнянь.

```
(%i2)  dfx:diff(f(x,y),x);
```

$$(\%o2)\quad 3\cdot x^2-9\cdot x+6$$

```
(%i3)  dfy:diff(f(x,y),y);
```

$$(\%o3)\quad 2\cdot y-4$$

```
(%i4)  s1:solve([dfx,dfy],[x,y]);
```

$$(\%o4)\quad [[x=1,y=2],[x=2,y=2]]$$

Маємо дві критичні точки: $(1,2)$ та $(2,2)$. Тепер шукаємо другі похідні.

```
(%i5)  dfxx:diff(dfx,x);
```

$$(\%o5)\quad 6\cdot x-9$$

```
(%i6)  dfxy:diff(dfx,y);
```

$$(\%o6)\quad 0$$

```
(%i7)  dfyy:diff(dfy,y);
```

$$(\%o7)\quad 2$$

Шукаємо значення других похідних у двох точках та дві комбінації $f''_{xx}\cdot f''_{yy}-(f''_{xy})^2$ у цих точках.

```
(%i8)  [A,B,C]:ev([dfxx,dfxy,dfyy],x=1,y=2);
```

$$(\%o8)\quad [-3,0,2]$$

```
(%i9)  A*C-B^2;
```

$$(\%o9)\quad -6$$

```
(%i10) [A,B,C]:ev([dfxx,dfxy,dfyy],x=2,y=2);
```

$$(\%o10)\quad [3,0,2]$$

```
(%i11) A*C-B^2;
```

(%o11) 6

Бачимо, що для першої точки $(1,2)$ комбінація рівна $-6 < 0$, тому вона не є ні максимумом, ні мінімумом. У другій точці $(2,2)$ комбінація рівна $6 > 0$ і $f''_{xx} = 3 > 0$, отже це точка мінімуму. Зобразимо графік поверхні (Рис. 7.5).

```
(%i12) plot3d(f(x,y),[x,0,5],[y,0,5],[legend,"f(x,y)"]);
```

(%o12) $[C : /Users/orreg/maxout.gnuplot]$

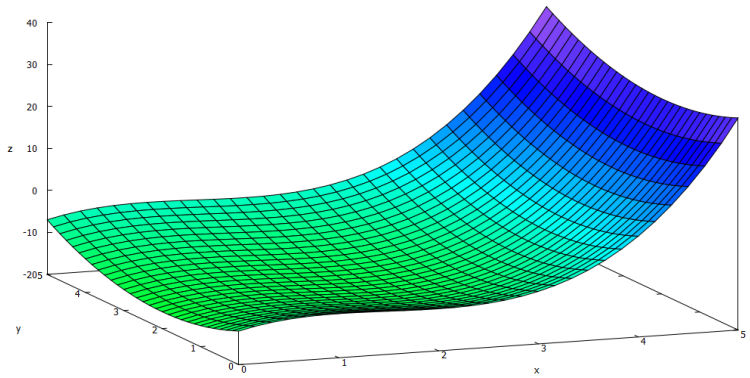


Рис. 7.5: Графік функції двох змінних

7.6 Розклад функції в ряд Тейлора

Для спеціалістів інженерного профілю дуже важливим є одночасне знаходження розв'язку в замкнутій аналітичній формі та отримання числових значень результату. Представлення функції у вигляді степеневого ряду дозволяє звести вивчення властивостей складної функції до вивчення цих властивостей у відповідного апроксимуючого поліномного розкладу. Заміна функцій на їх степеневі розклади допомагає вивченню границь, аналізу збіжності та розбіжності рядів та інтегралів, наближеному обчисленню інтегралів та розв'язанню диференціальних рівнянь.

Найважливішим таким інструментом є розклад функцій у вигляді рядів Тейлора: це степеневий ряд виду

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} \cdot (x - x_0)^k,$$

де числова функція f припускається визначеною в деякому околі точки x_0 та має в цій точці похідні всіх порядків, (k) позначає похідну k-го порядку. Якщо точка $x_0 = 0$, розклад називається рядом Маклорена.

Многочленами Тейлора для функції $f(x)$ порядку n називаються частинні суми ряду:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

В **Maxima** існує спеціальна команда, що дозволяє обчислити ряди та многочлени Тейлора для функції $f(x)$:

`taylor(f(x),x,a,n)`,

де x — змінна, по якій проводиться розклад, a — точка, в околі якої розкладається функція, n — порядок розкладу, він має бути представлений цілим додатнім числом.

Знайдемо роклади в ряд Маклорена важливих функцій:

```
(%i1)  taylor(%e^x,x,0,7);
```

$$(\%o1)/T/\ 1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}+\frac{x^6}{720}+\frac{x^7}{5040}+\ldots$$

```
(%i2)  taylor(sin(x),x,0,7);
```

$$(\%o2)/T/\ x-\frac{x^3}{6}+\frac{x^5}{120}-\frac{x^7}{5040}+\ldots$$

```
(%i3)  taylor(cos(x),x,0,7);
```

$$(\%o3)/T/\ 1-\frac{x^2}{2}+\frac{x^4}{24}-\frac{x^6}{720}+\ldots$$

```
(%i4)  taylor(log(1+x),x,0,7);
```

$$(\%o4)/T/\ x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\frac{x^5}{5}-\frac{x^6}{6}+\frac{x^7}{7}+\ldots$$

Оцінимо точність розкладу в ряд експоненти порівняно з її значенням в точці 1.

```
(%i5)  t:taylor(%e^x,x,0,9);
```

$$(\%o5)/T/\ 1+x+\frac{x^2}{2}+\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}+\frac{x^6}{720}+\frac{x^7}{5040}+\frac{x^8}{40320}+\frac{x^9}{362880}+\ldots$$

```
(%i6)  e9:ev(t,x=1);
```

$$(\%o6)/R/\ \frac{98641}{36288}$$

```
(%i7)  e9,numer;
```

$$(\%o7)\quad 2.718281525573192$$

```
(%i8)  bfloat(%e-e9);
```

$$(\%o8)\quad 3.028858530396942b-7$$

Бачимо, що дев'ять перших доданків забезпечують точність, більшу за 10^{-6}. Взагалі при використанні розкладу Тейлора для більш високих степенів точність наближення зростає, однак варто зауважити, що цей степінь не можна підвищувати необмежено (у зв'язку з накопиченням похибки у обчисленнях, що швидко зводить нанівець всі побудови розкладу, адже накопичена помилка дуже швидко стане набагато більшою за значення чергового члена розкладу в точці).

Разом з командою `taylor` для розкладу функцій та виразів в ряд використовується команда

`powerseries(f(x),x,a)` — будує розклад за змінною x в околі точки a, результатом виконання є побудова ряду Тейлора в загальному вигляді.

```
(%i9)  powerseries(sin(x),x,0);
```

$$(\%o9)\quad \sum_{i1=0}^{\infty}\frac{(-1)^{i1}\cdot x^{2\cdot i1+1}}{(2\cdot i1+1)!}$$

```
(%i10) powerseries(x/(1+x),x,0);
```

$$(\%o10)\quad x\cdot\sum_{i2=0}^{\infty}(-1)^{i2}\cdot x^{i2}$$

```
(%i11) powerseries(cos(x^3),x,0);
```

$$(\%o11) \quad \sum_{i3=0}^{\infty} \frac{(-1)^{i3} \cdot x^{6 \cdot i3}}{(2 \cdot i3)!}$$

► Цікавим є також відтворити класичне зображення із підручників математичного аналізу — порівняння точного графіку функції із розкладами її в ряд різних порядків. Зробимо це, для прикладу, із функцією $y = \cos x$.

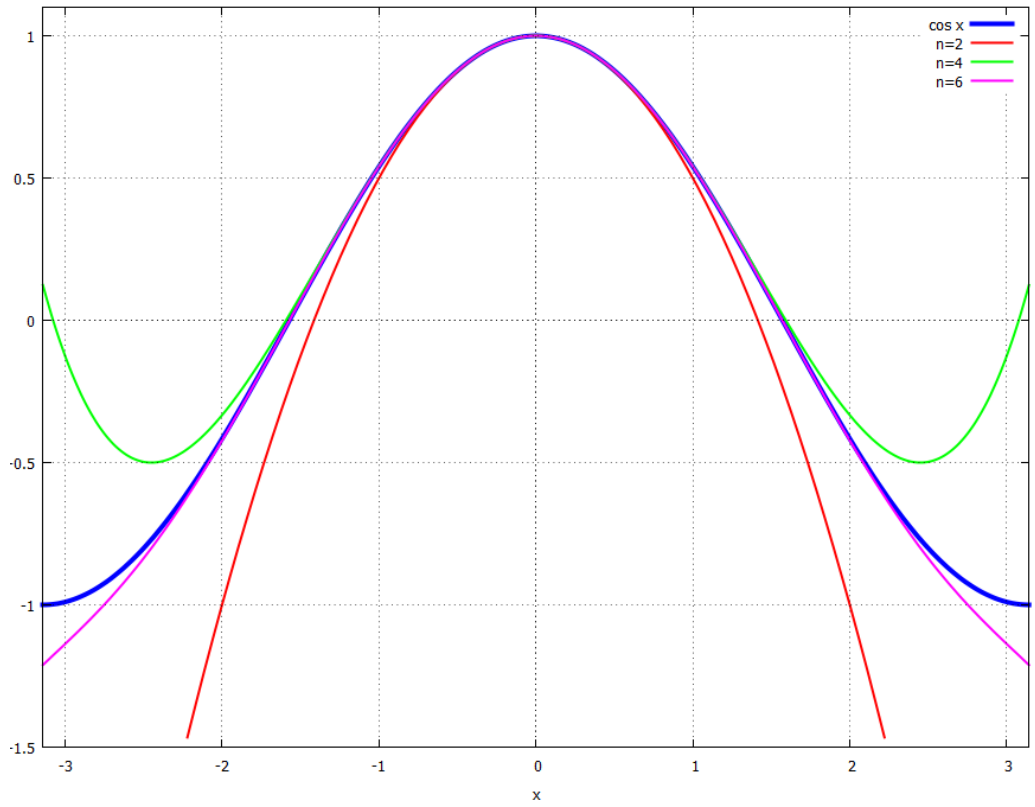


Рис. 7.6: Порівняння $y = \cos x$ з поліномами розкладу в точці $x = 0$.

```
(%i12) cos2:taylor(cos(x),x,0,2);
```

$$(\%o12)/T/1 - \frac{x^2}{2} + \ldots$$

```
(%i13) cos4:taylor(cos(x),x,0,4);
```

$$(\%o13)/T/1 - \frac{x^2}{2} + \frac{x^4}{24} + \ldots$$

```
(%i14) cos6:taylor(cos(x),x,0,6);
```

$$(\%o14)/T/1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \ldots$$

```
(%i15) plot2d([cos(x),cos2,cos4,cos6],[x,-%pi,%pi],[y,-1.5,1.1],
              [style,[lines,5,1],[lines,2,2],[lines,2,3],[lines,2,4]],
              [legend,"cos x","n=2","n=4","n=6"]);
```

Графік на Рис. 7.6.

У випадку розкладу функцій кількох змінних ряд Тейлора знаходиться за формулою

$$f(x,y)=\sum_{k=0}^{\infty}\left[(x-a)\frac{\partial}{\partial x}+(y-b)\frac{\partial}{\partial y}\right]^k f(a,b).$$

Синтаксис виклику **Maxima** наступний:

`taylor(f(x,y),[x,y],[a,b],n)` — розклад $f(x,y)$ в ряд Тейлора навколо точки (a,b) n-го порядку.

► Розкласти функцію $z=e^{-x^2-y^2}$ в ряд навколо точки $(1,1)$ порядку 2.

```
(%i1)  f(x,y):=exp(-x^2-y^2);
```

$$(\%o1)\quad \mathrm{f}\,(x,y):=\exp\left(-x^2-y^2\right)$$

```
(%i2)  taylor(f(x,y),[x,y],[1,1],2);
```

$$(\%o2)/\mathrm{T}/\ \frac{1}{e^2}-\frac{2\cdot(x-1)+2\cdot(y-1)}{e^2}+\frac{(x-1)^2+4\cdot(y-1)\cdot(x-1)+(y-1)^2}{e^2}$$

► Розкласти функцію $z=\sin(x+y)$ навколо точки $(0,0)$ порядку 5.

```
(%i3)  taylor(sin(x+y),[x,y],[0,0],5);
```

$$(\%o3)/\mathrm{T}/\ y+x-\frac{x^3+3\cdot y\cdot x^2+3\cdot y^2\cdot x+y^3}{6}+\frac{x^5+5\cdot y\cdot x^4+10\cdot y^2\cdot x^3+10\cdot y^3\cdot x^2+5\cdot y^4\cdot x+y^5}{120}$$

7.7 Диференціальні операції векторного аналізу

Для прямих обчислень, пов'язаних з операціями векторного аналізу, необхідно завантажити пакет `vect`. Крім того, у застосуванні операторів `div, curl, grad, laplacian` до деякої функції необхідно використовувати команду `express`. Нагадаємо, що оператори `div, curl` застосовуються до векторів (`curl` рівний оператору `rot` у німецькій традиції), а `grad, laplacian` — до скалярів. Процедура обчислень, наприклад градієнту функції, наступна: спочатку діємо оператором `grad` на функцію, потім командою `express` виражаємо отриманий вираз через прямі диференціальні операції, і нарешті командою `ev(...,diff)` знаходимо результат обчислень у вигляді компонент вектора. Дві останні операції можна провести одночасно.

```
(%i1)  load(vect);
(%i2)  F(x,y,z):=x^2*y+exp(y*z);
```

$$(\%o2)\quad \mathrm{F}\,(x,y,z):=x^2\cdot y+\exp\left(y\cdot z\right)$$

```
(%i3)  gradF:grad(F(x,y,z));
```

$$(\%o3)\quad \mathrm{grad}\left(e^{y\cdot z}+x^2\cdot y\right)$$

```
(%i4)  express(gradF);
```

$$(\%o4)\quad \left[\frac{d}{d\,x}\cdot\left(e^{y\cdot z}+x^2\cdot y\right),\frac{d}{d\,y}\cdot\left(e^{y\cdot z}+x^2\cdot y\right),\frac{d}{d\,z}\cdot\left(e^{y\cdot z}+x^2\cdot y\right)\right]$$

```
(%i5)  ev(%o4,diff);
```

$$(\%o5)\quad \left[2\cdot x\cdot y,z\cdot e^{y\cdot z}+x^2,y\cdot e^{y\cdot z}\right]$$

Ротор векторної функції:

```
(%i6)  G(x,y,z):=[x^2,2*y^2*z,%e^(x*z)];
```

$$(\%o6) \quad \mathrm{G}\,(x,y,z) := [x^2, 2\cdot y^2\cdot z, e^{x\cdot z}]$$

```
(%i7)  curlG:curl(G(x,y,z));
```

$$(\%o7) \quad \mathrm{curl}\left([x^2, 2\cdot y^2\cdot z, e^{x\cdot z}]\right)$$

```
(%i8)  ev(express(curlG),diff);
```

$$(\%o8) \quad [-2\cdot y^2, -z\cdot e^{x\cdot z}, 0]$$

Оператор Лапласа:

```
(%i9)  laplacian(x^2+2*y^2+3*z^2);
```

$$(\%o9) \quad \mathrm{laplacian}\left(3\cdot z^2 + 2\cdot y^2 + x^2\right)$$

```
(%i10) ev(express(%o9),diff);
```

$$(\%o10) \quad 12$$

Перевіримо відомі формули: rot grad $F(\vec{r}) = 0$, div rot $\vec{G}(\vec{r}) = 0$.

```
(%i16) div(curlG);
```

$$(\%o16) \quad 0$$

```
(%i17) curl(gradF);
```

$$(\%o17) \quad 0$$

7.8 Підсумовування рядів

Maxima реалізує числові ряди як один з видів списків, їх ми розглядали в Розділі 3. Нагадаємо синтаксис команди для пошуку суми ряду:

`sum(expr,k,k1,kN)` — Підсумовує ряд `expr` по індексу `k`, `k1` — номер першого члену сумації, `kN` — останнього (він може бути рівний нескінченності).

```
(%i1)  sum(1/3^k,k,1,7);
```

$$(\%o1) \quad \frac{1093}{2187}$$

```
(%i2)  sum(1/k,k,1,10);
```

$$(\%o2) \quad \frac{7381}{2520}$$

Якщо спрямувати кількість членів підсумовуваного ряду до нескінченності, **Maxima** просто поверне введене значення у вигляді символьного виразу.

```
(%i3)  sum(1/3^k,k,1,inf);
```

$$(\%o3) \quad \sum_{k=1}^{\infty} \frac{1}{3^k}$$

Щоб програма здійснила підсумовування, необхідно після введення команди `sum` долучити опцію `simpsum` через кому.

```
(%i4)  sum(1/3^k,k,1,inf),simpsum;
```

(%o4) $\frac{1}{2}$

► Таким чином можна знаходити суму широкого класу рядів. Розглянемо, наприклад, ряди виду $\sum_{k=0}^{\infty} \frac{1}{k^n}$ для різних n.

```
(%i5)  sum(1/k,k,1,inf),simpsum;
```

```
sum: sum is divergent.
 -- an error. To debug this try: debugmode(true);
```

```
(%i6)  sum(1/k^2,k,1,inf),simpsum;
```

(%o6) $\frac{\pi^2}{6}$

```
(%i7)  sum(1/k^3,k,1,inf),simpsum;
```

(%o7) $\zeta\,(3)$

```
(%i8)  sum(1/k^4,k,1,inf),simpsum;
```

(%o8) $\frac{\pi^4}{90}$

```
(%i9)  sum(1/k^5,k,1,inf),simpsum;
```

(%o9) $\zeta\,(5)$

Бачимо, що перший ряд розбіжний, як і має бути за інтегральною ознакою, для $n = 3; 5$ виникає ζ-функція Рімана.

Знакозмінні ряди **Maxima** залишає без змін і не обчислює їх, якщо ці ряди не можна звести до знакосталих:

```
(%i10) sum((-1)^k*(2*k+1)/2^k,k,0,inf),simpsum;
```

(%o10) $\sum_{k=0}^{\infty} \frac{(2\cdot k+1)\cdot(-1)^k}{2^k}$

```
(%i11) sum((-1)^k/2^k,k,0,inf),simpsum;
```

(%o11) $\frac{2}{3}$

Якщо сума ряду не зводиться до аналітичного вигляду, **Maxima** здійснює підсумовування з якою завгодно наперед заданою точністю, але інтерфейс **wxMaxima** виводить на екран тільки перші 16 цифр після коми. Це обмеження можна розширити, задавши значення опції `fpprec:N`, де `N` — кількість цифр після коми, після цього записавши команду `bfloat(expr)`.

```
(%i12) sum(1/k^k,k,1,10),numer;
```

(%o12) 1.291285997059043

```
(%i13) sum(1/k^k,k,1,100),numer;
```

(%o13) 1.291285997062663

```
(%i14) sum(1/k^k,k,1,1000),numer;
```
(%o14) 1.291285997062663

```
(%i15) fpprec:200;
```
(%o15) 200

```
(%i16) bfloat(sum(1/k^k,k,1,1000),numer);
```
(%o16) 1.291285997062663540407282590 5[143 цифр]79972917794827300902564923 06b0

Щоб побачити всі 143 цифри, зазначені у квадратних дужках, необхідно команду задавати у консольній версії **Maxima** або у інтерфейсі **XMaxima**.

7.9 Ряди Фур'є

Будь-яку функцію $f(x)$ на симетричному проміжку $(-L;L)$ можна розкласти в ряд виду

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{\pi nx}{L}+b_n\sin\frac{\pi nx}{L}\right),$$

де коефіцієнти розкладу a_0, a_n, b_n визначаються виразами

$$a_0=\frac{1}{L}\int_{-L}^{L}f(x)dx,\quad a_n=\frac{1}{L}\int_{-L}^{L}f(x)\cos\frac{\pi nx}{L}dx,\quad b_n=\frac{1}{L}\int_{-L}^{L}f(x)\sin\frac{\pi nx}{L}dx.$$

Подібно до ряду Тейлора, який розкладає функції у поліномні суми, ряд Фур'є створює розклад з тригонометричних сум. Особливо це буває корисним у теорії сигналів, коли необхідно представляти ступінчасті та лінійні функції у вигляді хвиль.

Maxima містить у своєму складі пакет `fourie`, який надає змогу знайти коефіцієнти розкладу a_n та b_n, та має ще ряд певних можливостей. Основні команди пакету:

`fourier(f(x),x,L)` —- повертає список коефіцієнтів Фур'є `F_list` для функції $f(x)$ по змінній x на проміжку $[-L;L]$;

`foursimp(F_list)` —- спрощує вирази коефіцієнтів у розкладі `F_list` з використанням значень $\sin\pi n$, $\cos\pi n$;

`fourexpand(F_list,x,L,N)` —- конструює ряд Фур'є із коефіцієнтів `F_list` кількістю N членів;

`totalfourier(f(x),x,L)` —- видає 7 елементів: три коефіцієнти a_0, a_n, b_n у загальному вигляді, ті ж коефіцієнти у спрощеному вигляді (командою `foursimp`), повний вираз ряду Фур'є у вигляді нескінченної суми.

▶ Розкладемо для прикладу у ряд Фур'є функцію $f(x)=x$.

```
(%i1) load(fourie);
(%i2) fourier(x,x,%pi);
```
(%t2) $a_0=0$

(%t3) $a_n=0$

(%t4) $$b_n=\frac{2\cdot\left(\frac{\sin(\pi\cdot n)}{n^2}-\frac{\pi\cdot\cos(\pi\cdot n)}{n}\right)}{\pi}$$

(%o4) $[\%t2,\%t3,\%t4]$

Як бачимо, усі коефіцієнти a_n рівні нулю, як і повинно бути для непарних функцій (для парних відповідно $b_n = 0$). Результат виражений у вигляді списку `F_list` трьох елементів a_0, a_n, b_n. Застосуємо команду спрощення

```
(%i5)  foursimp(%o4);
```

$$(\%t5) \quad a_0 = 0$$

$$(\%t6) \quad a_n = 0$$

$$(\%t7) \quad b_n = -\frac{2\cdot(-1)^n}{n}$$

$$(\%o7) \quad [\%t5, \%t6, \%t7]$$

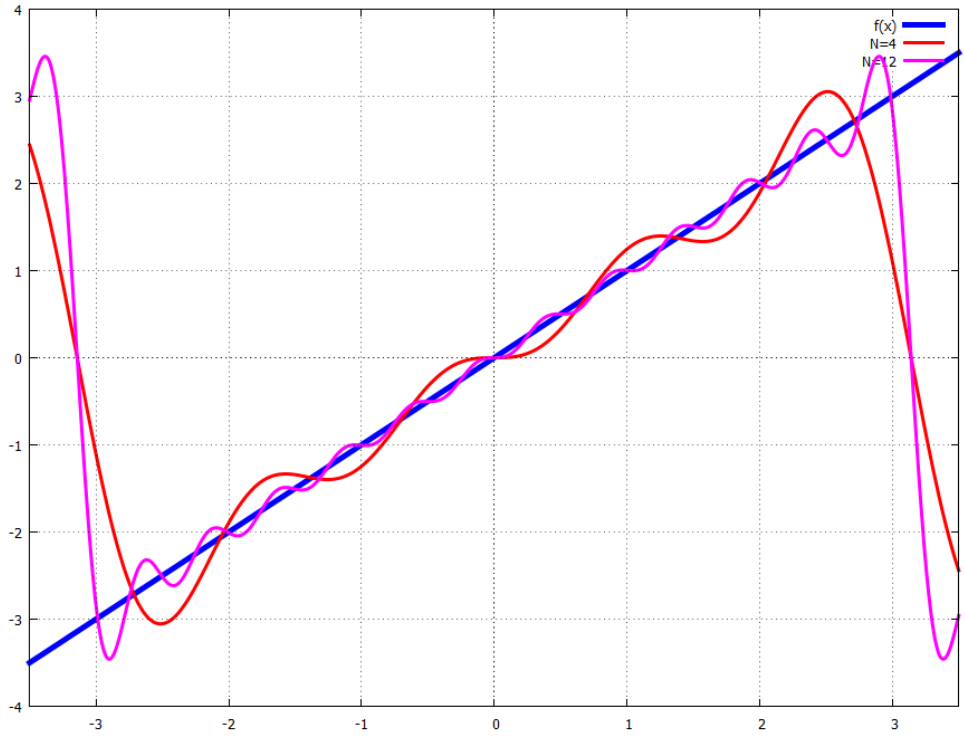


Рис. 7.7: Функція $f(x) = x$ та її розклади в ряд Фур'є.

Тепер створимо частинні суми ряду для $N = 4$ та $N = 12$:

```
(%i11) fourier4:fourexpand(%o4,x,%pi,4);
```

$$(\%o11) \quad -\frac{\sin(4\cdot x)}{2} + \frac{2\cdot\sin(3\cdot x)}{3} - \sin(2\cdot x) + 2\cdot\sin(x)$$

```
(%i12) fourier12:fourexpand(%o4,x,%pi,12);
```

$$(\%o12) \quad -\frac{\sin(12\cdot x)}{6} + \frac{2\cdot\sin(11\cdot x)}{11} - \frac{\sin(10\cdot x)}{5} + \frac{2\cdot\sin(9\cdot x)}{9} - \frac{\sin(8\cdot x)}{4} + \frac{2\cdot\sin(7\cdot x)}{7} - \frac{\sin(6\cdot x)}{3} + \frac{2\cdot\sin(5\cdot x)}{5} - \frac{\sin(4\cdot x)}{2} + \frac{2\cdot\sin(3\cdot x)}{3} - \sin(2\cdot x) + 2\cdot\sin(x)$$

Зобразимо вихідну функцію та її розклади для різних N (Рис.7.7):

```
(%i13) plot2d([x,fourier4,fourier12],[x,-3.5,3.5],
              [style,[lines,2,1],[lines,1,2],[lines,1,4]],
              [legend,"f(x)","N=4","N=12"]);
```

▶ Знайти повний розклад у ряд Фур'є функції $f(x) = x^2 + x$.

```
(%i14) totalfourier(x^2+x,x,%pi);
```

$$(\%t15) \quad a_n = \frac{\frac{2\cdot\pi^2\cdot\sin(\pi\cdot n)}{n} - \frac{4\cdot\sin(\pi\cdot n)}{n^3} + \frac{4\cdot\pi\cdot\cos(\pi\cdot n)}{n^2}}{\pi}$$

$$(\%t16) \quad b_n = \frac{\frac{2\cdot\sin(\pi\cdot n)}{n^2} - \frac{2\cdot\pi\cdot\cos(\pi\cdot n)}{n}}{\pi}$$

$$(\%t17) \quad a_0 = \frac{\pi^2}{3}$$

$$(\%t18) \quad a_n = \frac{4\cdot(-1)^n}{n^2}$$

$$(\%t19) \quad b_n = -\frac{2\cdot(-1)^n}{n}$$

$$(\%o19) \quad -2\cdot\left(\sum_{n=1}^{\infty}\frac{(-1)^n\cdot\sin(n\cdot x)}{n}\right) + 4\cdot\left(\sum_{n=1}^{\infty}\frac{(-1)^n\cdot\cos(n\cdot x)}{n^2}\right) + \frac{\pi^2}{3}$$

7.10 Завдання до Розділу 7

Знайти рівняння дотичної та нормалі до функції в точці, побудувати їх графіки

Функції, задані явно:

7.1. $y = x^{e^x}$; $x_0 = 1$;

7.2. $y = \ln x - x^2$; $x_0 = 4$;

7.3. $y = \operatorname{tg} x + \cos x$; $x_0 = \pi/2$;

7.4. $y = 1/x + x^2$; $x_0 = 2$;

7.5. $y = e^{\sqrt{x}}$; $x_0 = 2$;

7.6. $y = x^x$; $x_0 = 4$;

7.7. $y = (x^2 - 1)^3$; $x_0 = 2$;

7.8. $y = x\ln x$; $x_0 = 2$;

7.9. $y = \frac{3}{x^2+1}$; $x_0 = 3$;

7.10. $y = \frac{1}{\cos x}$; $x_0 = \pi/3$.

Функції, задані неявно:

7.11. $x^2 + y^2 = x^4 + y^4$; $x_0 = 1/2$;

7.12. $3x^2y^2 - 5x + \sin y = 3y - 1$; $x_0 = 0$;

7.13. $\ln y = \operatorname{arctg}\frac{y}{x}$; $x_0 = 2$;

7.14. $3x^2 - 5y^2 + 9x = 25 - 15y$; $x_0 = 2$.

7.15. $\cos x^2 = xe^y$; $x_0 = 0.3$.

7.16. $x^3y^3 - 2y = x$; $x_0 = 1$.

7.17. $(x + y)^2 = y - x$; $x_0 = -2$.

7.18. $x^4 + y^3 - 3x^2y = 0$; $x_0 = 1$.

7.19. $y^2 + x^3 - y^3 + 6 = 3y$; $x_0 = -1$.

7.20. $\sin y + x^2y^3 - \cos x = 2y$; $x_0 = 2$.

Функції, задані параметрично:

7.21. $\begin{cases} x(t) = 4 - 2t; \\ y(t) = 3 + 6t - 4t^2; \end{cases}$ $t_0 = 4.$

7.22. $\begin{cases} x(t) = \sqrt{t+1}; \\ y(t) = \frac{1}{t+1}; \end{cases}$ $t_0 = 1.8.$

7.23. $\begin{cases} x(t) = 3\sin t; \\ y(t) = -4\cos t; \end{cases}$ $t_0 = \pi.$

7.24. $\begin{cases} x(t) = 3\sin\left(\frac{t}{3}\right); \\ y(t) = -4\cos\left(\frac{t}{3}\right); \end{cases}$ $t_0 = 5.$

7.25. $\begin{cases} x(t) = 3 - 2\cos 3t; \\ y(t) = 1 + 4\sin 3t; \end{cases}$ $t_0 = 1.$

7.26. $\begin{cases} x(t) = 4\sin\left(\frac{t}{4}\right); \\ y(t) = 1 - 2\cos^2\left(\frac{t}{4}\right); \end{cases}$ $t_0 = \pi.$

7.27. $\begin{cases} x(t) = \sqrt{4 + \cos\left(\frac{5t}{2}\right)}; \\ y(t) = 1 + \frac{1}{3}\cos\left(\frac{5t}{2}\right) \end{cases}$ $t_0 = 1/5.$

7.28. $\begin{cases} x(t) = 2e^t; \\ y(t) = \cos\left(1 + e^{3t}\right); \end{cases}$ $t_0 = 1.$

7.29. $\begin{cases} x(t) = \frac{1}{2}e^{-3t}; \\ y(t) = e^{-6t} + 2e^{-3t} - 8; \end{cases}$ $t_0 = 2.$

7.30. $\begin{cases} x(t) = \sin 3t; \\ y(t) = e^{-3t}; \end{cases}$ $t_0 = 2.$

Функції, задані полярно:

7.31. $\rho = \dfrac{2}{\sin\theta - 3\cos\theta}; \quad \theta_0 = 2.$

7.32. $\rho = 1 + \cos\theta; \quad \theta_0 = 3.$

7.33. $\rho = \sin^2\theta; \quad \theta_0 = 1.$

7.34. $\rho = 1 + \dfrac{\pi^2}{\theta}; \quad \theta_0 = 2.$

7.35. $\rho = \dfrac{1}{\sqrt[3]{\cos\theta\sin^2\theta}}; \quad \theta_0 = 1.$

7.36. $\rho = e^{2\theta}; \quad \theta_0 = -1.$

7.37. $\rho = 2\cos 3\theta; \quad \theta_0 = \pi/6.$

7.38. $\rho = \dfrac{2}{\theta^2}; \quad \theta_0 = 1/2.$

7.39. $\rho = 3\theta^2; \quad \theta_0 = 1.$

7.40. $\rho = \sqrt{1+\theta^2}; \quad \theta_0 = \pi/4.$

Дослідити функцію та побудувати її графік

7.41. $y = \dfrac{1}{4}x^4 - \dfrac{1}{3}x^3 - x^2;$

7.42. $y = \dfrac{x}{(1-x^2)^2};$

7.43. $y = \dfrac{x-2}{\sqrt{x^2+1}};$

7.44. $y = 3x - x^3;$

7.45. $y = \sqrt{(x-1)(x-2)(x-3)};$

7.46. $y = \sin x \sin 3x; x \in [0, \pi];$

7.47. $y = \dfrac{e^x}{1+x};$

7.48. $y = \dfrac{\ln x}{\sqrt{x}};$

7.49. $y = \sqrt{1 - e^{-x^2}};$

7.50. $y = \ln(x + \sqrt{1+x^2});$

Розділ 8

Диференціальні рівняння

Система **Maxima** може успішно розв'язувати такі типи диференціальних рівнянь першого порядку: з відокремленими змінними, лінійні, нелінійні рівняння, однорідні, неоднорідні; типи рівнянь другого порядку: зі сталими коефіцієнтами, лінійні однорідні зі змінними коефіцієнтами, які можуть бути перетворені на рівняння зі сталими коефіцієнтами, рівнянням Ейлера, рівняння, що розв'язуються методом варіації сталої та рівняння, що допускають пониження порядку.

Розглянемо команди системи **Maxima** для пошуку розв'язків диференціальних рівнянь та їх систем у символьному вигляді.

`desolve(deqn,y(x))` — шукає часткові розв'язки лінійних диференціальних рівнянь першого та другого порядку.

`desolve([deqn1,...,deqnN],[y1,...,yN])` — шукає часткові розв'язки систем лінійних диференціальних рівнянь першого та другого порядку.

Робота цієї команди заснована на перетворенні Лапласа заданих дифрівнянь. Вона приймає два аргументи, перший з яких — рівняння або перелік рівнянь, другий — одна залежна змінна або список залежних змінних відповідно.

`desolve` знаходить розв'язок задачі Коші для відповідного рівняння, тому якщо значення залежної змінної або її похідної не задані, то у знайденому розв'язку вони подаються у вигляді `y(0)` або $\left.\frac{\mathtt{d}}{\mathtt{dx}}\mathtt{y(x)}\right|_{\mathtt{x=0}}$.

Значення функцій та похідних у початковій точці можна задати за допомогою функції `atvalue`, яка має наступний синтаксис:

`atvalue(f(x),x=a,A)` — присвоює значення `A` функції $f(x)$ в точці $x = a$;

`atvalue('diff(f(x),x),x=a,B)` — присвоює значення `B` похідній функції $f'(x)$ в точці $x = a$.

Необхідно підкреслити, що похідні в рівняннях та системах, розв'язуваних за допомогою цієї команди, повинні бути записані у вигляді `'diff(f(x),x)`, тобто оператор похідної у відкладеному вигляді (з апострофом), а аргумент функції виписаний явно.

Якщо команда `desolve` не може знайти розв'язок, вона повертає значення `false`.

`ode2(deqn,y,x)` — призначена для розв'язання звичайних лінійних диференціальних рівнянь першого та другого порядку.

Розв'язок знаходиться у загальній формі, він може повернутись як у явній, так і неявній формі. Тут у списку параметрів явним чином вказується залежна змінна, тому позначення типу $y(x)$ не потрібні: функція та змінна позначаються одиночними літерами.

Як і для звичайних рівнянь, є можливість перевірити розв'язок за допомогою підстановки. Довільна константа для рівнянь першого порядку позначається через `%c`. У рівняннях другого порядку константи позначаються як `%k1` та `%k2`. Якщо функція ode2 не може отримати розв'язок з якоїсь причини, вона повертає значення `false`, при цьому

видаляючи звіт про помилку.

Окрім `ode2` існує ще три команди для пошуку частинних розв'язків на основі отриманих загальних. Тобто ці команди, отримуючи конкретні умови щодо значення функції-розв'язку в заданій точці, знаходять виходячи з цих передумов відповідні значення констант інтегрування. Для розв'язку задач Коші (Intial Value Problems, IVP) використовуються функції `ic1` і `ic2` у випадку ЗДР першого та другого порядку відповідно. Крайові задачі (Boundary Value Problems, BVP) розв'язують з використанням функції `bc2`.

`ic1(solution,x=x0,y=y0)` — обробляє розв'язок диференціального рівняння першого порядку з початковими умовами. Приймає три аргументи: перший — це сам розв'язок, у формі, в якій він знаходиться функцією ode2, другий — це початкове значення незалежної змінної у вигляді $x = x_0$, третій — значення функції в цій точці $y_0 = y(x_0)$. Повертає частковий розв'язок, що проходить через точку із зазначеними координатами (x_0, y_0).

`ic2(solution,x=x0,y=y0,dy=dy0)` — призначена для обробки розв'язку диференціального рівняння другого порядку з початковими умовами. Приймає чотири аргументи: до аргументів команди `ic1` додається `dy0` - початкове значення першої похідної залежної змінної відносно незалежної змінної у формі $y'(x_0) = dy0$.

`bc2(solution,x1,y1,x2,y2)` — розв'язує крайову задачу для дифрівняння другого порядку. Тут `solution` — загальний розв'язок рівняння, який отримується функцією `ode2`; x_1 — задає значення незалежної змінної в першій межовій точці при $x = x_1$, y_1 — визначає значення залежної змінної в першій межовій точці у вигляді $y_1 = y(x_1)$. Вирази x_2 і y_2 задають аналогічні значення для цих змінних у другій межовій точці.

8.1 Диференціальні рівняння першого порядку

8.1.1 Рівняння з розділеними змінними

Це рівняння виду $y'(x) = f(x) \cdot g(y)$, **Maxima** може розв'язати їх різними способами.

▶ Знайти загальний і частинний розв'язок рівняння $y' = \dfrac{\sqrt{1-y^2}}{\sqrt{1-x^2}}$ при $y(0) = 1$.

```
(%i1)  deqn1:'diff(y,x)=sqrt(1-y^2)/sqrt(1-x^2);
```

$$(\%o1) \quad \frac{d}{d\,x} \cdot y = \frac{\sqrt{1-y^2}}{\sqrt{1-x^2}}$$

Загальний розв'язок:

```
(%i2)  sol1:ode2(deqn1,y,x);
```

$$(\%o2) \quad \operatorname{asin}(y) = \operatorname{asin}(x) + \%c$$

Частковий розв'язок при початкових умовах $x_0 = 0, y_0 = 1$:

```
(%i3)  ic1(sol1,x=0,y=1);
```

$$(\%o3) \quad \operatorname{asin}(y) = \frac{2 \cdot \operatorname{asin}(x) + \pi}{2}$$

▶ Ще приклад із використанням `desolve`: $y'' = x^2 y^2$, $y(0) = 1, y'(0) = 4$.

```
(%i4)  deqn2:'diff(y(x),x,2)=y^2*x^2;
```

$$(\%o4) \quad \frac{d^2}{d\,x^2} \cdot \mathrm{y}(x) = x^2 \cdot y^2$$

```
(%i5)  atvalue(y(x),x=0,1);
```

$$(\%o5) \quad 1$$

```
(%i6)  atvalue('diff(y(x),x),x=0,4);
```

(%o6) 4

```
(%i7)  desolve(deqn2,y(x));
```

(%o7) $\mathrm{y}\,(x) = \dfrac{x^4 \cdot y^2}{12} + 4 \cdot x + 1$

Якщо початкові умови, зазначені командою `atvalue`, не задати, система поверне розв'язок у наступному вигляді:

```
(%i1)  deqn2:'diff(y(x),x,2)=y^2*x^2;
```

(%o1) $\dfrac{d^2}{d\,x^2} \cdot \mathrm{y}\,(x) = x^2 \cdot y^2$

```
(%i2)  desolve(deqn2,y(x));
```

(%o2) $\mathrm{y}\,(x) = \dfrac{x^4 \cdot y^2}{12} + x \cdot \left(\dfrac{d}{d\,x} \cdot \mathrm{y}\,(x) \Big|_{x=0} \right) + \mathrm{y}\,(0)$

▶ Розв'язати один з видів рівняння Бернуллі $y' + 2e^x y = 2e^x \sqrt{y}$.

```
(%i7)  deqn4:'diff(y,x)+2*%e^x*y=2*%e^x*sqrt(y);
```

(%o7) $\dfrac{d}{d\,x} \cdot y + 2 \cdot e^x \cdot y = 2 \cdot e^x \cdot \sqrt{y}$

```
(%i8)  ode2(deqn4,y,x);
```

(%o8) $-\log\left(\sqrt{y} - 1\right) = e^x + \%c$

```
(%i9)  method;
```

(%o9) *separable*

▶ Розв'яжемо систему рівнянь

$$\begin{cases} f'(x) = g'(x) + \sin x; \\ g'(x) = 5f'(x) - \cos x; \end{cases} \qquad f(0) = 1; \quad g(0) = 7.$$

```
(%i1)  deqn1:'diff(f(x),x)='diff(g(x),x)+sin(x);
```

(%o1) $\dfrac{d}{d\,x} \cdot \mathrm{f}\,(x) = \dfrac{d}{d\,x} \cdot \mathrm{g}\,(x) + \sin\,(x)$

```
(%i2)  deqn2:'diff(g(x),x)=5*'diff(f(x),x)-cos(x);
```

(%o2) $\dfrac{d}{d\,x} \cdot \mathrm{g}\,(x) = 5 \cdot \left(\dfrac{d}{d\,x} \cdot \mathrm{f}\,(x) \right) - \cos\,(x)$

```
(%i4)  atvalue(f(x),x=0,1);atvalue(g(x),x=0,7);
```

(%o3) 1

(%o4) 7

```
(%i5)  desolve([deqn1,deqn2],[f(x),g(x)]);
```

(%o5) $[\mathrm{f}\,(x) = \dfrac{\sin\,(x)}{4} + \dfrac{\cos\,(x)}{4} + \dfrac{3}{4}, \mathrm{g}\,(x) = \dfrac{\sin\,(x)}{4} + \dfrac{5 \cdot \cos\,(x)}{4} + \dfrac{23}{4}]$

▶ Розв'язати задачу Коші для рівняння $xy' = y(1 + \ln y - \ln x), \quad y(1) = e^2$.

```
(%i1)  eqn5:x*'diff(y,x)=y*(log(y)-log(x)+1);
```

$$(\%o1) \quad x \cdot \left(\frac{d}{d\,x} \cdot y\right) = y \cdot (\log(y) - \log(x) + 1)$$

```
(%i3)  sol5:ode2(eqn5,y,x);
```

$$(\%o3) \quad x = \%c \cdot (\log(y) - \log(x))$$

```
(%i6)  ic1(sol5,x=1,y=%e^2);
```

$$(\%o6) \quad x = \frac{\log(y) - \log(x)}{2}$$

8.1.2 Однорідні рівняння

Це рівняння виду $y'(x) = f\left(\frac{y}{x}\right)$.

► Розв'язати рівняння $y' = \left(\frac{y}{x}\right)^2 + 2\frac{y}{x}$.

```
(%i1)  deqn1:'diff(y,x)=(y/x)^2+2*(y/x);
```

$$(\%o1) \quad \frac{d}{d\,x} \cdot y = \frac{y^2}{x^2} + \frac{2 \cdot y}{x}$$

```
(%i2)  ode2(deqn1,y,x);
```

$$(\%o2) \quad -\frac{x \cdot y + x^2}{y} = \%c$$

```
(%i3)  ic1(%o2,x=3,y=1);
```

$$(\%o3) \quad -\frac{x \cdot y + x^2}{y} = -12$$

```
(%i4)  solve(%o3,y);
```

$$(\%o4) \quad [y = -\frac{x^2}{x - 12}]$$

8.1.3 Рівняння в повних диференціалах

Це рівняння виду $P(x,y)dx + Q(x,y)dy = 0$.

Якщо виконується умова $\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}$, то ліва частина являє собою повний диференціал деякої функції. Якщо ж ця умова не виконується, але задовільняється теорема єдиності, то можна підібрати такий інтегруючий множник $\mu(x)$, який зводить ліву частину до повного диференціалу. Цей множник можна викликати командою `intfactor`, а метод розв'язання викликається командою `method`.

Для розв'язку в **Maxima** ліву частину небхідно представити у вигляді

$$P(x,y) + Q(x,y)\frac{dy}{dx} = 0.$$

► Розв'язати рівняння $2xy + x^2y + \frac{y^3}{3} + (x^2 + y^2)y' = 0$.

```
(%i1)  deqn1:(2*x*y+x^2*y+y^3/3)+(x^2+y^2)*'diff(y,x)=0;
```

$$(\%o1) \quad \left(y^2 + x^2\right) \cdot \left(\frac{d}{d\,x} \cdot y\right) + \frac{y^3}{3} + x^2 \cdot y + 2 \cdot x \cdot y = 0$$

```
(%i2)  ode2(deqn1,y,x);
```

$$(\%o2) \quad \frac{e^x \cdot y^3 + 3 \cdot x^2 \cdot e^x \cdot y}{3} = \%c$$

```
(%i3)  intfactor;
```

$$(\%o3) \quad e^x$$

```
(%i4)  method;
```

$$(\%o4) \quad exact$$

► Розв'язати рівняння $3x^2 + 6xy^2 + (6x^2y + 4y^3)y' = 0$.

```
(%i5)  deqn2:(3*x^2+6*x*y^2)+(6*x^2*y+4*y^3)*'diff(y,x)=0;
```

$$(\%o5) \quad \left(4 \cdot y^3 + 6 \cdot x^2 \cdot y\right) \cdot \left(\frac{d}{d\,x} \cdot y\right) + 6 \cdot x \cdot y^2 + 3 \cdot x^2 = 0$$

```
(%i6)  ode2(deqn2,y,x);
```

$$(\%o6) \quad y^4 + 3 \cdot x^2 \cdot y^2 + x^3 = \%c$$

```
(%i7)  intfactor;
```

$$(\%o7) \quad 1$$

```
(%i8)  method;
```

$$(\%o8) \quad exact$$

8.1.4 Рівняння Бернуллі

Це рівняння виду

$$y'(x) = \sum_i Q_i(x) \cdot y^{\alpha_i}, \quad \alpha_i = \text{const}.$$

► Розв'язати рівняння $y' = \frac{4}{x}y + x\sqrt{y}$.

```
(%i1)  deqn1:'diff(y,x)=4/x*y+x*sqrt(y);
```

$$(\%o1) \quad \frac{d}{d\,x} \cdot y = \frac{4 \cdot y}{x} + x \cdot \sqrt{y}$$

```
(%i2)  ode2(deqn1,y,x);
```

$$(\%o2) \quad y = x^4 \cdot \left(\frac{\log(x)}{2} + \%c\right)^2$$

```
(%i3)  method;
```

$$(\%o3) \quad bernoulli$$

► Розв'язати рівняння $y' + \frac{y}{x} = -xy^2$.

```
(%i4)  deqn2:'diff(y,x)+y/x=-x*y^2;
```

$$(\%o4) \quad \frac{d}{d\,x} \cdot y + \frac{y}{x} = -x \cdot y^2$$

```
(%i5)  ode2(deqn2,y,x);
```

$$(\%o5) \quad y = \frac{1}{x \cdot (x + \%c)}$$

```
(%i6)  method;
```

(%o6) *bernoulli*

▶ Розв'язати задачу Коші для рівняння $y' - 8y = x$, $y(0) = 2$.

```
(%i8)  deqn3:5*'diff(y(x),x)-8*y=x; atvalue(y(x),x=0,2);
```

$$(\%o7) \quad 5 \cdot \left(\frac{d}{d\,x} \cdot \mathrm{y}\,(x)\right) - 8 \cdot y = x$$

(%o8) 2

```
(%i9)  desolve(deqn3,y(x));
```

$$(\%o9) \quad \mathrm{y}\,(x) = \frac{8 \cdot x \cdot y}{5} + \frac{x^2}{10} + 2$$

```
(%i10) method;
```

(%o10) *bernoulli*

8.2 Диференціальні рівняння другого порядку

В **Maxima** за допомогою команди `ode2` можливе пряме розв'язання лише лінійних диференціальних рівнянь другого порядку $y'' + p(x)y' + q(x)y = r(x)$. Спочатку відшукується розв'язок однорідного рівняння у формі $y = k_1 y_1 + k_2 y_2$ (k_1, k_2 — довільні сталі), потім шукається частинний розв'язок методом варіації сталих.

8.2.1 Рівняння зі сталими коефіцієнтами

Це рівняння виду $y'' + ay' + by = r(x)$.

▶ Розв'язати неоднорідне рівняння загального виду $2y'' - y' - y = 4xe^{2x}$.

```
(%i1)  deqn1:2*'diff(y,x,2)-'diff(y,x)-y=4*x*exp(2*x);
```

$$(\%o1) \quad 2 \cdot \left(\frac{d^2}{d\,x^2} \cdot y\right) - \frac{d}{d\,x} \cdot y - y = 4 \cdot x \cdot e^{2 \cdot x}$$

```
(%i2)  ode2(deqn1,y,x);
```

$$(\%o2) \quad y = \frac{(20 \cdot x - 28) \cdot e^{2 \cdot x}}{25} + \%k1 \cdot e^{x} + \%k2 \cdot e^{-\frac{x}{2}}$$

Командою `yp` можна виділити частинний розв'язок.

```
(%i3)  yp;
```

$$(\%o3) \quad \frac{(20 \cdot x - 28) \cdot e^{2 \cdot x}}{25}$$

▶ Розв'язати неоднорідне рівняння з кратними коренями характеристичного рівняння $y'' - 2y' + y = xe^x$.

```
(%i4)  deqn2:'diff(y,x,2)-2*'diff(y,x)+y=x*exp(x);
```

$$(\%o4) \quad \frac{d^2}{d\,x^2} \cdot y - 2 \cdot \left(\frac{d}{d\,x} \cdot y\right) + y = x \cdot e^{x}$$

```
(%i5)  ode2(deqn2,y,x);
```

$$(\%o5)\quad y=\frac{x^3\cdot e^x}{6}+(\%k2\cdot x+\%k1)\cdot e^x$$

```
(%i6)  yp;
```

$$(\%o6)\quad \frac{x^3\cdot e^x}{6}$$

▶ Розв'язати неоднорідне рівняння з комплексними коренями $y''+y=x\sin x$.

```
(%i7)  deqn3:'diff(y,x,2)+y=x*sin(x);
```

$$(\%o7)\quad \frac{d^2}{d\,x^2}\cdot y+y=x\cdot\sin(x)$$

```
(%i8)  ode2(deqn3,y,x);
```

$$(\%o8)\quad y=\frac{2\cdot x\cdot\sin(x)+(1-2\cdot x^2)\cdot\cos(x)}{8}+\%k1\cdot\sin(x)+\%k2\cdot\cos(x)$$

```
(%i9)  yp;
```

$$(\%o9)\quad \frac{2\cdot x\cdot\sin(x)+(1-2\cdot x^2)\cdot\cos(x)}{8}$$

8.2.2 Рівняння зі змінними коефіцієнтами

Це рівняння виду $y''+p(x)y'+q(x)y=r(x)$.

▶ Розв'язати задачу Коші для рівняння $x^2y''-xy'=3x^3, y(1)=1, y'(1)=4$.

```
(%i1)  deqn1:x^2*'diff(y,x,2)-x*'diff(y,x)=3*x^3;
```

$$(\%o1)\quad x^2\cdot\left(\frac{d^2}{d\,x^2}\cdot y\right)-x\cdot\left(\frac{d}{d\,x}\cdot y\right)=3\cdot x^3$$

```
(%i2)  sol1:ode2(deqn1,y,x);
```

$$(\%o2)\quad y=x^3+\%k2\cdot x^2-\frac{\%k1}{2}$$

```
(%i3)  ic2(sol1,x=1,y=1,'diff(y,x)=4);
```

$$(\%o3)\quad y=x^3+\frac{x^2}{2}-\frac{1}{2}$$

▶ Розв'язати рівняння в загальному вигляді: $xy''+y'=x^2$.

```
(%i4)  deqn2:x*'diff(y,x,2)+'diff(y,x)=x^2;
```

$$(\%o4)\quad x\cdot\left(\frac{d^2}{d\,x^2}\cdot y\right)+\frac{d}{d\,x}\cdot y=x^2$$

```
(%i5)  ode2(deqn2,y,x);
```

$$(\%o5)\quad y=\%k1\cdot\log(x)+\frac{x^3}{9}+\%k2$$

▶ Розв'язати задачу Коші для рівняння $y''-7y'+6y=(x-2)e^x, y(0)=1, y'(0)=3$.

```
(%i6)  deqn3:'diff(y,x,2)-7*'diff(y,x)+6*y=(x-2)*%e^x;
```

$$(\%o6)\quad \frac{d^2}{d\,x^2}\cdot y-7\cdot\left(\frac{d}{d\,x}\cdot y\right)+6\cdot y=(x-2)\cdot e^x$$

```
(%i7)  sol3:ode2(deqn3,y,x);
```

$$(\%o7) \quad y = \%k1 \cdot e^{6 \cdot x} - \frac{(25 \cdot x^2 - 90 \cdot x - 18) \cdot e^x}{250} + \%k2 \cdot e^x$$

```
(%i8)  ic2(sol3,x=0,y=1,'diff(y,x)=3);
```

$$(\%o8) \quad y = \frac{41 \cdot e^{6 \cdot x}}{125} - \frac{(25 \cdot x^2 - 90 \cdot x - 18) \cdot e^x}{250} + \frac{3 \cdot e^x}{5}$$

8.3 Диференціальні рівняння з межовими умовами (крайові задачі)

Для задання межових умов при інтегруванні ДР другого порядку використовується команда `bc2`. Вона задає значення функції у двох точках, які дозволяють розв'язати систему рівнянь відносно двох невідомих сталих інтегрування `%k1` і `%k2`.

▶ Розв'язати рівняння $y'' + y \cdot (y')^3 = 0$, межові умови $y(0) = 1, y(1) = 3$.

```
(%i1)  deqn1:'diff(y,x,2)+y*'diff(y,x)^3=0;
```

$$(\%o1) \quad \frac{d^2}{d\,x^2} \cdot y + y \cdot \left(\frac{d}{d\,x} \cdot y\right)^3 = 0$$

```
(%i2)  sol1:ode2(deqn1,y,x);
```

$$(\%o2) \quad \frac{y^3 + 6 \cdot \%k1 \cdot y}{6} = x + \%k2$$

```
(%i3)  bc2(sol1,x=0,y=1,x=1,y=3);
```

$$(\%o3) \quad \frac{y^3 - 10 \cdot y}{6} = x - \frac{3}{2}$$

▶ Розв'язати рівняння $y'' + y = x$, межові умови $y(0) = 0, y(4) = 1$.

```
(%i4)  deqn2:'diff(y,x,2)+y=x;
```

$$(\%o4) \quad \frac{d^2}{d\,x^2} \cdot y + y = x$$

```
(%i5)  sol2:ode2(deqn2,y,x);
```

$$(\%o5) \quad y = \%k1 \cdot \sin(x) + \%k2 \cdot \cos(x) + x$$

```
(%i6)  bc2(sol2,x=0,y=0,x=4,y=1);
```

$$(\%o6) \quad y = x - \frac{3 \cdot \sin(x)}{\sin(4)}$$

8.4 Пакет розширення `contrib_ode`

У системі **Maxima** є додатковий пакет `contrib_ode`, який дозволяє знаходити розв'язки нелінійних диференціальних рівнянь першого та другого порядку з розширеними можливостями. З допомогою `contrib_ode` можливим є розв'язання рівняння Клеро, Лагранжа, Ріккаті та ін. В загальному випадку результат — список розв'язків. Для деяких рівнянь (зокрема Ріккаті) розв'язок виводиться у формі іншого ЗДР — результату заміни змінних. Функция `contrib_ode` реалізує методи факторизації (factorization), Клеро (Clairault), Лагранжа (Lagrange), Ріккаті (Riccati), Абеля (Abel) та метод симетрії Лі

(Lie symmetry method). Для його використання необхідно підключити пакет за допомогою команди `load(contrib_ode)`.

Синтаксис команди такий же, як і в `ode2`:

`contrib_ode(deqn,y,x)`.

Якщо диференціальне рівняння система може розв'язати, вона повертає розв'язок або список розв'язків, причому кожен з них може бути представлене в одному з таких видів:

— шукана функція виражена у явному вигляді;

— шукана функція виражена у неявному вигляді;

— розв'язок одержано у параметричному вигляді з параметром `%t`;

— диференціальне рівняння перетворюється на інше диференціальне рівняння відносно функції `%u`.

▶ Розв'язати рівняння $x(y')^2 - \dfrac{y'}{1+xy} + y = 0$.

```
(%i1)  load(contrib_ode);

(%i2)  deqn1:x*'diff(y,x)^2-(1+x*y)*diff(y,x)+y=0;
```

$$(\%o2) \quad x \cdot \left(\frac{d}{d\,x} \cdot y\right)^2 + y = 0$$

```
(%i3)  contrib_ode(deqn1,y,x);
```

$$(\%t3) \quad x \cdot \left(\frac{d}{d\,x} \cdot y\right)^2 + y = 0$$

```
 first order equation not linear in y'
```

$$(\%o3) \quad [x^2 \cdot \left(y^2 + (2 \cdot x + 2 \cdot \%c) \cdot y + x^2 - 2 \cdot \%c \cdot x + \%c^2\right) = 0]$$

```
(%i4)  method;
```

(%o4) *lagrange*

▶ Розв'язати рівняння $(y')^2 + xy' - y = 0$.

```
(%i5)  deqn2:'diff(y,x)^2+x*'diff(y,x)-y=0;
```

$$(\%o5) \quad \left(\frac{d}{d\,x} \cdot y\right)^2 + x \cdot \left(\frac{d}{d\,x} \cdot y\right) - y = 0$$

```
(%i6)  contrib_ode(deqn2,y,x);
```

$$(\%t6) \quad \left(\frac{d}{d\,x} \cdot y\right)^2 + x \cdot \left(\frac{d}{d\,x} \cdot y\right) - y = 0$$

```
 first order equation not linear in y'
```

$$(\%o6) \quad [y = \%c \cdot x + \%c^2, y = -\frac{x^2}{4}]$$

```
(%i7)  method;
```

(%o7) *clairault*

▶ Розв'язати рівняння $(1 + y')x + (y')^2 - y = 0$.

```
(%i8)  deqn3:(1+'diff(y,x))*x+('diff(y,x))^2-y=0;
```

$$(\%o8) \quad \left(\frac{d}{d\,x} \cdot y\right)^2 + x \cdot \left(\frac{d}{d\,x} \cdot y + 1\right) - y = 0$$

```
(%i9)  contrib_ode(deqn3,y,x);
```

$$(\%t9) \quad \left(\frac{d}{d\,x}\cdot y\right)^2 + x\cdot\left(\frac{d}{d\,x}\cdot y + 1\right) - y = 0$$

```
first order equation not linear in y'
```

$$(\%o9) \quad [[x = e^{-\%t}\cdot\left(\%c - 2\cdot(\%t-1)\cdot e^{\%t}\right), y = (\%t+1)\cdot x + \%t^2]]$$

```
(%i10) method;
```

(%o10) *lagrange*

В деяких випадках можливий розв'язок лише в параметричному вигляді.

▶ Розв'язати рівняння $(1+y')x + (y')^2 - y = 0$.

```
(%i11) deqn4:'diff(y,x)=(x+y)^2;
```

$$(\%o11) \quad \frac{d}{d\,x}\cdot y = (y+x)^2$$

```
(%i12) contrib_ode(deqn4,y,x);
```

$$(\%o12) \quad [[x = \%c - \operatorname{atan}\left(\sqrt{\%t}\right), y = -x - \sqrt{\%t}], [x = \operatorname{atan}\left(\sqrt{\%t}\right) + \%c, y = \sqrt{\%t} - x]]$$

```
(%i13) method;
```

(%o13) *lagrange*

Пакет `contrib_ode` дозволяє успішно розв'язувати диференціальні рівняння, які не обчислюються `ode2` безпосередньо, наприклад узагальнені однорідні рівняння. В такому випадку цей пакет використовує методи Абеля та симетрії Лі.

▶ Розв'язати рівняння $(2x - y + 4)y' + (x - 2y + 5) = 0$.

```
(%i14) deqn5:(2*x-y+4)*'diff(y,x)+(x-2*y+5)=0;
```

$$(\%o14) \quad (-y + 2\cdot x + 4)\cdot\left(\frac{d}{d\,x}\cdot y\right) - 2\cdot y + x + 5 = 0$$

```
(%i15) contrib_ode(deqn5,y,x);
```

$$(\%o15) \quad [\frac{\log\left(3 - \frac{2\cdot(2\cdot x+4)-x-5}{-y+2\cdot x+4}\right) - 3\cdot\log\left(1 - \frac{2\cdot(2\cdot x+4)-x-5}{-y+2\cdot x+4}\right) + 2\cdot\log\left(-\frac{2\cdot(2\cdot x+4)-x-5}{4\cdot(-y+2\cdot x+4)}\right)}{2} =$$

$$\log(x+1) + \%c]$$

```
(%i16) method;
```

(%o16) *abel2*

▶ Розв'язати рівняння $y' = \dfrac{1 - 3x - 3y}{1 + x + y}$.

```
(%i17) deqn6:'diff(y,x)=(1-3*x-3*y)/(1+x+y);
```

$$(\%o17) \quad \frac{d}{d\,x}\cdot y = \frac{-3\cdot y - 3\cdot x + 1}{y + x + 1}$$

```
(%i18) contrib_ode(deqn6,y,x);
```

$$(\%o18) \quad [\frac{2\cdot\log(y + x - 1) + y + 3\cdot x}{2} = \%c]$$

```
(%i19) method;
```

(%o19) *lie*

8.5 Числові методи

В ряді випадків відшукати символьний розв'язок ДР в достатньо компактному вигляді неможливо, тому необхідно скористатись числовими способами. Загалом спроба розв'язати диференціальні рівняння числовими методами веде давню історію, протягом якої з'явились методи Ейлера (він же Рунге–Кутти 1-го порядку), Ейлера–Коші (він же Рунге–Кутти 2-го порядку), і власне Рунге–Кутти 4-го порядку. Останній вважається достатньо точним для отримання надійного розв'язку. Підвищення порядку традиційно нівелюється набіганням похибки обчислень, яка за розміром починає збігатись з різницевим членом, тому зупиняються на порядку 4. Є, однак, задачі специфічного вигляду (так звані жорсткі задачі, stiff problems), в яких на певних інтервалах інтегрування все ж доцільно використовувати 5-ий порядок розкладу. Принципова схема побудови такого розв'язку була розроблена Фельбергом, тому цей спосіб називається методом Рунге–Кутти–Фельберга 4-5 порядку.

Для побудови числового розв'язку диференціального рівняння першого порядку воно має бути записане у вигляді $\frac{dy}{dx} = f(x,y)$. У випадку автономної системи — має бути у вигляді

$$\begin{cases} \frac{dx}{dt} = G(x,y); \\ \frac{dy}{dt} = H(x,y). \end{cases}$$

Автономність системи означає відсутність явної залежності від змінної t у правих частинах рівнянь.

Як було зазначено, метод РК4 придатний до розв'язання лише ЗДР першого порядку, однак є спосіб поширення його і на вищі порядки, а саме запровадженням проміжної змінної. Для прикладу, зробивши заміну $\dot{y} = z$, $\ddot{y} = \dot{z}$, можна одне рівняння другого порядку перетворити на систему двох рівнянь першого порядку, до розв'язання якої цей метод цілком застосовний. Докладніше ми ознайомимося з цим у Розділах 10 та 11.

8.5.1 Метод Рунге–Кутти 4-го порядку `rk`

Maxima включає в себе пакет розширення `dynamics`, що дозволяє проінтегрувати системи ДР методом Рунге–Кутти. Окрім цього, пакет `dynamics` містить ряд функцій для побудови різноманітних фракталів.

Метод Рунге–Кутти 4-го порядку реалізує команда `rk`. Синтаксис виклику:

`rk(right_deqn,y,y0,[x,x0,x1,h])` — для окремого рівняння;

`rk([right_deqn],[x,y],[x0,y0],[t,t0,t1,h])` — для автономної системи рівнянь.

Тут `right_deqn` — права частина рівняння або їх список, `y` або `[x,y]` — залежна змінна або їх список, `y0` або `[x0,y0]` — початкові значення залежної змінної, `[x,x0,x1,h]` або `[t,t0,t1,h]` — оголошення незалежної змінної, її початкове, кінцеве значення, та крок обчислень.

Для демонстрації методу розв'яжемо наступну задачу.

▶ Розв'язати задачу Коші для рівняння $y' = y - x, \quad y(0) = 1.5$. Методом Рунге–Кутти знайти графічний розв'язок на відрізку $x \in [0,1]$, крок $h = 0.1$.

Розв'яжемо спочатку це рівняння командою `ode2`:

```
(%i1)  deqn1:'diff(y,x)=y-x;
```

$$(\%o1) \quad \frac{d}{d\,x} \cdot y = y - x$$

```
(%i3)  ode2(deqn1,y,x); ic1(%,x=0,y=1.5);
```

$$(\%o2) \quad y = \left(\%c - (-x-1)\cdot e^{-x}\right)\cdot e^{x}$$

```
rat: replaced 0.5 by 1/2 = 0.5
```

(%o3) $$y = \frac{e^x + 2\cdot x + 2}{2}$$

```
(%i4) sol1:rhs(%o3);
```

(%o4) $$\frac{e^x + 2\cdot x + 2}{2}$$

Тепер скористаємось числовим розв'язком.

```
(%i5) load(dynamics);
(%i6) sol2:rk(y-x,y,1.5,[x,0,1,0.1]);
```

(%o6) *list*

```
(%i7) plot2d([sol1,[discrete,sol2]],[x,0,1],[style,[lines,3,1],
        [points,3,2,1]],[legend,false]);
```

(%o7) $[C : /Users/orreg/maxout.gnuplot]$

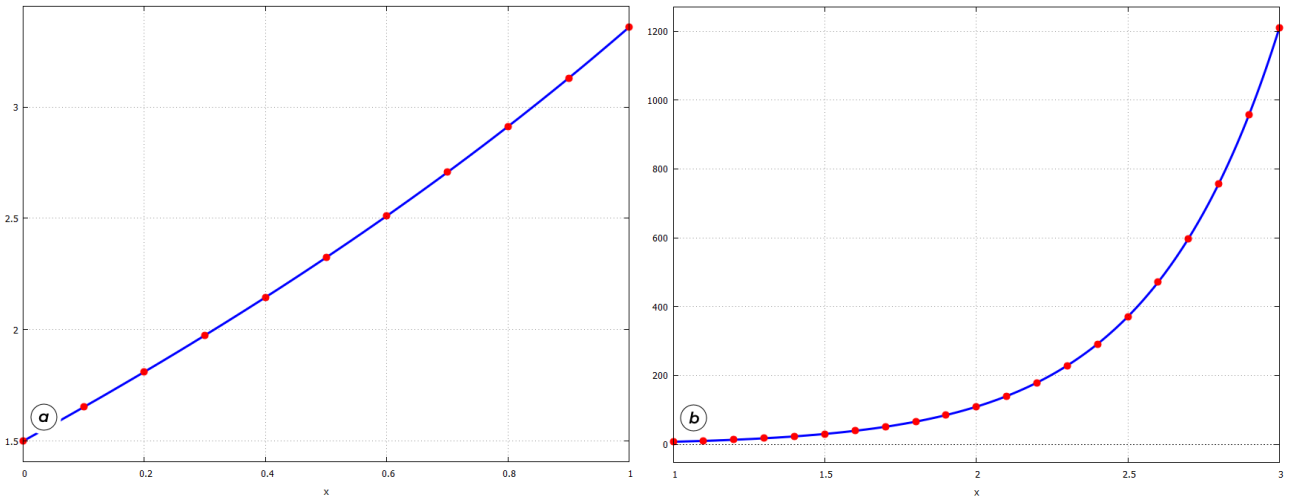


Рис. 8.1: Графіки прямого та числового розв'язків ДР за методом Рунге-Кутти.

Графіки розв'язків зображені на Рис. 8.1,(а). Бачимо, що всі точки числового розв'язку чітко лежать на кривій, що підкреслює точність обраного методу.

▶ Розглянемо рівняння, яке ми вже розв'язали раніше: $xy' = y(1+\ln y - \ln x)$, $y(1) = e^2$. Нагадаємо, що тоді ми отримали розв'язок $x = \dfrac{\ln y - \ln x}{2}$, яке зводиться до вигляду $y = xe^{2x}$. Знайдемо графічний розв'язок методом Рунге–Кутти на відрізку $x \in [1, 3]$, крок $h = 0.1$.

```
(%i8) rk6:rk((y/x)*(1+log(y)-log(x)),y,%e^2,[x,1,3,0.1]);
```

(%o8) *list*

```
(%i9) plot2d([x*%e^(2*x),[discrete,rk6]],[x,1,3],[style,[lines,3,1],
        [points,3,2,1]],[legend,false]);
```

(%o9) $[C : /Users/orreg/maxout.gnuplot]$

Графіки розв'язків зображені на Рис. 8.1,(б), і знову ми бачимо, що вони збігаються. Розглянемо дещо складніший випадок.

▶ Розв'язати задачу Коші для рівняння $y' = \dfrac{1-3x-3y}{1+x+y}, y(-5) = 8$. Зобразити прямий та числовий розв'язок цього рівняння методом Рунге–Кутти.

Спочатку розв'яжемо це рівняння з врахуванням початкових умов.

```
(%i1)  load(contrib_ode);
(%i2)  deqn6:'diff(y,x)=(1-3*x-3*y)/(1+x+y);
```

$$(\%o2) \quad \frac{d}{d\,x}\cdot y = \frac{-3\cdot y - 3\cdot x + 1}{y + x + 1}$$

```
(%i3)  contrib_ode(deqn6,y,x);
```

$$(\%o3) \quad [\frac{2\cdot\log\left(y+x-1\right)+y+3\cdot x}{2} = \%c]$$

```
(%i4)  ic1((%o3),x=-5,y=8);
```

$$(\%o4) \quad [\frac{2\cdot\log\left(y+x-1\right)+y+3\cdot x}{2} = \frac{2\cdot\log\left(2\right)-7}{2}]$$

Як бачимо, розв'язок ми отримали у вигляді співвідношення, яке відносно y розв'язати неможливо. Щоб зобразити графік, необхідно звернутись до команди для неявно заданих функцій.

```
(%i5)  load(implicit_plot);
(%i6)  implicit_plot(%o4,[x,-5,5],[y,-10,10], [nticks,200]);
```

rat: replaced 0.6931471805599453 by 13614799/19642003 = 0.693147180559946

(%o6) *done*

Результат зображений на Рис. 8.2(a).

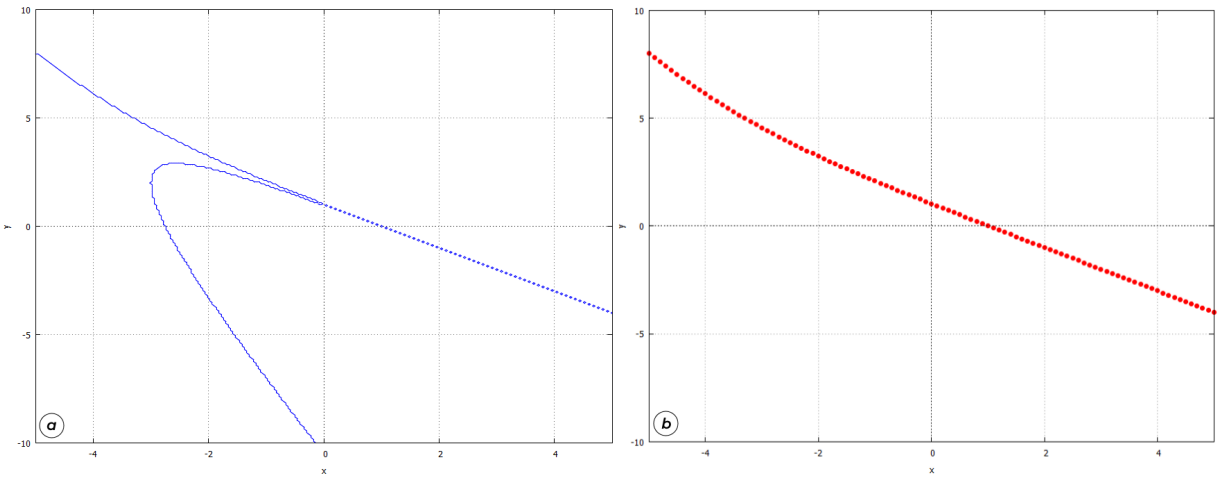


Рис. 8.2: Порівняння аналітичного та числового розв'язків.

Звернемось тепер до числового методу. Побудуємо інтегральну криву, обчислену за методом Рунге–Кутти.

```
(%i7)  load(dynamics);
(%i8)  sol2:rk((1-3*x-3*y)/(1+x+y),y,8,[x,-5,5,0.1]);
```

(%o9) *list*

```
(%i9)  plot2d([discrete,sol2],[x,-5,5],[y,-10,10],[style,
              [points,2,2,1]],[legend,false]);
```

(%o9) $[C : /Users/orreg/maxout.gnuplot]$

Результат зображений на Рис. 8.2(б). З порівняння графіків видно, що команда `rk` добре опрацювала верхню частину розв'язків, яка фактично збігається з аналітично визначеною, але нижня частина зовсім не опрацьована, її розв'язки «випали з уваги» Maxima.

► Розв'яжемо автономну систему рівнянь

$$\begin{cases} \frac{dx}{dt} = 4 - x^2 - 4y^2; \\ \frac{dy}{dt} = y^2 - x^2 + 1; \end{cases} \qquad t \in [0;4]; x(0) = -1,25; y(0) = 0,75.$$

Виберемо крок інтегрування 0,02.

```
(%i10) sol:rk([4-x^2-4*y^2,y^2-x^2+1],[x,y],[-1.25,0.75],[t,0,4,0.02]);
```

(%o8) *list*

Для побудови графіка перетворимо отриманий список, створивши окремо список значень t (`tlist`), x (`xlist`), y (`ylist`). Потім побудуємо разом два графіки: $x(t)$ та $y(t)$, та заразом відобразимо фазову траєкторію $y(x)$.

```
(%i11) tlist:makelist(sol[k][1],k,1,length(sol))$
       xlist:makelist(sol[k][2],k,1,length(sol))$
       ylist:makelist(sol[k][3],k,1,length(sol))$
(%i12) plot2d([[discrete,tlist,xlist],[discrete,tlist,ylist]],
              [legend,false]);
```

(%o12) $[C : /Users/orreg/maxout.gnuplot]$

```
(%i13) plot2d([discrete,xlist,ylist],
              [legend,false]);
```

(%o13) $[C : /Users/orreg/maxout.gnuplot]$

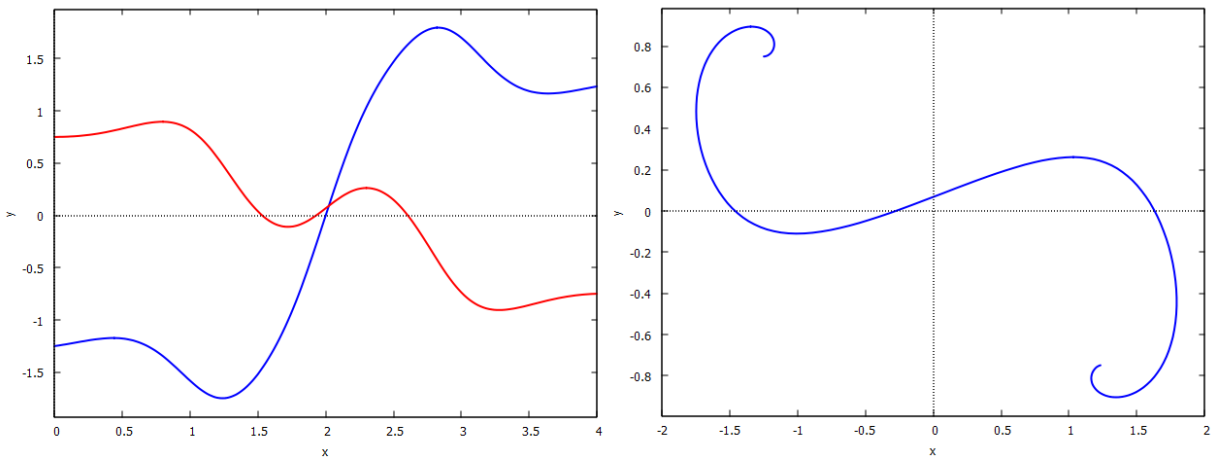


Рис. 8.3: Графіки розв'язку параметричної системи рівнянь та її фазова траєкторія.

Графік зображений на Рис. 8.3.

8.5.2 Метод Рунге–Кутти–Фельберга 4-5 порядку `rkf45`

Щоб скористатись цим методом, необхідно завантажити відповідний пакет командою `load(rkf45)`. Синтаксис наступний:

`rkf45(right_deqn,y,y0,[x,x0,x1],opts)` — для окремого рівняння;

`rkf45([right_deqn],[x,y],[x0,y0],[t,t0,t1],opts)` — для системи рівнянь.

Тут `right_deqn` — права частина ДР або їх список (нагадаємо, що рівняння повинно бути записане у вигляді $\frac{dy}{dx} = f(x,y)$), `y` або `[x,y]` — залежна змінна або їх список, `y0` або `[x0,y0]` — початкові значення залежної змінної, `[x,x0,x1]` або `[t,t0,t1]` — оголошення незалежної змінної, її початкове та кінцеве значення, `opts` — додаткові опції.

Розглянемо опції `opts` докладніше. Їх значення записуються у форматі `opt=flag`.

`full_solution`: значення `true` повертає список всіх точок інтегрування, обраних алгоритмом. За замовчуванням стоїть у положенні `false` — тоді видаються лише фінальні результуючі точки.

`absolute_tolerance`: задає верхню межу абсолютної похибки, за замовчуванням стоїть значення 10^{-6}. Зменшення цього числа, наприклад заданням `absolute_tolerance=10e-8`, збільшить кількість отриманих точок інтегрування.

`max_iterations`: максимальна кількість ітерацій (за замовчуванням 10^4).

`h_start`: задає початковий крок інтегрування (за замовчуванням $1/100$ від заданого інтервалу).

`report`: значення `true` задає повернення звіту про обчислення.

Як видно, на відміну від команди `rk`, у функції `rkf45` відсутнє задання кроку інтегрування. Розмір цього кроку обчислюється автоматично, в залежності від досягненняя заданої абсолютної похибки: це змінна величина, що безпосередньо відображається у різній кількості точок на різних проміжках заданого інтервалу, яка в свою чергу є наслідком звернення до точності 4-го чи 5-го порядку у схемі Рунге–Кутти–Фельберга.

Продемонструємо ці відмінності на прикладі.

▶ Розв'язати диференціальне рівняння безпосередньо, методом `rk`, та методом `rkf45`:

$$\frac{dy}{dx} = -3xy^2 + \frac{1}{x^3+1}; \quad y(0) = 0.$$

```
(%i1)  load(contrib_ode);
(%i2)  deqn2:'diff(y,x)=-3*x*y^2+1/(x^3+1);
```

$$(\%o2) \quad \frac{d}{d\,x} \cdot y = \frac{1}{x^3+1} - 3 \cdot x \cdot y^2$$

```
(%i4)  sol2:contrib_ode(deqn2,y,x)$ sol3:ic1(sol2,x=0,y=0);
```

$$(\%o4) \quad [y = \frac{x}{x^3+1}]$$

```
(%i5)  f:rhs(sol3[1]);
```

$$(\%o5) \quad \frac{x}{x^3+1}$$

Отримали точний розв'язок.

```
(%i6)  right_deqn2:rhs(deqn2);
```

$$(\%o6) \quad \frac{1}{x^3+1} - 3 \cdot x \cdot y^2$$

```
(%i7)  load(dynamics);
(%i8)  sol4:rk(right_deqn2,y,0,[x,0,5,0.1]);
```

$$(\%o8) \quad list$$

Отримали розв'язок за допомогою `rk`.

```
(%i9)  load(rkf45);
(%i10) sol5:rkf45(right_deqn2,y,0,[x,0,5]);
```

(%o10) *list*

Отримали розв'язок за допомогою `rkf45`.
Зобразимо тепер на точній інтегральній кривій ці результати (Рис. 8.4).

```
(%i11) plot2d([f,[discrete,sol4]],[x,0,5],[style,[lines,2,1],
              [points,3,2,1]],[legend,false]);
(%i12) plot2d([f,[discrete,sol5]],[x,0,5],[style,[lines,2,1],
              [points,3,2,1]],[legend,false]);
```

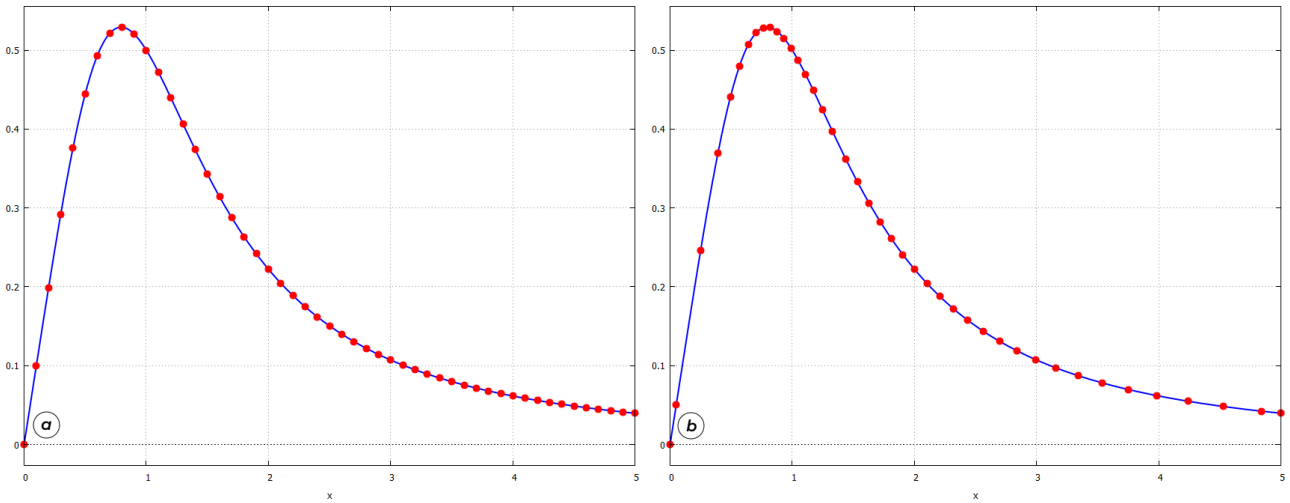


Рис. 8.4: Результати інтегрування ДР за методом `rk` (а) та `rkf45` (б).

З порівняння графіків чітко видно різницю між двома методами: у `rk` всі проміжки по змінній x між точками однакові, тоді як у `rkf45` присутня різна густина точок — поблизу екстремуму їх стає більше, тоді як на пологих місцях менше. Ця різниця відіграє велику роль при дослідженні так званих систем з пороговим ефектом — тобто таких кривих, у яких з'являється значний ступінчастий підйом при незначній зміні малого параметра.

▶ Розглянемо рівняння

$$\frac{dg}{dt} = s - 1.51g + 3.03\frac{g^2}{g^2+1}; \quad g(0) = 0; \quad t \in [0, 100].$$

Це математична модель біохімічного механізму, що має назву «генетичний перемикач». Здавалося б, у цьому рівнянні від сталого параметра s мало що залежить. Однак найменша зміна його від значення $s = 0.202$ до $s = 0.206$ докорінно змінює поведінку системи: з'являється пороговий ефект, який і лежить в природі цього «перемикача». Розв'яжемо задане рівняння з цими значеннями параметрів.

```
(%i1)  right_deqn2:0.202-1.51*g+3.03*(g^2)/(g^2+1);
```

(%o1) $$\frac{3.03 \cdot g^2}{g^2+1} - 1.51 \cdot g + 0.202$$

```
(%i2)  load(rkf45);
(%i3)  sol202:rkf45(right_deqn2,g,0,[t,0,100]);
```

(%o3) *list*

```
(%i4)  right_deqn3:0.206-1.51*g+3.03*(g^2)/(g^2+1);
```

$$(\%o4) \quad \frac{3.03 \cdot g^2}{g^2+1} - 1.51 \cdot g + 0.206$$

```
(%i5)  sol206:rkf45(right_deqn3,g,0,[t,0,100]);
```

$$(\%o5) \quad list$$

```
(%i6)  plot2d([[discrete,sol202],[discrete,sol206]],
              [t,0,100],[style,[lines,2,1],[lines,2,2]],
              [legend,"s=0.202","s=0.206"]);
```

$$(\%o6) \quad [C : /Users/orreg/maxout.gnuplot]$$

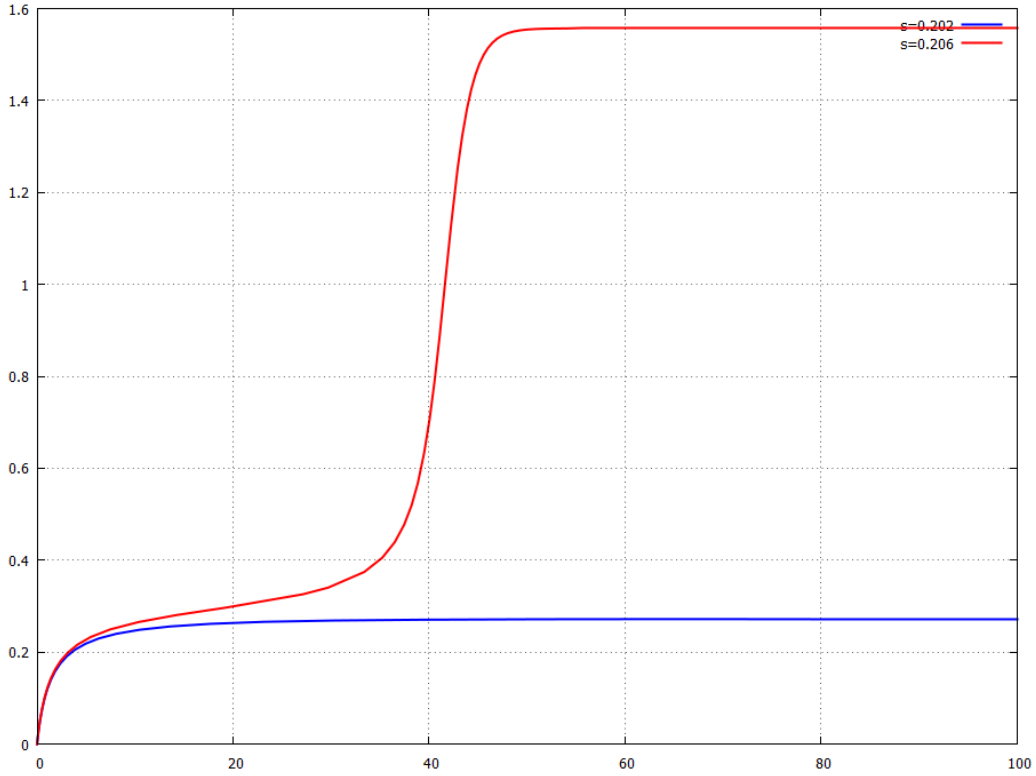


Рис. 8.5: Дослідження порогового ефекту.

При викликанні докладного звіту `report=true` система доповість, що у випадку $s = 0.206$ було змінено та переобчислено 14 поганих кроків (bad steps). Тобто на етапі обрахунку на порогових відрізках **Maxima** виявила, що порядку 4 методу Рунге–Кутти буде недостатньо для заданої точності, та переобрахувала ці точки, збільшивши їх кількість та відповідно звернувшись до 5-го порядку. Цю обставину видно навіть із кількості обчислених точок: для $s = 0.202$ їх 24, для $s = 0.206$ їх 62.

З огляду на таку розумну поведінку алгоритму можна констатувати, що метод `rkf45` є значно надійнішим при розв'язанні ДР числовими методами, аніж метод `rk`.

Зобразимо наочно кількість та розміщення точок інтегрування обох методів для системи з пороговим ефектом (Рис. 8.6). Видно, що на «цікавий» інтервал сходинки `rk` витратив приблизно 10 точок, тоді як у `rkf45` на це пішло до 50 точок.

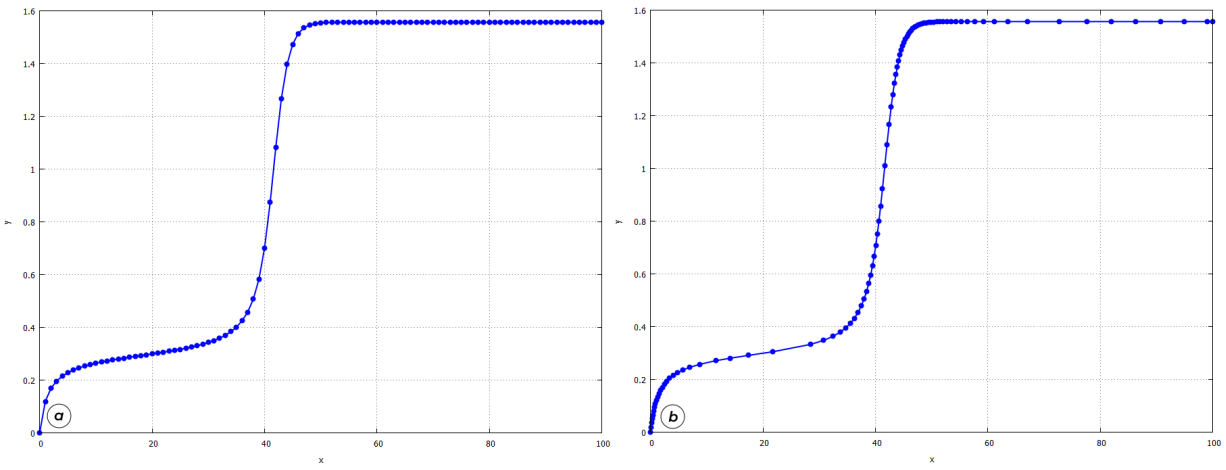


Рис. 8.6: Розміщення точок інтегрування для методу `rk` (а) та `rkf45` (б), загальна кількість точок однакова, $n = 100$.

8.6 Завдання до Розділу 8

Розв'язати задачу Коші для диференціальних рівнянь

8.1. $y'' - y' + x^2 = \sin x; \quad y(0) = 1, y'(0) = 4.$

8.2. $y'' - y' \cdot x^2 = 0; \quad y(0) = 1, y'(0) = 2.$

8.3. $xy'' - 0.3y' + x = 0; \quad y(0) = 2, y'(0) = -2.$

8.4. $y'' - 0.1y' + e^x y = 0; \quad y(0) = 4, y'(0) = 1.$

8.5. $y'' + 3y' - 4y = e^{-4x}; \quad y(0) = 6, y'(0) = 2.$

8.6. $y'' + y = x^3; \quad y(0) = 4, y'(0) = -1.$

8.7. $y'' - y + \cos x = x^2; \quad y(0) = 1, y'(0) = 3.$

8.8. $y'' + y' + y = \sin^2 x; \quad y(0) = 0, y'(0) = 6.$

8.9. $y'' + 5y' - y = e^x; \quad y(0) = 1, y'(0) = 1.$

8.10. $y'' + (2x + 3)y' = 0; \quad y(0) = 2, y'(0) = 1.$

Розв'язати рівняння за методом Рунге-Кутти. Розв'язок представити графічно

8.11. $y' + xy \sin 5x = e^{0.1x^2}; \quad x \in [0, 4], y(0) = 1, h = 0.01.$

8.12. $y' + y^4 = 1 - x; \quad x \in [0, 1], y(0) = 0, h = 0.01.$

8.13. $y' + e^{0.1x} y = \sin xy; \quad x \in [0, 10], y(0) = 1, h = 0.01.$

8.14. $y' + xy \cos 3x = e^{0.3x}; \quad x \in [0, 4], y(0) = 2, h = 0.01.$

8.15. $y' + x^y = \sin 2x; \quad x \in [0, 5], y(0) = 1, h = 0.01.$

8.16. $y' + y^4 \ln(x^2 + 5) = 1 - x^5 y; \quad x \in [0, 2.5], y(0) = 3, h = 0.01.$

8.17. $y' \lg y^3 = y \sin(x^2); \quad x \in [-1, 2], y(-1) = 20, h = 0.01.$

8.18. $y' + x^4 \sqrt[4]{y^5} = y \cos(x + 5); \quad x \in [-1, 4], y(-1) = 15, h = 0.01.$

8.19. $y' + xy \cos(4x^2) = e^{0.1x}; \quad x \in [0, 4], y(0) = 1, h = 0.01.$

8.20. $y' + y\sqrt[3]{x^2} = \cos 2x \ln x; \quad x \in [1, 9], y(1) = 2, h = 0.01.$

Знайти розв'язок крайової задачі

8.21. $y'' - 3y' + 2y = x\cos x; \quad y(0) = 1, y(2) = 4.$

8.22. $y'' + 4y' + 3y = x^2 e^x; \quad y(1) = 4, y(4) = 10.$

8.23. $y'' + 5y' - y = x\cos x; \quad y(0) = 1, y(5) = 2.$

8.24. $y'' - 7y' + 12y = e^x \cos x; \quad y(0) = 6, y(4) = 10.$

Знайти розв'язок системи рівнянь

8.25. $\begin{cases} x'(t) = 2x + y; \\ y'(t) = 3x + 4y. \end{cases}$

8.26. $\begin{cases} x'(t) = x - y; \\ y'(t) = y - 4x. \end{cases}$

8.27. $\begin{cases} x'(t) = 8y - x; \\ y'(t) = x + y. \end{cases}$

8.28. $\begin{cases} x'(t) = y + 2e^t; \\ y'(t) = x + t^2. \end{cases}$

8.29. $\begin{cases} x'(t) = y - 5\cos t; \\ y'(t) = 2x + y. \end{cases}$

8.30. $\begin{cases} x'(t) = 3x + 2y + 4e^{5t}; \\ y'(t) = x + 2y. \end{cases}$

Розділ 9

Інтегральне числення

9.1 Неозначені інтеграли

Хоч процес знаходження інтегралу може розглядатись як протилежний до диференціювання, на практиці знаходження інтегралів є набагато складнішим за знаходження похідних. У **Maxima** за інтегрування відповідає основна команда `integrate`:

`integrate(f(x),x)` — шукає неозначений інтеграл за змінною x;

`integrate(f(x),x,a,b)` — шукає означений інтеграл в межах $x \in [a,b]$.

Для цього оператора присутня також відкладена форма із апострофом `'integrate`, яка буває потрібна, коли підінтегральний вираз залежить від параметрів, що поки не обчислені. Значення параметрів може бути застосоване командою `ev(iexpr,a=A,b=B...)`. `integrate` може задавати питання для уточнення вигляду параметрів. Найперше він звертає увагу на функцію `assume` (припустити), якщо ж таких умов немає, то виникають питання, на які можна відповідати `yes` (так), `no` (ні), `pos` (додатній), `neg` (від'ємний), `zero` (нульовий).

► Знайти інтеграл $\int \sin^3 x dx$.

```
(%i1)  integrate(sin(x)^3,x);
```

$$(\%o1) \quad \frac{\cos(x)^3}{3} - \cos(x)$$

Як бачимо, **Maxima** виводить лише змістовну частину інтегрування, константу C необхідно додавати самостійно.

► Знайти інтеграл $\int x(b^2 - x^2)^{-1/2} dx$.

```
(%i1)  integrate(x/sqrt(b^2-x^2),x);
```

$$(\%o1) \quad -\sqrt{b^2 - x^2}$$

У програмі **Maxima** існує спеціальна команда, яка використовує формулу інтегрування частинами

$$\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx.$$

Щоб використати її, необхідно завантажити додатковий пакет `load(antid)`. Синтаксис команди:

`antidiff(expr,x,v(x))` — тут `expr` підінтегральний вираз, x змінна інтегрування, $v(x)$ функція, яка звільняється від диференціювання.

```
(%i1)  load(antid);
(%i2)  A:u(x)*diff(v(x),x);
```

$$(\%o2) \quad \mathrm{u}(x) \cdot \left(\frac{d}{d\,x} \cdot \mathrm{v}(x) \right)$$

```
(%i3)  antidiff(A,x,v(x));
```

$$(\%o3)\quad \mathrm{u}\left(x\right)\cdot\mathrm{v}\left(x\right)-\int\mathrm{v}\left(x\right)\cdot\left(\frac{d}{d\,x}\cdot\mathrm{u}\left(x\right)\right)dx$$

```
(%i4)  expr1:exp(z(x))*diff(z(x),x)*sin(x);
```

$$(\%o4)\quad e^{\mathrm{z}\left(x\right)}\cdot\sin\left(x\right)\cdot\left(\frac{d}{d\,x}\cdot\mathrm{z}\left(x\right)\right)$$

```
(%i5)  antidiff(expr1,x,z(x));
```

$$(\%o5)\quad e^{\mathrm{z}\left(x\right)}\cdot\sin\left(x\right)-\int e^{\mathrm{z}\left(x\right)}\cdot\cos\left(x\right)dx$$

Ще одна корисна функція — заміна змінних у інтегральному виразі, вона здійснюється за допомогою команди

`changevar(iexpr,f(x,t),t,x)` — де `iexpr` інтеграл у відкладеному вигляді, $f(x,t)$ функція, яка пов'язує стару і нову змінні у форматі $f(x,t)=0$, t нова змінна, x стара змінна.

```
(%i6)  B:'integrate(x*exp(-x^2),x);
```

$$(\%o6)\quad \int x\cdot e^{-x^2}dx$$

```
(%i7)  changevar(B,x^2-z,z,x);
```

$$(\%o7)\quad \frac{\int e^{-z}dz}{2}$$

```
(%i8)  C:'integrate(sin(x)^5*cos(x),x);
```

$$(\%o8)\quad \int\cos\left(x\right)\cdot\sin\left(x\right)^5dx$$

```
(%i9)  changevar(C,sin(x)-t,t,x);
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

$$(\%o9)\quad \int t^5dt$$

Інтегрування тригонометричних та логарифмічних виразів:

```
(%i1)  integrate(sin(x)*sin(2*x)*sin(3*x),x);
```

$$(\%o1)\quad \frac{\cos\left(6\cdot x\right)}{24}-\frac{\cos\left(4\cdot x\right)}{16}-\frac{\cos\left(2\cdot x\right)}{8}$$

```
(%i2)  integrate(1/cos(x)^3,x);
```

$$(\%o2)\quad \frac{\log\left(\sin\left(x\right)+1\right)}{4}-\frac{\log\left(\sin\left(x\right)-1\right)}{4}-\frac{\sin\left(x\right)}{2\cdot\sin\left(x\right)^2-2}$$

```
(%i3)  integrate(x^3*log(x),x);
```

$$(\%o3)\quad \frac{x^4\cdot\log\left(x\right)}{4}-\frac{x^4}{16}$$

При інтегруванні дробово-раціональних виразів першою метою є представлення виразу у вигляді елементарних дробів, які утворюються із знаменника розбиттям останнього на множники. Тут корисною є команда `partfrac`, яка здійснює таку операцію.

▶ Знайти інтеграл $\int -\dfrac{xdx}{x^3+4x^2+5x+2}$.

```
(%i4)  K:-x/(x^3+4*x^2+5*x+2);
```

$$(\%o4) \quad -\frac{x}{x^3+4\cdot x^2+5\cdot x+2}$$

```
(%i5)  partfrac(K,x);
```

$$(\%o5) \quad \frac{2}{x+2}-\frac{2}{x+1}+\frac{1}{(x+1)^2}$$

```
(%i6)  integrate(%o5,x);
```

$$(\%o6) \quad 2\cdot\log(x+2)-2\cdot\log(x+1)-\frac{1}{x+1}$$

```
(%i7)  integrate(K,x);
```

$$(\%o7) \quad 2\cdot\log(x+2)-2\cdot\log(x+1)-\frac{1}{x+1}$$

Якщо у знаменнику складний вираз вищих степенів, у нагоді для розкладу його на множники також стануть команди `gfactor`, `allroots` та подібні.

9.2 Означені інтеграли

У синтаксисі виклику необхідно вказувати межі, виділені комами. Якщо інтеграл невласний, межі позначаються `inf` $(+\infty)$, `minf` $(-\infty)$.

▶ Знайти інтеграл $\int_0^{+\infty}\dfrac{dx}{1+x^2}$.

```
(%i1)  integrate(1/(1+x^2),x,0,inf);
```

$$(\%o1) \quad \frac{\pi}{2}$$

▶ Знайти інтеграл $\int_0^{4\pi} x^2\sin^3 x dx$.

```
(%i2)  integrate(sin(x)^3*x^2,x,0,4*%pi);
```

$$(\%o2) \quad -\frac{288\cdot\pi^2-40}{27}-\frac{40}{27}$$

9.2.1 Спеціальні функції інтегрування

При інтегруванні деяких функцій часто зустрічалися випадки, коли неозначений інтеграл неможливо знайти в явному вигляді, однак технічні та наукові потреби вимагали роботу саме з такими виразами. Такі інтеграли були добре вивчені, досліджені та протабульовані, а самі вирази отримали назву спеціальних інтегральних функцій.

Знайдемо у **Maxima** наступний інтеграл: $\int_{-\infty}^{+\infty} x^2e^{-x^2}dx$

```
(%i1)  integrate (x^2*exp(-x^2),x,minf,inf);
```

$$(\%o1) \quad \frac{\sqrt{\pi}}{2}$$

Тепер знайдемо такий же, але неозначений:

```
(%i2)  integrate (x^2*exp(-x^2),x);
```

$$(\%o2) \quad \frac{\sqrt{\pi}\cdot\operatorname{erf}(x)}{4}-\frac{x\cdot e^{-x^2}}{2}$$

Бачимо, що у виразі з'явилась функція `erf(x)` — це добре відомий «інтеграл помилок» Гаусса

$$\texttt{erf(t)} = \frac{2}{\sqrt{\pi}} \int_0^t e^{-z^2} dz.$$

Ще декілька спеціальних функцій, що можуть виникати при інтегруванні:

$$\texttt{gamma(z)} = \int_0^{+\infty} t^{z-1} e^{-t} dt \qquad \Gamma - \text{функція Ейлера;}$$

$$\texttt{gamma\_incomplete(a,z)} = \int_z^{+\infty} t^{a-1} e^{-t} dt \qquad \text{неповна } \Gamma - \text{функція Ейлера;}$$

$$\texttt{beta(r,s)} = \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)} \qquad \text{B} - \text{функція;}$$

$$\texttt{zeta(s)} = \sum_{n=1}^{\infty} \frac{1}{n^s} \qquad \zeta - \text{функція Рімана;}$$

$$\texttt{elliptic\_f(phi,m)} = \int_0^{\phi} \frac{dx}{\sqrt{1 - m\sin^2 x}} \qquad \text{еліптичний інтеграл першого роду;}$$

$$\texttt{elliptic\_m(phi,m)} = \int_0^{\phi} \sqrt{1 - m\sin^2 x} dx \qquad \text{еліптичний інтеграл другого роду.}$$

Існує ще багато спеціальних функцій, які виникають при знаходженні інтегралів чи розв'язанні диференціальних рівнянь, зацікавленому читачеві варто звернутись до досить обширної літератури з цього питання.

9.2.2 Інтегрування з `defint` та `ldefint`

У знаходженні означених інтегралів існують допоміжні команди `defint` та `ldefint`. Синтаксис задання аналогічний до `integrate`.

Команда `defint(f(x),x,a,b` — шукає інтеграл звичним для нас способом, тобто знаходить спочатку неозначений інтеграл, потім підставляє в нього межі (як не дивно, `integrate` діє іншим, відомим лише розробникам, чином).

Команда `ldefint(f(x),x,a,b` — шукає інтеграл за процедурою `defint`, але при підставлянні меж знаходить границі $\lim_{x\to a}$ та $\lim_{x\to b}$. Якщо проінтегрований вираз «гарний», то результат не буде відрізнятись від `defint` чи `integrate`, однак при розбіжному інтегралі чи безмежних межах виведення результату буде в іншому вигляді. Коли для розбіжного інтегралу `integrate` просто проконстатує цей факт, `ldefint` покаже корисну інформацію, яким саме чином інтеграл буде розбігатись.

► Знайти інтеграл $\int_0^{+\infty} x^{-3} dx$.

```
(%i3)  integrate(1/x^3,x,0,inf);
defint: integral is divergent.
```

— an error. To debug this try: debugmode(true);

```
(%i4)  ldefint(1/x^3,x,0,inf);
```

$$(\%o4) \qquad \frac{\lim_{x\to 0} \frac{1}{x^2}}{2}$$

► Знайти інтеграл $\int_0^{+\infty} \ln x dx$.

```
(%i5)  integrate(log(x),x,0,inf);
defint: integral is divergent.
```

— an error. To debug this try: debugmode(true);

```
(%i6)  ldefint(log(x),x,0,inf);
```

$$(\%o6) \qquad \lim_{x\to\infty} x\cdot\log(x)-x$$

9.2.3 Інтеграли, що залежать від параметра

Задамо **Maxima** наступний інтеграл: $\int_{-\infty}^{+\infty}\frac{dx}{b^2+x^2}$.

```
(%i1)  integrate(1/(b^2+x^2),x,0,inf);
Is b zero or nonzero? nonzero;
Is b positive or negative? pos;
```

$$(\%o1) \qquad \frac{\pi}{2\cdot b}$$

Як бачимо, нам довелось давати відповіді на питання системи щодо вигляду параметра b, оскільки від цього залежить результат виведення. Наприклад, інтеграл $\int_0^{+\infty} e^{-ax}dx$ при $a>0$ збіжний, при $a\le 0$ — розбіжний.

```
(%i2)  integrate(%e^(-a*x),x,0,inf);
Is a positive, negative or zero? pos;
```

$$(\%o2) \qquad \frac{1}{a}$$

```
(%i3)  integrate(%e^(-a*x),x,0,inf);
Is a positive, negative or zero? neg;
Is e^a-1 positive, negative or zero? neg
defint: integral is divergent.
```

— an error. To debug this try: debugmode(true);

Щоб не відповідати на додаткові питання, на параметри можна накласти умови. Це здійснюється командою `assume(ineq)`, де `ineq` — відповідна нерівність.

```
(%i1)  assume(a>1);
```

$$(\%o1) \qquad [a>1]$$

```
(%i4)  integrate(x^a/(x+1)^(5/2),x,0,inf);
Is a an integer? no;
Is 2a-1 positive, negative or zero? neg;
```

$$(\%o4) \qquad \text{beta}\left(\frac{3}{2}-a,a+1\right)$$

Скасувати умову дозволяє команда `forget(ineq)`.

9.3 Площа між двома кривими

Площа поверхні між двома кривими $f(x)$ та $g(x)$: $S = \int_a^b (f(x) - g(x))dx$, де a та b точки їх перетину. Щоб знайти ці точки, потрібно розв'язати рівняння $f(x) = g(x)$. Вимога додатності площі зумовлює те, що $f(x)$ має знаходитись вище за $g(x)$, і за цією обставиною необхідно уважно слідкувати, оскільки на проміжку $[a, b]$ точок перетину може бути кілька. В задачах такого типу найперше потрібно зобразити графіки кривих.

► Знайти площу між кривими $f(x) = \sqrt{x}$ та $g(x) = x^{3/2}$.

Зобразимо ці криві (Рис. 9.1). Видно, що $f(x)$ знаходиться вище за $g(x)$.

```
(%i1)  plot2d([sqrt(x),x^(3/2)], [x,0,1.2],[style,[lines,2,1],
           [lines,2,2]],[plot_format, gnuplot]);
```

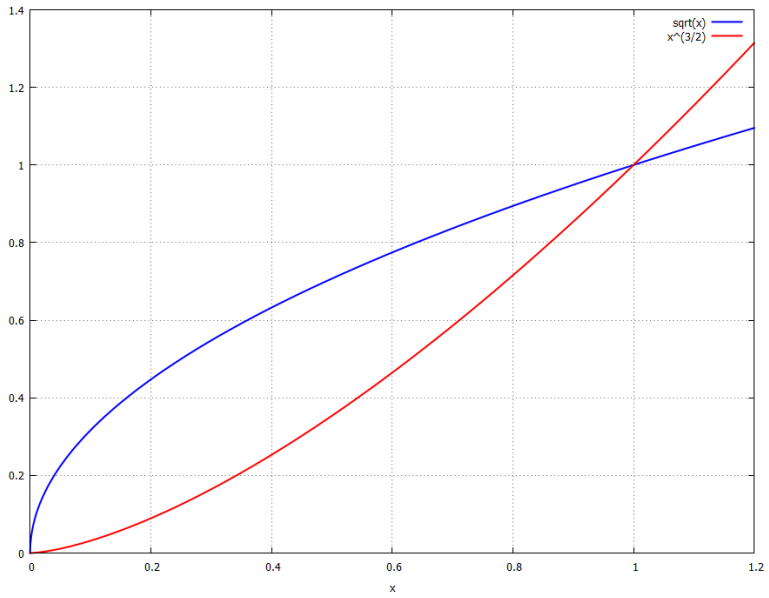


Рис. 9.1: Графіки кривих $f(x)$ та $g(x)$.

Точки перетину:

```
(%i2)  f(x):=sqrt(x);
```

(%o2) $\mathrm{f}(x) := \sqrt{x}$

```
(%i3)  g(x):=x^(3/2);
```

(%o3) $\mathrm{g}(x) := x^{\frac{3}{2}}$

```
(%i4)  solve(f(x)=g(x),x);
```

(%o4) $[x = 0, x = 1]$

Площа поверхні:

```
(%i5)  integrate(f(x)-g(x),x,0,1);
```

(%o5) $\frac{4}{15}$

► Знайти площу між кривими $f(x) = \frac{3}{10}x^5 - 3x^4 + 11x^3 - 18x^2 + 12x + 1$ та $g(x) = -4x^3 + 28x^2 - 56x + 32$.

Зобразимо ці криві (Рис. 9.2). Видно, що $f(x)$ та $g(x)$ мають дві точки перетину, і положення однієї кривої відносно іншої змінюється.

```
(%i1)  f(x):=(3/10)*x^5-3*x^4+11*x^3-18*x^2+12*x+1;
```

$$(\%o1) \quad \mathrm{f}(x) := \frac{3 \cdot x^5}{10} - 3 \cdot x^4 + 11 \cdot x^3 - 18 \cdot x^2 + 12 \cdot x + 1$$

```
(%i2)  g(x):=-4*x^3+28*x^2-56*x+32;
```

$$(\%o2) \quad \mathrm{g}(x) := -4 \cdot x^3 + 28 \cdot x^2 - 56 \cdot x + 32$$

```
(%i3)  plot2d([f(x),g(x)],[x,-1,5],[y,-10,10],
              [style,[lines,2,1],[lines,2,2]],
              [legend,"f(x)","g(x)"],[axes,solid]);
```

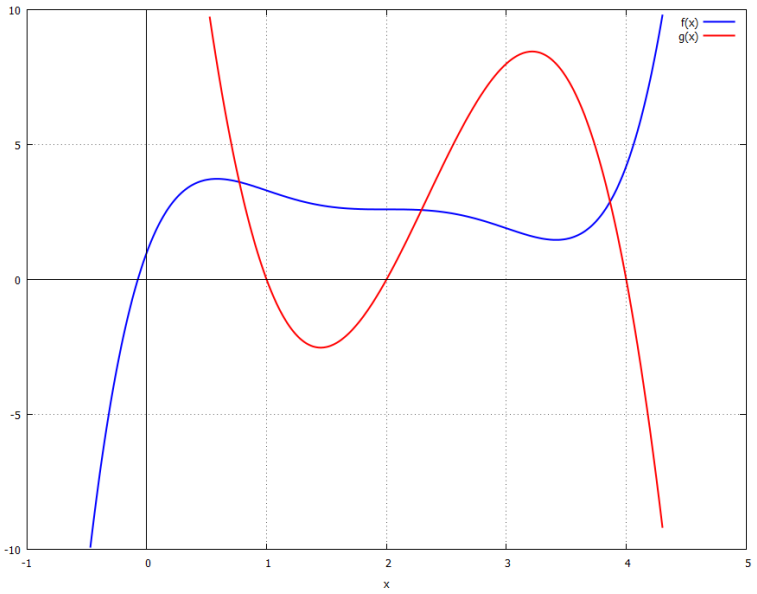


Рис. 9.2: Площа між кривими $f(x)$ та $g(x)$.

Щоб знайти точки перетину, розв'яжемо рівняння $f(x) = g(x)$.

```
(%i4)  solve(f(x)=g(x),x);
```

$$(\%o4) \quad [0 = 3 \cdot x^5 - 30 \cdot x^4 + 150 \cdot x^3 - 460 \cdot x^2 + 680 \cdot x - 310]$$

```
(%i5)  h(x):=3*x^5-30*x^4+150*x^3-460*x^2+680*x-310;
```

$$(\%o5) \quad \mathrm{h}(x) := 3 \cdot x^5 - 30 \cdot x^4 + 150 \cdot x^3 - 460 \cdot x^2 + 680 \cdot x - 310$$

Як бачимо, утворилось рівняння 5-го степеня, яке нерозв'язне в радикалах. Але **Maxima** дозволяє знайти числові розв'язки. Із графіка видно, що точки перетину містяться в проміжках $[0, 1]$, $[2, 3]$, та $[3, 4]$. Ця інформація дозволяє нам застосувати команду `find_root`.

```
(%i6)  x1:find_root(h(x),x,0,1);
```

$$(\%o6) \quad 0.772058304527811$$

```
(%i7)  x2:find_root(h(x),x,2,3);
```

$$(\%o7) \quad 2.291819210962955$$

```
(%i8)  x3:find_root(h(x),x,3,4);
```

(%o8) 3.865127100061793

```
(%i9)  [y1,y2,y3]:map(f,[x1,x2,x3]);
```

(%o9) [3.613992056691179, 2.575784006305817, 2.882949345140559]

Тепер, нарешті, можемо обчислити площу S, попередньо виконавши команду `ratprint:false`, щоб **Maxima** не намагалась перетворити вирази з плаваючою комою у раціональні:

```
(%i10) ratprint:false;
```

(%o10) false

```
(%i11) S:integrate(f(x)-g(x),x,x1,x2)+integrate(g(x)-f(x),x,x2,x3);
```

(%o11) $$\frac{169540440064218273521}{13902320084620535442}$$

```
(%i12) S,numer;
```

(%o12) 12.19511844298367

9.4 Повторні інтеграли

У **Maxima** можливо знайти повторні інтеграли якої завгодно розмірності, використовуючи вкладення команди `integrate` одна в одну. Звісно, в першу чергу нас будуть цікавити подвійні та потрійні інтеграли. Припустимо, необхідно знайти інтеграл по області V:

$$\iiint_V F(x,y,z)dV = \int_{x_1}^{x_2}\left[\int_{y_1(x)}^{y_2(x)}\left(\int_{z_1(x,y)}^{z_2(x,y)} F(x,y,z)dz\right)dy\right]dx.$$

На мові **Maxima** реалізація такого інтегралу наступна:
`integrate(integrate(integrate(F(x,y,z),z,z1(x,y),z2(x,y)),y,y1(x),y2(x)),x,x1,x2)`.

Дослідниками також розроблені додаткові пакети для обчислень криволінійних та поверхневих інтегралів різних видів, зокрема `Math214` (докладніше з цим можна ознайомитись на сайті The Maximalist). Можливості цього пакету досить широкі: знаходження площ чи об'ємів складних поверхонь, кривини та довжини різноманітних кривих, операції векторного аналізу з використанням теорем Гаусса та Стокса.

► Знайти інтеграл
$\iiint_V (x^2+2y^2+3z^2)dV$, де $V:\{-\sqrt{x^2+y^2}\le z\le\sqrt{x^2+y^2};\ -x\le y\le x;\ 1\le x\le 2\}$.

```
(%i1)  integrate(integrate(integrate(x^2+2*y^2+3*z^2,z,
                                     -sqrt(x^2+y^2),sqrt(x^2+y^2)),
                           y,-(x),(x)),x,1,2);
```

Is x `positive, negative or zero? pos;`

(%o1) $$\frac{31\cdot\left(5\cdot\operatorname{asinh}(1)+17\cdot\sqrt{2}\right)}{10}$$

9.4.1 Площа еліпса

Рівняння еліпса має вигляд $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$. Покладемо $a>b$ та виразимо x через y у першому квадранті: $x=\frac{a}{b}\sqrt{b^2-y^2}$. Тоді чверть площі еліпса має вигляд

$$\frac{S}{4}=\iint_D dxdy=\int_0^b\left(\int_0^{\frac{a}{b}\sqrt{b^2-y^2}}dx\right)dy.$$

Запрограмуємо задачу на мові **Maxima**, використавши вкладення одного інтегралу в інший для реалізації повторного інтегралу:

```
(%i13) s1:solve((x/a)^2+(y/b)^2=1,x);
```

$$(\%o13)\quad [x=-\frac{a\cdot\sqrt{b^2-y^2}}{b},x=\frac{a\cdot\sqrt{b^2-y^2}}{b}]$$

```
(%i14) x1:rhs(s1[2]);
```

$$(\%o14)\quad \frac{a\cdot\sqrt{b^2-y^2}}{b}$$

```
(%i15) integrate(integrate(1,x,0,x1),y,0,b);
```

$$(\%o15)\quad \frac{\pi\cdot a\cdot b}{4}$$

Звідси загальна площа еліпса $S=\pi ab$.

9.4.2 Момент інерції еліпсоїда

Нехай маємо однорідний еліпсоїд обертання з півосями a,b,c:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1;\quad -a\le x\le a;\quad -b\le y\le b;\quad -c\le z\le c.$$

Кожен елемент об'єму $dV=dxdydz$ має масу $dm=\rho dV$. Загальна маса об'єкту m, загальний об'єм $V=\frac{4}{3}\pi abc$. Обчислимо момент інерції еліпсоїда відносно осі z, що шукається за формулою

$$I_z=\iiint_V(x^2+y^2)dm=\rho\iiint_V(x^2+y^2)dxdydz.$$

Запишемо межі інтегрування, які знаходяться із проєктування поверхні на відповідні осі та площини:

$$\begin{cases}-a\le x\le a;\\ -b\sqrt{1-(x/a)^2}\le y\le b\sqrt{1-(x/a)^2};\\ -c\sqrt{1-(x/a)^2-(x/b)^2}\le z\le c\sqrt{1-(x/a)^2-(x/b)^2}.\end{cases}$$

Тоді задача на мові **Maxima** виглядатиме наступним чином:

```
(%i1)  F(x,y,z):=x^2+y^2;
```

$$(\%o1)\quad \mathrm{F}\,(x,y,z):=x^2+y^2$$

```
(%i2)  V:(4/3)*%pi*a*b*c;
```

$$(\%o2)\quad \frac{4\cdot\pi\cdot a\cdot b\cdot c}{3}$$

```
(%i3)  %rho:m/V;
```

$$(\%o3)\quad \frac{3\cdot m}{4\cdot\pi\cdot a\cdot b\cdot c}$$

```
(%i4)  x1:-a;
```

$$(\%o4)\quad -a$$

```
(%i5)  x2:a;
```

$$(\%o5)\quad a$$

```
(%i6)  y1(x):=-b*sqrt(1-(x/a)^2);
```

$$(\%o6)\quad \mathrm{y1}\,(x) := -b\cdot\sqrt{1-\left(\frac{x}{a}\right)^2}$$

```
(%i7)  y2(x):=b*sqrt(1-(x/a)^2);
```

$$(\%o7)\quad \mathrm{y2}\,(x) := b\cdot\sqrt{1-\left(\frac{x}{a}\right)^2}$$

```
(%i8)  z1(x,y):=-c*sqrt(1-(x/a)^2-(y/b)^2);
```

$$(\%o8)\quad \mathrm{z1}\,(x,y) := -c\cdot\sqrt{1-\left(\frac{x}{a}\right)^2-\left(\frac{y}{b}\right)^2}$$

```
(%i9)  z2(x,y):=c*sqrt(1-(x/a)^2-(y/b)^2);
```

$$(\%o9)\quad \mathrm{z2}\,(x,y) := c\cdot\sqrt{1-\left(\frac{x}{a}\right)^2-\left(\frac{y}{b}\right)^2}$$

```
(%i10) Iz:%rho*integrate(integrate(integrate
       (F(x,y,z),z,z1(x,y),z2(x,y)),y,y1(x),y2(x)),x,x1,x2);
Is a positive, negative or zero? pos;
Is b positive, negative or zero? pos;
Is (x-a)(x+a) positive, negative or zero? neg;
```

$$(\%o10)\quad \frac{(8\cdot\pi\cdot a^5\cdot b^3+8\cdot\pi\cdot a^7\cdot b)\cdot m}{40\cdot\pi\cdot a^5\cdot b}$$

```
(%i11) rat(Iz);
```

$$(\%o11)/\mathrm{R}/\quad \frac{(b^2+a^2)\cdot m}{5}$$

Бачимо, що отриманий результат збігається із класичним, одержаним аналітично «на папері».

9.5 Числове інтегрування

За наявності складних випадків, або при інтегруванні узагальнених чи спеціальних функцій, **Maxima** не може відобразити результат інтегрування у вигляді елементарних функцій або числа. Тоді доводиться інтегрувати числовими методами. **Maxima** містить два основних способи числового інтегрування: команди сімейства `quadpack` та метод `romberg`.

`quadpack` має довгу та шановану історію, інтегрування цим методом було написане на мові FORTRAN ще 1973 року. Згодом код був адаптований та включений до **Maxima**. Команди цього сімейства: `quad_qag`, `quad_qags`, `quad_qagi`, `quad_qawc`, `quad_qawf`, `quad_qawo`, `quad_qagp`. Кожна команда створена для цілком окремого випадку функцій з певними властивостями, тому вміле застосування їх приносить максимальний результат.

Ми розглянемо дві команди (з іншими читач може ознайомитись у Довідці **Maxima**):

`quad_qags(f(x),x,a,b,[opts])` — числово шукає інтеграл $f(x)$ на обмеженому проміжку $[a,b]$. Необов'язкові команди `[opts]` регулюють величину відносної помилки апроксимації (за замовчуванням 10^{-8}), число підінтервалів розбиття (за замовчуванням 200).

`quad_qagi(f(x),x,a,b,[opts])` — числово шукає інтеграл $f(x)$ на проміжку $[a,b]$, одне зі значень якого (a чи b) може бути нескінченність.

▶ Числово обчислити інтеграл $\int_0^3 xe^{\sqrt{x}}dx$.

```
(%i1)  quad_qags(x*exp(sqrt(x)),x,0,3);
```

(%o1) $[18.6521959729504, 1.474278886087793 \cdot 10^{-7}, 147, 0]$

У відповіді з'явився список з чотирьох елементів: власне саме значення інтегралу, відносна похибка обчислення, число проміжних інтегральних значень, код помилки (0 означає проблем в інтегруванні не було).

Цікаво порівняти результат з командою `integrate`:

```
(%i2)  integrate(x*exp(sqrt(x)),x,0,3),numer;
```

(%o2) 18.65219597319475

Різниця між двома значеннями $2.443591995415772 \cdot 10^{-10}$.

▶ Числово знайти інтеграл $\int_0^{+\infty} x^3e^{-3x}$.

```
(%i3)  quad_qagi (x^3*exp(-3*x), x, 0, inf);
```

(%o3) $[0.07407407407407407, 1.419030764062393 \cdot 10^{-10}, 105, 0]$

Порівняємо з `integrate`:

```
(%i4)  integrate(x^3*exp(-3*x), x, 0, inf);
```

(%o4) $\frac{2}{27}$

```
(%i5)  2/27,numer;
```

(%o5) 0.07407407407407407

Як бачимо, знову збіг двох значень дуже точний.

Ще одна команда для числового інтегрування, що використовує метод Ромберга: `romberg(f(x),x,a,b)`.

Точність цього способу задається глобальними змінними `rombergabs` — абсолютна похибка апроксимації, та `rombergtol` — відносна похибка апроксимації. Команда `rombergit` задає кількість поділів навпіл початкового інтервалу під час інтегрування, тобто якщо прийняти `rombergit:10`, кількість проміжків стає $2^{10} = 1024$.

▶ Знайти інтеграл за методом Ромберга $\int_3^{10} \left(\frac{1}{(x-1)^2+0.01} + \frac{1}{(x-2)^3+0.001}\right) dx$.

```
(%i25) f(x):=1/((x-1)^2+1/100)+1/((x-2)^3+1/1000);
```

(%o25) $\mathrm{f}\left(x\right) := \frac{1}{\left(x-1\right)^2+\frac{1}{100}} + \frac{1}{\left(x-2\right)^3+\frac{1}{1000}}$

```
(%i26) rombergtol:1e-6;
```
(%o26) $1.0 \cdot 10^{-6}$

```
(%i27) rombergit:10;
```
(%o27) 10

```
(%i28) int1:romberg(f(x),x,3,10);
```
(%o28) 0.8804650492843361

```
(%i31) int2:integrate(f(x),x,3,10),numer;
```
(%o31) 0.8804650492560029

```
(%i32) int1-int2;
```
(%o32) $2.833322465534138 \cdot 10^{-11}$

Точність цього методу теж дуже висока.

9.6 Завдання до Розділу 9

В поданих інтегралах здійснити заміну змінної, після чого обчислити інтеграл по новій змінній

9.1. $\int \frac{2}{x^2-2x-5}dx; \quad x-1 \to z$

9.2. $\int xe^{-\sqrt{x}}dx; \quad \sqrt{x} \to t$

9.3. $\int x^2 \ln x dx; \quad \ln x \to y$

9.4. $\int \sin^2 x \cos^2 x dx; \quad \sin x \to z$

9.5. $\int \frac{e^x}{x}dx; \quad e^x \to t$

9.6. $\int x^2 \sin\sqrt{x}dx; \quad \sqrt{x} \to y$

9.7. $\int \frac{dx}{5x^3-14x^2+13x-4}; \quad x-1 \to z$

9.8. $\int \frac{dx}{x\sqrt{\ln x}}; \quad 1/x \to t$

9.9. $\int \sqrt{x}e^{-x^2}dx; \quad \sqrt{x} \to y$

9.10. $\int \mathrm{sh}^2 x \,\mathrm{ch}^3 x dx; \quad \mathrm{ch}\, x \to z$

Обчислити площу між заданими кривими

9.11. $f(x)=\frac{2}{x}; \quad g(x)=\frac{4}{x^2}; \quad x \in [1,4]$

9.12. $f(x)=x^5; \quad g(x)=x^3;$

9.13. $f(x)=4\sin x;$
$g(x)=\frac{1}{20}(x+6)(x+2)(x-1)(x+5);$

9.14. $f(x)=x;$
$g(x)=x+\sin^2 x; \quad$ один цикл

9.15. $f(x)=5\cos x;$
$g(x)=x^4-x^3+2x; \quad x \in [-2,2]$

9.16. $f(x)=x^3-x-4; \quad g(x)=x^4-20;$

9.17. $f(x)=3\sin^3 x;$
$g(x)=8\cos^3 x; \quad$ один цикл

9.18. $x-y-1=0; \quad y^2=2x+1;$

9.19. $f(x)=e^{\sqrt{x}}; \quad g(x)=5\sin x;$

9.20. $f(x)=\cos\frac{1}{x}; \quad g(x)=x;$
остання замкнена фігура.

Знайти числово інтеграли за методом `quad_pack` та методом `romberg`, порівняти їх значення

9.21. $\int_0^3 e^{-x^3} dx;$

9.22. $\int_1^2 \sin(x^3) dx;$

9.23. $\int_2^5 \frac{\sin x}{x} dx;$

9.24. $\int_0^1 \sin(\cos x) dx;$

9.25. $\int_0^4 \sqrt{1+\sin^2 x} dx;$

9.26. $\int_1^5 \frac{1}{\sqrt{1+\sin^2 x}} dx;$

9.27. $\int_1^3 \sin \sqrt[3]{x} dx;$

9.28. $\int_1^{10} \sin(\ln x) dx;$

9.29. $\int_1^7 \frac{1}{x^5 - 3x^3 + 22} dx;$

9.30. $\int_1^2 \frac{x}{\cos x} dx;$

Розділ 10

Коливні системи

Коливний рух дуже поширений у фізиці. Рівняннями коливань описуються різноманітні періодичні явища, хвильові процеси, хаотичні процеси, хімічні реакції тощо. Вивчення відповідних рівнянь та їх властивостей є важливим для базового розуміння фізики та суміжних наук.

10.1 Математичний маятник

Розглянемо стандартну модель математичного маятника: точковий вантаж маси m підвішений на нерозтяжній нитці довжини l та відведений від положення рівноваги на початковий кут θ_0, початкова швидкість рівна v_0.

У класичній механіці одним з центральних рівнянь є рівняння Лагранжа другого роду для узагальненої координати q. За цю координату ми оберемо змінний кут θ між ниткою та положенням рівноваги, незалежною змінною є час t.

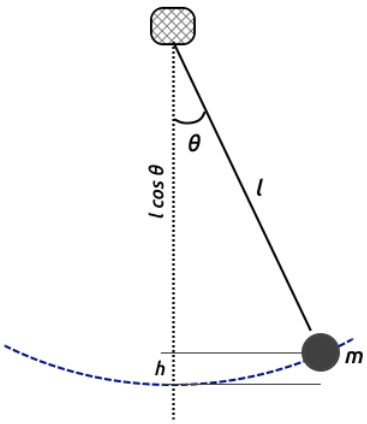


Рис. 10.1: Модель математичного маятника.

Запровадимо функцію Лагранжа

$$L = T - U,$$

де $T = \dfrac{mv^2}{2} = \dfrac{m}{2}(v_0 + l\dot{q})^2$ — кінетична енергія, $U = mgh = mg(l - l\cos q) = mgl(1 - \cos q)$ — потенціальна енергія точки. Тепер можемо записати рівняння Лагранжа, яке буде описувати зміну узагальненої координати від часу:

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0.$$

Запишемо ці вступні дані на мові **Maxima**.

Задаємо залежність узагальненої координати від часу:

```
(%i1)  depends(q,t);
```

(%o1) $[\mathrm{q}(t)]$

Кінетична енергія, у заданні якої використовуємо відкладену похідну `'diff`:

```
(%i2)  T:(m/2)*(v0+l*'diff(q,t))^2;
```

(%o2) $$\frac{m\cdot\left(v0+l\cdot\left(\frac{d}{dt}\cdot q\right)\right)^2}{2}$$

Потенціальна енергія:

```
(%i3)  U:m*g*l*(1-cos(q));
```

$$(\%o3) \quad g \cdot l \cdot m \cdot (1 - \cos(q))$$

Компонуємо лагранжіан системи:

```
(%i4)  Lagr:T-U;
```

$$(\%o4) \quad \frac{m \cdot \left(v0 + l \cdot \left(\frac{d}{dt} \cdot q\right)\right)^2}{2} - g \cdot l \cdot m \cdot (1 - \cos(q))$$

Записуємо рівняння Лагранжа у загальному вигляді, не обчислюючи поки похідних:

```
(%i5)  Deqn1:'diff(Lagr,q)-'diff(('diff(Lagr,'diff(q,t))),t)=0;
```

$$(\%o5) \quad \frac{d}{d\,q} \cdot \left(\frac{m \cdot \left(v0 + l \cdot \left(\frac{d}{dt} \cdot q\right)\right)^2}{2} - g \cdot l \cdot m \cdot (1 - \cos(q))\right)$$
$$- \frac{d^2}{d\left(\frac{d}{dt} \cdot q\right) \cdot d\,t} \cdot \left(\frac{m \cdot \left(v0 + l \cdot \left(\frac{d}{dt} \cdot q\right)\right)^2}{2} - g \cdot l \cdot m \cdot (1 - \cos(q))\right) = 0$$

Тепер отримуємо рівняння руху, задавши команду `nouns` провести всі обчислення у явному вигляді:

```
(%i6)  Deqn2:ev(Deqn1,nouns);
```

$$(\%o6) \quad - l^2 \cdot m \cdot \left(\frac{d^2}{d\,t^2} \cdot q\right) - g \cdot l \cdot m \cdot \sin(q) = 0$$

Зводимо рівняння до спрощеного вигляду, скоротивши його на величину $-ml^2$:

```
(%i7)  Deqn3:ratsimp(Deqn2/(-m*l^2));
```

$$(\%o7) \quad \frac{l \cdot \left(\frac{d^2}{d\,t^2} \cdot q\right) + g \cdot \sin(q)}{l} = 0$$

Зробивши заміну $\frac{g}{l} = \omega^2$, приводимо рівняння руху до звичного у фізиці вигляду:

```
(%i8)  Deqn4:ratsubst(%omega^2,g/l,Deqn3);
```

$$(\%o8) \quad \frac{d^2}{d\,t^2} \cdot q + \omega^2 \cdot \sin(q) = 0$$

Отримане рівняння є ЗДР другого порядку, нелінійним та однорідним. Початкові умови його: $q|_{t=0} = q_0$, $\dot{q}|_{t=0} = v_0/l$. Спробуємо напряму розв'язати його стандартною командою `ode2`.

```
(%i9)  ode2(Deqn4,q,t);
```

$$(\%o9) \quad \left[-\frac{\int \frac{1}{\sqrt{\cos(q) - \%k1}}dq}{\sqrt{2} \cdot \omega} = t + \%k2, \frac{\int \frac{1}{\sqrt{\cos(q) - \%k1}}dq}{\sqrt{2} \cdot \omega} = t + \%k2\right]$$

Як бачимо, система не змогла розв'язати рівняння руху, оскільки для цього необхідно знайти інтеграл виду $\int \frac{1}{\sqrt{\cos q - \%k1}}dq$. За формулою подвійного кута $\cos 2\alpha = 1 - 2\sin^2\alpha$ він зводиться до еліптичного інтегралу першого роду (який не виражається в елементарних функціях), тому така обставина не повинна дивувати.

Розв'яжемо це рівняння числово. Для цього використаємо метод Рунге-Кутти з командою `rk`, або його вдосконалений варіант `rkf45`.

Задамо значення параметрів:

```
(%i10) g:9.81; l:1; %omega:sqrt(g/l);
(%o10)  9.81
(%o11)  1
(%o12)  3.132091952673165
```

Перш ніж використати пакет `rkf45`, зауважимо, що він може застосовуватись лише у випадку ЗДР першого порядку, тоді як у нас другий. Вихід із такого ускладнення наступний: перетворимо рівняння другого порядку у систему двох пов'язаних рівнянь першого порядку запровадженням проміжної функції (кутової швидкості $\dot{q} = \varepsilon$):

$$\begin{cases} \dfrac{dq}{dt} = \varepsilon; \\ \dfrac{d\varepsilon}{dt} = -\omega^2 \sin q. \end{cases}$$

Приймемо за початкові точки, з якими буде інтегруватись рівняння, $q = \pi/6$, $\varepsilon = 1$.

```
(%i13) load(rkf45);
(%i14) sol45:rkf45([%epsilon,-%omega^2*sin(q)],
                   [q,%epsilon],[%pi/6,1],[t,0,10]);
(%o14)  list
```

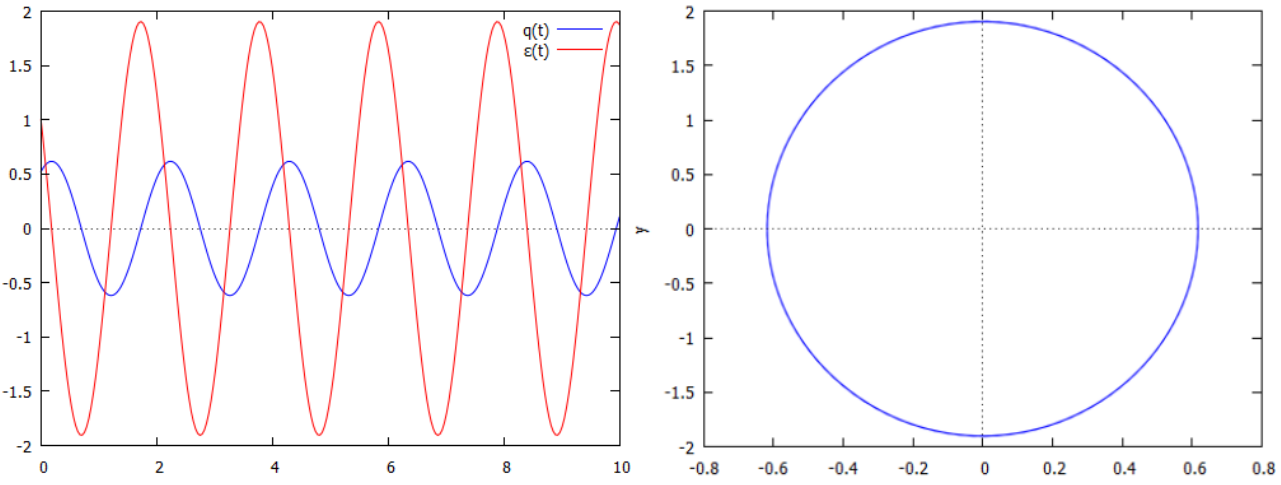


Рис. 10.2: Графіки залежностей $q(t)$ та $\varepsilon(t)$ та фазова діаграма (q, ε) для початкового кута $q_0 = \pi/6$, початкової швидкості $v_0 = 1$.

Отримали список, що складається з великої кількості трійок елементів: незалежна змінна t, дві залежні змінні q та ε. Виділимо окремо кожну з цих величин, сформувавши три окремі списки `t_list`, `q_list`, `%epsilon_list`.

```
(%i15) t_list:makelist(sol45[k][1],k,1,length(sol45))$
       q_list:makelist(sol45[k][2],k,1,length(sol45))$
       %epsilon_list:makelist(sol45[k][3],k,1,length(sol45))$
```

Тепер зобразимо графіки залежності $q(t)$, $\varepsilon(t)$ та графік фазової діаграми руху системи у просторі (q, ε) (Рис.10.2).

```
(%i16) plot2d([[discrete,t_list,q_list],
               [discrete,t_list,%epsilon_list]],
               [legend,"q(t)","%epsilon(t)"]);
(%i17) plot2d([[discrete,q_list,%epsilon_list]],[legend,false]);
```

Розглянемо отримані результати. Видно, що графіки для q та $\dot{q} = \varepsilon$ знаходяться чітко у протифазі: максимуми та мінімуми однієї функції припадають на нулі іншої. Це узгоджується з механікою коливального процесу: при максимальному відхиленні кута швидкість кульки (як кутова, так і лінійна) буде рівна нулю, при нульовому відхиленні швидкість руху буде максимальною.

При погляді на створені зображення можна інтерпретувати їх як дві синусоїдні лінії та еліпс (не коло через різний масштаб на осях). Це помилкове враження, хоч і дуже близьке до істини. Дослідимо причини такого співпадіння. Як ми бачили раніше, при прямому розв'язку рівняння руху виникли інтеграли виду $\int \frac{1}{\sqrt{\cos q - \%k1}} dq$. Здійснимо розклад функції $\cos q$ в ряд поблизу точки рівноваги $q = 0$.

```
(%i18) taylor(cos(q),q,0,4);
```

$$(\%o18)/T/1 - \frac{q^2}{2} + \frac{q^4}{24} + \ldots$$

Виберемо для наближення перші два члени розкладу $\cos q = 1 - q^2/2$ та підставимо в цей інтеграл.

```
(%i19) integrate(1/sqrt(1-q^2-%k1),q);
Is %k1-1 positive or negative?neg;
```

$$(\%o19) \quad \operatorname{asin}\left(\frac{q}{\sqrt{1-\%k1}}\right)$$

Звідси бачимо, що справді залежність $q(t)$ є синусоїдною, а взявши похідну по t, залежність $\varepsilon(t)$ виявляється косинусоїдною, відповідно фазова діаграма у параметричному представленні стає еліпсом. Справедливість таких висновків буде залежати від величини обраного початкового кута, він має бути якомога ближчий до нуля. Прийнято вважати, що кута q_0 меншого за $45° = \pi/4$ достатньо для того, щоб розглядати коливання як гармонічні.

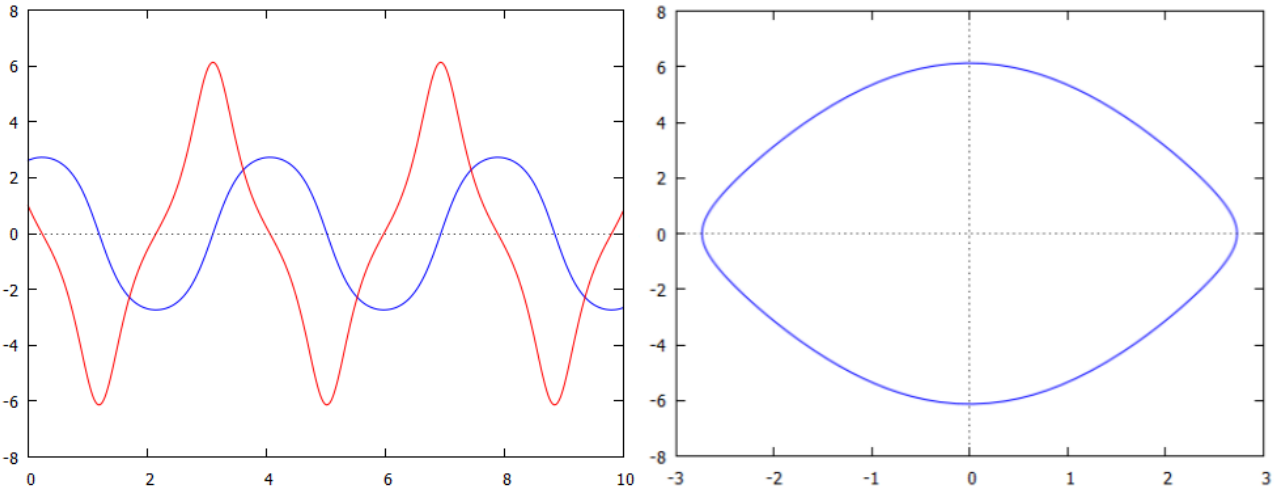


Рис. 10.3: Графіки $q(t)$, $\varepsilon(t)$ та фазова діаграми для початкового кута $q_0 = \pi/1.05$.

Покладемо для прикладу дуже великий кут відхилення, близький до π, (зокрема $q = \pi/1.05$) та задамо його у числовий розв'язок рівняння коливань. Тоді система видасть графіки, зображені на Рис.10.3.

Як видно, такі коливання вже зовсім не схожі на гармонічні. Фазова діаграма схожа швидше на дві з'єднані параболи, ніж на еліпс.

Досі ми не заторкали вплив початкової швидкості на рух системи, покладаючи її рівною одиниці. Зміна цього фактору в певних межах не дуже сильно впливає на вигляд інтегральних кривих, аж поки він не досягне певного критичного значення. Якщо кінетична енергія переважить потенціальну енргію у найбільш високій точці (тобто на висоті $h = 2l$), кулька зробить повний оберт та буде обертатись завжди, коливальний процес зникне. Розрахуємо цю максимальну швидкість.

```
(%i20) vmax:sqrt(2*g*2*l);
```

(%o20) 6.26418390534633

Для довжини підвісу $l = 1$ числові значення швидкостей v та ε будуть збігатися. Підставивши значення $\varepsilon = 6.5$ у вираз для `sol45`, отримаємо картину на Рис.10.4. Бачимо, що кут $q(t)$ збільшується до нескінченності, швидкість $\varepsilon(t)$ змінюється хвилеподібно. Фазова діаграма (q, ε) вже не є замкненою.

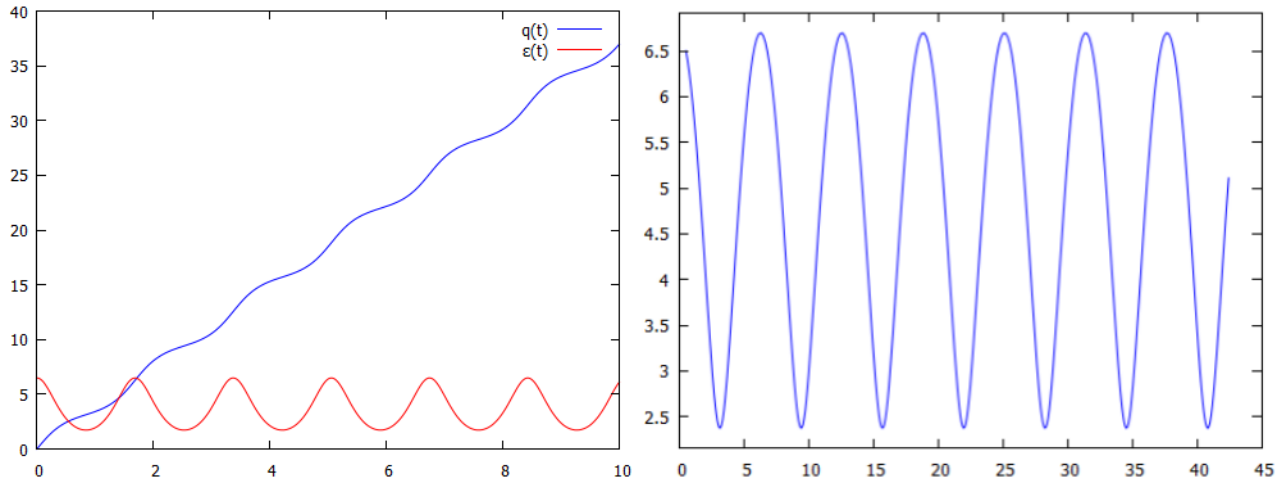


Рис. 10.4: Графіки $q(t)$, $\varepsilon(t)$ та фазова діаграма для швидкості кульки $v_0 = 6.5$ м/с.

Чисті гармонічні коливання виникають, якщо у базовому рівнянні руху `Deqn4` зробити розклад синуса поблизу точки рівноваги та взяти перший значущий член.

```
(%i21) taylor(sin(q),q,0,3);
```

(%o21)/T/ $q - \dfrac{q^3}{6} + \ldots$

Тоді рівняння претворюється на наступне:

```
(%i22) Deqn5:'diff(q,t,2)+%omega^2*q=0;
```

(%o22) $\dfrac{d^2}{dt^2} \cdot q + \omega^2 \cdot q = 0$

Розв'яжемо його із вказаними вище початковими умовами ($q|_{t=0} = q_0$, $\dot{q}|_{t=0} = v_0/l$):

```
(%i23) Sol5:ode2(Deqn5,q,t);
Is %omega zero or nonzero?nonzero;
```

(%o23) $q = \%k1 \cdot e^{i \cdot \omega \cdot t} + \%k2 \cdot e^{-i \cdot \omega \cdot t}$

```
(%i24) ic2(Sol5,t=0,q=q0,'diff(q,t)=v0/l);
```

$$(\%o24) \quad q = -\frac{i \cdot e^{i \cdot \omega \cdot t} \cdot (v0 + i \cdot \omega \cdot l \cdot q0)}{2 \cdot \omega \cdot l} - \frac{i \cdot e^{-i \cdot \omega \cdot t} \cdot (i \cdot \omega \cdot l \cdot q0 - v0)}{2 \cdot \omega \cdot l}$$

Система видала розв'язок у комплексній формі, тому переводимо її у алгебричну, відразу задавши спрощення тригонометричних функцій:

```
(%i25) trigsimp(rectform(%));
```

$$(\%o25) \quad q = \frac{\sin(\omega \cdot t) \cdot v0 + \omega \cdot l \cdot q0 \cdot \cos(\omega \cdot t)}{\omega \cdot l}$$

Розділивши весь вираз на ωl, отримуємо:

```
(%i26) partfrac(%,%omega);
```

$$(\%o26) \quad q = \frac{\sin(\omega \cdot t) \cdot v0}{\omega \cdot l} + q0 \cdot \cos(\omega \cdot t)$$

Це і є повний розв'язок рівняння руху у випадку наближення до коливань на малі кути. Залежність кута та кутової швидкості від часу є синусоїдною, відповідно фазова діаграма руху буде справжнім еліпсом (Рис.10.5).

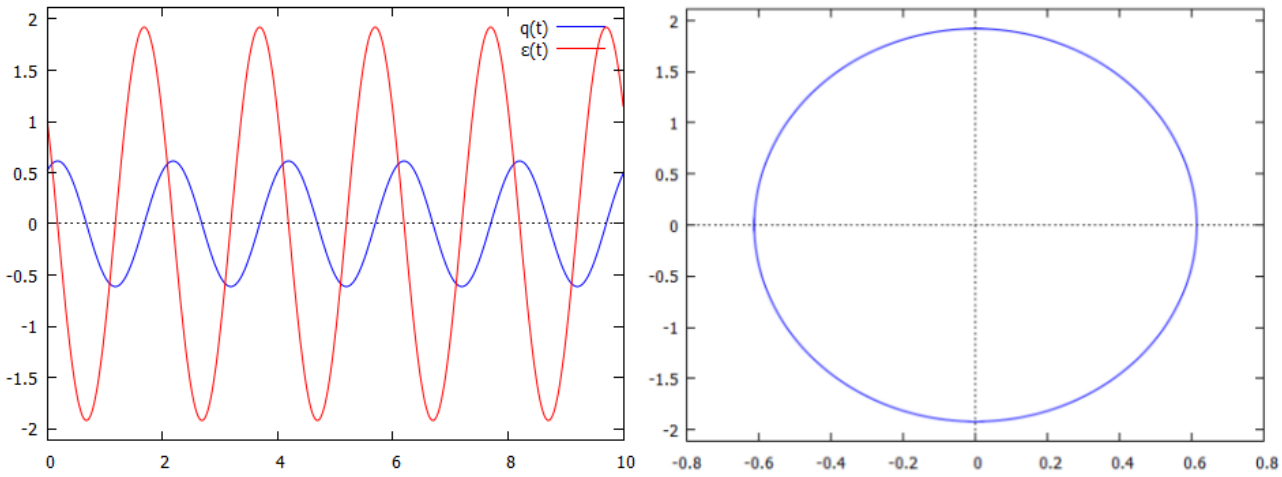


Рис. 10.5: Графіки $q(t)$, $\varepsilon(t)$ та фазова діаграма для гармонічних коливань.

```
(%i27) v0:1;
```

$$(\%o27) \quad 1$$

```
(%i28) q0:%pi/6;
```

$$(\%o28) \quad \frac{\pi}{6}$$

```
(%i29) q(t):=(v0/(%omega*l))*sin(%omega*t)+q0*cos(%omega*t);
```

$$(\%o29) \quad \mathrm{q}(t) := \frac{v0 \cdot \sin(\omega \cdot t)}{\omega \cdot l} + q0 \cdot \cos(\omega \cdot t)$$

```
(%i30) %epsilon(t):=l*diff(q(t),t);
```

$$(\%o30) \quad \varepsilon(t) := l \cdot \mathrm{diff}(\mathrm{q}(t), t)$$

```
(%i31) plot2d([q(t),%epsilon(t)],[t,0,10],[legend,"q(t)","%epsilon(t)"]);
```

```
(%i32) plot2d([parametric,q(t),%epsilon(t),[t,0,10]],[legend,false]);
```

Як видно, ці графіки повністю аналогічні до тих, які були отримані числово з інтегрування початкового рівняння руху при $q_0 = \pi/6$, $v_0 = 1$, і це підтверджує ті міркування, що при малих коливаннях рух системи можна цілком справедливо розглядати як гармонічний.

Зауважимо, що точно таке ж рівняння виникає при розгляді коливань вантажка на пружині, і обмеження, які накладаються при його виведенні, мають аналогічний характер. Закон Гука пропорційності сили пружності до видовження $F = -k\Delta x$ справедливий лише при невеликих зміщеннях Δx, якщо ж пружину видовжити занадто сильно, значну роль почнуть відігравати ефекти непружної деформації.

Розглянемо питання про період коливань нашої системи. Відома «шкільна» формула дає для періоду вираз $T = 2\pi/\omega = 2\pi\sqrt{l/g}$, однак це справедливо лише для чистих гармонічних коливань. Звернемось до розв'язку початкового рівняння руху:

$$-\frac{1}{\sqrt{2}\omega}\int\frac{1}{\sqrt{\cos q - \%k1}}dq = t + \%k2.$$

Тут присутні дві сталі інтегрування, як і годиться у розв'язку ЗДР другого порядку. Покладемо $v_0 = 0$, тоді використавши початкову умову $q(0) = q_0$ та провівши певні елементарні маніпуляції (докладніше див. [4]), можна через половинний кут звести цей інтеграл до вигляду

$$t = -\frac{1}{\sqrt{2}\omega}\int_{q_0}^{q}\frac{dq}{\sqrt{\cos q - \cos q_0}} = -\frac{1}{2\omega}\int_{q_0}^{q}\frac{dq}{\sqrt{\sin^2\frac{q_0}{2} - \sin^2\frac{q}{2}}} =$$

$$= \frac{1}{2\omega}\int_{q}^{q_0}\frac{dq}{\sin\frac{q_0}{2}\sqrt{1 - \dfrac{\sin^2\frac{q}{2}}{\sin^2\frac{q_0}{2}}}}.$$

Здійснивши заміну $\sin x = \dfrac{\sin\frac{q}{2}}{\sin\frac{q_0}{2}}$, та позначивши $k = \sin^2\frac{q_0}{2}$, цей інтеграл зводиться до повного еліптичного інтегралу першого роду $\displaystyle\int_0^{\pi/2}\frac{dx}{\sqrt{1 - k\sin^2 x}} = K(k)$. Розглянемо коливання від початкового відхилення $q = q_0$ до положення рівноваги $q = 0$. За цей час кулька зробить $1/4$ коливання від завершеного періоду, тому:

$$\frac{T}{4} = \frac{1}{\omega}\int_0^{\pi/2}\frac{dx}{\sqrt{1 - \sin^2\frac{q_0}{2}\sin^2 x}}.$$

У системі **Maxima** є вбудована функція для повного еліптичного інтегралу

$$\texttt{elliptic\_kc(k)} = \int_0^{\pi/2}\frac{dx}{\sqrt{1 - k\sin^2 x}}.$$

Це дає нам змогу записати кінцеву формулу для періоду

$$T = \frac{4}{\omega}\cdot\texttt{elliptic\_kc}(\sin^2\tfrac{q_0}{2}).$$

Задамо цю залежність у **Maxima**:

```
(%i33) T(q0):=(4/%omega)*elliptic_kc(sin(q0/2)^2);
```

$$(\%o33)\quad \mathrm{T}\,(q0) := \frac{4\cdot\mathrm{elliptic\_kc}\left(\sin\left(\frac{q0}{2}\right)^2\right)}{\omega}$$

Знайдемо справжній період коливань для кута $q_0 = \pi/6$:

```
(%i34) T(%pi/6),numer;
(%o34)  2.04098988951913
```

Тепер знайдемо період коливань у гармонічному наближенні:

```
(%i35) T_harm:2*%pi/%omega,numer;
(%o35)  2.006066680710647

(%i36) T(%pi/6)-T_harm,numer;
(%o36)  0.03492320880848343
```

Як бачимо, різниця значень становить 0.035 с при періоді 2 с, таке відхилення експериментально можна знайти лише при дуже точних вимірюваннях. Якщо ж задати великий початковий кут $q_0 = \pi/1.05$, значення періоду буде суттєво більшим за гармонічне:

```
(%i37) T(%pi/1.05),numer;
(%o37)  5.08840732181855
```

Для вивчення графіків коливань та фазових діаграм звернемося ще до інструменту `plotdf`, який вбудований у **Maxima**. Знайдемо поля напрямків та інтегральні криві для базового рівняння руху. Так само як у випадку числового інтегрування командою `rkf45`, ми повинні представити його у вигляді системи двох пов'язаних рівнянь для q та ε, початкові значення оберемо $q_0 = \pi/6 \approx 0.5$, $\varepsilon_0 = 4$. Повзунком задамо зміну параметра довжини підвісу $l = 1 \ldots 5$, тоді при сталій початковій швидкості буде певна критична довжина, при якій маятник почне обертатись.

```
(%i38) load(plotdf)$
(%i39) plotdf([%epsilon,-(9.81/l)*sin(q)],[q,%epsilon],
              [trajectory_at,0.5,4],[parameters,"l=3"],
              [tstep,0.05],[nsteps,300],[sliders,"l=1:5"])$
```

Після виконання команди виникає вікно $(q;\varepsilon)$ з напрямками розв'язків диференціального рівняння та червоним виділена траєкторія інтегральної кривої для заданих початкових умов. Клацаючи лівою кнопкою миші всередині даної діаграми на якусь точку, автоматично відбувається інше задання q_0, ε_0 та будується ще одна інтегральна крива. Пересування повзунка знизу змінює параметр l, при його русі вигляд заданої інтегральної кривої змінюється, як показано на Рис.10.6. Зверху справа міститься значок з кривими, при його натисненні виникає ще одне вікно з залежностями $q(t)$, $\varepsilon(t)$.

Видно, що отримані графіки повністю аналогічні тим, які отримані нами раніше за допомогою прямого числового інтегрування, і це підтверджує те, що інструмент `plotdf` є дуже потужним та інформативним. Може виникнути питання, чому ж одразу було не задати цю команду та отримати за секунду ті результати, які виводились нами довгим та складним шляхом. Відповіддю є та обставина, що ці графіки може «читати» лише підготовлена особа, яка пройшла та осягнула всю описану вище теорію.

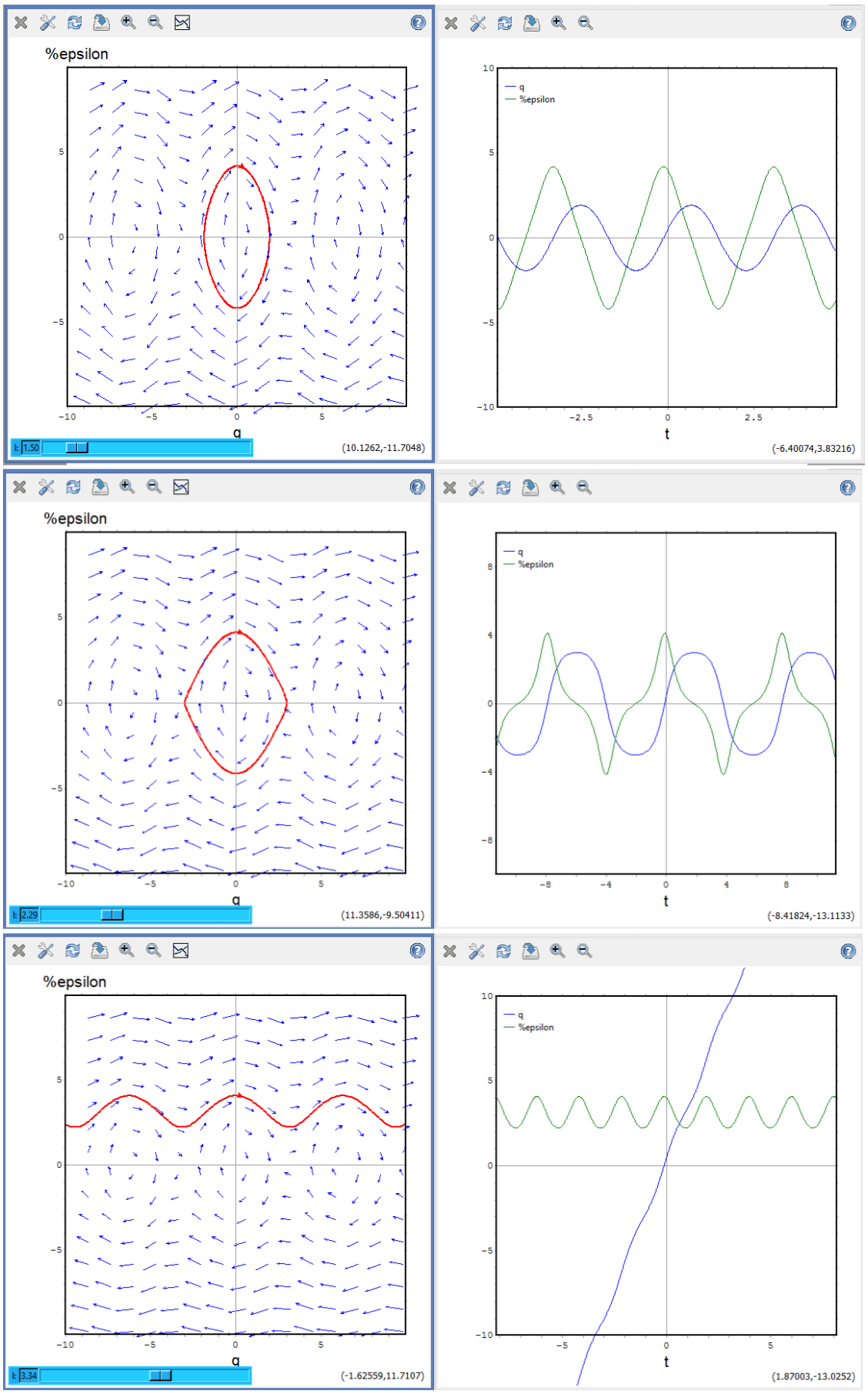


Рис. 10.6: Фазові діаграми та графіки руху для q та ε при різних значенях параметра l.

Розглянемо анімацію вільних гармонічних коливань. Код команди для зображення анімації поданий нижче, із необхідними коментарями. Зауважимо лише, що у заданні координат точки $x = l\cos q$, $y = l\sin q$ необхідно до кута зміщення q додати $-\pi/2$, щоб коливання відбувались у звичній для нас орієнтації.

```
(%i40) /* Загальний розв'язок */
       q(t):=(sin(%omega*t)*v0)/(%omega*l)+q0*cos(%omega*t)$
       /* Початкові умови */
       %omega:1$ v0:0$ q0:%pi/6$ l:1$
       /* Розбиття проміжку коливання на 40 частин */
       lst:makelist((7*%pi/6+i*%pi/60),i,0,40)$
       with_slider_draw(
       /* З'єднання прямого та оберненого списку */
       k,append(lst,reverse(lst)),
       /* Без позначок на осях, кількість точок розрахунку 100 */
       ytics=false,xtics=false,nticks=100,
       /* Задання зображення траєкторії */
       proportional_axes = xy,line_width = 3,line_type  = dashes,
       parametric(l*cos(t),l*sin(t),t,7*%pi/6,11*%pi/6),
       /* Задання верхньої нерухомої точки */
       color=grey,point_type=5,point_size=6,
       points([[0,0]]),
       /* Задання точки коливання */
       color=navy,point_type=filled_circle,point_size=6,
       points([l*cos(-%pi/2+q(k))],[l*sin(-%pi/2+q(k))]),
       /* Задання коливання підвісу */
       line_type=solid,head_length=0.01,color=black,head_angle=1,
       vector([0,0],[l*cos(-%pi/2+q(k)),l*sin(-%pi/2+q(k))]))$
```

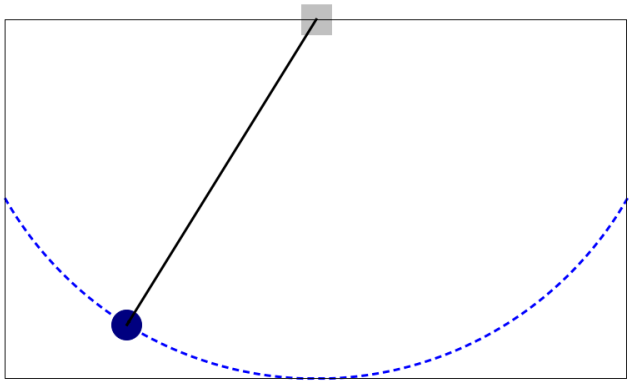


Рис. 10.7: Кадр анімації математичного маятника.

10.2 Маятник із згасанням

При русі маятника у в'язкому середовищі з'являється додаткова сила, що діє в протилежному до руху напрямку. В широкому колі явищ ця сила буде пропорційна до швидкості об'єкта та обумовлює появу доданка в рівнянні руху $bv = bl\dot{q}$ (у випадку пружинного маятника $bk\dot{x}$). Запишемо рівняння руху у вигляді

$$m\frac{d^2}{dt^2}(lq) + b\frac{d}{dt}(lq) + mgq = 0,$$

де кожен доданок має розмірність сили, тоді другий компонент якраз враховує силу тертя. Розділивши на ml, маємо

$$\ddot{q} + \frac{b}{m}\dot{q} + \frac{g}{l}q = 0.$$

Здійснимо стандартну заміну $\omega^2 = g/l$, тоді прийдемо до кінцевого рівняння

$$\frac{d^2q}{dt^2} + \frac{b}{m}\frac{dq}{dt} + \omega^2 q = 0.$$

Його можна розв'язати аналітично. Задання прямого розв'язку у **Maxima** дає нам:

```
(%i1)  Deqn6:'diff(q,t,2)+(b/m)*'diff(q,t)+%omega^2*sin(q)=0;
```

$$(\%o1) \quad \frac{d^2}{d\,t^2}\cdot q + \frac{b\cdot\left(\frac{d}{d\,t}\cdot q\right)}{m} + \omega^2\cdot q = 0$$

```
(%i2)  ode2(Deqn6,q,t);
Is (2·ω·m − b)·(2·ω·m + b)  positive, negative or zero?pos;
```

$$(\%o2) \quad q = e^{-\frac{b\cdot t}{2\cdot m}}\cdot\left(\%k1\cdot\sin\left(\frac{\sqrt{4\cdot\omega^2-\frac{b^2}{m^2}}\cdot t}{2}\right)+\%k2\cdot\cos\left(\frac{\sqrt{4\cdot\omega^2-\frac{b^2}{m^2}}\cdot t}{2}\right)\right)$$

```
(%i3)  ode2(Deqn6,q,t);
Is (2·ω·m − b)·(2·ω·m + b)  positive, negative or zero?zero;
```

$$(\%o3) \quad q = (\%k2\cdot t + \%k1)\cdot e^{-\frac{b\cdot t}{2\cdot m}}$$

```
(%i4)  ode2(Deqn6,q,t);
Is  (2·ω·m − b)·(2·ω·m + b)  positive, negative or zero?neg;
```

$$(\%o4) \quad q = \%k1\cdot e^{\frac{\left(\sqrt{\frac{b^2}{m^2}-4\cdot\omega^2}-\frac{b}{m}\right)\cdot t}{2}} + \%k2\cdot e^{\frac{\left(-\frac{b}{m}-\sqrt{\frac{b^2}{m^2}-4\cdot\omega^2}\right)\cdot t}{2}}$$

Графік першого розв'язку зображений на Рис.10.8, за $m = 1$, $\omega = 1$, $b = 0.2$. При погляді на отримані розв'язки можна бачити, що їхній вигляд залежить від комбінації $4m^2\omega^2 - b^2$, і покладаючи її більшою нуля, нулю та меншою нуля, вирази для $q(t)$ будуть щоразу іншими. Для полегшення сприймання розв'язків здійснимо деякі перетворення вихідного рівняння, а саме перейдемо до нової змінної — безрозмірного часу $\tau = \omega t$. Тоді, розділивши все рівняння на ω^2, отримаємо

$$\frac{d^2q}{d(\omega t)^2} + \frac{b}{m\omega}\frac{dq}{d(\omega t)} + q = 0,$$

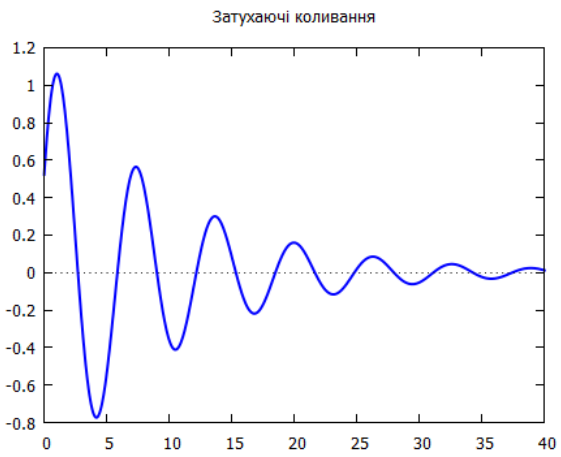


Рис. 10.8: Коливання маятника із згасанням.

звідси, позначивши $b/(m\omega) = 2\gamma$, можна записати безрозмірне рівняння

$$\frac{d^2q}{d\tau^2} + 2\gamma\frac{dq}{d\tau} + q = 0.$$

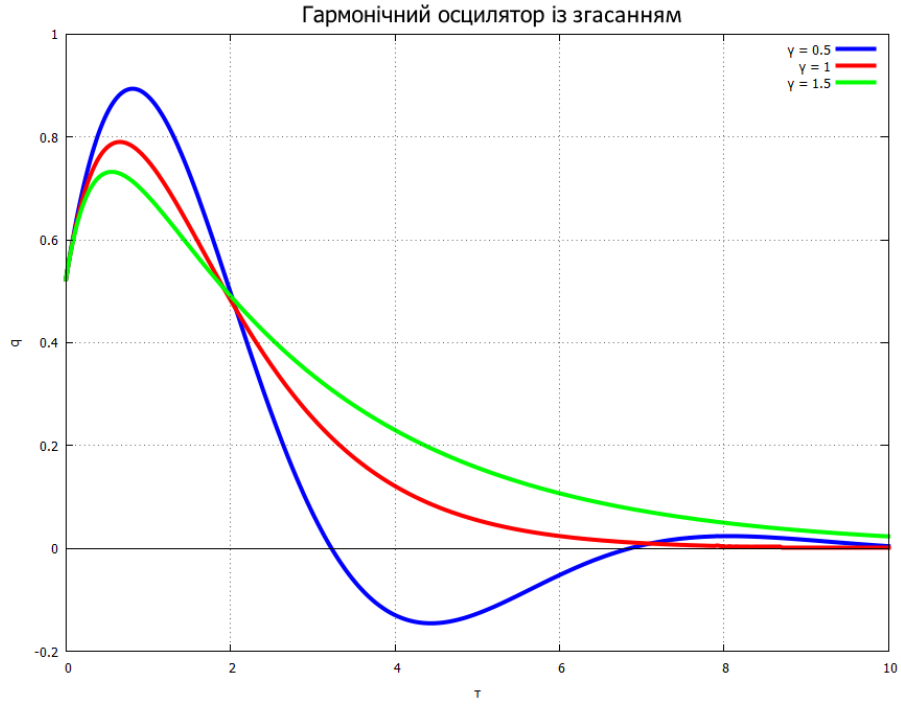


Рис. 10.9: Графіки прямих розв'язків для згасаючих коливань.

Умова для $4m^2\omega^2 - b^2$ переходить в умову $\gamma < 1, \gamma = 1, \gamma > 1$. Покладемо для прикладу $\gamma = 0.5; 1; 1.5$, початкові умови $q = \pi/6$, $\dot{q} = 1$. Знайдемо прямі розв'язки для безрозмірного рівняння руху та побудуємо їх графіки (Рис.10.9).

```
(%i5)  Deqn6a:'diff(q,%tau,2)+2*%gamma*'diff(q,%tau)+q=0;
```

$$(\%o5) \quad \frac{d^2}{d\tau^2}\cdot q + 2\cdot\gamma\cdot\left(\frac{d}{d\tau}\cdot q\right) + q = 0$$

```
(%i6)  for i thru 3 do
       q[i]:rhs(ic2(ode2(subst(%gamma=i/2,Deqn6a),q,%tau),
                    %tau=0,q=%pi/6,'diff(q,%tau)=1))$
(%i7)  for i:1 thru 3 do
       display(q[i]);
```

$$q_1 = e^{-\frac{\tau}{2}}\cdot\left(\frac{\left(\sqrt{3}\cdot\pi + 4\cdot 3^{\frac{3}{2}}\right)\cdot\sin\left(\frac{\sqrt{3}\cdot\tau}{2}\right)}{18} + \frac{\pi\cdot\cos\left(\frac{\sqrt{3}\cdot\tau}{2}\right)}{6}\right)$$

$$q_2 = \left(\frac{(\pi+6)\cdot\tau}{6} + \frac{\pi}{6}\right)\cdot e^{-\tau}$$

$$q_3 = \frac{\left(\left(3\cdot\sqrt{5}+5\right)\cdot\pi + 12\cdot\sqrt{5}\right)\cdot e^{\frac{(\sqrt{5}-3)\cdot\tau}{2}}}{60} - \frac{\left(\left(3\cdot\sqrt{5}-5\right)\cdot\pi + 12\cdot\sqrt{5}\right)\cdot e^{\frac{(-\sqrt{5}-3)\cdot\tau}{2}}}{60}$$

(%o7) *done*

```
(%i8)  plot2d([q[1],q[2],q[3]],[%tau,0,10],
       [style,[lines,4]],[xlabel,"%tau"],[ylabel,"q"],
       [title,"Гармонічний осцилятор із затуханням"],
       [axes,solid],[legend,"%gamma = 0.5","%gamma = 1","%gamma = 1.5"])$
```

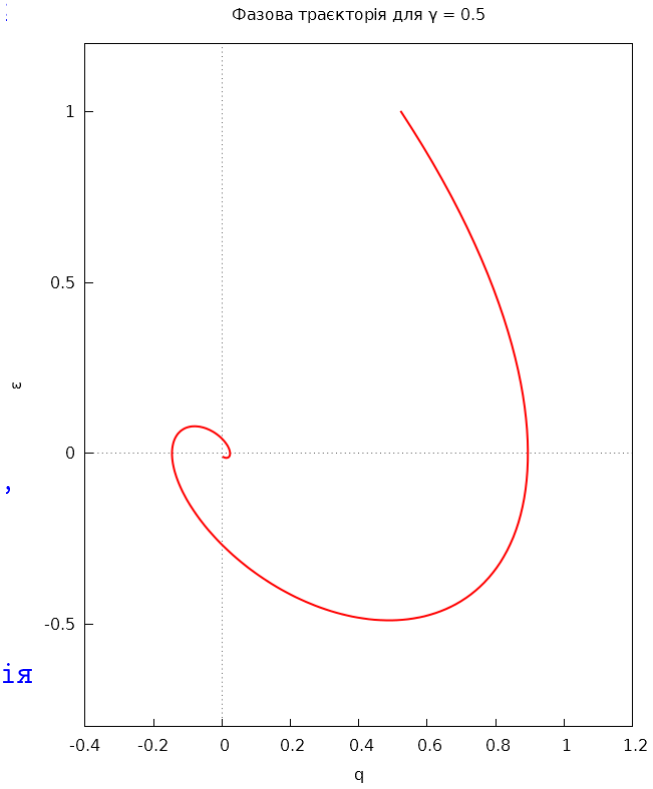
Отримані графіки відображають цікаву обставину: найшвидше до нуля іде крива з $\gamma = 1$. Випадок $\gamma < 1$ означає, що система встигає зробити повне коливання, після чого

повертається до положення рівноваги, $\gamma > 1$ означає, що коливання взагалі не відбувається, і система повільно йде до стану рівноваги. Саме тому інженери при виготовленні демпферів (пристроїв, що гасять шкідливі коливання) підбирають такі їх параметри, при яких $\gamma = 1$.

Зобразимо фазову діаграму для випадку $\gamma = 0.5$:

```
(%i9)  for i:1 thru 3 do
       %epsilon[i]:diff(q[i],%tau);
(%o9)    done

(%i10) load(draw)$
(%i11) wxdraw2d(
          xlabel = "q",
          ylabel = "%epsilon",
          nticks = 300,
          line_width = 2,
          color  = red,
          xrange = [-0.4,1.2],
          yrange = [-0.8,1.2],
       parametric(q[1],%epsilon[1],
                  %tau,0,10),
          xaxis = true,
          yaxis = true,
          proportional_axes = xy,
          title = "Фазова траєкторія
                 для %gamma = 0.5"
       )$
```



Із діаграми видно, що траєкторія руху у фазовому просторі закінчується в центрі. Побудувавши аналогічні графіки для (q_2, ε_2), (q_3, ε_3), можна переконатись, що вони теж ідуть до центральної точки $(0, 0)$, і це цілком зрозуміло, оскільки будь-які коливання із згасанням закінчуються у стані з нульовою швидкістю та нульовим відхиленням.

Повністю аналогічні графіки дає нам і функція `plotdf`. Клацнувши кнопкою миші та додаючи ще інтегральні траєкторії, бачимо, що всі вони закінчуються в центрі.

```
(%i12) load(plotdf);
(%i13) plotdf(
              [%epsilon,-2*0.5*%epsilon-q],[q,%epsilon],
              [trajectory_at,0.5,1],[direction,both],[q,-0.4,1.2],
              [%epsilon,-0.8,1.2],[nsteps,400],[tstep,0.01]
             );
```

Ралізуємо тепер анімацію коливального руху із тертям. Основа коду вже була записана раніше для випадку гармонічних коливань, необхідно лише замінити вираз для $q(t)$ на $q(\tau) = q(\omega t)$ та взяти один з трьох отриманих розв'язків для різних γ. Також потрібно сформувати лише прямий список у розбитті кутів на частини, оскільки зворотнього руху вже не відбуватиметься, коливання затихнуть у положенні рівноваги. Для $\gamma = 1$ код приведений нижче.

```
(%i14) load(draw)$
```

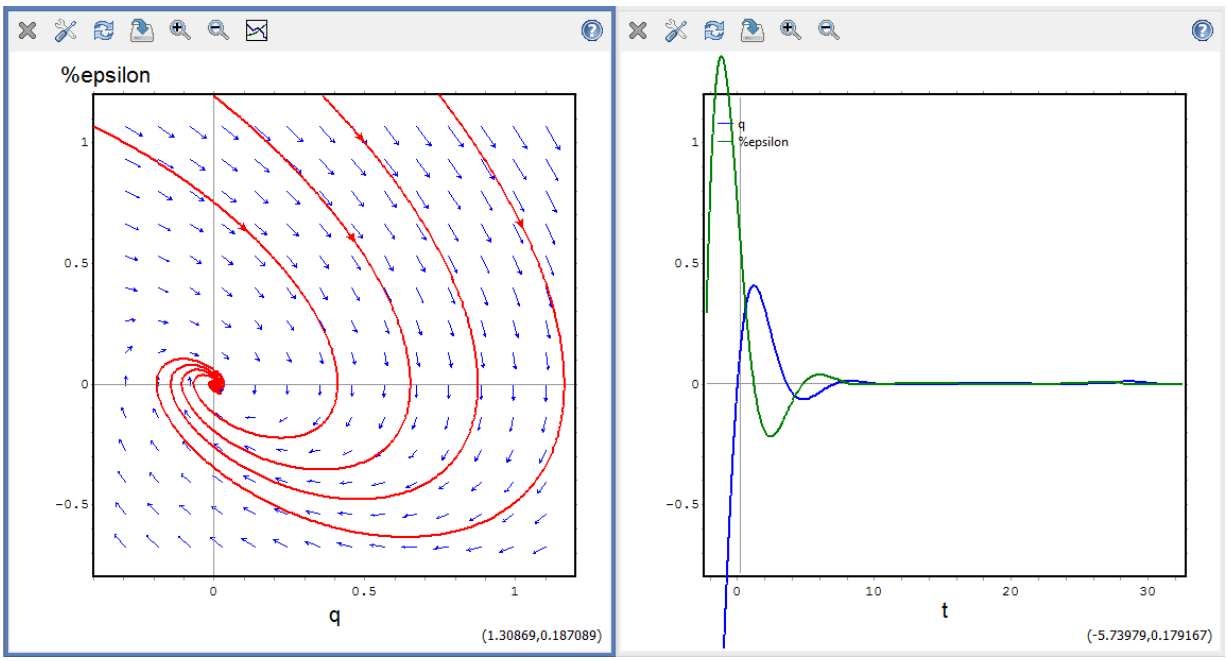


Рис. 10.10: Фазова діаграма для $\gamma = 0.5$.

```
(%i15) q(t):=((((%pi+6)*%omega*t)/6+%pi/6)*%e^(-%omega*t)$
       %omega:1$ v0:0$ q0:%pi/6$ l:1$
       lst:makelist((7*%pi/6+i*%pi/60),i,0,40)$
       with_slider_draw(
       k,lst,
       ytics=false,xtics=false,nticks=100,
       proportional_axes = xy,line_width = 3,line_type  = dashes,
       parametric(l*cos(t),l*sin(t),t,7*%pi/6,11*%pi/6),
       color=grey,point_type=5,point_size=6,
       points([[0,0]]),
       color=navy,point_type=filled_circle,point_size=6,
       points([l*cos(-%pi/2+q(k))],[l*sin(-%pi/2+q(k))]),
       line_type=solid,head_length=0.01,color=black,head_angle=1,
       vector([0,0],[l*cos(-%pi/2+q(k)),l*sin(-%pi/2+q(k))]))$
```

10.3 Вимушені коливання математичного маятника

Якщо на коливальну систему діє сила, яка має власну частоту коливань, виникають ефекти накладання двох рухів. Нехай ця сила коливається із частотою Ω та має амплітудне значення F, тоді можна записати наступне рівняння руху:

$$m\frac{d^2}{dt^2}(lq) + mgq = F\cos\Omega t.$$

Дослідимо ці ефекти у **Maxima**.

```
(%i1)  ratprint:false;
(%i2)  depends(q,t);
(%o2)     [q(t)]
```

```
(%i3)  Deqn7:m*'diff(l*q,t,2)+m*g*q=F*cos(%Omega*t);
```

$$(\%o3)\qquad m\cdot\left(\frac{d^2}{d\,t^2}\cdot(l\cdot q)\right)+g\cdot m\cdot q=\cos(\Omega\cdot t)\cdot F$$

Приведемо рівняння до звичного у фізиці, здійснивши заміни $\omega^2=g/l$, $A=F/(ml)$.

```
(%i4)  ratsimp(Deqn7/(m*l));
```

$$(\%o4)\qquad \frac{\frac{d^2}{d\,t^2}\cdot(l\cdot q)+g\cdot q}{l}=\frac{\cos(\Omega\cdot t)\cdot F}{l\cdot m}$$

```
(%i5)  ratsubst(%omega^2,g/l,ev(%,diff));
```

$$(\%o5)\qquad \frac{d^2}{d\,t^2}\cdot q+\omega^2\cdot q=\frac{\cos(\Omega\cdot t)\cdot F}{l\cdot m}$$

```
(%i6)  Deqn7a:ratsubst(A,F/(m*l),%);
```

$$(\%o6)\qquad \frac{d^2}{d\,t^2}\cdot q+\omega^2\cdot q=\cos(\Omega\cdot t)\cdot A$$

Розв'язуємо рівняння із початковими значеннями $q_0=\pi/6$, $\dot{q}_0=1$.

```
(%i7)  sol7:ode2(Deqn7a,q,t);
Is ω zero or nonzero? nonzero;
```

$$(\%o7)\qquad q=-\frac{\cos(\Omega\cdot t)\cdot A}{\Omega^2-\omega^2}+\%k1\cdot e^{i\cdot\omega\cdot t}+\%k2\cdot e^{-i\cdot\omega\cdot t}$$

```
(%i8)  sol7a:trigsimp(rectform(ic2(sol7,t=0,q=%pi/6,'diff(q,t)=1)));
```

$$(\%o8)\qquad q=-\frac{(6\cdot\omega\cdot\cos(\Omega\cdot t)-6\cdot\omega\cdot\cos(\omega\cdot t))\cdot A+(6\cdot\omega^2-6\cdot\Omega^2)\cdot\sin(\omega\cdot t)}{6\cdot\omega\cdot\Omega^2-6\cdot\omega^3}-$$
$$-\frac{(\pi\cdot\omega^3-\pi\cdot\omega\cdot\Omega^2)\cdot\cos(\omega\cdot t)}{6\cdot\omega\cdot\Omega^2-6\cdot\omega^3}$$

Для графічного представлення покладемо $\omega=1$, $A=1$ та розглянемо рух при різних Ω. Приймемо послідовно $\Omega=4.5$ та $\Omega=1.2$.

```
(%i9)  Q1:sol7a,A=1,%omega=1,%Omega=4.5;
```

$$(\%o9)\qquad q=-0.0086580086\cdot(6\cdot\cos(4.5\cdot t)-115.5\cdot\sin(t)-19.25\cdot\pi\cdot\cos(t)-6\cdot\cos(t))$$

```
(%i10) plot2d(rhs(Q1),[t,0,15*%pi],[style,[lines,2]],
       [xlabel,""],[ylabel,""]);
(%i11) Q2:sol7a,A=1,%omega=1,%Omega=1.2;
```

$$(\%o11)\qquad q=-0.37878787\cdot(6\cdot\cos(1.2\cdot t)-2.64\cdot\sin(t)-0.44\cdot\pi\cdot\cos(t)-6\cdot\cos(t))$$

```
(%i12) plot2d(rhs(Q2),[t,0,94],[style,[lines,2]],
       [xlabel,""],[ylabel,""]);
```

Отримані графіки зображені на Рис.10.11. Як видно на Рис.10.11(а), при великих Ω вплив вимушуючої сили на рух є мінімальним. Траєкторія коливань у певних місцях стає трохи деформованою, але всі основні характеристики коливального руху (амплітуда і частота) фактично зберігаються. Це пояснюється тим, що при дуже швидких осциляціях вимушуючої сили її середнє значення за період основних коливань $T=2\pi/\omega$ стає близьким до нуля.

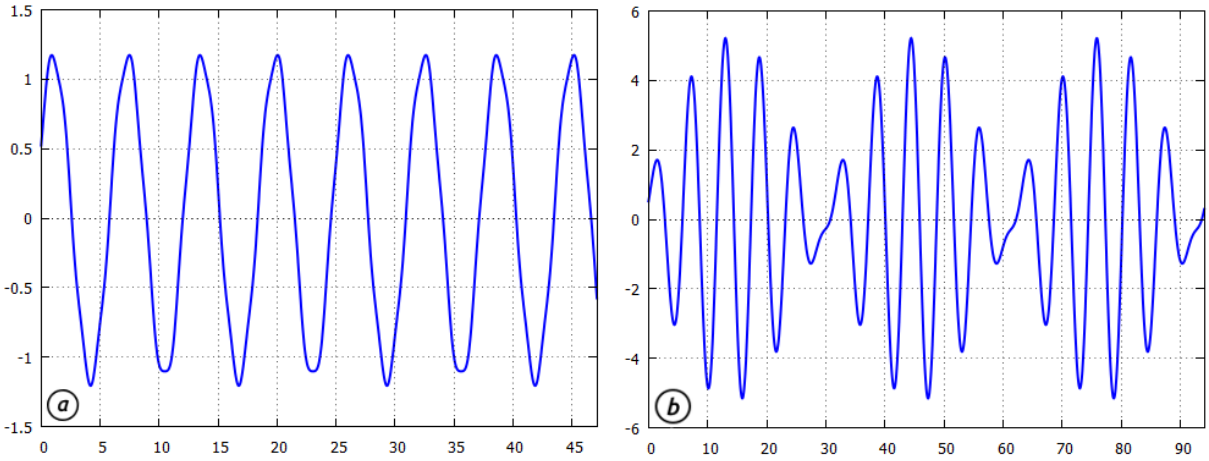


Рис. 10.11: Графіки коливань для $\Omega = 4.5$ (а) та $\Omega = 1.2$ (б), при $\omega = 1$.

Зовсім інша ситуація відбувається при приблизній рівності частот основної системи та вимушуючої сили. Коливання встигають провзаємодіяти та накластись одне на інше, що продукує картину на Рис.10.11(б). Виникає природне питання, що ж буде відбуватись, коли дві частоти зрівняються.

Просто покласти $\Omega = \omega$ у розв'язку неможливо внаслідок появи ділення на 0, тому знайдемо границю виразу для q при $\Omega \to \omega$ та зобразимо її графік (Рис.10.12).

```
(%i13) Q3:limit(sol7a,%Omega,%omega),A=1,%omega=1;
```

$$(\%o13) \quad q = -\frac{-6 \cdot t \cdot \sin(t) - 12 \cdot \sin(t) - 2 \cdot \pi \cdot \cos(t)}{12}$$

```
(%i14) plot2d(rhs(Q3),[t,0,94],[style,[lines,2]],
       [xlabel,""],[ylabel,""]);
```

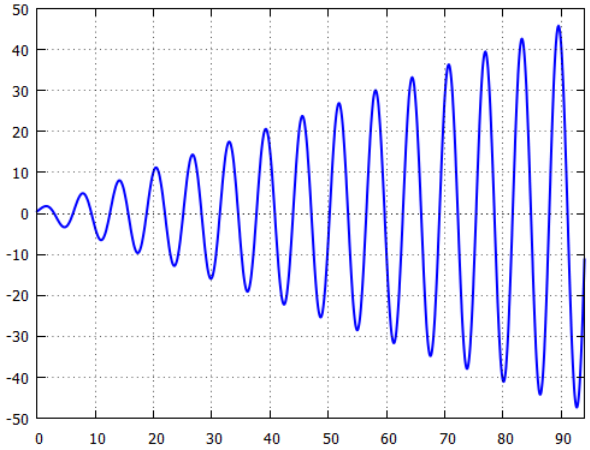


Рис. 10.12: Графік коливань при $\Omega = \omega$.

Із наявності множника t у чисельнику стає зрозумілим, що при $t \to \infty$ відхилення q також піде до нескінченності, причому осцилюючим способом. Для математичного маятника це означатиме, що кут відхилення буде весь час збільшуватись із прискоренням, тобто маятник почне обертатись із все більшою швидкістю. Для пружинного маятника (у якого роль q відіграє відхилення x вантажка) це означатиме, що амплітуда коливань буде весь час збільшуватись до нескінченності. У реальних фізичних випадках обидва варіанти призводять до руйнування всієї системи. Таке явище отримало назву резонанс, а процес підходу змінної частоти Ω до основної частоти ω називається биттям.

Зазвичай виникнення резонансу у механічних системах є досить шкідливим явищем, тому намагаються всіляко його уникнути. Наприклад, широко відомою є заборона марширувати в один крок колонам на мосту, також не дозволяється проїжджати там автомобілям, особливо вантажним, із певною встановленою швидкістю.

10.4 Вимушені коливання математичного маятника зі згасанням

Скомбінуємо попередно розглянуті випадки, додавши до згасаючих коливань вимушуючу силу синусоїдального вигляду. Тоді рівняння руху буде

$$m\frac{d^2}{dt^2}(lq) + b\frac{d}{dt}(lq) + mgq = F\cos\Omega t,$$

де Ω — чатота додаткової сили. Задавши це рівняння для розв'язку у **Maxima**, отримаємо повну відповідь, яка буде дуже довгою та складною для аналізу. Тому спочатку зробимо певні підготовчі дії для спрощення вигляду рівняння. Розділивши на ml та позначивши $\omega^2 = g/l$, $2\gamma = b/m$, $B = F/ml$, маємо

$$\frac{d^2q}{dt^2} + 2\gamma\frac{dq}{dt} + \omega^2 q = B\cos\Omega t.$$

Запровадимо безрозмірні величини: $\tau = \omega t$, $\delta = \gamma/\omega$, $W = \Omega/\omega$. Без втрати загальності можна покласти $B = 1$, здійснивши масштабне перетворення. Тоді, розділивши на ω^2, отримаємо рівняння

$$\frac{d^2q}{d\tau^2} + 2\delta\frac{dq}{d\tau} + q = \cos W\tau.$$

Задавши прямий розв'язок рівняння, маємо

```
(%i1) Deqn8:'diff(q,%tau,2)+2*%delta*'diff(q,%tau)+q=cos(W*%tau);
```

$$(\%o1)\quad \frac{d^2}{d\,\tau^2}\cdot q + 2\cdot\delta\cdot\left(\frac{d}{d\,\tau}\cdot q\right) + q = \cos\left(\tau\cdot W\right)$$

```
(%i2) ode2(Deqn8,q,%tau);
```

```
Is (δ − 1) · (δ + 1)  positive, negative or zero?pos;
```

$$(\%o2)\quad q = \frac{2\cdot\delta\cdot W\cdot\sin\left(\tau\cdot W\right) + \left(1 - W^2\right)\cdot\cos\left(\tau\cdot W\right)}{W^4 + \left(4\cdot\delta^2 - 2\right)\cdot W^2 + 1} +$$

$$\%k1\cdot e^{\frac{\left(\sqrt{4\cdot\delta^2-4}-2\cdot\delta\right)\cdot\tau}{2}} + \%k2\cdot e^{\frac{\left(-\sqrt{4\cdot\delta^2-4}-2\cdot\delta\right)\cdot\tau}{2}}$$

```
(%i3) ode2(Deqn8,q,%tau);
```

```
Is (δ − 1) · (δ + 1)  positive, negative or zero?zero;
```

$$(\%o3)\quad q = \frac{2\cdot\delta\cdot W\cdot\sin\left(\tau\cdot W\right) + \left(\delta^2 - W^2\right)\cdot\cos\left(\tau\cdot W\right)}{W^4 + 2\cdot\delta^2\cdot W^2 + \delta^4} + \left(\%k2\cdot\tau + \%k1\right)\cdot e^{-\delta\cdot\tau}$$

```
(%i4) ode2(Deqn8,q,%tau);
```

```
Is (δ − 1) · (δ + 1)  positive, negative or zero?neg;
```

$$(\%o4)\quad q = \frac{2\cdot\delta\cdot W\cdot\sin\left(\tau\cdot W\right) + \left(1 - W^2\right)\cdot\cos\left(\tau\cdot W\right)}{W^4 + \left(4\cdot\delta^2 - 2\right)\cdot W^2 + 1} +$$

$$e^{-\delta\cdot\tau}\cdot\left(\%k1\cdot\sin\left(\frac{\sqrt{4 - 4\cdot\delta^2}\cdot\tau}{2}\right) + \%k2\cdot\cos\left(\frac{\sqrt{4 - 4\cdot\delta^2}\cdot\tau}{2}\right)\right)$$

Вигляд розв'язку залежить від знаку виразу $(\delta^2 - 1)$, тому послідовно покладемо δ рівним $0.5; 1; 1.5$.

```
(%i5)  for i:1 thru 3 do
       q[i]:rhs(ic2(ode2(subst(%delta=i/2,Deqn8),q,%tau),
                %tau=0,q=%pi/6,'diff(q,%tau)=1));
(%o5)  done
```

Розглянемо для прикладу випадок $\delta = 0.5$ при високому значенні безрозмірної частоти $W = 5$, це означає, що кутова частота вимушуючої сили в п'ять разів більша за власну кутову частоту осцилятора.

```
(%i6)  Q1:subst(W=5,q[1]);
```

$$(\%o6)\quad e^{-\frac{\tau}{2}}\cdot\left(\frac{\left(625\cdot(\pi+12)+\pi+25\cdot(-\pi-18)+6\right)\cdot\sin\left(\frac{\sqrt{3}\cdot\tau}{2}\right)}{1202\cdot 3^{\frac{3}{2}}}+\right.$$
$$\left.\frac{\left(626\cdot\pi+25\cdot(6-\pi)-6\right)\cdot\cos\left(\frac{\sqrt{3}\cdot\tau}{2}\right)}{3606}\right)+\frac{5\cdot\sin(5\cdot\tau)-24\cdot\cos(5\cdot\tau)}{601}$$

```
(%i7)  %Epsilon1:diff(Q1,%tau)$
```

Побудуємо графіки залежностей кутового зміщення та кутової швидкості (Рис.10.13).

```
(%i8)  plot2d([Q1,%Epsilon1],[%tau,0,12],[nticks,100],
       [style,[lines,2]],[legend," Q1(%tau) "," E1(%tau) "],
       [axes,solid],[xlabel,"%tau"],[ylabel,"Q1 E1"]);
```

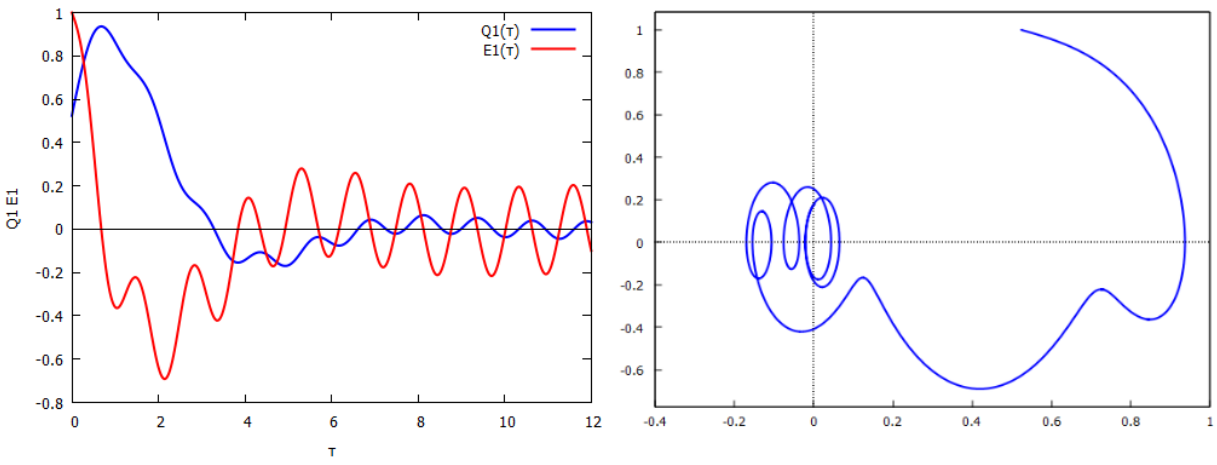


Рис. 10.13: Графіки для безрозмірних величин Q_1, E_1 та фазова діаграма.

Аналізуючи графік, бачимо, що через певний проміжок часу рушійна сила домінує над рухом лінійного осцилятора, який починає коливатися на вимушуючій частоті Ω. Це домінування, очевидно, не має нічого спільного з величиною амплітуди сили A, і такий же результат отримається за будь-якого A. Зробивши аналогічні дії з виразами q_2 та q_3 (при $\delta = 1$ та $\delta = 1.5$), можна побудувати відповідні графіки та переконатись, що даний ефект спостерігається і в цих випадках. Пояснюється це тим, що власні коливання у в'язкому середовищі з тертям завжди згасають до нуля, тоді як вимушуюча сила буде діяти завжди, і зрештою об'єкт почне коливатись лише під дією такої сили.

Сформуємо діаграму фазового простору для ранньої історії руху (Рис.10.13(б)). Зауважимо, що командою `plotdf` ми не можемо скористатись через присутність у правій

частині залежної змінної t. З діаграми видно, що система переходить на стабільний коливальний рух починаючи приблизно зі значення $q = 0$. Збільшуючи час у заданні діаграми, можна побачити, що фазовий портрет стабілізується та переходить у еліпс, тобто коливання системи стає монотонним.

```
(%i9)  load(draw);
(%i10) draw2d(nticks=500, line_width=2, xaxis=true, yaxis=true,
       yrange=[-0.7,0.4],xrange=[-0.4,1],parametric(Q1,%Epsilon1,%tau,0,9));
```

10.5 Осцилятор Ван дер Поля

Одним з найбільш відомих прикладів системи з нелінійним загасанням є осцилятор Ван дер Поля. Він описує коливальну систему у в'язкому середовищі, у якій сила опору пропорційна до зміщення в другому степені. Цей осцилятор відноситься до автоколивальних систем, тобто таких, в яких існує можливість періодичного асимптотично стійкого руху. Траєкторії руху у фазовому просторі представляють собою замкнутий контур, названий граничним циклом, до якого незалежно від початкових умов «притягуються» траєкторії з деякого околу. Загалом рівняння Ван дер Поля має наступний вигляд:

$$\frac{d^2q}{dt^2} + \mu(q^2 - 1)\frac{dq}{dt} + q = 0.$$

Спроба прямого розв'язку не приводить до успіху. Скористаємось командою `rkf45` для числового знаходження характеристик руху.

```
(%i1)  Deqn9:'diff(q,t,2)+%mu*(q^2-1)*'diff(q,t)+q=0;
```

$$(\%o1) \quad \frac{d^2}{dt^2}\cdot q + \mu\cdot\left(q^2-1\right)\cdot\left(\frac{d}{dt}\cdot q\right) + q = 0$$

```
(%i2)  ode2(Deqn9,q,t);
(%o2)  false

(%i3)  load(rkf45)$ %mu:4$
(%i4)  sol9:rkf45([%epsilon,-%mu*(q^2-1)*%epsilon-q],
                  [q,%epsilon],[0.75,0],[t,0,20])$
(%i5)  t_list:makelist(sol9[k][1],k,1,length(sol9))$
       q_list:makelist(sol9[k][2],k,1,length(sol9))$
       %epsilon_list:makelist(sol9[k][3],k,1,length(sol9))$
(%i6)  plot2d([[discrete,t_list,q_list],
              [discrete,t_list,%epsilon_list]],
              [legend,"q(t)","%epsilon(t)"]);
(%i7)  plot2d([[discrete,q_list,%epsilon_list]],
              [legend,false],[style,[lines,2,4]])$
```

Для $|x| > 1$ доданок $\mu(x^2 - 1)$ є додатним, тож він діє на коливальну систему як тертя або опір, забираючи енергію з системи. Однак якщо $|x| < 1$, коефіцієнт згасання від'ємний, тоді цей доданок діє як «негативний опір», що постачає енергію системі. Саме ця обставина робить такі коливання стійкими. Для прикладу, у маятнику Фруда енергія надається системі за посередництвом рівномірного обертання навколо осі.

Використання команди `plotdf` показує, що інтегральні криві, обрані поблизу фазової діаграми, завжди сходяться до спільного атрактора, зображеного на Рис.10.14. Високе

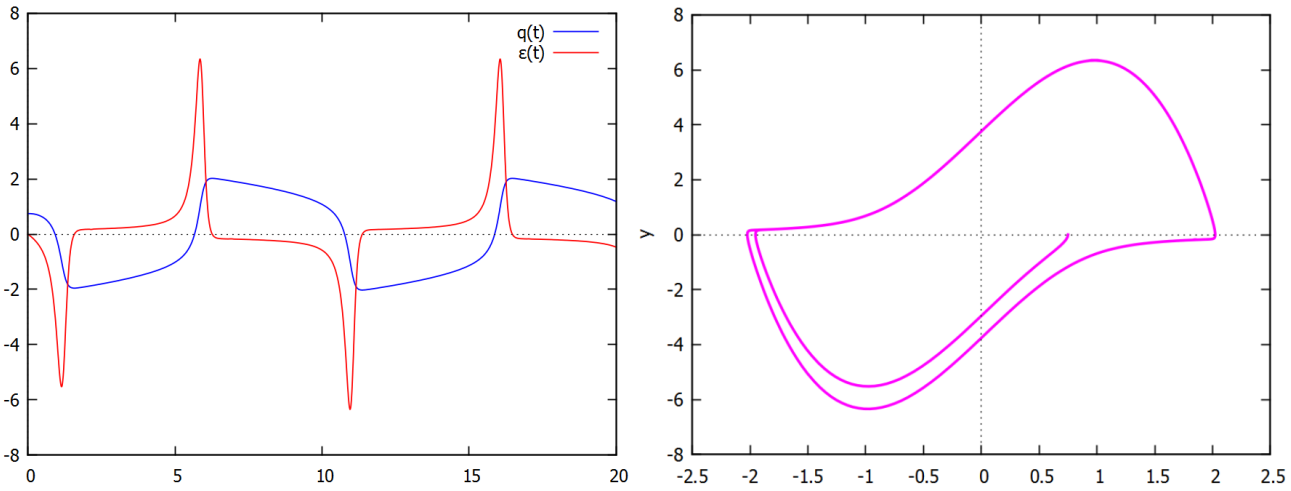


Рис. 10.14: Графіки для $q(t)$, $\varepsilon(t)$ та фазова діаграма осцилятора Ван дер Поля.

значення μ робить задачу Коші для ЗДР жорсткою і складно інтегровною, тому беремо достатньо невелике $\mu = 4$. Можливість існування періодичного асимптотично стійкого руху, що зображається ізольованою замкнутою траєкторією у фазовому просторі, до якої згодом притягуються траєкторії з околу незалежно від початкових умов, забезпечується тільки в нелінійних дисипативних системах.

10.6 Подвійний математичний маятник

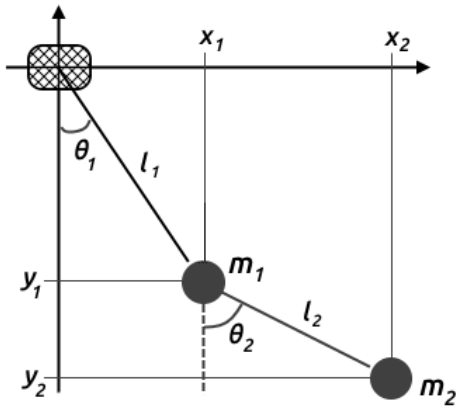


Рис. 10.15: Модель подвійного математичного маятника.

Розглянемо динамічну систему, що складається з двох послідовно з'єднаних математичних маятників. Будемо вважати, що нитки довжинами l_1 та l_2 нерозтяжні, маси тягарців m_1 та m_2. За узагальнені координати оберемо відповідні кути відхилення кожної кульки від положення рівноваги: $\theta_1 = q_1$, $\theta_2 = q_2$.

Із наведеного на Рис.10.15 зображення видно, що просторові координати кульок наступні:

$$\begin{cases} x_1 = l_1 \sin\theta_1; \\ y_1 = -l_1 \cos\theta_1; \\ x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2; \\ y_2 = -l_1 \cos\theta_1 - l_2 \cos\theta_2. \end{cases}$$

Кінетична енергія кожного тягарця

$$T_1 = \frac{m_1}{2}\left(\dot{x}_1^2 + \dot{y}_1^2\right); \quad T_2 = \frac{m_2}{2}\left(\dot{x}_2^2 + \dot{y}_2^2\right),$$

Потенціальна енергія:

$$U_1 = mgy_1; \quad U_2 = mgy_2.$$

Записавши лагранжіан системи

$$L = T - U = (T_1 + T_2) - (U_1 + U_2),$$

можна скласти рівняння Лагранжа для кожного тягарця окремо, система цих рівнянь і буде описувати рух кульок:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} = 0.$$

Реалізуємо це на мові **Maxima**.

Задаємо залежності узагальнених координат від часу і просторові координати:

```
(%i1) depends([q1,q2],t);
```

$$(\%o1) \quad [q1(t), q2(t)]$$

```
(%i2) x1(q1):=l1*sin(q1);
```

$$(\%o2) \quad x1(q1) := l1 \cdot \sin(q1)$$

```
(%i3) y1(q1):=-l1*cos(q1);
```

$$(\%o3) \quad y1(q1) := -l1 \cdot \cos(q1)$$

```
(%i4) x2(q1,q2):=l1*sin(q1)+l2*sin(q2);
```

$$(\%o4) \quad x2(q1, q2) := l1 \cdot \sin(q1) + l2 \cdot \sin(q2)$$

```
(%i5) y2(q1,q2):=-l1*cos(q1)-l2*cos(q2);
```

$$(\%o5) \quad y2(q1, q2) := -l1 \cdot \cos(q1) - l2 \cdot \cos(q2)$$

Задаємо потенціальну енергію:

```
(%i6) U:m1*g*y1(q1)+m2*g*y2(q1,q2);
```

$$(\%o6) \quad g \cdot m2 \cdot (-l2 \cdot \cos(q2) - l1 \cdot \cos(q1)) - g \cdot l1 \cdot m1 \cdot \cos(q1)$$

Кінетична енергія першої кульки:

```
(%i7) T1:(m1/2)*(('diff(x1(q1),t))^2+('diff(y1(q1),t))^2);
```

$$(\%o7) \quad \frac{m1 \cdot \left(\left(\frac{d}{dt} \cdot (l1 \cdot \sin(q1))\right)^2 + \left(\frac{d}{dt} \cdot (-l1 \cdot \cos(q1))\right)^2\right)}{2}$$

```
(%i8) ev(T1,nouns);
```

$$(\%o8) \quad \frac{m1 \cdot \left(l1^2 \cdot \sin(q1)^2 \cdot \left(\frac{d}{dt} \cdot q1\right)^2 + l1^2 \cdot \cos(q1)^2 \cdot \left(\frac{d}{dt} \cdot q1\right)^2\right)}{2}$$

Кінетична енергія другої кульки

```
(%i9) T2:(m2/2)*(('diff(x2(q1,q2),t))^2+('diff(y2(q1,q2),t))^2);
```

$$(\%o9) \quad \frac{m2 \cdot \left(\left(\frac{d}{dt} \cdot (l2 \cdot \sin(q2) + l1 \cdot \sin(q1))\right)^2 + \left(\frac{d}{dt} \cdot (-l2 \cdot \cos(q2) - l1 \cdot \cos(q1))\right)^2\right)}{2}$$

```
(%i10) ev(T2,nouns);
```

$$(\%o10) \quad (m2 \cdot (\left(l2 \cdot \sin(q2) \cdot \left(\frac{d}{dt} \cdot q2\right) + l1 \cdot \sin(q1) \cdot \left(\frac{d}{dt} \cdot q1\right)\right)^2 + \left(l2 \cdot \cos(q2) \cdot \left(\frac{d}{dt} \cdot q2\right) + l1 \cdot \cos(q1) \cdot \left(\frac{d}{dt} \cdot q1\right)\right)^2))/ \cdot 2 \cdot$$

Загальна кінетична енергія:

```
(%i11) T:T1+T2;
```

$$(\%o11)\quad \frac{m2\cdot\left(\left(\frac{d}{dt}\cdot(l2\cdot\sin(q2)+l1\cdot\sin(q1))\right)^2+\left(\frac{d}{dt}\cdot(-l2\cdot\cos(q2)-l1\cdot\cos(q1))\right)^2\right)}{2}$$
$$+\frac{m1\cdot\left(\left(\frac{d}{dt}\cdot(l1\cdot\sin(q1))\right)^2+\left(\frac{d}{dt}\cdot(-l1\cdot\cos(q1))\right)^2\right)}{2}$$

Формуємо лагранжіан системи та складаємо рівняння руху для обох кульок. Для спрощення кінцевого результату використовуємо команди `trigreduce` та `trigsimp`, щоб вирази не були дуже громіздкими.

```
(%i12) Lagr:T-U;
(%i13) Deqn10a:'diff(Lagr,q1)-'diff('diff(Lagr,'diff(q1,t)),t)=0;
(%i14) Deqn10a2:trigreduce(trigsimp(ev(Deqn10a,nouns)));
```

$$(\%o14)\quad l1\cdot l2\cdot m2\cdot\left(\frac{d}{dt}\cdot q2\right)^2\cdot\sin(q2-q1)-l1\cdot l2\cdot m2\cdot\left(\frac{d^2}{dt^2}\cdot q2\right)\cdot\cos(q2-q1)$$
$$-{l1}^2\cdot m2\cdot\left(\frac{d^2}{dt^2}\cdot q1\right)-{l1}^2\cdot m1\cdot\left(\frac{d^2}{dt^2}\cdot q1\right)-g\cdot l1\cdot m2\cdot\sin(q1)-g\cdot l1\cdot m1\cdot\sin(q1)=0$$

```
(%i15) Deqn10b:'diff(Lagr,q2)-'diff('diff(Lagr,'diff(q2,t)),t)=0;
(%i16) Deqn10b2:trigreduce(trigsimp(ev(Deqn10b,nouns)));
```

$$(\%o16)\quad -l1\cdot l2\cdot m2\cdot\left(\frac{d}{dt}\cdot q1\right)^2\cdot\sin(q2-q1)-l1\cdot l2\cdot m2\cdot\left(\frac{d^2}{dt^2}\cdot q1\right)\cdot\cos(q2-q1)$$
$$-l2^2\cdot m2\cdot\left(\frac{d^2}{dt^2}\cdot q2\right)-g\cdot l2\cdot m2\cdot\sin(q2)=0$$

Диференціальні рівняння `Deqn10a2` та `Deqn10b2` і є шуканими рівняннями руху. Вони другого порядку, нелінійні, неоднорідні. Прямий розв'язок їх неможливий, однак можна спробувати розв'язати їх числово методом Рунге-Кутти. Для початку виділимо окремо другі похідні $\ddot{q}_1$ та $\ddot{q}_2$.

```
(%i17) Sol10:solve([Deqn10a2,Deqn10b2],[('diff(q1,t,2)),('diff(q2,t,2))]);
```

Це необхідно для того, щоб сформувати з системи двох ЗДР другого порядку систему з чотирьох ЗДР першого порядку, до яких метод Рунге-Кутти може бути застосовний. Запровадимо кутові швидкості $\varepsilon_1=\dot{q}_1$, $\varepsilon_2=\dot{q}_2$, тоді можна записати наступну систему:

$$\begin{cases}\dot{q}_1=\varepsilon_1;\\ \dot{\varepsilon}_1=\texttt{Sol10[1]};\\ \dot{q}_2=\varepsilon_2;\\ \dot{\varepsilon}_2=\texttt{Sol10[2]}.\end{cases}$$

Для числового розв'язку задаємо початкові умови та характеристики системи:

```
(%i24) m1:1$ m2:1.5$ l1:0.4$ l2:0.6$ g:9.81$ t_start:0$t_end:8$
(%i28) q1_start:float(%pi/8)$ q2_start:float(%pi/4)$
       %epsilon1_start:0$ %epsilon2_start:0$
```

Формуємо систему чотирьох диференціальних рівнянь:

```
(%i29) equs : ev([%epsilon1,%epsilon2,rhs(Sol10[1][1]),
                  rhs(Sol10[1][2])],'diff(q1,t)=%epsilon1,
                  'diff(q2,t)=%epsilon2)$
```

Застосуємо тепер метод Рунге-Кутти `rkf45`:

```
(%i30) load(rkf45);
(%i31) sol:rkf45(equs,[q1, q2, %epsilon1, %epsilon2],
                 [q1_start,q2_start,%epsilon1_start,%epsilon2_start],
                 [t,t_start,t_end],report=true) $
```

Система видала звіт, що для інтегрування було використано 845 точок, причому у 33 випадках був здійснений перерахунок, оскільки точність обчислень у них була меншою ніж задана за замовчуванням. Результатом розрахунків є список із 845 п'ятірок елементів $(t, q_1, q_2, \varepsilon_1, \varepsilon_2)$. Щоб опрацювати їх графічно, виділимо списки для кожної з цих величин окремо.

```
(%i36) t_list:makelist(sol[k][1],k,1,length(sol))$
       q1_list:makelist(sol[k][2],k,1,length(sol))$
       q2_list:makelist(sol[k][3],k,1,length(sol))$
       %epsilon1_list:makelist(sol[k][4],k,1,length(sol))$
       %epsilon2_list:makelist(sol[k][5],k,1,length(sol))$
```

Тепер можемо відобразити графіки залежностей динамічних величин від часу та накреслити фазові портрети системи.

Залежності узагальнених координат q_1 та q_2 від часу (Рис.10.16):

```
(%i37) wxplot2d([[discrete,t_list,q1_list],[discrete,t_list,q2_list]],
       [style,[lines,2]],[legend,"q1(t)","q2(t)"]);
```

Залежності кутових швидкостей ε_1 та ε_2 від часу (Рис.10.17):

```
(%i38) wxplot2d([[discrete,t_list,%epsilon1_list],
                 [discrete,t_list,%epsilon2_list]],
                 [style,[lines,2],[lines,2]],
                 [legend,"%epsilon1(t)","%epsilon2(t)"]);
```

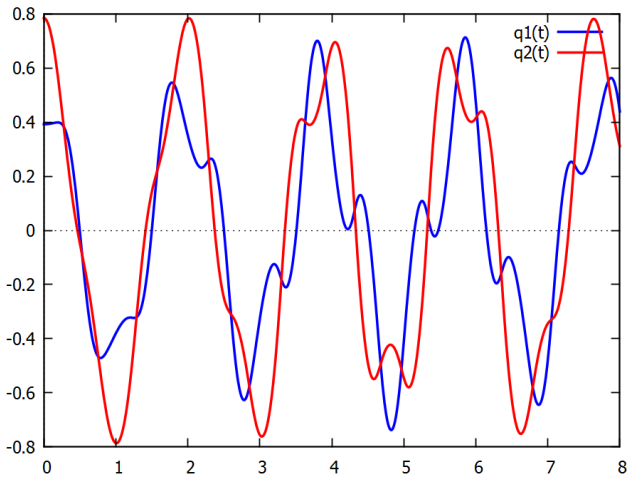


Рис. 10.16: Залежності $q_1(t)$, $q_2(t)$.

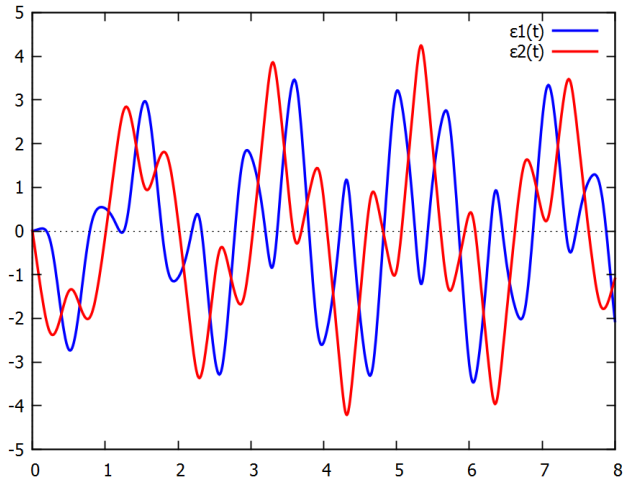


Рис. 10.17: Залежності $\varepsilon_1(t)$, $\varepsilon_2(t)$.

Залежність $q_1(t)$ від $q_2(t)$ як параметрична крива (Рис.10.18):

```
(%i39) wxplot2d([[discrete,q1_list,q2_list]],
                [style,[lines,2,4]],[legend,false]);
```

Залежність $\varepsilon_1(t)$ від $\varepsilon_2(t)$ як параметрична крива (Рис.10.19):

```
(%i40) wxplot2d([[discrete,%epsilon1_list,%epsilon2_list]],
                [style,[lines,2,4]],[legend,false]);
```

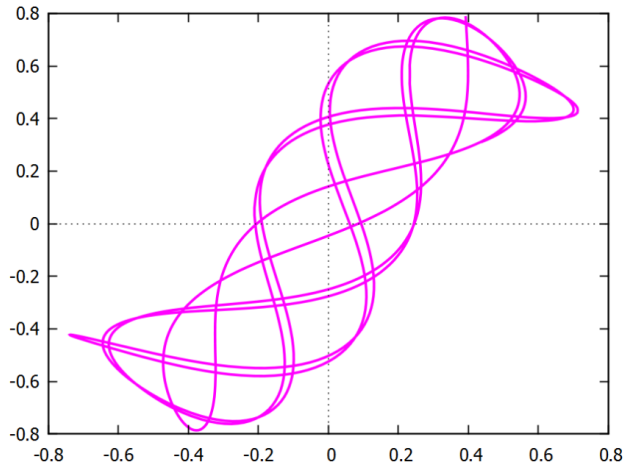


Рис. 10.18: $q_1(t)$ та $q_2(t)$ як параметрична крива.

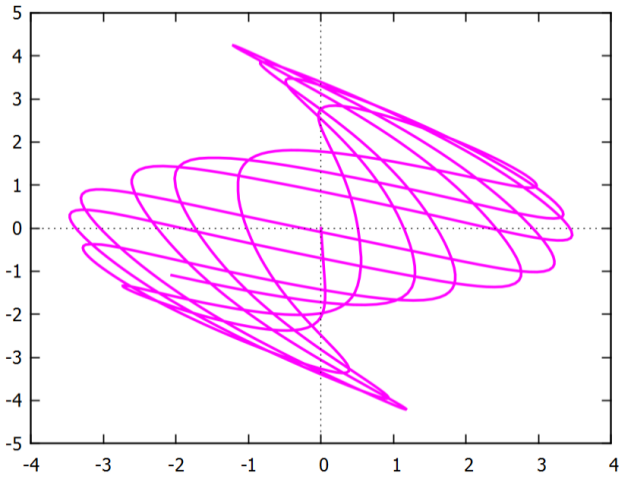


Рис. 10.19: $\varepsilon_1(t)$ та $\varepsilon_2(t)$ як параметрична крива.

Фазовий портрет для $q_1(t)$ та $\varepsilon_1(t)$ (Рис.10.20):

```
(%i41) wxplot2d([[discrete,q1_list,%epsilon2_list]],
                [style,[lines,2]],[legend,false]);
```

Фазовий портрет для $q_2(t)$ та $\varepsilon_2(t)$ (Рис.10.21):

```
(%i42) wxplot2d([[discrete,q2_list,%epsilon2_list]],
                [style,[lines,2]],[legend,false]);
```

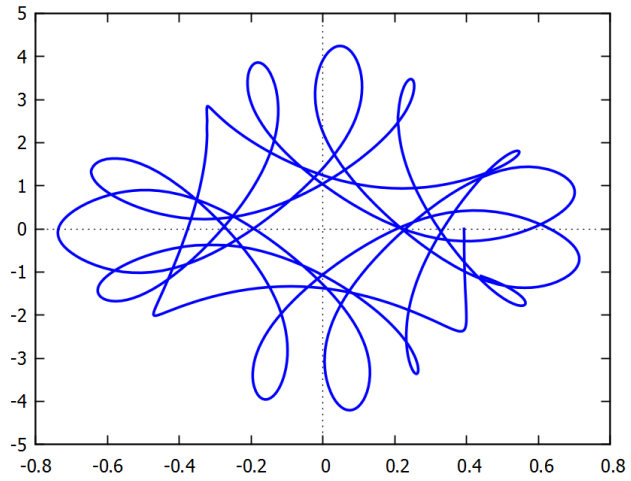


Рис. 10.20: Фазова діаграма для $q_1(t)$ та $\varepsilon_1(t)$.

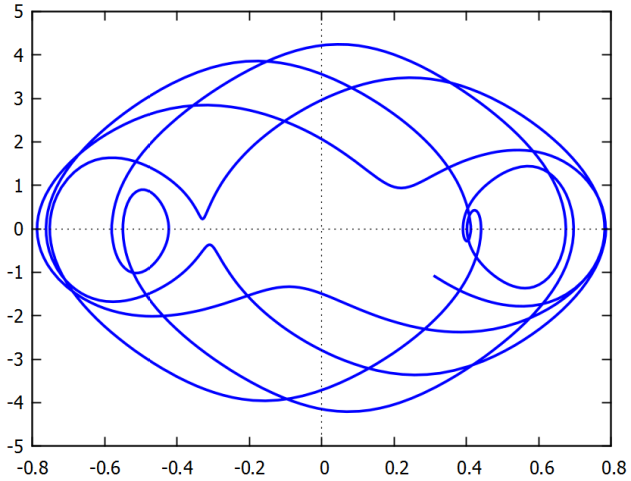


Рис. 10.21: Фазова діаграма для $q_2(t)$ та $\varepsilon_2(t)$.

Для анімації коливання знайдемо реальні просторові координати кульок:

```
(%i43) x1_list:l1*sin(q1_list)$
       y1_list:-l1*cos(q1_list)$
       x2_list:l1*sin(q1_list)+l2*sin(q2_list)$
       y2_list:-l1*cos(q1_list)-l2*cos(q2_list)$
```

Кожна пара (x_1, y_1), (x_2, y_2) задає положення відповідної точки у просторі. Вибираючи із списків почергово ці величини та зображаючи їх на координатній площині, можна сформувати послідовність «знімків» загальної картини коливальної системи. Цих кадрів буде досить велика кількість (845), тому щоб не перевантажувати оболонку **wxMaxima**, відразу створимо файл .gif, де і буде відображений коливальний процес.

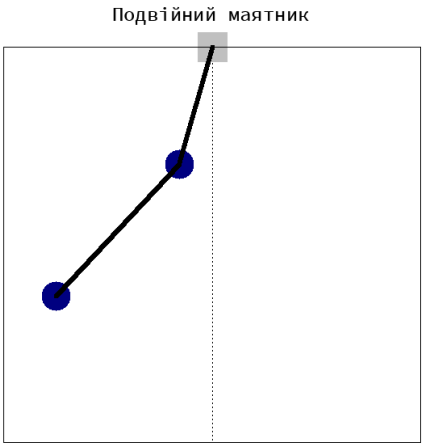


Рис. 10.22: Кадр анімації коливань подвійного маятника.

```
(%i44) load(draw)$
       with_slider_draw(
       k,makelist(i,i,1,length(sol)),
       terminal   = animated_gif,
       file_name  = "Double_Pend_3",
       dimensions = [600, 600],
       xaxis = true, yaxis = true,
       ytics = false,xtics = false,nticks = 100,
       xrange = [-0.7,0.7],
       yrange = [-1.3,0],
       color=grey,point_type=5,point_size=6,
       points([[0,0]]),
       color=navy,point_type=filled_circle,point_size=6,
       points([[x1_list[k],y1_list[k]]]),
       color=navy,point_type=filled_circle,point_size=6,
       points([[x2_list[k],y2_list[k]]]),
       line_type=solid,line_width=5,head_length=0.01,
       color=black,head_angle=1,
       vector([0,0],[x1_list[k],y1_list[k]]),
       line_type=solid,line_width=5,head_length=0.01,
       color=black,head_angle=1,
       vector([x1_list[k],y1_list[k]],
              [x2_list[k]-x1_list[k],y2_list[k]-y1_list[k]]),
       font = "Arial Narrow",font_size = 22,color = black,
       title = "Подвійний математичний маятник"
       )$
```

Розділ 11

Задачі класичної механіки

11.1 Задача Плеяд

Зоряний кластер Плеяд (українська назва Стожари) має діаметр в 12 світлових років. Від нас Плеяди віддалені на відстань в 410 світлових років, вони вважаються порівняно молодим скупченням зірок, вік яких знаходиться в діапазоні 75-100 мільйонів років. Плеяди давно відомі як фізично пов'язана група об'єктів, а не результат випадкової проекції різновіддалених зір. Велику цікавість цей кластер викликає в першу чергу тим, що він є другим після групи Ґіад найближчим до Землі зірковим скупченням. Плеяди та Ґіади є ідеальним ґрунтом для моделювання еволюції зір, оскільки всі їхні зірки мають однаковий вік і склад, але демонструють широкий діапазон мас. Ці два скупчення єдині, відстань до яких можна виміряти на поверхні Землі безпосередньо за їхнім паралактичним зміщенням. Виміряні значення лежать в основі діаграми Герцшпрунга-Рассела, за якою визначають відстані до всіх інших зоряних скупчень. Науковці екстраполюють отриману шкалу від розсіяних зіркових скупчень до галактик і галактичних скупчень, побудувавши шкалу космічних відстаней. Зрештою, знання астрономів про вік і розвиток Всесвіту у великій мірі залежать від знання відстані до зіркового скупчення Плеяд, тому вони є своєрідним еталоном, котрий добре вивчений та змодельований.

Задача Плеяд — це модельна задача небесної механіки семи зірок, що рухаються в одній площині. Математично вона описується системою з 14 диференціальних рівнянь другого порядку разом з 28 початковими умовами, що визначають початкове положення та швидкість для кожного небесного тіла. Задача не жорстка, але досить складна, її можна легко розширити на будь-яку кількість об'єктів, не обов'язково в одній площині. Звісно, реальний рух зірок відбувається у тривимірному просторі, але нас в першу чергу цікавитиме проєкція цього руху на площину небесної сфери.

Нехай маємо рух кількох масивних тіл на площині. Запишемо другий закон Ньютона для i-го тіла, та вираз для сили притягання між двома тілами i та j, що взаємодіють гравітаційно:

$$\vec{F_i} = m_i \frac{d\vec{p_i}}{dt} \quad \vec{F}_{ij} = G\frac{m_i m_j}{|\vec{r_i} - \vec{r_j}|^3}(\vec{r_i} - \vec{r_j}).$$

У системі багатьох тіл сила, що діє на окремий об'єкт, є векторною сумою всіх сил, що діють на нього від усіх інших об'єктів. Тому можемо записати $\vec{F_i} = \sum_{i \neq j} \vec{F}_{ij}$, що після покладання $G = 1$ та розділення змінних нам дає

$$\begin{cases} \ddot{x}_i = \sum_{i \neq j} \dfrac{m_j(x_i - x_j)}{\left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{3/2}}; \\ \ddot{y}_i = \sum_{i \neq j} \dfrac{m_j(y_i - y_j)}{\left((x_i - x_j)^2 + (y_i - y_j)^2\right)^{3/2}}. \end{cases}$$

Початкові дані для цього руху беруться із роботи [10]. Числа означають положення кожного тіла відносно певної умовної нерухомої точки всередині скупчення та їх швидкості в момент часу $t = 0$. Маси розглядуваних зірок є фактично однаковими, вони міняються від 3 до 6 мас Сонця. Ми покладемо масу кожної зірки рівною її номеру $m_j = j$, на вигляд розв'язку це вплине мінімально.

$$\begin{cases} x_i|_{t=0} = [3, 3, -1, -3, 2, -2, 2]^T; \\ y_i|_{t=0} = [3, -3, 2, 0, 0, -4, 4]^T; \\ \dot{x}_i|_{t=0} = [0, 0, 0, 0, 0, 1.75, -1.5]^T; \\ \dot{y}_i|_{t=0} = [0, 0, 0, -1.25, 1, 0, 0]^T. \end{cases}$$

Щоб перейти від рівнянь другого порядку до системи з 28 рівнянь першого порядку, запишемо їх у наступній формі:

$$\begin{pmatrix} z \\ w \end{pmatrix}' = \begin{pmatrix} w \\ f(z) \end{pmatrix},$$

де z позначає набір всіх координат x_i та y_i, $f(z)$ позначає набір правих частин рівнянь для $\ddot{x}_i$ та $\ddot{y}_i$, w проміжна змінна, яка є лінійною швидкістю кожного об'єкту.

Переведемо ці вхідні дані на мову **Maxima** та побудуємо розв'язки.

Задаємо вирази для гравітаційних рівнянь руху (команда `concat` з'єднує будь-які списки):

```
(%i1)  fx(i):=sum(if i#j then j*(concat(x,j)-concat(x,i)) /
                  ((concat(x,i)-concat(x,j))^2+
                  (concat(y,i)-concat(y,j))^2)^1.5
                else 0, j,1,7);
```

$$(\%o1)\quad \mathrm{fx}\,(i) := \sum_{j=1}^{7} \text{if } i \# j \text{ then } \frac{j \cdot (\mathrm{concat}\,(x,j) - \mathrm{concat}\,(x,i))}{\left((\mathrm{concat}\,(x,i) - \mathrm{concat}\,(x,j))^2 + (\mathrm{concat}\,(y,i) - \mathrm{concat}\,(y,j))^2\right)^{1.5}}$$

$$\text{else } 0$$

```
(%i2)  fy(i):=sum(if i#j then j*(concat(y,j)-concat(y,i)) /
                  ((concat(x,i)-concat(x,j))^2+
                  (concat(y,i)-concat(y,j))^2)^1.5
                else 0, j,1,7);
```

$$(\%o2)\quad \mathrm{fy}\,(i) := \sum_{j=1}^{7} \text{if } i \# j \text{ then } \frac{j \cdot (\mathrm{concat}\,(y,j) - \mathrm{concat}\,(y,i))}{\left((\mathrm{concat}\,(x,i) - \mathrm{concat}\,(x,j))^2 + (\mathrm{concat}\,(y,i) - \mathrm{concat}\,(y,j))^2\right)^{1.5}}$$

$$\text{else } 0$$

Формуємо список з 28 правих частин рівнянь руху, запровадивши додаткові змінні w_x, w_y (команда `flatten` робить із списків різних рівнів вкладеності один великий список першого рівня, де перелічені всі елементи):

```
(%i3)  equs:flatten([makelist(concat(wx,i),i,1,7),
                     makelist(concat(wy,i),i,1,7),
                     makelist(fx(i),i,1,7),
                     makelist(fy(i),i,1,7)])$
```

Формуємо список із 28 невідомих функцій, відносно яких потрібно розв'язати рівняння руху:

```
(%i4)  funcs:flatten([makelist(concat(x,i),i,1,7),
                      makelist(concat(y,i),i,1,7),
                      makelist(concat(wx,i),i,1,7),
                      makelist(concat(wy,i),i,1,7)]);
```

Записуємо початкові значення функцій та часовий проміжок інтегрування:

```
(%i5)  init:[3,3,-1,-3,2,-2,2,
             3,-3,2,0,0,-4,4,
             0,0,0,0,0,1.75,-1.5,
             0,0,0,-1.25,1,0,0]$
(%i7)  t_start:0$ t_end:3$
```

Завантажуємо пакет `rkf45` та застосовуємо його:

```
(%i8)  load(rkf45);
(%i9)  sol1:rkf45(equs,funcs,init,[t,t_start,t_end],report=true)$
```

Із відгуку системи дізнаємось, що було обрано 1143 точки інтегрування, з максимальною похибкою 10^{-6} на кожному кроці. Результати повернулись у вигляді списку із 1143 елементів, кожен з яких є списком із 29 елементів (час t, сім координат x_i, сім координат y_i, сім x-компонент швидкості w_{xi}, сім y-компонент швидкості w_{yi}). Для опрацювання даних виділимо окремо списки для часу та всіх просторових координат.

```
(%i10) t_list:makelist(sol1[k][1],k,1,length(sol1))$
(%i17) x1_list:makelist(sol1[k][2],k,1,length(sol1))$
       x2_list: ...
(%i24) y1_list:makelist(sol1[k][9],k,1,length(sol1))$
       y2_list: ...
```

Позначимо окремо початкові точки, з яких стартував рух кожної зірки:

```
(%i25) init_pnts:[discrete,makelist([init[i],init[i+7]],i,1,7)]$
```

Тепер можемо зобразити траєкторії руху кожної зорі (Рис.11.1(а)). Як бачимо, на графіку присутня область, на якій траєкторії дуже переплітаються. Клацнувши правою кнопкою миші на відповідній частині малюнка, можна виділити цю область, тоді вона автоматично збільшується (Рис.11.1(б)).

```
(%i26) plot2d([init_pnts,
               [discrete,x1_list,y1_list],
               [discrete,x2_list,y2_list],
               [discrete,x3_list,y3_list],
               [discrete,x4_list,y4_list],
               [discrete,x5_list,y5_list],
               [discrete,x6_list,y6_list],
               [discrete,x7_list,y7_list]],
               [x,-4,4],[y,-6,6],[style,[points,2,5,7],
               [lines,2,1],[lines,2,2],[lines,2,3],
               [lines,2,4],[lines,2,5],[lines,2,6],[lines,2,2]],
               [legend,"Стартові точки","Зоря 1","Зоря 2","Зоря 3",
                "Зоря 4","Зоря 5","Зоря 6","Зоря 7"])$
```

Задавши подібну команду, де у парах замість x чи y стоять часові координати, можна зобразити часову еволюцію відповідних просторових компонент $x(t)$, $y(t)$ (Рис.11.2).

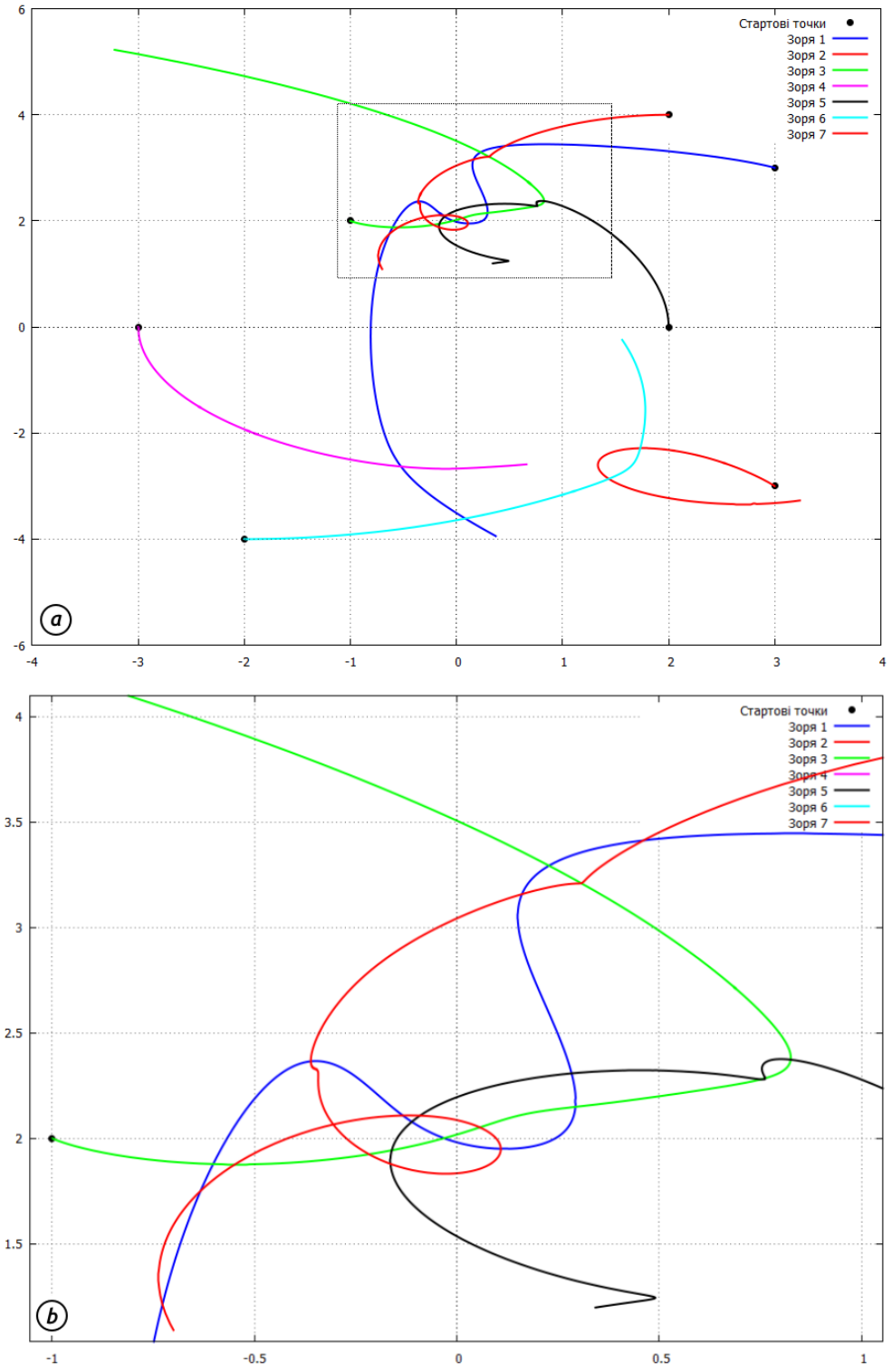


Рис. 11.1: Траєкторії руху зірок при чисельному інтегруванні (а), збільшена частина області з переплетеними траєкторіями (б).

Використавши додаткову опцію `absolute_tolerance`, можна змінювати величину похибки та підвищувати точність інтегрування, однак платою за це буде значно більше навантаження на процесор та збільшення часу обчислень. Наприклад, точність 10^{-10} замість 10^{-6} призведе до того, що час обробки зміниться від 5 с до 10 год. Втім, задачу чисельного інтегрування руху кількох тіл для реальних потреб космічних розрахунків обчислюють з точністю 10^{-16}, що для звичайних комп'ютерів вимагало б років безперервної роботи.

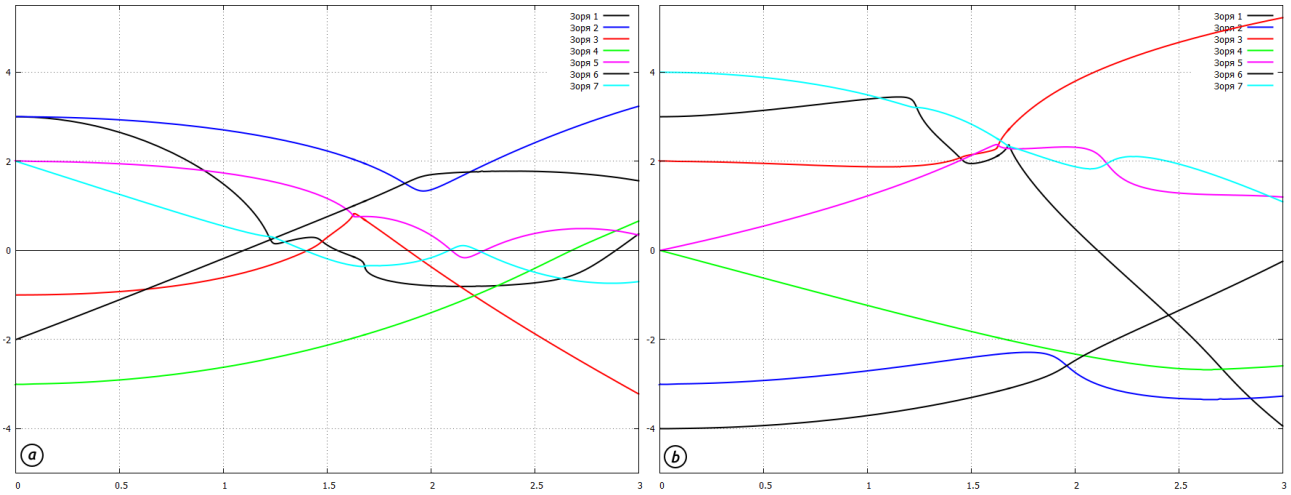


Рис. 11.2: Часова залежність $x(t)$ (а) та $y(t)$ (б) координат для зір скупчення Плеяд.

11.2 Модельні задачі кінематики

Продемонструємо, як можна використати застосунок **Maxima** для розв'язання та графічного представлення отриманих розв'язків у задачах кінематики. Візьмемо до розгляду кілька завдань зі «Збірника задач з механіки» (М. В. Ваврух та ін., Львів, 2017) [5].

► Хлопчик стоїть на відстані a перед стінкою та кидає в неї із висоти h м'ячик із швидкістю v_0 та початковим кутом α, який пружно відбивається. Знайти відстань до хлопчика, на якій м'ячик впаде на землю, та зобразити траєкторію його руху в залежності від величини a.

Задамо початкову швидкість у проєкціях на координатні осі та знайдемо вирази для $x(t)$ та $y(t)$ інтегруванням відповідних проєкцій.

```
(%i1) ratprint:false;
```
(%o1) false

```
(%i2) v_x(t):=v_0*cos(%alpha);
```
(%o2) $\mathrm{v\_x}\,(t) := v\_0 \cdot \cos\,(\alpha)$

```
(%i3) v_y(t):=v_0*sin(%alpha)-g*t;
```
(%o3) $\mathrm{v\_y}\,(t) := v\_0 \cdot \sin\,(\alpha) - g \cdot t$

```
(%i4) x(t):=integrate(v_x(%xi),%xi,0,t);
```
(%o4) $\mathrm{x}\,(t) := \int_0^t \mathrm{v\_x}\,(\xi)\,d\xi$

```
(%i5) y(t):=h+integrate(v_y(%xi),%xi,0,t);
```
(%o5) $\mathrm{y}\,(t) := h + \int_0^t \mathrm{v\_y}\,(\xi)\,d\xi$

```
(%i6) x(t);
```
(%o6) $\cos\,(\alpha) \cdot t \cdot v\_0$

```
(%i7)  y(t);
```

$$(\%o7) \quad \frac{2\cdot\sin(\alpha)\cdot t\cdot v\_0 - g\cdot t^2}{2} + h$$

Визначимо тепер повний час руху до падіння t_{full}, він знаходиться з умови $y(t) = 0$, із двох розв'язків квадратного рівняння обираємо додатний.

```
(%i8)  solve(y(t)=0,t);
```

$$(\%o8) \quad [t = -\frac{\sqrt{\sin(\alpha)^2\cdot v\_0^2 + 2\cdot g\cdot h} - \sin(\alpha)\cdot v\_0}{g},$$

$$t = \frac{\sqrt{\sin(\alpha)^2\cdot v\_0^2 + 2\cdot g\cdot h} + \sin(\alpha)\cdot v\_0}{g}]$$

```
(%i9)  t_full:rhs(second(solve(y(t)=0,t)));
```

$$(\%o9) \quad \frac{\sqrt{\sin(\alpha)^2\cdot v\_0^2 + 2\cdot g\cdot h} + \sin(\alpha)\cdot v\_0}{g}$$

Зараз врахуємо, що перед хлопчиком стоїть стінка. При ударові об неї м'яч зазнає пружного відбиття; це означає, що x-компонента вектора швидкості міняється на протилежну, тоді як y-компонента залишається без змін. Фактично така обставина проявляється як зміна $x(t)$ на $-x(t)$, але із зсувом вправо на величину $2a$, щоб рух продовжувався із тієї самої точки.

```
(%i10) xa(t):=if t<t_a then x(t) else 2*a-x(t);
```

$$(\%o10) \quad \mathrm{xa}(t) := \mathrm{if}\ t < t\_a\ \mathrm{then}\ \mathrm{x}(t)\ \mathrm{else}\ 2\cdot a - \mathrm{x}(t)$$

```
(%i11) ya(t):=y(t);
```

$$(\%o11) \quad \mathrm{ya}(t) := \mathrm{y}(t)$$

```
(%i12) solve(x(t)=a,t);
```

$$(\%o12) \quad [t = \frac{a}{\cos(\alpha)\cdot v\_0}]$$

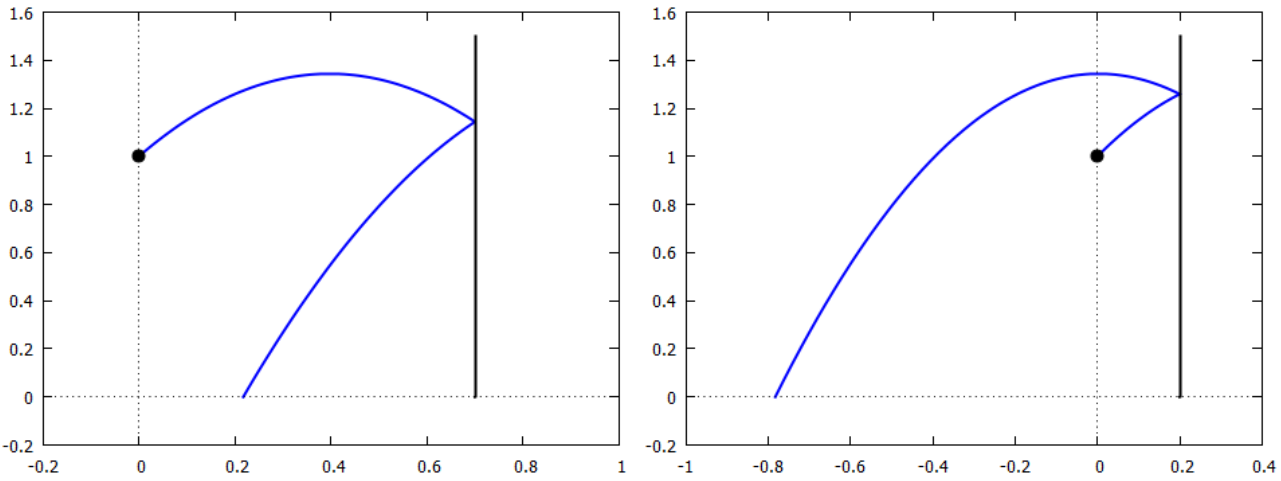


Рис. 11.3: Траєкторія м'яча при $a = 0.7$ м та $a = 0.2$ м.

Новий розділяючий параметр t_a — це час польоту до стінки, що визначається з умови $x(t) = a$. Тепер можемо записати умову для кінцевої x-координати м'яча, модуль якої і буде відстанню до хлопчика: $l = |x_a(t_{full})|$.

Надамо конкретні числові значення для побудови графіка та розрахунку l.

```
(%i16) v_0:3$ g:9.81$ %alpha:%pi/3$ h:1$
(%i18) a:0.2; t_a:(rhs(first(solve(x(t)=a,t))))$
(%i19) l:float(abs(xa(t_full))),a=0.2;
(%o19)  0.7824539819480768
```

Зобразимо траєкторію м'яча, виділивши його початкове положення (Рис.11.3).

```
(%i20) plot2d([[parametric,xa(t),ya(t),[t,0,t_full]],
               [discrete,[[a,0],[a,1.5]]],[discrete,[[0,h]]],
               [discrete,[[a,y(t_a)]]]],[x,-1,0.4],
               [style,[lines,2],[lines,2,5],[points,3,5,1]],
               [legend,false],[xlabel," "],[ylabel," "])$
```

► З гармати випустили послідовно два снаряди з однаковою швидкістю $v_0 = 300$ м/с, один під кутом $\alpha = \pi/3$, інший під кутом $\beta = \pi/5$. Знайти інтервал між пострілами, при якому снаряди зіткнуться між собою (опір повітря відсутній).

Задамо у **Maxima** початкові дані: проєкції швидкостей на координатні осі для першого та другого снарядів.

```
(%i1)  ratprint:false;
(%o1)   false

(%i2)  v1_x(t):=v0*cos(%alpha);
(%o2)   v1_x(t) := v0 · cos(α)

(%i3)  v1_y(t):=v0*sin(%alpha)-g*t;
(%o3)   v1_y(t) := v0 · sin(α) − g · t

(%i4)  v2_x(t):=v0*cos(%beta);
(%o4)   v2_x(t) := v0 · cos(β)

(%i5)  v2_y(t):=v0*sin(%beta)-g*t;
(%o5)   v2_y(t) := v0 · sin(β) − g · t
```

Координати снарядів отримуються інтегруванням відповідних проєкцій швидкостей.

```
(%i6)  x1(t):=integrate(v1_x(%xi),%xi,0,t);
(%o6)   x1(t) := ∫_0^t v1_x(ξ) dξ

(%i7)  y1(t):=integrate(v1_y(%xi),%xi,0,t);
(%o7)   y1(t) := ∫_0^t v1_y(ξ) dξ
```

```
(%i8)  x2(t):=integrate(v2_x(%xi),%xi,0,t);
```

$$(\%o8) \quad \mathrm{x2}\,(t) := \int_0^t \mathrm{v2\_x}\,(\xi)\,d\xi$$

```
(%i9)  y2(t):=integrate(v2_y(%xi),%xi,0,t);
```

$$(\%o9) \quad \mathrm{y2}\,(t) := \int_0^t \mathrm{v2\_y}\,(\xi)\,d\xi$$

Знайдемо загальні часи руху двох тіл, які шукаються з умов $y(t) = 0$:

```
(%i10) T_1:rhs(first(solve(y1(t)=0,t)));
```

$$(\%o10) \quad \frac{2 \cdot \sin(\alpha) \cdot v0}{g}$$

```
(%i11) T_2:rhs(first(solve(y2(t)=0,t)));
```

$$(\%o11) \quad \frac{2 \cdot \sin(\beta) \cdot v0}{g}$$

Нехай після пострілу першого снаряду минув час τ, після якого здійснили другий постріл. Позначимо час руху до зіткнення t. Отримаємо систему двох рівнянь, один з коренів якої τ дає розв'язок задачі: $\{x_1(\tau + t) = x_2(t); y_1(\tau + t) = y_2(t)\}$.

```
(%i12) sol1:solve([x1(%tau+t)=x2(t),y1(%tau+t)=y2(t)],[%tau,t]);
```

$$(\%o12) \quad [[\tau = -\frac{(2 \cdot \cos(\alpha) \cdot \sin(\beta) - 2 \cdot \sin(\alpha) \cdot \cos(\beta)) \cdot v0}{(\cos(\beta) + \cos(\alpha)) \cdot g},$$
$$t = -\frac{\left(2 \cdot \cos(\alpha)^2 \cdot \sin(\beta) - 2 \cdot \cos(\alpha) \cdot \sin(\alpha) \cdot \cos(\beta)\right) \cdot v0}{\left(\cos(\beta)^2 - \cos(\alpha)^2\right) \cdot g}],$$
$$[\tau = 0, t = 0]]$$

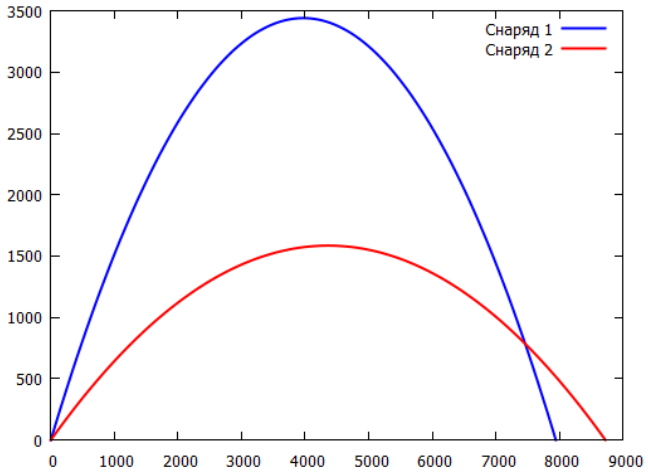


Рис. 11.4: Траєкторії руху снарядів.

Уведемо числові значення змінних та знайдемо конкретний вигляд для τ.

```
(%i16) v0:300$ g:9.81$ %alpha:%pi/3$ %beta:%pi/5$
(%i17) float(sol1[1]),v0:300, g:9.81, %alpha:%pi/3, %beta:%pi/5;
```

$$(\%o17) \quad [\tau = 19.00422913577016, t = 30.74948867166716]$$

Як видно, час між пострілами становить 19 с, а загальний час до зіткнення 50 с. Зобразимо тепер траєкторії руху снарядів (Рис.11.4).

```
(%i19) plot2d([[parametric,x1(t),y1(t),[t,0,T_1]],
               [parametric,x2(t),y2(t),[t,0,T_2]]],
       [style,[lines,2],[lines,2]],[legend,"Снаряд 1","Снаряд 2"]);
```

Побудований графік надає нам досить цінної додаткової інформації. Видно, що дальність польоту двох снарядів приблизно однакова, 8-9 км, тоді як максимальна висота підняття відрізняється більш ніж удвічі: 3.5 км у першого проти 1.5 км у другого снаряда.

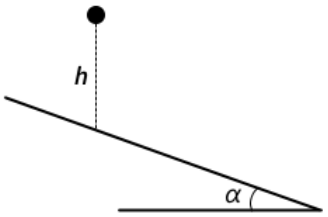


Рис. 11.5: Кулька на похилій поверхні.

► Кулька почала падати зі стану спокою на гладку поверхню, нахилену під кутом α до горизонту. Пролетівши відстань h, вона відбилась від поверхні з коефіцієнтом пружності k. На якій відстані від місця падіння кулька відіб'ється другий раз?

Із Рис.11.5 видно, що кут нахилу площини α є кутом між прямою падіння кульки та нормаллю до площини похилої у точці дотику. Це наводить на думку запровадити систему координат, вісь x якої буде напрямлена вздовж похилої, а вісь y вздовж нормалі, причому початок помістимо у точку падіння (Рис.11.6). Тоді прискорення вільного падіння $\vec{g}$ буде мати вже дві компоненти, вздовж обох осей:

$$g_x = g\sin\alpha; \quad g_y = -g\cos\alpha.$$

Позначимо швидкість, з якою кулька досягне площини, через v_0. Вона знаходиться із закону збереження енергії $\frac{mv^2}{2} = mgh$. Після першого відбиття внаслідок втрати енергії y-компоненту швидкості необхідно домножити на k, тоді як x-компонента залишається практично незмінною (це проявляється зокрема у тому, що кут відбивання м'яча від землі при повздовжньому ударі стає щоразу меншим).

Отже задача звелась до наступної: дослідити рух тіла, що починає рухатись із початковою швидкістю v_0, під кутом до горизонту $90°-\alpha$, на яке діє сила тяжіння із проєкціями прискорення g_x та g_y.

Запишемо значення відповідних швидкостей та інтегруванням знайдемо координати.

```
(%i1)  ratprint:false;
```
(%o1) false

```
(%i2)  g_x:g*sin(%alpha);
```
(%o2) $\sin(\alpha)\cdot g$

```
(%i3)  g_y:-g*cos(%alpha);
```
(%o3) $-\cos(\alpha)\cdot g$

```
(%i4)  v1_x(t):=v_0*cos(%pi/2-%alpha)+g_x*t;
```
(%o4) $\mathrm{v1\_x}(t) := v\_0\cdot\cos\left(\frac{\pi}{2}-\alpha\right)+g\_x\cdot t$

```
(%i5)  v1_y(t):=k*(v_0*sin(%pi/2-%alpha)+g_y*t);
```
(%o5) $\mathrm{v1\_y}(t) := k\cdot\left(v\_0\cdot\sin\left(\frac{\pi}{2}-\alpha\right)+g\_y\cdot t\right)$

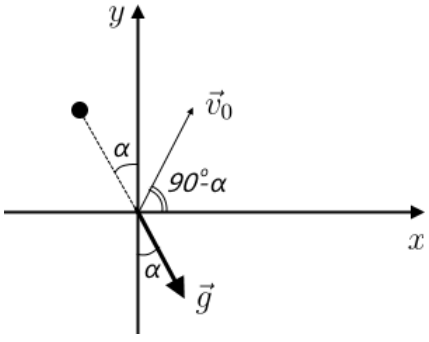


Рис. 11.6: Система координат для кульки.

```
(%i6)  x1(t):=integrate(v1_x(%xi),%xi,0,t);
```

$$(\%o6)\quad \mathrm{x1}\,(t) := \int_0^t \mathrm{v1\_x}\,(\xi)\,d\xi$$

```
(%i7)  y1(t):=integrate(v1_y(%xi),%xi,0,t);
```

$$(\%o7)\quad \mathrm{y1}\,(t) := \int_0^t \mathrm{v1\_y}\,(\xi)\,d\xi$$

```
(%i8)  x1(t);
```

$$(\%o8)\quad \frac{2\cdot\sin(\alpha)\cdot t\cdot v\_0+\sin(\alpha)\cdot g\cdot t^2}{2}$$

```
(%i9)  y1(t);
```

$$(\%o9)\quad \frac{k\cdot(2\cdot\cos(\alpha)\cdot t\cdot v\_0-\cos(\alpha)\cdot g\cdot t^2)}{2}$$

Тепер обчислимо час руху на першій арці, який знаходиться з умови $y_1(t) = 0$.

```
(%i10) sol2:solve(y1(t)=0,t);
```

$$(\%o10)\quad [t=\frac{2\cdot v\_0}{g}, t=0]$$

```
(%i11) T1:rhs(first(sol2));
```

$$(\%o11)\quad \frac{2\cdot v\_0}{g}$$

Знайдемо початкову швидкість v_0:

```
(%i12) v_0:rhs(second(solve(m*v^2/2=m*g*h,v)));
```

$$(\%o12)\quad \sqrt{2}\cdot\sqrt{g\cdot h}$$

Відстань, яку пролетіла кулька:

```
(%i13) L1:x1(T1),v0=sqrt(2)*sqrt(g*h);
```

$$(\%o13)\quad 8\cdot\sin(\alpha)\cdot h$$

Числове значення відстані:

```
(%i14) ev(L1,%alpha=%pi/8,h=0.5),numer;
```

$$(\%o14)\quad 1.530733729460359$$

Задамо початкові значення для побудови графіка:

```
(%i17) h:0.5$ g:9.81$ %alpha:%pi/8$ k=0.9$
```

Числове значення часу руху вздовж першої арки:

```
(%i18) T1a:ev(T1),h=0.5, g=9.81, %alpha=%pi/8, k=0.9,numer;
```

$$(\%o18)\quad 0.638550856814101$$

Тепер можемо зобразити рух кульки. Для побудови графіка використаємо команду `transform`, яка здійснює перетворення координат за заданим сценарієм (у нашому випадку це поворот кривої руху та площини відбиття на кут α).

```
(%i19) load(draw);
```

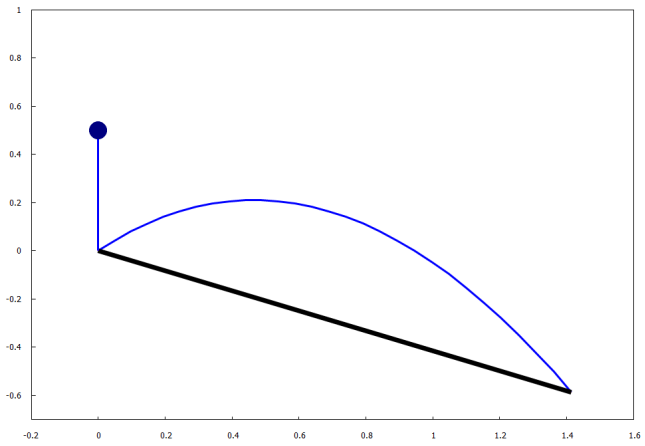


Рис. 11.7: Перша арка руху кульки при відбитті від похилої площини.

```
(%i20) figure1:gr2d(
           line_width = 3,
           proportional_axes = xy,
           transform  = [cos(%alpha)*x + sin(%alpha)*y,
                         -sin(%alpha)*x + cos(%alpha)*y, x, y],
       parametric(x1(t),y1(t),t,0,T1a),
           color = black, line_width = 7,
       polygon([[0,0],[x1(T1a),0]]),
           color = blue, line_width = 3,
           transform  = [cos(0)*x + sin(0)*y,
                         -sin(0)*x + cos(0)*y, x, y],
       polygon([[0,0.5],[0,0]]),
           point_size = 4, point_type = filled_circle, color = navy,
           transform  = [cos(0)*x + sin(0)*y,
                         -sin(0)*x + cos(0)*y, x, y],
       points([[0,0.5]]),
       xrange = [-0.2,1.6],
       yrange = [-0.7,1]
       )$
       draw(figure1), %alpha=%pi/8$
```

Результат зображений на Рис.11.7.

Можна також здійснити розрахунок траєкторії польоту на другій арці. Для цього ввідними даними будуть проекції швидкості $v_x = v_{1x}(T_1)$ та $v_y = kv_{1y}(T_1)$, при інтегруванні координат беруться початкові значення $x_1(T_1)$, $y_1(T_1)$. Розрахунки провадяться за такою ж схемою. Цікавою особливістю такої системи є те, що час руху по другій арці за умови $k = 1$ буде таким же, як і по першій, хоч відстань польоту буде більшою, про що читачеві рекомендується переконатись самостійно.

▶ Точка A знаходиться на ободі колеса радіуса $r = 0.5$ м, яке котиться без проковзування по горизонтальній поверхні зі швидкістю $v = 1$ м/с (Рис.11.8). Знайти модуль та напрям вектора прискорення, а також шлях S, який ця точка проходить між двома послідовними доторками до поверхні.

Точка A бере участь у двох рухах: обертання відносно центра колеса C, та рівномірний рух вздовж осі Ox. Цю обставину можна змоделювати так: загальний радіус-вектор точки є сумою двох радіус-векторів — точки A відносно точки C та точки C відносно

точки O.

Додамо ще кілька зауважень. По-перше, стандартний напрям повороту кута (проти годинникової стрілки) та напрям повороту колеса є протилежними, тому необхідно у формулах вживати $-\omega t$; по-друге, початкове положення точки задається кутом $\pi/2$. Ці міркування є неважливими для розв'язку «на папері», оскільки кінцевий результат не залежить від цих обставин, однак при моделюванні руху точки їх необхідно враховувати.

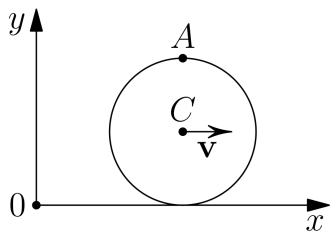


Рис. 11.8: Точка на обертальному колесі.

```
(%i1)  ratprint:false;
```

(%o1) false

```
(%i2)  R_CA(t):=[r*cos(-%omega*t+%pi/2),r*sin(-%omega*t+%pi/2)];
```

$$(\%o2)\quad \mathrm{R\_CA}\,(t) := [r\cdot\cos\left(-\omega\cdot t+\frac{\pi}{2}\right), r\cdot\sin\left(-\omega\cdot t+\frac{\pi}{2}\right)]$$

```
(%i3)  R_OC(t):=[v*t,r];
```

$$(\%o3)\quad \mathrm{R\_OC}\,(t) := [v\cdot t, r]$$

```
(%i4)  R_OA(t):=R_CA(t)+R_OC(t);
```

$$(\%o4)\quad \mathrm{R\_OA}\,(t) := \mathrm{R\_CA}\,(t) + \mathrm{R\_OC}\,(t)$$

Знайдемо повну швидкість та прискорення точки A:

```
(%i5)  v_A(t):=diff(R_OA(t),t);
```

$$(\%o5)\quad \mathrm{v\_A}\,(t) := \mathrm{diff}\,(\mathrm{R\_OA}\,(t), t)$$

```
(%i6)  v_A(t);
```

$$(\%o6)\quad [v+\omega\cdot r\cdot\cos(\omega\cdot t), -\omega\cdot r\cdot\sin(\omega\cdot t)]$$

```
(%i7)  a_A(t):=diff(v_A(t),t);
```

$$(\%o7)\quad \mathrm{a\_A}\,(t) := \mathrm{diff}\,(\mathrm{v\_A}\,(t), t)$$

```
(%i8)  a_A(t);
```

$$(\%o8)\quad [-\omega^2\cdot r\cdot\sin(\omega\cdot t), -\omega^2\cdot r\cdot\cos(\omega\cdot t)]$$

Модуль прискорення $|\vec{a}| = \sqrt{a_1^2 + a_2^2}$:

```
(%i9)  a=trigsimp(sqrt(a_A(t)[1]^2+a_A(t)[2]^2));
```

$$(\%o9)\quad a = \omega^2\cdot|r|$$

Як бачимо, отримали добре знаний результат $a = \omega^2 r = v^2/r$, який справедливий при рівномірному русі центра колеса вздовж осі Ox. Знайдемо тепер шлях, пройдений точкою A, він рівний інтегралу від швидкості за період обертання $T = 2\pi/\omega$:

```
(%i10) S:integrate(sqrt((v_A(t)[1])^2+(v_A(t)[2])^2),t,0,2*%pi/%omega),
                    v=%omega*r;
Is ω positive or negative?pos;
```

Is $\cos(new - var - 34506) + 1.0$ positive or negative?pos;

(%o10) $8 \cdot |r|$

Надамо параметрам певних значень для побудови графіку траєкторії (Рис.11.9) та числового розрахунку S.

```
(%i13) v:1$ r:0.5$ %omega:2$
(%i14) plot2d([parametric,R_OA(t)[1],R_OA(t)[2],[t,0,1.5*%pi]],
              [x,-0.2,5],[same_xy],[style,[lines,2]],
              [legend,false],[xlabel,""],[ylabel,""]);
```

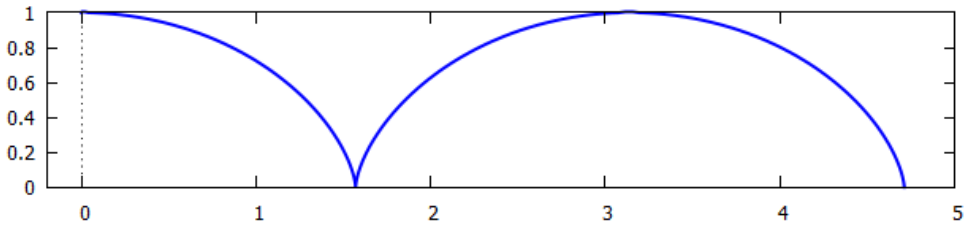


Рис. 11.9: Траєкторія руху точки A на колесі.

```
(%i15) S, r=0.5;
```

(%o18) 4.0

Отримана траєкторія руху — це циклоїда.

Зробимо ще анімацію руху точки по колесу (Рис.11.10). Змоделюємо його колом із змінним центром у точці $O_1(x - vt; y - r)$.

```
(%i16) with_slider_draw(
       k,makelist(1.5*%pi*i/40,i,0,40),nticks=100,
           point_type = filled_circle, point_size = 3, color = navy,
       points([R_OA(k)[1]],[R_OA(k)[2]]),
           line_width = 4, color = blue,
       parametric(R_OA(t)[1],R_OA(t)[2],t,0,k),
           line_width = 3, color = black, nticks = 100,
       implicit((x-v*k)^2+(y-r)^2=r^2,x,0,1.5*%pi,y,0,1),
       xrange=[0,1.5*%pi], yrange=[-0.2,1.2],
       xaxis = true, proportional_axes = xy, delay = 2)$
```

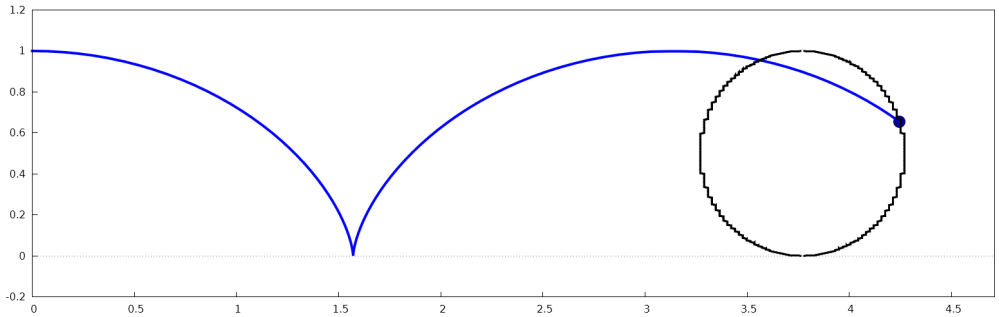


Рис. 11.10: Кадр анімації руху колеса.

Розділ 12

Еволюційні процеси

12.1 Особливі точки систем автономних ЗДР

Найбільш цікаві результати щодо моделювання властивостей динамічних систем отримані на моделях із двох диференціальних рівнянь, які допускають якісне дослідження за допомогою методу фазової площини. Розглянемо систему двох автономних (це означає, що у правій частині немає явної залежності від часу) звичайних диференціальних рівнянь загального виду

$$\begin{cases} \dot{x} = P(x, y); \\ \dot{y} = Q(x, y), \end{cases}$$

де $P(x, y), Q(x, y)$ — неперервні функції, визначені в деякій ділянці G евклідової площини. Область G може бути як необмеженою, так і скінченною. Якщо змінні (x, y) мають конкретний біологічний зміст (концентрація речовин, чисельність популяції видів тощо), найчастіше область G є позитивним квадрантом правої півплощини: $\{0 \leq x \leq \infty; 0 \leq y \leq \infty\}$. Концентрації речовин або чисельності видів можуть бути обмежені зверху об'ємом посудини або площею ареалу проживання. Тоді область значень змінних має вигляд $\{0 \leq x \leq x_0; 0 \leq y \leq y_0\}$.

Розглянемо площину з осями координат, на яких відкладено значення змінних x, y. Сукупність точок (x, y) на цій фазовій площині, положення яких відповідає станам системи в процесі зміни в часі змінних $x(t), y(t)$, називається фазовою траєкторією, а напрямки цих змін задають векторне поле. Множина фазових траєкторій при різних початкових значеннях змінних дає легко доступний для огляду та аналізу «портрет» системи. Побудова фазового портрета дозволяє зробити висновки про характер зміни величин x, y без знання аналітичних розв'язків вихідної системи рівнянь.

Завдання побудови векторного поля спрощується, якщо отримати вираз фазових траєкторій в аналітичному вигляді. Для цього розділимо друге із рівнянь системи на перше:

$$\frac{dy}{dx} = \frac{Q(x, y)}{P(x, y)}.$$

Розв'язання цього рівняння в неявному вигляді $F(x, y) = C$, де C — стала інтегрування, задає сімейство інтегральних кривих вихідної системи, які і будуть фазовими траєкторіями. Фазові траєкторії фактично будуть лініями рівня інтегральної функції F.

Нехай динамічна система знаходиться у стані рівноваги, це означає що змін у часі не відбувається: $\{\dot{x} = 0; \dot{y} = 0\}$. Розв'язавши цю систему, можна знайти рівноважні значення x та y, які називаються особливими точками. Стійкість особливих точок визначається тим, чи піде від неї точка M при малому відхиленні від стаціонарного стану (нестійка),

чи навпаки, буде наближатись (стійка). Математично стійкість особливих точок формулюється наступним чином: стан рівноваги є стійким, якщо для будь-якої заданої області відхилень від стану рівноваги D можна вказати область E, яка оточує стан рівноваги і має ту властивість, що жодна траєкторія, яка починається всередині області D, ніколи не досягне межі E.

Для великого класу систем, характер поведінки яких не змінюється при малій зміні виду рівнянь, інформацію про тип поведінки на околі стаціонарного стану можна отримати, досліджуючи не вихідну, а спрощену лінеаризовану систему. Розглянемо систему двох лінійних рівнянь

$$\begin{cases} \dot{x} = ax + by; \\ \dot{y} = cx + dy. \end{cases}$$

Загальний розв'язок шукатимемо у вигляді $x = Ae^{\lambda t}; y = Be^{\lambda t}$. Підстановка у систему нам дає

$$\begin{cases} \lambda A = aA + bB; \\ \lambda B = cA + dB, \end{cases} \qquad \begin{cases} (a-\lambda)A + bB = 0; \\ cA + (d-\lambda)B = 0. \end{cases}$$

Алгебрична система рівнянь такого типу має ненульовий розв'язок лише в тому випадку, якщо її детермінант, складений з коефіцієнтів при невідомих, рівний нулю:

$$\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0.$$

Розкриваючи цей визначник, отримаємо характеристичне рівняння системи

$$\lambda^2 - (a+d)\lambda + (ad - bc) = 0,$$

яке дає нам корені

$$\lambda_{1,2} = \frac{a+d}{2} \pm \sqrt{\frac{(a+d)^2 - 4(ad-bc)}{4}}.$$

Характеристика стійкості даної системи буде залежати від вигляду величин λ:

∘ дійсні $\lambda_1 < 0; \lambda_2 < 0$ — особлива точка типу *стійкий вузол*, фазові траєкторії параболічного типу, які сходяться до вузла;

∘ дійсні $\lambda_1 > 0; \lambda_2 > 0$ — особлива точка типу *нестійкий вузол*, фазові траєкторії параболічного типу, які розходяться від вузла;

∘ дійсні $\lambda_1 > 0; \lambda_2 < 0$, або $\lambda_1 < 0; \lambda_2 > 0$ — особлива точка типу *сідло*, вона завжди нестійка, фазові траєкторії гіперболічного типу, які оминають вузол і розходяться на нескінченність;

∘ комплексно спряжені $\lambda_1 = \alpha + i\beta; \lambda_2 = \alpha - i\beta$, $\mathrm{Re}\ \lambda = \alpha < 0$ — особлива точка типу *стійкий фокус*, фазові траєкторії спірального типу, які сходяться до вузла;

∘ комплексно спряжені $\lambda_1 = \alpha + i\beta; \lambda_2 = \alpha - i\beta$, $\mathrm{Re}\ \lambda = \alpha > 0$ — особлива точка типу *нестійкий фокус*, фазові траєкторії спірального типу, які розходяться від вузла;

∘ чисто уявні $\lambda_1; \lambda_2$ — особлива точка типу *центр*, вона завжди стійка, фазові траєкторії еліптичного типу, які обертаються навколо вузла.

П'ять типів стану рівноваги жорсткі, їх характер не змінюється за досить малих змін правих частин вихідної системи рівнянь. При цьому малими мають бути зміни не лише правих частин, а й їх похідних першого порядку. Шостий стан рівноваги — центр — нежорсткий. При малих змінах параметрів правої частини рівнянь він перетворюється на стійкий або нестійкий фокус.

Якщо праві частини вихідної системи є нелінійними виразами, необхідно провести процедуру лінеаризації рівнянь. Запровадимо узагальнену координату $z = \begin{pmatrix} x \\ y \end{pmatrix}$ та узагальну функцію $F(z) = \begin{pmatrix} Q(z) \\ P(z) \end{pmatrix}$ Тоді вихідна система приймає вигляд

$$\dot{z} = F(z).$$

Нехай точка z_0 є особливою точкою вихідної системи, що означає $F(z_0) = 0$. Ми припускаємо, що функції f і g принаймні C^1-гладкі (тобто мають неперервні частинні похідні) і отже відображення F є диференційовним. Розкладемо його поблизу особливої точки:

$$F(z) = F(z_0) + \left.\frac{\partial F}{\partial z}\right|_{z=z_0} \cdot (z - z_0) + o(|z - z_0|).$$

Тут $\frac{\partial F}{\partial z} = \mathrm{J}$ — матриця Якобі відображення F, тобто

$$\mathrm{J} = \begin{pmatrix} f'_x(x_0, y_0) & f'_y(x_0, y_0) \\ g'_x(x_0, y_0) & g'_y(x_0, y_0) \end{pmatrix}.$$

Відкинувши малий доданок $o(|z - z_0|)$ (поблизу особливої точки він практично нульовий), та позначивши $\{x - x_0 = u, y - y_0 = v\}$ (фактично це означає перенесення початку координат в особливу точку), можна записати

$$\dot{z} = \mathrm{J} \cdot (z - z_0), \quad \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \mathrm{J} \cdot \begin{pmatrix} u \\ v \end{pmatrix}.$$

Отримали тепер систему лінійних рівнянь, яку вже можна аналізувати описаними вище методами. Як бачимо, тип особливої точки залежить від власних значень матриці Якобі лінеаризованої системи у самих цих точках.

12.2 Логістичне рівняння

Загалом питання чисельності населення було розглянуте ще Томасом Мальтусом (1798), який запропонував прийняти збільшення кількості населення пропорційним до самого населення: $\frac{dN}{dt} = kN$, що після інтегрування дає експоненціальний закон зростання $N = N_0 e^{kt}$. Ці прості міркування склали велике враження на наукову спільноту світу, ставши предметом обговорення вчених, філософів, політиків. Достатньо лиш відмітити, що під їх впливом писали свої роботи творці сучасної біології та економіки Ч. Дарвін та Дж. Кейнс. Звісно, така груба модель з одним параметром не може гарно описувати реальність, тому її необхідно було ускладнювати та розвивати.

12.2.1 Неперервний випадок: рівняння Вергалста

У довгостроковій перспективі модель Мальтуса не виглядає правдоподібною: вона ігнорує обмежувальні фактори, такі як розповсюдження хвороб або брак ресурсів. Просту модифікацію, яка враховує ці фактори, можна отримати заміною сталого k на параметр, залежний від N: $k = a - bN$, тоді рівняння приймає вигляд

$$\frac{dN(t)}{dt} = (a - bN(t))N(t),$$

де $a > b > 0$. Розділивши на a та здійснивши заміну $N(t) = \frac{a}{b}x(t)$, $a = k$, отримаємо

$$\frac{dx(t)}{dt} = kx(t) \cdot (1 - x(t)),$$

де k — позитивна константа, що характеризує ступінь зростання. Це рівняння, назване логістичним, запропонував П. Вергалст у 1838 р.

Застосуємо **Maxima** щоб його розв'язати.

```
(%i1) deqn1:'diff(x,t)=k*x*(1-x);
```

$$(\%o1) \quad \frac{d}{dt} \cdot x = k \cdot (1 - x) \cdot x$$

```
(%i2) ode2(deqn1,x,t);
```

$$(\%o2) \quad \frac{\log(x) - \log(x-1)}{k} = t + \%c$$

```
(%i3) ic1(%,t=0,x=x0),logcontract;
```

$$(\%o3) \quad \frac{\log\left(\frac{x}{x-1}\right)}{k} = \frac{\log\left(\frac{x0}{x0-1}\right) + k \cdot t}{k}$$

```
(%i4) solve(%,x);
```

$$(\%o4) \quad [x = \frac{e^{k \cdot t} \cdot x0}{(e^{k \cdot t} - 1) \cdot x0 + 1}]$$

Отримали розв'язок, що залежить від x_0 та k. Перетворимо його у функцію від t.

```
(%i5) define(x(t),rhs(first(%o4)));
```

$$(\%o5) \quad \mathrm{x}(t) := \frac{e^{k \cdot t} \cdot x0}{(e^{k \cdot t} - 1) \cdot x0 + 1}$$

Поглянувши на вигляд одержаної функції, можна відмітити, що вона прямує до 1 при $t \to \infty$. Зобразимо це графічно, надавши різні значення для k, та покладемо $x_0 = 0.1$.

```
(%i6) plot2d(makelist(subst([k=d*0.3,x0=0.1],x(t)),d,1,5),
      [t,0,15],[style,[lines,2]],
      [legend,"k=0.3","k=0.6","k=0.9","k=1.2","k=1.5"],
      [gnuplot_preamble,"set key right bottom"]);
```

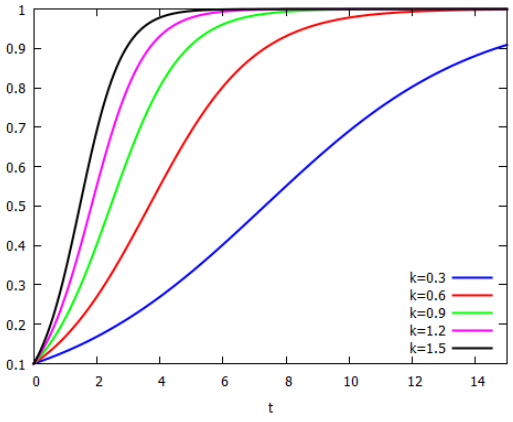


Рис. 12.1: Логістичні криві при різних k.

Чим більше k, тим швидше логістична крива виходить на плато (досягає свого максимального значення та фактично не змінюється). В 1920 році американський вчений Р. Пірл провів широкомасштабні дослідження з підрахування чисельності плодових мух в лабораторних умовах. Всі вони підтвердили S-подібний закон зростання популяції, тому логістичне рівняння вважається досить точною моделлю чисельності, але за умови нескінченних зовнішніх ресурсів та відсутності конкуренції між представниками групи. Цю криву можна застосувати і для опису динаміки народонаселення, якщо врахувати, що розвиток продуктивності та охорони здоров'я змінює величину параметрів, тому це призводить до перескоку на нову криву, із новими значеннями k та x_0.

12.2.2 Дискретний випадок: рівняння Мея

Рівняння Вергалста не видозмінювалось аж до 1976 р., коли математик Р. Мей вирішив застосувати його для моделювання кількості комах чи інших тварин, які розмножуються не неперервно протягом року, а лише у певні часові проміжки, в цілком визначені дні року (взимку попереднє покоління комах повністю вимирає, а влітку з відкладених яєць з'являється нове покоління). Тоді виникає дискретне рівняння вигляду

$$x_{n+1} = rx_n(1 - x_n).$$

Індекс n відіграє роль номера року, x_n — відносна чисельність популяції. Функція $y = x(1-x)$ є квадратичною, з вершиною у точці $(0.5; 0.25)$, тому умова $0 < x_n < 1$ зумовлює для r проміжок $0 < r < 4$.

Здавалося б, перетворення неперервного рівняння на дискретне нічого особливого не змінює у поведінці розв'язків, оскільки останні є підмножиною перших. Однак виявилось, що це враження оманливе, і перехід до дискретизації викриває нові, цікаві особливості моделі.

Знайдемо точку рівноважного стану — тобто такого, при якому чисельність популяції залишається незмінною. Позначимо її буквою c.

```
(%i1)  solve(c=r*c*(1-c),c);
```

$$(\%o1) \quad [c = \frac{r-1}{r}, c = 0]$$

Із теорії неперервного логістичного рівняння можна було б очікувати, що послідовність x_n буде обмежена зверху та при $n \to \infty$ прямуватиме до c, однак такий сценарій буде відбуватись не для всіх r.

Запровадимо тепер оператор еволюції $F(r, x) = rx(1 - x)$, та використаємо пакет `dynamics` у **Maxima**, з яким ми мали справу раніше при числовому інтегруванні ЗДР. Цей пакет містить інструменти для дослідження еволюції систем у різних представленнях, фракталів та ін. Команда

`evolution(F,x0,n,opts)` — зображає $n + 1$ точку на двовимірному графіку, де x-координати точок є цілими числами $0, 1, 2, ...n$ а y-координати – відповідні значення $x(n)$ послідовності, визначеної за рекурентним співвідношенням $x_{n+1} = F(x_n)$. Для прикладу, при $F = \cos x$ та $x_0 = 2$ будемо мати $x_4 = \cos(\cos(\cos(\cos 2)))$.

Початкове значення x_0 оберемо випадковим чином. Будемо надавати r різні значення та дослідимо поведінку x_n.

```
(%i2)  F(r,x):=r*x*(1-x);
```

$$(\%o2) \quad \mathrm{F}\,(r, x) := r \cdot x \cdot (1 - x)$$

```
(%i3)  load(dynamics);
(%i4)  x0:random(1.0);
```

$$(\%o4) \quad 0.9138095996128959$$

```
(%i5)  evolution(F(0.25,x),x0,15);
```

Отримане зображення на Рис.12.2 показує, що популяція вимирає, причому ця обставина ніяк не залежить від початкового значення x_0. Загалом в залежності від значення r можливі наступні сценарії поведінки розв'язків:

- при $0 < r \leq 1$ розв'язки ідуть до нуля, тобто популяція зникає при будь-якому значенні x_0 (Рис.12.2);

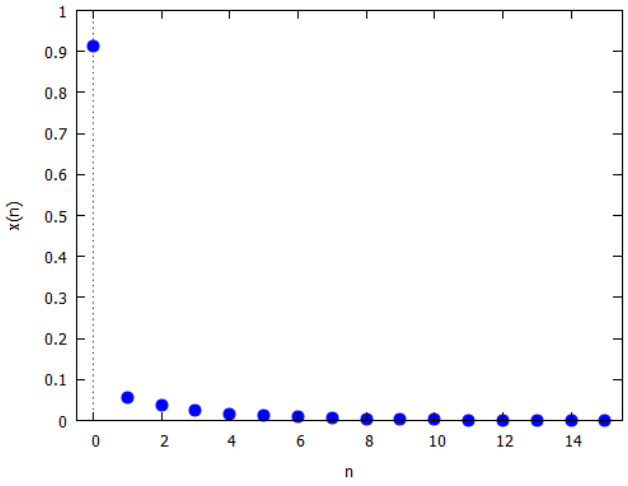


Рис. 12.2: $0 < r \leq 1$.

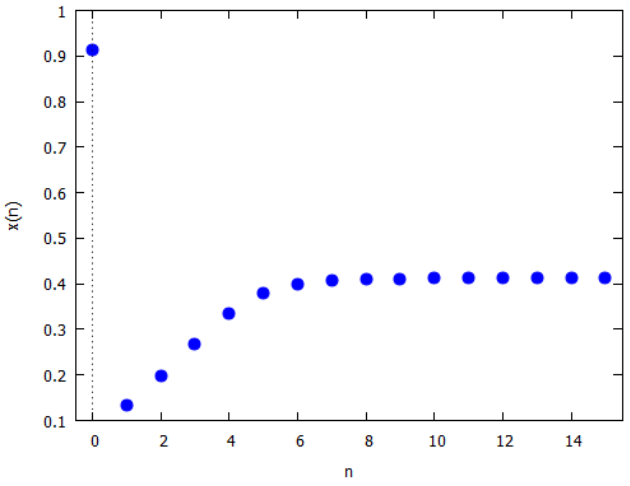


Рис. 12.3: $1 < r \leq 2$.

- при $1 < r \leq 2$ розв'язки швидко виходять на плато рівноважного значення $c = \frac{r-1}{r}$ незалежно від x_0 (Рис.12.3);

```
(%i6)  evolution(F(1.7,x),x0,15);
```

- при $2 < r \leq 3$ розв'язки теж виходять на рівноважне значення $c = \frac{r-1}{r}$, при цьому коливаючись навколо нього (Рис.12.4);

```
(%i7)  evolution(F(2.7,x),x0,15);
```

- при $3 < r \leq 3.44949$ кількість популяції осцилює поблизу двох значень, котрі майже не залежать від початкового x_0. Ці значення залежні від r: $x_{\pm} = \frac{1}{2r}\left(r+1 \pm \sqrt{(r-3)(r+1)}\right)$ (Рис.12.5);

```
(%i8)  evolution(F(3.25,x),x0,25);
```

- при $3.44949 < r \leq 3.54409$ кількість популяції осцилює вже поблизу чотирьох значень, які є коренями рівняння 4-го степеня і знаходяться числово (Рис.12.6);

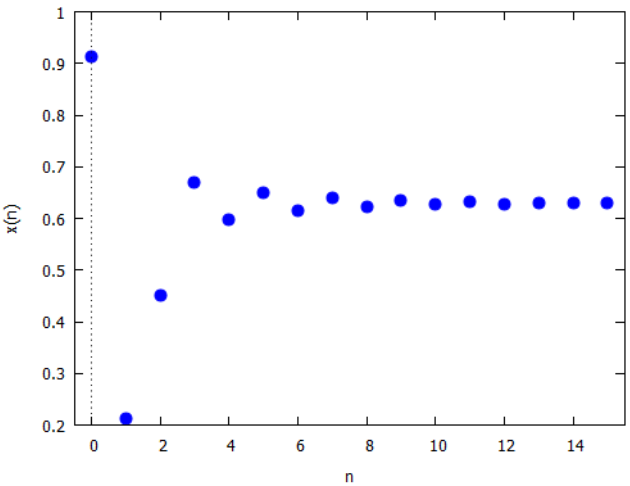


Рис. 12.4: $2 < r \leq 3$.

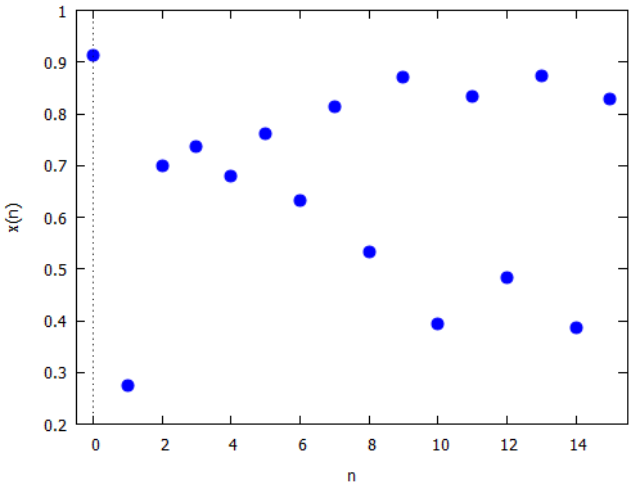


Рис. 12.5: $3 < r \leq 3.44949$.

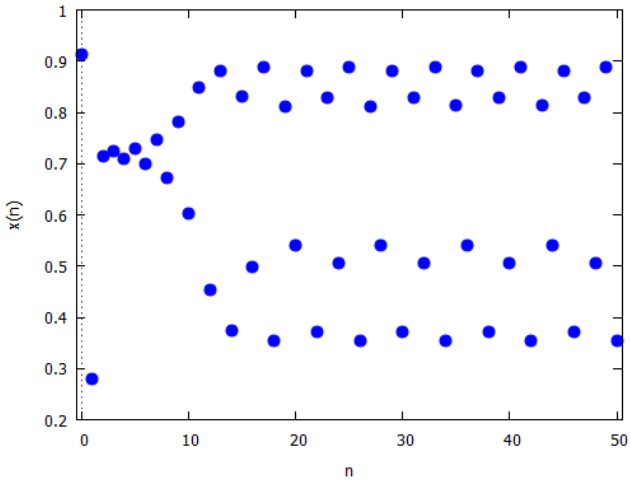


Рис. 12.6: $3.44949 < r \leq 3.54409$.

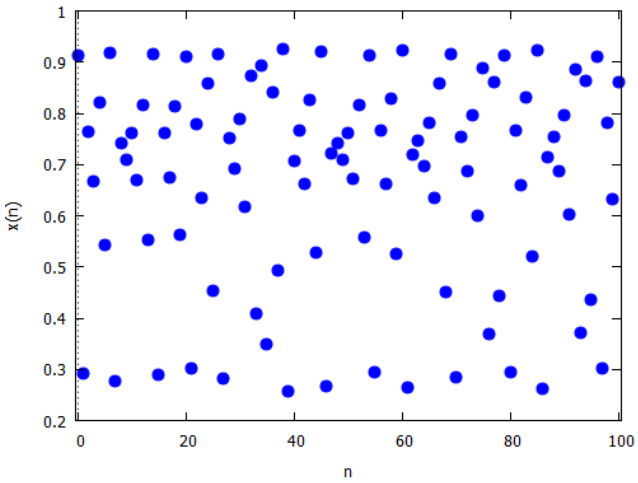


Рис. 12.7: $r > 3,56995$.

```
(%i9)  evolution(F(3.5,x),x0,50);
```

- при $3.54409 < r \leq 3.56995$ кількість популяції осцилює поблизу 8, 16, 32,... значень. Довжини інтервалів параметру r, які задають коливання певного періоду, швидко зменшуються, але відношення довжин двох послідовних біфуркаційних інтервалів наближається до сталої Фейгенбаума $\delta = 4.669201$;

- при $r > 3,56995$ ніякого впорядкованого процесу вже немає (Рис.12.7). З майже всіх початкових умов неможливо отримати коливання скінченного періоду. Невеликі зміни початкової чисельності популяції x_0 дають дуже відмінні результати еволюції, що є основною характеристикою хаосу.

```
(%i10) evolution(F(3.7,x),x0,100);
```

Біологічний зміст роздвоєння кількості популяції та її коливання поблизу якогось стаціонарного значення наступний: за сприятливих природних умов, коли r перевищує 3, комахи починають інтенсивно розмножуватись і виходять на більший корінь роздвоєного періоду. Це призводить до значного збільшення їх чисельності, що при обмежених ресурсах викликає високу смертність. Отже на наступний рік кількість комах зменшується і виходить на менший корінь. Цей процес за ідеально тотожних умов буде завжди повторюватись. Звісно, у природі такого точного дублювання не спостерігається, тому і чисельність може коливатись поблизу різних віток стаціонарних значень: кілька років популяція перебуває на одній вітці, потім перестрибує на іншу. Суттєві зміни природних умов можуть сильно змістити чисельність із верхніх стаціонарних гілок на нижні, або навпаки.

Ми бачили, що поведінка орбіт для логістичної карти залежить від параметра r. Фактично, починаючи з $r = 3$ і далі, кількість періодичних точок зростає у вигляді «каскаду подвоєння періоду». Вивчати зміну структури періодичних орбіт можна за допомогою так званих біфуркаційних діаграм. У них значення параметра r представляються на горизонтальній осі. Для кожної з цих величин визначаються значення відповідних притягуючих стаціонарних точок. **Maxima** реалізує ці графіки вбудованою командою

`orbits(F,x0,n1,n2,[r,r1,r2,rstep],opts)` — зображає діаграму орбіт для сімейства одновимірних дискретних динамічних систем з одним параметром r, який приймає значення в інтервалі від `r1` до `r2` з кроком `rstep`; `n1` та `n2` позначають номери ітерацій застосування функції `F` на `x0`.

Щоб спостерегти, для прикладу, 8 розгалужень, будемо змінювати r від 2.5 до 3.55 (Рис.12.8).

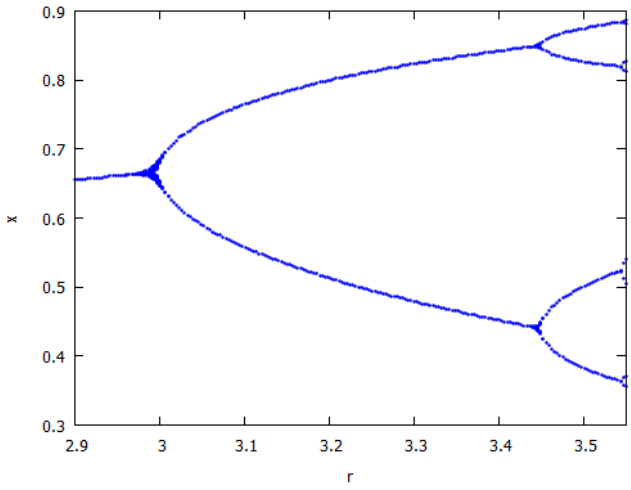


Рис. 12.8: Розгалуження періоду 8.

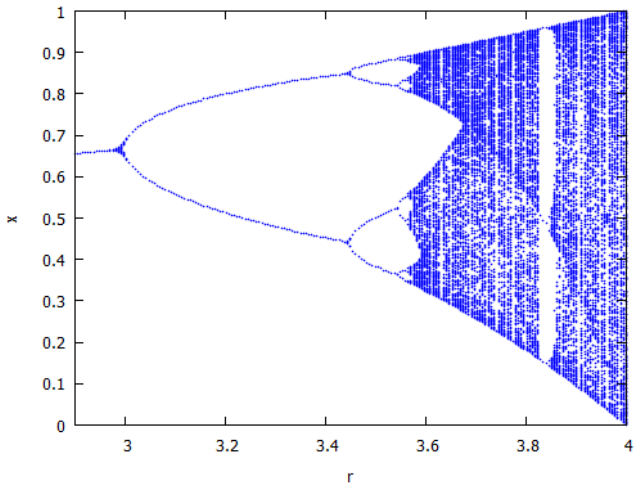


Рис. 12.9: Повна біфуркаційна діаграма.

```
(%i11)  orbits(F(r,x),x0,150,300,[r,2.5,3.55],[style,[points,0.5]]);
```

Розгалуження на 8 віток видно справа на самому краю. Можна було б чіткіше зобразити їх, змінивши масштаб та обраховувати r від 3.5 до 3.55 (залишимо це читачу). Повна біфуркаційна діаграма, зі зміною r до крайнього значення 4, відображена на Рис.12.9. Як видно, досить несподівано з'являються острівки порожніх місць, тобто існують такі значення чисельності, котрих популяція ніколи не зможе досягнути.

```
(%i12)  orbits(F(r,x),x0,150,300,[r,2.9,4],[style,[points,0.2]]);
```

Отримана таким чином повна діаграма має особливу властивість: вона самоподібна. Це означає, що після зміни масштабу отримане зображення має таку саму структуру, що й вихідне. Мірою самоподібності множини є її фрактальна розмірність, яка на відміну від розмірності векторного простору, може бути нецілим числом. Тому біфуркаційна діаграма є різновидом фракталу.

12.3 Моделі взаємодіючих популяцій

Розглядаючи рівняння динаміки популяцій, ми зверталу увагу в першу чергу на людський соціум, у якого є зовнішній ресурс у вигляді їжі, та природня смертність від хвороб чи інших членів того ж соціуму. Повноцінної взаємодії з іншими популяціями істот не відбувається, оскільки людина стоїть на верхній ланці біологічного розвитку. Якщо ж розглянути який-небудь замкнутий біоценоз із взаємодією різних видів, виникають цікаві особливості поведінки їх чисельності.

12.3.1 Модель Лотки–Вольтерри

Вперше математична модель «хижак-жертва» була отримана А. Лоткою (1925), який використовував для опису динаміки взаємодіючих біологічних популяцій. Трохи пізніше і незалежно від Лотки аналогічні (і більш складні) моделі були розроблені італійським математиком В. Вольтеррою (1926), глибокі дослідження якого в галузі екологічних проблем

заклали фундамент математичної теорії біологічних популяцій або так званої математичної екології. Сама робота Вольтерри була виконана з метою пояснення циклічних змін, які спостерігаються в популяціях акул і промислових риб в Адріатичному морі.

Позначимо за x кількість травоїдних тварин (жертви, для прикладу кролики), а за y — кількість м'ясоїдних (хижаки, для прикладу лисиці). Зростання популяції жертв відповідатиме моделі Мальтуса, тобто це зростання буде експоненційним без хижаків. Якщо ж не буде кроликів, то популяція лисиць буде нульовою, оскільки їм не буде чим харчуватися. Відповідно до цих висновків та позначень, можемо записати систему диференціальних рівнянь, що виражає чисельність популяції обох видів:

$$\begin{cases} \frac{dx}{dt} = ax - pxy; \\ \frac{dy}{dt} = -by + qxy. \end{cases}$$

Ця сукупність називається системою Лотки–Вольтерри. У ній: a — швидкість зростання чисельності травоїдних без хижаків; b — швидкість скорочення чисельності м'ясоїдних без травоїдних; p і q — швидкість, з якою зустрічі хижаків з жертвами видаляють травоїдних з популяції, і швидкість, з якою ці зустрічі дозволяють хижакам додавати чисельність своєї популяції.

Із рівнянь видно, що будь-яка зміна чисельності травоїдних впливає на чисельність м'ясоїдних і навпаки, тому дві популяції слід розглядати разом. Якщо популяція травоїдних збільшується, ймовірність зустрічей хижак-жертва зростає, і, відповідно (через деякий час), зростає популяція хижаків. Але зростання популяції хижаків призводить до скорочення популяції травоїдних (також після деякої затримки), що веде до зниження чисельності потомства хижаків, а це підвищує кількість травоїдних і так далі.

Якщо покласти p та q рівними нулю (взаємодії не відбувається), для обох видів маємо експоненціальний закон зміни чисельності, тільки для кроликів він додатній, їх кількість іде до нескінченності, а для лисиць від'ємний, вони швидко вимирають.

Аналітично ця система є нерозв'язною, це можна зробити лише числовими методами. Втім, перший інтеграл її можна знайти прямим інтегруванням.

Записуємо два рівняння вихідної системи:

```
(%i1) depends([x,y],t);
```
(%o1) $[\mathrm{x}(t), \mathrm{y}(t)]$

```
(%i2) deqn1:'diff(x,t)=a*x-p*x*y;
```
(%o2) $\frac{d}{dt} \cdot x = a \cdot x - p \cdot x \cdot y$

```
(%i3) deqn2:'diff(y,t)=-b*y+q*x*y;
```
(%o3) $\frac{d}{dt} \cdot y = q \cdot x \cdot y - b \cdot y$

Спроба прямого розв'язання системи командою `desolve` призводить до появи узагальнених інтегральних функцій з перетвореннями Лапласа, тому такий розв'язок мало придатний для аналізу. Поділимо друге рівняння на перше та знайдемо перший інтеграл:

```
(%i4) eq1:ode2('diff(y,x)=rhs(deqn2)/rhs(deqn1),y,x);
```
(%o4) $a \cdot \log(y) - p \cdot y = -b \cdot \log(x) + q \cdot x + \%c$

```
(%i5) eq2:solve(eq1,%c);
```
(%o5) $[\%c = a \cdot \log(y) - p \cdot y + b \cdot \log(x) - q \cdot x]$

Запровадимо нову функцію $F(x, y)$, лінії рівня якої будуть фазовими траєкторіями системи.

```
(%i6)  F(x,y):=a*log(y)-p*y+b*log(x)-q*x;
```

$$(\%o6) \quad \mathrm{F}(x, y) := a \cdot \log(y) - p \cdot y + b \cdot \log(x) - q \cdot x$$

Для подальшого аналізу та побудови графіків задамо конкретні значення параметрів:

```
(%i7)  F1(x,y):=at(F(x,y),[a=0.2,b=0.3,p=1.5,q=2]);
```

$$(\%o7) \quad \mathrm{F1}(x, y) := \mathrm{at}(\mathrm{F}(x, y), [a = 0.2, b = 0.3, p = 1.5, q = 2])$$

```
(%i8)  F1(x,y);
```

$$(\%o8) \quad 0.2 \cdot \log(y) - 1.5 \cdot y + 0.3 \cdot \log(x) - 2 \cdot x$$

Тепер зобразимо цю функцію разом з її лініями рівня (Рис. 12.10).

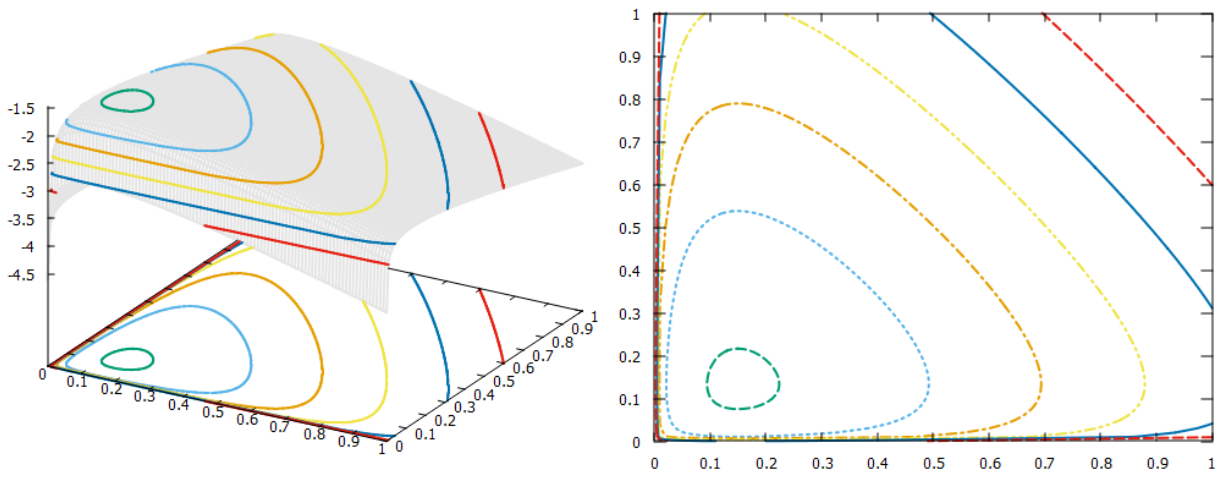


Рис. 12.10: Поверхня функції $F(x, y)$ з лініями рівня.

```
(%i9)  load(draw)$
(%i10) draw3d(
       color = gray90,
       nticks = 500,
       line_width = 2,
       contour_levels = [-3,0.3,-1.5],
       contour = both,
       xu_grid = 100,
       yv_grid = 100,
       surface_hide = true,
       explicit(F1(x,y),x,0.0001,1,y,0.0001,1))$
```

Для зручності побудови ми задали початкову точку не 0, а 0.0001, оскільки логарифм нуля не існує. Збоку розміщений графік лише проєкцій ліній рівня, який отримується такою ж командою, але зі зміною `contour = map`.

Як видно, всі фазові траєкторії є замкнутими кривими, які обертаються навколо центральної точки. Щось подібне ми вже спостерігали у випадку коливань маятника, і це наводить на думку про періодичність процесу. Для кожного зрізу інтегральної функції x та y набувають певного мінімального та максимального значення, весь час повторюючись.

Замкнуті криві не дуже схожі на еліпс, тому коливальний процес хоч і періодичний, але не синусоїдний.

Здійснимо аналіз особливих точок системи та здійснимо її лінеаризацію. Знайдемо спочатку особливі точки:

```
(%i11) sol1:solve([rhs(deqn1)=0,rhs(deqn2)=0],[x,y]);
```

(%o11) $[[x=0,y=0],[x=\frac{b}{q},y=\frac{a}{p}]]$

Як бачимо, їх дві: $M_1(0;0)$, $M_2(\frac{b}{q};\frac{a}{p})$.

```
(%i12) x0_1:rhs(sol1[1][1])$
       y0_1:rhs(sol1[1][2])$
       x0_2:rhs(sol1[2][1])$
       y0_2:rhs(sol1[2][2])$
```

Матриця Якобі:

```
(%i13) J:matrix([diff(rhs(deqn1),x),diff(rhs(deqn1),y)],
                [diff(rhs(deqn2),x),diff(rhs(deqn2),y)]);
```

(%o13) $\begin{pmatrix} a-p\cdot y & -p\cdot x \\ q\cdot y & q\cdot x-b \end{pmatrix}$

Тепер необхідно здійснити аналіз стійкості кожної особливої точки, вона залежить від власних значень матриці Якобі у даній точці.

```
(%i14) J0_1:J,x=x0_1,y=y0_1;
```

(%o14) $\begin{pmatrix} a & 0 \\ 0 & -b \end{pmatrix}$

```
(%i15) eigenvalues(J0_1);
```

(%o15) $[[-b,a],[1,1]]$

Перша точка M_1 дає два дійсні значення, протилежні за знаком. Це сідлова точка, вона нестійка.

```
(%i16) J0_2:J,x=x0_2,y=y0_2;
```

(%o16) $\begin{pmatrix} 0 & -\frac{b\cdot p}{q} \\ \frac{a\cdot q}{p} & 0 \end{pmatrix}$

```
(%i17) eigenvalues(J0_2);
```

(%o17) $[[-\sqrt{-a\cdot b},\sqrt{-a\cdot b}],[1,1]]$

Друга точка M_2 дає два чисто уявні власні значення. Це особлива точка типу центр, вона завжди стійка. Фазова траєкторія кружляє навколо цієї точки. Якщо уважно подивитись на лінії контурів функції F, то можна помітити, що всі криві обертаються навколо центральної точки $x=b/q=0.15$, $y=a/p=0.1333$.

Знайдемо період коливань системи. Лінеаризована система у загальному випадку для будь-якого значення (x_0,y_0) має вигляд:

```
(%i18) depends([u,v],t);
```

(%o18) $[\mathrm{u}(t),\mathrm{v}(t)]$

```
(%i19) matrix(['diff(u,t)],['diff(v,t)])=J.matrix([u],[v]),x=x0,y=y0;
```

$$(\%o19)\quad \begin{pmatrix} \frac{d}{dt}\cdot u \\ \frac{d}{dt}\cdot v \end{pmatrix} = \begin{pmatrix} u\cdot(a-p\cdot y0)-p\cdot v\cdot x0 \\ q\cdot u\cdot y0+v\cdot(q\cdot x0-b) \end{pmatrix},$$

а у випадку точки M_2:

```
(%i20) matrix(['diff(u,t)],['diff(v,t)])=J.matrix([u],[v]),x=x0_2,y=y0_2;
```

$$(\%o20)\quad \begin{pmatrix} \frac{d}{dt}\cdot u \\ \frac{d}{dt}\cdot v \end{pmatrix} = \begin{pmatrix} -\frac{b\cdot p\cdot v}{q} \\ \frac{a\cdot q\cdot u}{p} \end{pmatrix}$$

Отримали систему двох рівнянь

$$\begin{cases} \dot{u} = -\dfrac{bp}{q}v; \\ \dot{v} = \dfrac{aq}{p}u. \end{cases}$$

Після диференціювання першого рівняння та підстановки $\dot{v}$ із другого, маємо рівняння осцилятора

$$\ddot{u} + ab\cdot u = 0.$$

Циклічна частота його $\omega^2 = ab$, а період коливань буде рівний $T = \dfrac{2\pi}{\sqrt{ab}}$.

Здійснимо нарешті прямий числовий розв'язок вихідної системи, щоб оцінити залежність $x(t); y(t)$. Зробимо це за допомогою команди `rk`, збільшивши кількість кроків інтегрування до 1000.

```
(%i21) load(dynamics);
(%i22) deqn1a:deqn1,a=0.2,b=0.3,p=1.5,q=2;
```

$$(\%o22)\quad \frac{d}{dt}\cdot x = 0.2\cdot x - 1.5\cdot x\cdot y$$

```
(%i23) deqn2a:deqn2,a=0.2,b=0.3,p=1.5,q=2;
```

$$(\%o23)\quad \frac{d}{dt}\cdot y = 2\cdot x\cdot y - 0.3\cdot y$$

```
(%i24) sol2:rk([rhs(deqn1a),rhs(deqn2a)],[x,y],[0.2,0.4],[t,0,100,0.1])$
(%i25) t_list2:makelist(sol2[k][1],k,1,length(sol2))$
       x_list2:makelist(sol2[k][2],k,1,length(sol2))$
       y_list2:makelist(sol2[k][3],k,1,length(sol2))$
```

Відображення числових результатів — Рис.12.11.

```
(%i28) plot2d([0.2/1.5,[discrete,[[0.15,0],
              [0.15,0.45]]],[discrete,[[0.2,0.4]]],
              [discrete,x_list2,y_list2]],[x,0,0.4],
              [style,[lines,2,4],[lines,2,3],
              [points,4,5,1],[lines,3,1]],
       [legend,"y0","x0","Стартова точка","Фазова траєкторія"]);
```

```
(%i29) plot2d([0.2/1.5,0.3/2,
               [discrete,t_list2,x_list2],
               [discrete,t_list2,y_list2]],
               [x,0,100],
               [style,[lines,1,4],[lines,1,3],
               [lines,3,1],[lines,3,2]],
               [xlabel,"час"],[ylabel,"чисельність"],
               [legend,"y0","x0","Кролики","Лисиці"]);
```

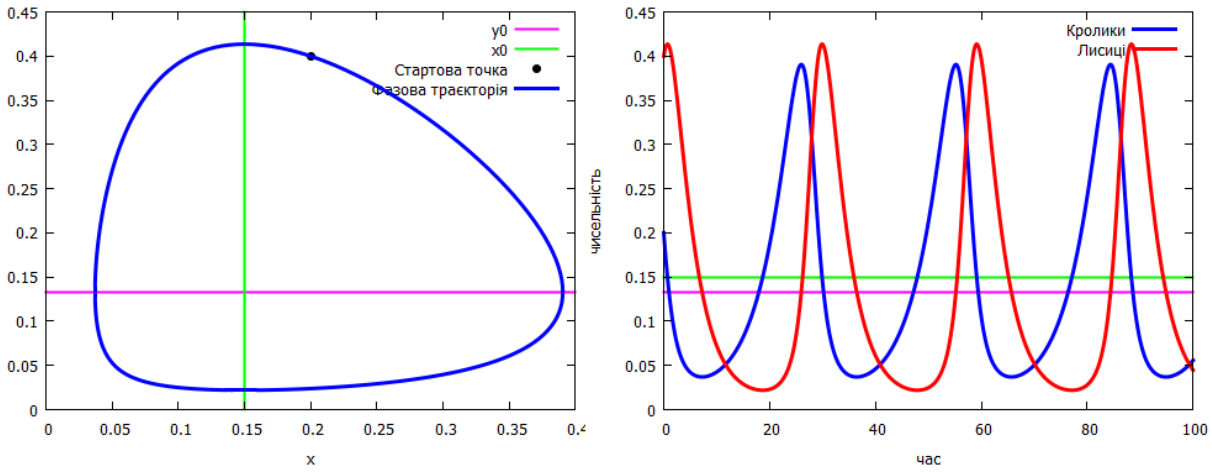


Рис. 12.11: Числовий розв'язок вихідної системи: фазова траєкторія із початковою точкою (а); часова залежність $x(t)$, $y(t)$ (б).

На обох графіках додатково зображені прямі рівноважного стану, навколо яких відбуваються коливання. Перетин їх на фазовій площині утворює особливу стаціонарну точку — центр. Тип еволюції системи є стійким, тобто він повторюється та ніколи не відходить від стаціонарної точки на нескінченність.

Такі ж самі результати отримались би з використанням команди `plotdf`, у чому читач може переконатись самостійно:

```
(%i30) load(plotdf);

(%i31) plotdf([rhs(deqn1a),rhs(deqn2a)],[x,y],
       [trajectory_at,0.2,0.4],[direction,both],[x,0.0001,0.45],
       [y,0.0001,0.45],[nsteps,400],[tstep,0.01]);
```

12.3.2 Модель із внутрішньовидовою конкуренцією

Хоч модель Лотки–Вольтерри, розглянута попередньо, і є застосовною до реальних явищ, проте вона має два принципові та взаємопов'язані недоліки. Один із них математичного, інший біологічного характеру. По-перше, при будь-яких скільки завгодно малих збуреннях фазових координат цикл, за яким відбуваються коливання, буде змінюватися. Також зміни у правій частині рівнянь Лотки–Вольтерри можуть призвести до зміни типу особливої точки, отже і характеру фазових траєкторій. З точки зору біології, недоліком даної моделі є неврахування природних чинників співіснування пари популяцій: конкуренція всередині виду, обмеженість ресурсів, насичення тощо.

Розглянемо природне узагальнення класичної моделі Лотки–Вольтерри:

$$\begin{cases} \dot{x} = ax - p_1xy - p_2x^2; \\ \dot{y} = -by + q_1xy - q_2y^2, \end{cases}$$

де з'явились нові параметри p_2 та q_2 — відповідно скорочення популяції жертв та хижаків за рахунок внутрішньовидової конкуренції. Усі параметри додатні, разом з величинами x та y.

Задамо ці дані у систему.

```
(%i1)  depends([x,y],t);
```

(%o1) $[\mathrm{x}(t), \mathrm{y}(t)]$

```
(%i2)  deqn1:'diff(x,t)=a*x-p1*x*y-p2*x^2;
```

(%o2) $$\frac{d}{dt} \cdot x = -p1 \cdot x \cdot y - p2 \cdot x^2 + a \cdot x$$

```
(%i3)  deqn2:'diff(y,t)=-b*y+q1*x*y-q2*y^2;
```

(%o3) $$\frac{d}{dt} \cdot y = -q2 \cdot y^2 + q1 \cdot x \cdot y - b \cdot y$$

Спроба знайти перший інтеграл не призводить до успіху (пакет `contrib_ode` теж не може проінтегрувати дане ЗДР):

```
(%i4)  eq1:rhs(deqn2)/rhs(deqn1);
```

(%o4) $$\frac{-q2 \cdot y^2 + q1 \cdot x \cdot y - b \cdot y}{-p1 \cdot x \cdot y - p2 \cdot x^2 + a \cdot x}$$

```
(%i5)  eq2:ode2('diff(y,x)=eq1,y,x);
```

(%o5) false

Знаходимо особливі точки та запроваджуємо матрицю Якобі:

```
(%i6)  sol1:solve([rhs(deqn1)=0,rhs(deqn2)=0],[x,y]);
```

(%o6) $$[[x = 0, y = 0], [x = \frac{a}{p2}, y = 0], [x = 0, y = -\frac{b}{q2}],$$
$$[x = \frac{a \cdot q2 + b \cdot p1}{p2 \cdot q2 + p1 \cdot q1}, y = \frac{a \cdot q1 - b \cdot p2}{p2 \cdot q2 + p1 \cdot q1}]]$$

```
(%i7)  J:matrix([diff(rhs(deqn1),x),diff(rhs(deqn1),y)],
                [diff(rhs(deqn2),x),diff(rhs(deqn2),y)]);
```

(%o7) $$\begin{pmatrix} -p1 \cdot y - 2 \cdot p2 \cdot x + a & -p1 \cdot x \\ q1 \cdot y & -2 \cdot q2 \cdot y + q1 \cdot x - b \end{pmatrix}$$

Маємо чотири особливих точки M_1, M_2, M_3, M_4. Вигляд власних значень J у цих точках встановлює характер особливих точок.

```
(%i15) x0_1:rhs(sol1[1][1])$
       y0_1:rhs(sol1[1][2])$
       x0_2:rhs(sol1[2][1])$
       y0_2:rhs(sol1[2][2])$
       x0_3:rhs(sol1[3][1])$
       y0_3:rhs(sol1[3][2])$
       x0_4:rhs(sol1[4][1])$
       y0_4:rhs(sol1[4][2])$
```

Поглянувши на вираз для y_4, можна помітити, що його знак залежить від вигляду $aq_1 - bp_2$. Оскільки величина y_4 є чисельністю популяції, необхідно прийняти її більшою нуля. Внесемо у систему також, що всі інші параметри теж більші нуля.

```
(%i22) assume(a*q1>b*p2)$
       assume(a>0)$
       assume(b>0)$
       assume(p1>0)$
       assume(p2>0)$
       assume(q1>0)$
       assume(q2>0)$
```

Дізнаємось власні значення матриці Якобі в особливих точках. Щоб встановити їх знак, скористаємось командою `signum` — вона виводить 1, якщо число більше нуля, 0 якщо рівне нулю та -1 якщо число менше нуля.

```
(%i23) J0_1:J,x=x0_1,y=y0_1;
```

$$(\%o23) \quad \begin{pmatrix} a & 0 \\ 0 & -b \end{pmatrix}$$

```
(%i24) %lambda_1:eigenvalues(J0_1);
```

$$(\%o24) \quad [[-b, a], [1, 1]]$$

```
(%i25) signum([%lambda_1[1][1],%lambda_1[1][2]]);
```

$$(\%o25) \quad [-1, 1]$$

Як видно, власні значеня у першій точці M_1 є дійсними та з протилежними знаками, отже це нестійке сідло.

```
(%i26) J0_2:J,x=x0_2,y=y0_2;
```

$$(\%o26) \quad \begin{pmatrix} -a & -\frac{a \cdot p1}{p2} \\ 0 & \frac{a \cdot q1}{p2} - b \end{pmatrix}$$

```
(%i27) %lambda_2:eigenvalues(J0_2);
```

$$(\%o27) \quad [[\frac{a \cdot q1 - b \cdot p2}{p2}, -a], [1, 1]]$$

```
(%i28) signum([%lambda_2[1][1],%lambda_2[1][2]]);
```

$$(\%o28) \quad [1, -1]$$

Особлива точка M_2 є знову сідловою.

```
(%i29) J0_3:J,x=x0_3,y=y0_3;
```

$$(\%o29) \quad \begin{pmatrix} \frac{b \cdot p1}{q2} + a & 0 \\ -\frac{b \cdot q1}{q2} & b \end{pmatrix}$$

```
(%i30) %lambda_3:eigenvalues(J0_3);
```

$$(\%o30) \quad [[\frac{a \cdot q2 + b \cdot p1}{q2}, b], [1, 1]]$$

```
(%i31) signum([%lambda_3[1][1],%lambda_3[1][2]]);
```

$$(\%o31) \quad [1, 1]$$

Особлива точка M_3 — нестійкий вузол, фазові траєкторії параболічного типу.

Знайдемо тепер власні значення у точці M_4:

```
(%i32) J0_4:fullratsimp(J),x=x0_4,y=y0_4;
```

$$(\%o32)\quad \begin{pmatrix} -\frac{a\cdot p2\cdot q2+b\cdot p1\cdot p2}{p2\cdot q2+p1\cdot q1} & -\frac{a\cdot p1\cdot q2+b\cdot p1^2}{p2\cdot q2+p1\cdot q1} \\ \frac{a\cdot q1^2-b\cdot p2\cdot q1}{p2\cdot q2+p1\cdot q1} & -\frac{(a\cdot q1-b\cdot p2)\cdot q2}{p2\cdot q2+p1\cdot q1} \end{pmatrix}$$

```
(%i33) %lambda_4:eigenvalues(J0_4);
(%i34) signum(%lambda_4[1][1]);
(%i35) signum(%lambda_4[1][2]);
```

Повні вирази для λ_4 є довгими та складними для аналізу. Присутність коренів не дає змоги визначити знак власних значень, оскільки вони можуть бути комплексними. Тому здійснимо наступне: підставимо у матрицю `J0_4` вирази для a та b, знайдені із умови стаціонарності вихідної системи у точках x_0, y_0. Оскільки це чисельності популяцій, вони також повинні бути додатними.

```
(%i36) J0_4a:subst([a=p1*y0+p2*x0,b=q1*x0-q2*y0],J0_4)$
(%i37) fullratsimp(J0_4a);
```

$$(\%o37)\quad \begin{pmatrix} -p2\cdot x0 & -p1\cdot x0 \\ q1\cdot y0 & -q2\cdot y0 \end{pmatrix}$$

```
(%i38) assume(x0>0)$
       assume(y0>0)$
(%i39) %lambda_4a:eigenvalues(J0_4a);
```

$$(\%o39)\quad [[-\frac{\sqrt{q2^2\cdot y0^2+(-2\cdot p2\cdot q2-4\cdot p1\cdot q1)\cdot x0\cdot y0+p2^2\cdot x0^2}+q2\cdot y0+p2\cdot x0}{2},$$
$$\frac{\sqrt{q2^2\cdot y0^2+(-2\cdot p2\cdot q2-4\cdot p1\cdot q1)\cdot x0\cdot y0+p2^2\cdot x0^2}-q2\cdot y0-p2\cdot x0}{2}],[1,1]]$$

```
(%i40) signum([%lambda_4a[1][1],%lambda_4a[1][1]]);
```

$$(\%o40)\quad [-1,-1]$$

Отже, в результаті отримали стійку особливу точку — це стійкий вузол, якщо підкореневий вираз більший нуля, або стійкий фокус, якщо він менший нуля (тоді власні значення стають комплексними з від'ємними дійсними частинами). Всі фазові траєкторії сходяться до центру.

Продемонструємо це за допомогою команди `plotdf`, зобразивши окремі траєкторії клацанням миші в різних областях сцени (Рис.12.12). При наданні певних числових значень параметрам не слід забувати про умову $aq_1 - bp_2 > 0$.

```
(%i41) load(plotdf);
(%i42) deqn1a:deqn1,a=0.3,p1=0.2,p2=0.1;
```

$$(\%o42)\quad \frac{d}{dt}\cdot x = -0.2\cdot x\cdot y - 0.1\cdot x^2 + 0.3\cdot x$$

```
(%i43) deqn2a:deqn2,b=0.2,q1=0.1,q2=0.1;
```

$$(\%o43)\quad \frac{d}{dt}\cdot y = -0.1\cdot y^2 + 0.1\cdot x\cdot y - 0.2\cdot y$$

```
(%i44) plotdf([rhs(deqn1a),rhs(deqn2a)],[x,y],[tstep,0.01],
       [x,0,4],[y,0,1],[direction,both],[nsteps,300]);
```

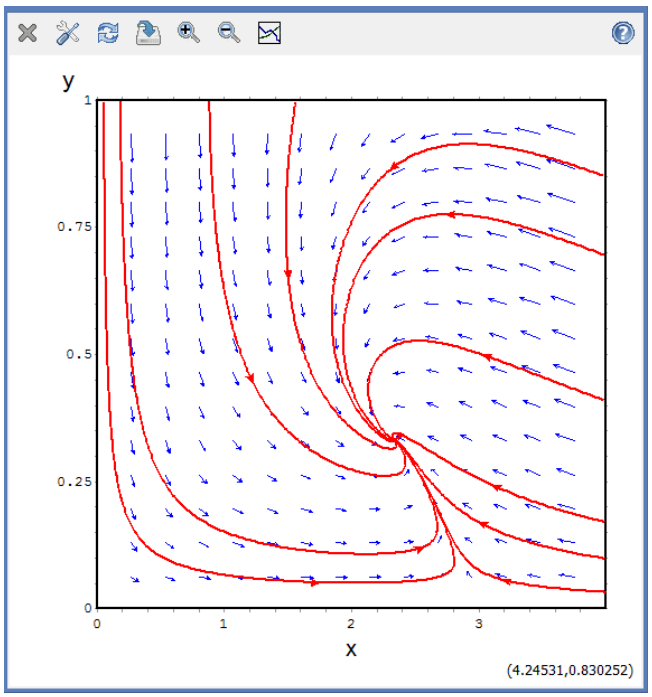


Рис. 12.12: Фазові траєкторії для стійкої особливої точки M_4.

12.4 Модель Лоренца

Перша модель хаотичної поведінки з утворенням дивних атракторів була запропонована Е. Лоренцом в 1963 р. Він досліджував конвективні процеси повітряних мас поблизу земної поверхні, котрі утворювали атмосферні фронти. Спрощена лінеаризована система рівнянь Нав'є-Стокса привела до наступних співвідношень:

$$\begin{cases} \dot{x} = -\sigma x + \sigma y; \\ \dot{y} = rx - y - xz; \\ \dot{z} = xy - bz. \end{cases}$$

Всі параметри $(\sigma,\, r,\, b)$ є позитивними.

Задамо ці рівняння у **Maxima** та проведемо їх лінеаризацію поблизу стаціонарних точок.

```
(%i1) depends([x,y,z],t);
```

$$(\%o1) \quad [\mathrm{x}\,(t)\,,\mathrm{y}\,(t)\,,\mathrm{z}\,(t)]$$

```
(%i2) deqn1:'diff(x(t),t)=-%sigma*x+%sigma*y;
```

$$(\%o2) \quad \frac{d}{dt}\cdot \mathrm{x}\,(t) = \sigma\cdot y - \sigma\cdot x$$

```
(%i3) deqn2:'diff(y(t),t)=r*x-y-x*z;
```

$$(\%o3) \quad \frac{d}{dt}\cdot \mathrm{y}\,(t) = -x\cdot z - y + r\cdot x$$

```
(%i4) deqn3:'diff(z(t),t)=x*y-b*z;
```

$$(\%o4) \quad \frac{d}{dt}\cdot \mathrm{z}\,(t) = x\cdot y - b\cdot z$$

```
(%i5)  sol1:solve([rhs(deqn1)=0,rhs(deqn2)=0,rhs(deqn3)=0],[x,y,z]);
```

(%o5) $$[[x=\sqrt{b\cdot r-b}, y=\frac{b\cdot r-b}{\sqrt{b}\cdot\sqrt{r-1}}, z=r-1],$$
$$[x=-\sqrt{b\cdot r-b}, y=-\frac{b\cdot r-b}{\sqrt{b}\cdot\sqrt{r-1}}, z=r-1],$$
$$[x=0, y=0, z=0]]$$

Як виявилось, маємо три набори особливих точок M_1, M_2, M_3, тому виділимо їх окремо:

```
(%i14) x0_1:rhs(sol1[1][1])$
       y0_1:rhs(sol1[1][2])$
       z0_1:rhs(sol1[1][3])$
       x0_2:rhs(sol1[2][1])$
       y0_2:rhs(sol1[2][2])$
       z0_2:rhs(sol1[2][3])$
       x0_3:rhs(sol1[3][1])$
       y0_3:rhs(sol1[3][2])$
       z0_3:rhs(sol1[3][3])$
(%i17) assume(%sigma>0)$
       assume(r>0)$
       assume(b>0)$
```

Запишемо матрицю Якобі у кожній з них:

```
(%i18) J:matrix([diff(rhs(deqn1),x),diff(rhs(deqn1),y),diff(rhs(deqn1),z)],
                [diff(rhs(deqn2),x),diff(rhs(deqn2),y),diff(rhs(deqn2),z)],
                [diff(rhs(deqn3),x),diff(rhs(deqn3),y),diff(rhs(deqn3),z)]);
```

(%o18) $$\begin{pmatrix} -\sigma & \sigma & 0 \\ r-z & -1 & -x \\ y & x & -b \end{pmatrix}$$

```
(%i19) J0_1:J,x=x0_1,y=y0_1,z=z0_1;
```

(%o19) $$\begin{pmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & -\sqrt{b\cdot r-b} \\ \frac{b\cdot r-b}{\sqrt{b}\cdot\sqrt{r-1}} & \sqrt{b\cdot r-b} & -b \end{pmatrix}$$

```
(%i20) J0_2:J,x=x0_2,y=y0_2,z=z0_2;
```

(%o20) $$\begin{pmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & \sqrt{b\cdot r-b} \\ -\frac{b\cdot r-b}{\sqrt{b}\cdot\sqrt{r-1}} & -\sqrt{b\cdot r-b} & -b \end{pmatrix}$$

```
(%i21) J0_3:J,x=x0_3,y=y0_3,z=z0_3;
```

(%o21) $$\begin{pmatrix} -\sigma & \sigma & 0 \\ r & -1 & 0 \\ 0 & 0 & -b \end{pmatrix}$$

Найперше проаналізуємо стаціонарну точку $M_3(0;0;0)$. Знайдемо власні значення матриці `J0_3`:

```
(%i22) eigenvalues(J0_3);
```

(%o22) $$[[-\frac{\sqrt{4\cdot\sigma\cdot r+\sigma^2-2\cdot\sigma+1}+\sigma+1}{2}, \frac{\sqrt{4\cdot\sigma\cdot r+\sigma^2-2\cdot\sigma+1}-\sigma-1}{2}, -b],$$

$[1, 1, 1]]$

Бачимо, що перше та третє власні числа є від'ємними, а друге буде таким при $\sqrt{4\sigma r + \sigma^2 - 2\sigma + 1} - \sigma - 1 < 0$, що відразу дає нам $r < 1$. Отже, за цієї умови точка M_3 є стійкою стаціонарною точкою типу вузол. При $r > 1$ один з коренів стає додатнім, і ця точка перетворюється в нестійке сідло.

Звернемося до точок M_1 та M_2. Пряме задання команди `eigenvalues` видає дуже довгі вирази, які практично неможливо проаналізувати. Тому підемо іншим шляхом, а саме знайдемо відповідні характеристичні поліноми кожної з матриць `J0_1` та `J0_2`.

```
(%i23) factor(charpoly(J0_1,%lambda));
```

$$(\%o23) \quad -\left(2\cdot\sigma\cdot b\cdot r+\lambda\cdot b\cdot r+\lambda\cdot\sigma\cdot b-2\cdot\sigma\cdot b+\lambda^2\cdot b+\lambda^2\cdot\sigma+\lambda^3+\lambda^2\right)$$

```
(%i24) factor(charpoly(J0_2,%lambda));
```

$$(\%o24) \quad -\left(2\cdot\sigma\cdot b\cdot r+\lambda\cdot b\cdot r+\lambda\cdot\sigma\cdot b-2\cdot\sigma\cdot b+\lambda^2\cdot b+\lambda^2\cdot\sigma+\lambda^3+\lambda^2\right)$$

Ці вирази повністю однакові, тому і типи точок теж будуть однакові. Розглянемо вільний член у цих виразах, який не містить λ. За другою теоремою Вієта він рівний добутку трьох коренів:

$$\lambda_1\lambda_2\lambda_3 = 2\sigma b(1-r).$$

Цей добуток, а отже і знаки трьох коренів, знову залежить від співвідношення r та 1. При $r < 1$ добуток додатній, отже один з коренів має бути додатнім, а інші два або від'ємні, або комплексно спряжені. Тоді точки M_1 та M_2 будуть нестійкими вузлами або нестійкими фокусами. При $r > 1$ добуток від'ємний, отже може реалізуватись два варіанти: один з коренів від'ємний, а два інші або додатні, або комплексно спряжені (нестійке сідло); всі три корені дійсні та від'ємні (стійкий вузол). У досить широкому діапазоні значень для $r > 1$ відбувається саме останній варіант, що і призводить до стійкості системи, фазові траєкторії якої носять назву «дивних атракторів Лоренца».

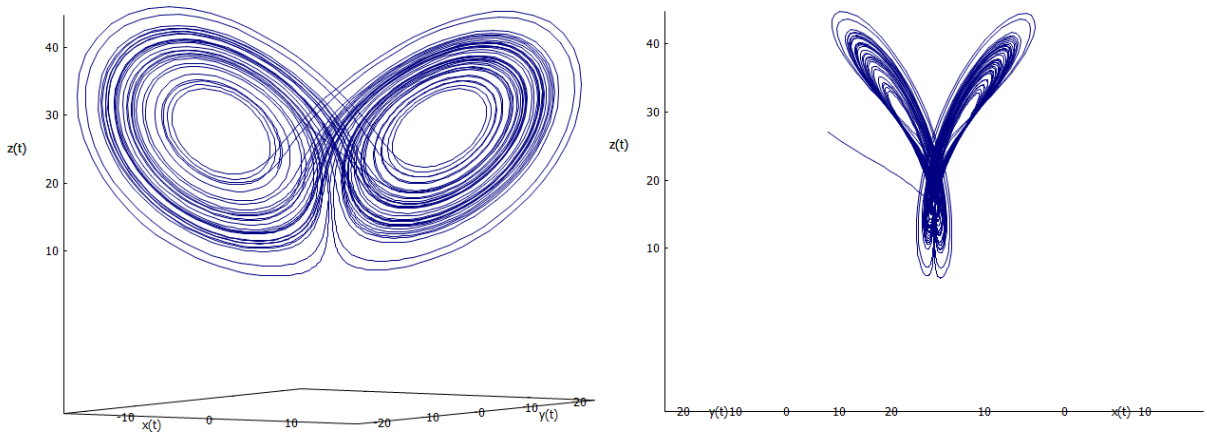


Рис. 12.13: Дивний атрактор Лоренца – прямий та бічний вигляд.

Зобразимо ці атрактори. Для конкретних значень параметрів традиційно обирають $\sigma = 10$, $b = \frac{8}{3}$, $r = 28$. Знайдемо числовий розв'язок вихідної системи з цими початковими даними.

```
(%i25) %sigma:10$ r:28$ b:8/3$
(%i26) load(dynamics);
```

```
(%i27) sol2:rk([rhs(deqn1),rhs(deqn2),rhs(deqn3)],
                [x,y,z],[-8,8,27],[t,0,50,0.01])$
(%i28) t_list:makelist(sol2[k][1],k,1,length(sol2))$
       x_list:makelist(sol2[k][2],k,1,length(sol2))$
       y_list:makelist(sol2[k][3],k,1,length(sol2))$
       z_list:makelist(sol2[k][4],k,1,length(sol2))$
```

Графік обрахованих фазових траєкторій – на Рис.12.13.

Фраза «порядок у хаосі» зустрічається часто. Це означає, що в деяких системах, незважаючи на те, що інтегральні траєкторії можуть здаватися непередбачуваними і дуже складними, все ж існує певна закономірність. Наприклад, у системі Лоренца чутливість до початкових умов не дозволяє робити точні прогнози щодо розвитку, але можна стверджувати, що в довгостроковій перспективі орбіта буде обмежена областю простору, зайнятою дивним атрактором.

12.5 Завдання до Розділу 12

Системи автономних ЗДР

Дослідити систему автономних лінійних ЗДР та визначити тип особливих точок в залежності від значення параметра a. Побудувати поле напрямків системи із слайдером зміни параметра a, виділити на полі кілька інтегральних кривих та продемонструвати їх трансформацію та залежність спрямування від значення a (скористатись відомостями із Параграфу 5.3).

12.1. $\begin{cases} \dot{x} = 2x - 4y; \\ \dot{y} = ax - 6y. \end{cases}$

12.2. $\begin{cases} \dot{x} = 6x - 3y; \\ \dot{y} = ax + 4y. \end{cases}$

12.3. $\begin{cases} \dot{x} = x + 6y; \\ \dot{y} = 2x + ay. \end{cases}$

12.4. $\begin{cases} \dot{x} = -4x + ay; \\ \dot{y} = x - 5y. \end{cases}$

12.5. $\begin{cases} \dot{x} = -3x + y; \\ \dot{y} = ax + 2y. \end{cases}$

12.6. $\begin{cases} \dot{x} = 2x + 4y; \\ \dot{y} = ax - 5y. \end{cases}$

12.7. $\begin{cases} \dot{x} = -3x - 4y; \\ \dot{y} = ax + 3y. \end{cases}$

12.8. $\begin{cases} \dot{x} = ax - 2y; \\ \dot{y} = -3x + 2y. \end{cases}$

12.9. $\begin{cases} \dot{x} = 5x + ay; \\ \dot{y} = -2x + 3y. \end{cases}$

12.10. $\begin{cases} \dot{x} = -3x - 4y; \\ \dot{y} = 5x + ay. \end{cases}$

Знайти особливі точки системи автономних нелінійних ЗДР, лінеаризувати систему рівнянь в їх околі, визначити тип кожної особливої точки.

12.11. $\begin{cases} \dot{x} = 0.2x - 0.4y - 0.1x^2; \\ \dot{y} = -x + 0.6y - 0.2y^2. \end{cases}$

12.12. $\begin{cases} \dot{x} = 0.6x - 0.3y - 0.2x^2; \\ \dot{y} = -0.5x + 0.4y - 0.1y^2. \end{cases}$

12.13. $\begin{cases} \dot{x} = x + 0.6y - 0.3x^2; \\ \dot{y} = 0.2x + 0.4y + 0.6xy. \end{cases}$

12.14. $\begin{cases} \dot{x} = -0.4x + 0.1y + 0.3xy; \\ \dot{y} = 0.1x - 0.5y - 0.7x^2. \end{cases}$

12.15. $\begin{cases} \dot{x} = -0.3x + 0.4y + 0.2x^2; \\ \dot{y} = 0.6x + 0.2y - 0.3x^2. \end{cases}$

12.16. $\begin{cases} \dot{x} = 0.2x - 0.4y - 0.1xy; \\ \dot{y} = -0.5x - 0.6y - 0.2y^2. \end{cases}$

12.17. $\begin{cases} \dot{x} = -0.3x - 0.4y + 0.2y^2; \\ \dot{y} = 0.5x - 0.6y + 0.1xy. \end{cases}$

12.18. $\begin{cases} \dot{x} = 0.5x - 0.2y - 0.2x^2; \\ \dot{y} = -0.3x + 0.2y + 0.6y^2. \end{cases}$

12.19. $\begin{cases} \dot{x} = -x + 0.5y - 0.1xy; \\ \dot{y} = 0.2x + 0.3y + 0.2x^2. \end{cases}$

12.20. $\begin{cases} \dot{x} = -0.3x - 0.4y + 0.6x^2; \\ \dot{y} = 0.5x + 0.1y - 0.5xy. \end{cases}$

Розділ 13

Вступ у числові методи розв'язання ДР

Розв'язання диференціальних рівнянь та рівнянь математичної фізики становлять предмет окремих фундаментальних курсів математичного аналізу. Лише невелика частина цих задач допускає прямий аналітичний розв'язок; значний пласт цікавих для практичного застосування випадків можуть бути вирішені лише за допомогою наближених способів, що й обумовлює розвинення числових методів ще з часів Ньютона та Ейлера. Докладному опису цих методів присвячена велика кількість літератури, тому ми не будемо детально обговорювати походження та виведення відповідних виразів, а зосередимось на безпосередньому їх використанні та реалізації у програмному середовищі **Maxima**.

13.1 Задачі Коші для ЗДР першого порядку

Звичайне диференціальне рівняння першого порядку має вигляд

$$\frac{dy}{dx} = f(x, y).$$

Рішенням такого рівняння є нескінченна кількість інтегральних кривих, множина яких є загальним розв'язком ЗДР. Умова проходження інтегральної кривої через певну точку $(x_0; y_0)$ фіксує константу інтегрування, тим самим утворюючи частковий розв'язок ЗДР.

Задача знаходження часткового розв'язку ЗДР $y(x)$, що задовільняє початковій умові $y(x_0) = y_0$, називається задачею Коші. В англомовній літературі цей термін рідко використовується, натомість вживається усталена назва IVP — Initial Value Problem. Зауважимо, що задача Коші також поширюється на рівняння вищих порядків похідної (ступеня n), для відшукання частинних розв'язків яких необхідно вказувати значення похідних у точці x_0 для величин $y(x_0), y'(x_0), ...y^{(n-1)}(x_0)$.

В основі числового розв'язку ЗДР ключовим поняттям є так звана дискретизація системи. Цей метод відноситься також до всіх способів розв'язання диференціальних рівнянь вищих порядків та диференціальних рівнянь у частинних похідних. Тому необхідно докладно розглянути його спочатку на простіших прикладах, та пізніше розвинути до більш складних.

Нехай задана задача Коші для ЗДР

$$y'(x) = 2y - e^x, \quad y(0) = 4.$$

Знайдемо її розв'язок у **Maxima**.

```
(%i1)  deqn1:'diff(y,x)=2*y-%e^x;
```

$$(\%o1) \quad \frac{d}{d\,x} \cdot y = 2 \cdot y - e^x$$

```
(%i2)  ode2(deqn1,y,x);
```

(%o2) $y = \left(e^{-x} + \%c\right) \cdot e^{2 \cdot x}$

```
(%i3)  sol1:ic1(%,x=0,y=4);
```

(%o3) $y = 3 \cdot e^{2 \cdot x} + e^{x}$

Тепер розв'яжемо цю ж задачу різними числовими засобами.

13.1.1 Метод Ейлера

Це найдавніший метод, який заклав базис для подальших ідей та вдосконалень. Суть його полягає в тому, щоб розбити інтервал інтегрування на скінченну кількість проміжків та замінити криву $y(x)$ на дотичні до неї у вузлових точках.

На осі Ox запровадимо рівномірну сітку з кроком $h > 0$, тобто розглянемо систему точок $x_i = i \cdot h$, $i = 0, 1, 2, \ldots$. Кожна з цих точок називається вузловою. Якщо кількість проміжків рівна n, то кількість точок буде $n+1$. Запишемо загальне рівняння ЗДР першого порядку в околі кожної вузлової точки, замінивши похідну різницевим відношенням:

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i).$$

Звідси значення функції y_{i+1} визначається формулою

$$y_{i+1} = y_i + h \cdot f(x_i, y_i).$$

Метод Ейлера відноситься до явних однокрокових методів: кожне наступне значення функції залежить від попереднього. Його ще називають методом Рунге-Кутта 1-го порядку. Точність цього методу невелика, про що зараз переконаємось.

Будемо розв'язувати рівняння на проміжку $[0; 2]$. Крок інтегрування оберемо 0.1. Задамо ці значення в систему, використавши команду `floor` — виводить перше ціле число, яке буде більшим за введене (оскільки відношення $(b-a)/h$ не обов'язково буває цілим).

```
(%i6)  a:0$ b:2$ h:0.1$
(%i7)  n:floor((b-a)/h);
```

(%o7) 20

Тепер створимо два порожніх одновимірних масива розміру $n + 1$ (`flonum` позначає тип змінних у масиві — числа з плаваючою точкою).

```
(%i8)  X1:make_array(flonum, n+1)$
(%i9)  Y1:make_array(flonum, n+1)$
```

Задаємо значення для початкових елементів у масивах:

```
(%i12) X1[0]:a$ X1[1]:a+h$ Y1[0]:4$
```

Заповнимо масив `X1` його значеннями — вузловими точками.

```
(%i13) for i:2 thru n do
       (X1[i]:X1[i-1]+h);
```

(%o13) *done*

Нарешті, використавши формулу Ейлера, заповнюємо масив `Y1`.

```
(%i14) for i:1 thru n do
       (Y1[i]: float (Y1[i-1]+h*(2*Y1[i-1]-exp(X1[i-1]))));
(%o14)  done
```

Отримали два набори значень `X1` та `Y1`, які є координатними компонентами точок шуканої інтегральної кривої. Зобразимо на одному рисунку точний розв'язок (`Exact`) та отриманий числовий розв'язок (`Euler`) — Рис.13.1.

```
(%i15) wxdraw2d(
       line_width = 3, color = navy, key = "Exact", key_pos=top_left,
           explicit(rhs(sol1),x,0,2),
       point_size = 1, points_joined = true, point_type = filled_circle,
       color = red, key = "Euler",
           points(X1,Y1)
       )$
```

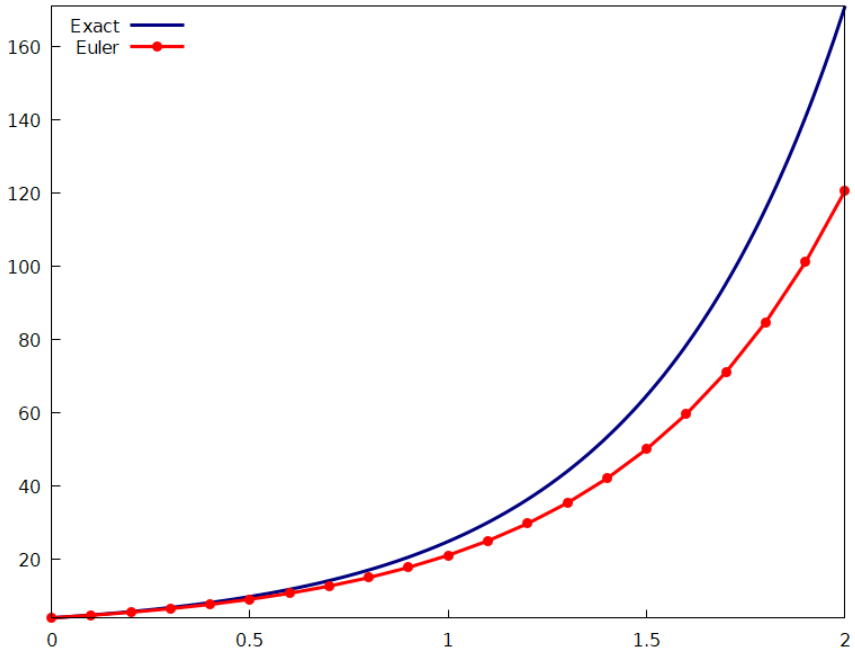


Рис. 13.1: Точний розв'язок ЗДР та числовий за методом Ейлера.

Як бачимо, крива Ейлера лежить нижче за точну криву. Так буде відбуватись для опуклих донизу функцій, оскільки дотичні в кожній точці завжди лежатимуть нижче кривої. Відповідно для опуклих доверху функцій все буде навпаки — крива Ейлера буде зверху.

13.1.2 Метод Ейлера–Коші

Це більш вдосконалений варіант методу Ейлера, особливістю якого є те, що значення правої частини рівняння обчислюється не лише у вузлових точках, а й у середині кожного проміжка. Його ще називають методом Рунге–Кутти 2-го порядку. Розрахункова формула наступна:

$$y_{i+1} = y_i + \frac{h}{2} \cdot \left(f(x_i, y_i) + f(x_{i+1}, y_i + h \cdot f(x_i, y_i)) \right).$$

Створимо три одновимірних масиви: `X2`, `Y2` та допоміжний `Z2`.

```
(%i16) X2:make_array(flonum, n+1)$
(%i17) Y2:make_array(flonum, n+1)$
(%i18) Z2:make_array(flonum, n+1)$
```

Задаємо початкові значення та заповнюємо вузлові точки масиву `X2`.

```
(%i22) X2[0]:a$ X2[1]:a+h$ Y2[0]:4$ Z2[0]:4$

(%i23) for i:2 thru n do
       (X2[i]:X2[i-1]+h)$
```

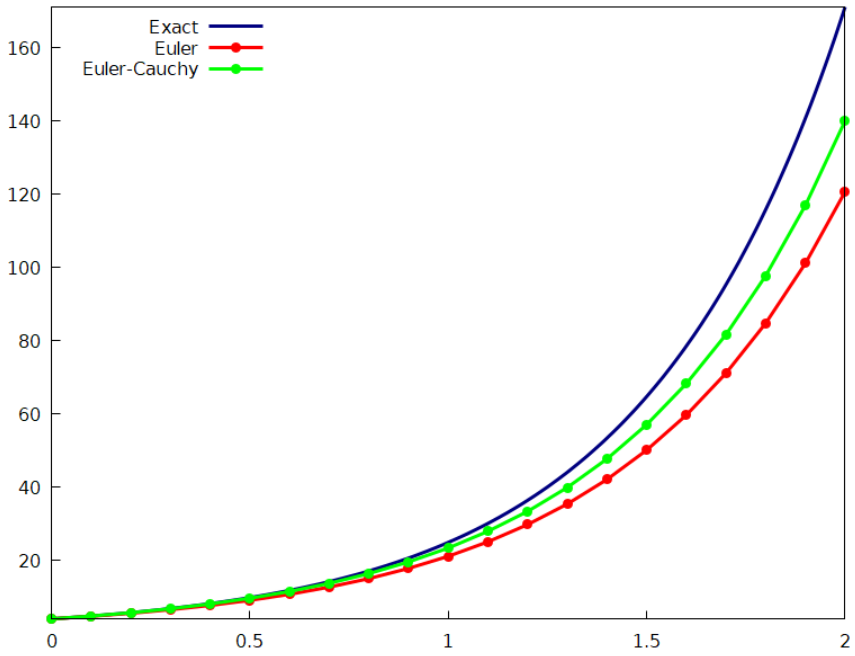


Рис. 13.2: Точний розв'язок ЗДР, числовий за методом Ейлера та за методом Ейлера–Коші.

Тепер, використавши формулу Ейлера–Коші, заповнюємо відповідно `Z2` та `Y2`.

```
(%i24) for i:1 thru n do
       (
       Z2[i]:float(Y2[i-1]+h*(2*Y1[i-1]-exp(X1[i-1]))),
       Y2[i]:float(Y2[i-1]+(h/2)*(2*Y1[i-1]-
                  exp(X1[i-1])+2*Z2[i]-exp(X2[i])))
       )$
```

Отримані точки нанесемо на графік поряд з двома попередніми кривими (Рис. 13.2).

```
(%i25) wxdraw2d(
       line_width = 3, color = navy, key = "Exact", key_pos=top_left,
           explicit(rhs(sol1),x,0,2),
       point_size = 1, points_joined = true, point_type = filled_circle,
       color = red, key = "Euler",
           points(X1,Y1),
       point_size = 1, points_joined = true, point_type = filled_circle,
       color = green, key = "Euler-Cauchy",
           points(X2,Y2)
       );
```

Як добре видно, даний метод досить точніший за попередній.

13.1.3 Метод Рунге–Кутти 4-го порядку

Цей метод найчастіше використовується в реальних розрахунках, оскільки точність його достатньо висока. Ми вже неодноразово використовували РК4 у попередніх обчисленнях, у системі **Maxima** існує вбудована команда `rk` для його виклику. Реалізуємо однак метод самостійно, створенням відповідної програми.

Розрахункові формули наступні:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4);$$

$$\begin{cases} k_1 = hf(x_i, y_i); \\ k_2 = hf(x_i + h/2, y_i + k_1/2); \\ k_3 = hf(x_i + h/2, y_i + k_2/2); \\ k_4 = hf(x_i + h, y_i + k_3). \end{cases}$$

Сворюємо необхідні масиви, тепер їх буде шість.

```
(%i26) k1:make_array(flonum, n+1)$
(%i27) k2:make_array(flonum, n+1)$
(%i28) k3:make_array(flonum, n+1)$
(%i29) k4:make_array(flonum, n+1)$
(%i30) X3:make_array(flonum, n+1)$
(%i31) Y3:make_array(flonum, n+1)$
```

Задаємо початкові значення та заповнюємо `X3` вузловими точками.

```
(%i34) X3[0]:a$ X3[1]:a+h$ Y3[0]:4$
(%i35) k1[0]:h*(2*Y3[0]-exp(X3[0]));
(%o35)  0.7000000000000001

(%i36) for i:2 thru n do
       (X3[i]:X3[i-1]+h)$
```

Створюємо цикл для обчислення `Y3` через проміжні масиви `k1-k4`.

```
(%i37) for i:1 thru n do
       (
       k1[i-1]:h*(2*Y3[i-1]-exp(X3[i-1])),
       k2[i-1]:h*(2*(Y3[i-1]+(1/2)*k1[i-1])-exp(X3[i-1]+h/2)),
       k3[i-1]:h*(2*(Y3[i-1]+(1/2)*k2[i-1])-exp(X3[i-1]+h/2)),
       k4[i-1]:h*(2*(Y3[i-1]+k3[i-1])-exp(X3[i-1]+h)),
       Y3[i]:Y3[i-1]+(1/6)*(k1[i-1]+2*k2[i-1]+2*k3[i-1]+k4[i-1])
       )$
```

Нарешті виводимо зображення обчислених точок поряд з трьома графіками, отриманими вище. При цьому виявляється, що крива Рунге–Кутти практично повністю накладається на точну криву, тому для наочності покладемо `Exact` широкою сірою лінією (Рис.13.3).

```
(%i38) wxdraw2d(
       line_width = 10, color = gray80, key = "Exact", key_pos=top_left,
           explicit(rhs(sol1),x,0,2),
       line_width = 2, point_size = 1, points_joined = true,
       point_type = filled_circle, color = red, key = "Euler",
           points(X1,Y1),
       line_width = 2, point_size = 1, points_joined = true,
       point_type = filled_circle, color = green, key = "Euler-Cauchy",
           points(X2,Y2),
       line_width = 2, point_size = 1, points_joined = true,
       point_type = filled_circle, color = blue, key = "Runge-Kutta 4",
           points(X3,Y3)
       );
```

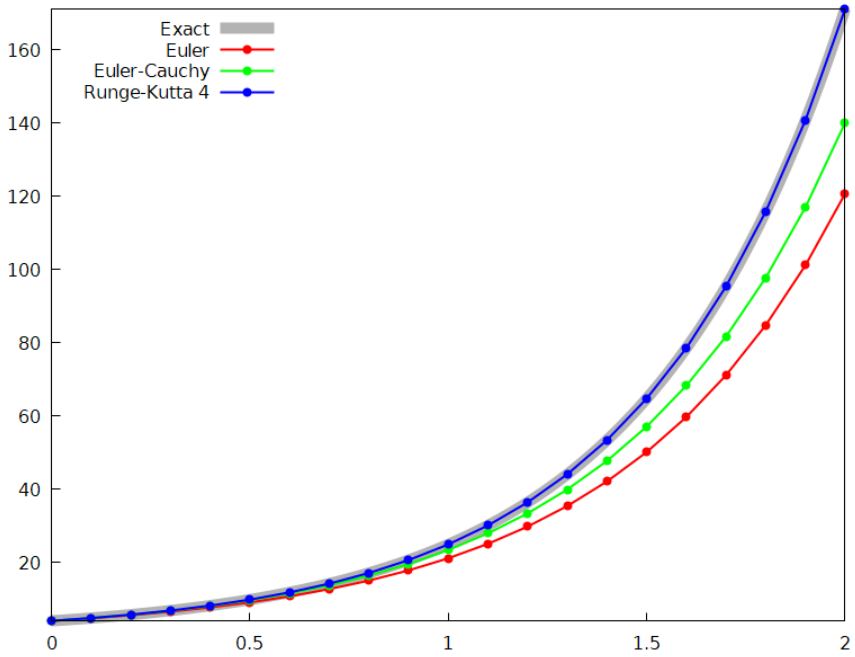


Рис. 13.3: Точний розв'язок, числовий за методом Ейлера, за методом Ейлера–Коші та за методом Рунге–Кутти 4-го порядку.

Як видно з побудованих графіків, метод РК4 дає найкращий результат при числовому розв'язанні. Він цілком задовольняє потребам у точності за умови пологості інтегральної кривої, коли немає порогових ефектів або розривів. При виникненні таких критичних випадків небхідно підніматись вже до 5-го порядку точності, використовуючи метод Рунге–Кутти–Фельберга `rkf45`. Втім, існують навіть методи 8-го порядку для досягнення необхідної точності при вирішенні екстремальних задач.

13.2 Крайові задачі для ЗДР другого порядку

Диференціальне рівняння другого порядку має загальний вигляд

$$y''(x) = f(x, y, y').$$

При інтегруванні його з'являються дві невідомі константи, що дає нам нескінченну кількість розв'язків. Щоб знайти частковий розв'язок, необхідно використати умови значень

функції або її похідної на краях проміжка інтегрування $[a;b]$, що й обумовлює назву крайової задачі (в англомовній літературі BVP — Boundary Value Problem).

Загалом широке коло рівнянь другого порядку є лінійними, тому зосередимось на їх вивченні. Такі рівняння завжди можна представити у вигляді

$$y'' + p(x)y' + q(x)y = r(x).$$

Функції $p(x)$, $q(x)$, $r(x)$ вважаються неперервними на проміжку $[a;b]$. Умови знаходження частинного розв'язку можуть бути представлені трьома способами:

∘ крайові умови першого роду

$$y(a) = A; \quad y(b) = B;$$

∘ крайові умови другого роду

$$y'(a) = A; \quad y'(b) = B;$$

∘ крайові умови третього роду (змішані)

$$\begin{cases} \alpha_1 y(a) + \alpha_2 y'(a) = A; \\ \beta_1 y(b) + \beta_2 y'(b) = B. \end{cases}$$

Як видно, крайові умови третього роду містять у собі дві перших при певному значенні коефіцієнтів. Однак перші дві виділяються окремо, тому що у реальних задачах фактично лише вони і виникають.

В процесі дискретизації у вузлових точках похідну функції ми заміняли виразом $(y_{i+1} - y_i)/h$. Подібним чином утворюється вираз для другої похідної:

$$y''(x_i) = \frac{1}{h^2}\left(y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right), \quad \text{або} \quad y_i'' = \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right).$$

Тоді загальне рівняння буде мати вигляд

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p(x_i)\frac{y_{i+1} - y_i}{h} + q(x_i)y_i = r(x_i), \quad i = 1, ...n-1;$$

а відповідні крайові умови стають наступними:

$$y_0 = A; \quad y_n = B; \qquad \text{(першого роду)}$$

$$\frac{1}{h}(y_1 - y_0) = A; \quad \frac{1}{h}(y_n - y_{n-1}) = B; \qquad \text{(другого роду)}$$

$$\begin{cases} \alpha_1 y_0 + \alpha_2 \dfrac{1}{h}(y_1 - y_0) = A; \\ \beta_1 y_n + \beta_2 \dfrac{1}{h}(y_n - y_{n-1}) = B. \end{cases} \qquad \text{(третього роду)}$$

В результаті маємо систему з $n+1$ рівнянь для $n+1$ невідомих величин y_i, $i = 0, 1, ...n$. Розв'язувати цю систему можна різними способами: за методом Гаусса, матричним методом тощо. Як бачимо, кількість рівнянь системи рівна кількості точок розбиття. Звідси висновок, що цю кількість потрібно тримати в розумних межах, для розв'язання «на папері» її рідко покладали більшою 10, оскільки розв'язати систему з такої кількості рівнянь стає дуже непросто. Але з появою комп'ютерів ця трудність стала вже несуттєвою.

13.2.1 Крайові умови першого роду

Вирішимо наступну крайову задачу:

$$xy'' + y = 0; \quad y(1) = 1, y(2) = 2.$$

Задамо спочатку рівняння у **Maxima** та спробуємо знайти точний розв'язок.

```
(%i1) deqn1:'diff(y,x,2)*x+y=0;
```

$$(\%o1) \quad x \cdot \left(\frac{d^2}{d\,x^2} \cdot y\right) + y = 0$$

```
(%i2) ode2(deqn1,y,x);
```

(%o2) false

Стандартна команда не дає результату, тому використаємо додатковий пакет `contrib_ode`.

```
(%i3) load(contrib_ode);
(%i4) contrib_ode(deqn1,y,x);
```

$$(\%o4) \quad [y = \text{bessel\_y}\left(1, 2\cdot\sqrt{x}\right)\cdot \%k2\cdot\sqrt{x} + \text{bessel\_j}\left(1, 2\cdot\sqrt{x}\right)\cdot \%k1\cdot\sqrt{x}]$$

Розв'язок виявився комбінацією спеціальних функцій Бесселя різного типу. Застосуємо тепер умови крайової задачі та зобразимо на рисунку його графік (Рис.13.4).

```
(%i5) sol1:bc2(%o4[1],x=1,y=1,x=2,y=2);
```

$$(\%o5) \quad y = \frac{\left(2\cdot\text{bessel\_y}\left(1,2\right) - \sqrt{2}\cdot\text{bessel\_y}\left(1,2^{\frac{3}{2}}\right)\right)\cdot\text{bessel\_j}\left(1,2\cdot\sqrt{x}\right)\cdot\sqrt{x}}{\sqrt{2}\cdot\text{bessel\_y}\left(1,2\right)\cdot\text{bessel\_j}\left(1,2^{\frac{3}{2}}\right) - \sqrt{2}\cdot\text{bessel\_j}\left(1,2\right)\cdot\text{bessel\_y}\left(1,2^{\frac{3}{2}}\right)} -$$
$$\frac{\left(2\cdot\text{bessel\_j}\left(1,2\right) - \sqrt{2}\cdot\text{bessel\_j}\left(1,2^{\frac{3}{2}}\right)\right)\cdot\text{bessel\_y}\left(1,2\cdot\sqrt{x}\right)\cdot\sqrt{x}}{\sqrt{2}\cdot\text{bessel\_y}\left(1,2\right)\cdot\text{bessel\_j}\left(1,2^{\frac{3}{2}}\right) - \sqrt{2}\cdot\text{bessel\_j}\left(1,2\right)\cdot\text{bessel\_y}\left(1,2^{\frac{3}{2}}\right)}$$

```
(%i6) plot2d(rhs(sol1),[x,0,10],[style,[lines,3]]);
```

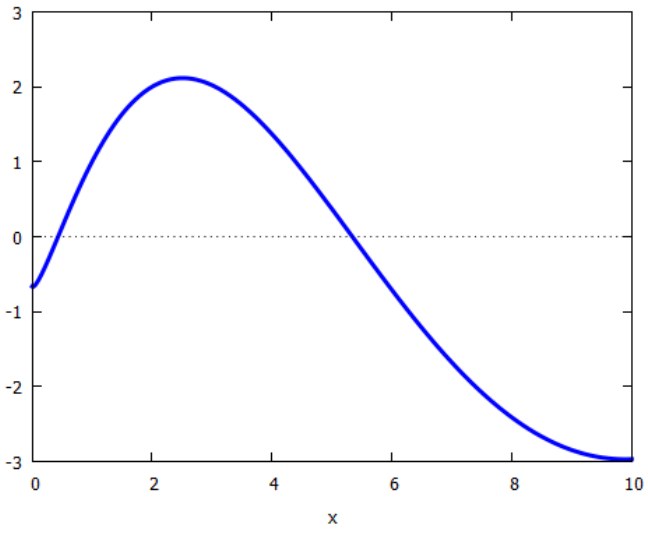


Рис. 13.4: Точний розв'язок крайової задачі.

Переходимо тепер до числового розв'язку. Покладемо крок розбиття рівним 0.2 і внесемо необхідні функції та дані у систему.

```
(%i9) p(x):=1$ q(x):=1/x$ r(x):=0$
```

```
(%i12) a:1$ b:2$ h:0.2$
(%i14) A:1$ B:2$
(%i15) n:floor((b-a)/h);
```

(%o15) 5

Проміжків у нас 5, отже вузлових точок та рівнянь у системі буде 6. Перейдемо тепер до дискретизації задачі. Формуємо необхідні масиви та заповнюємо їх відповідні відомі елементи.

```
(%i16) P:make_array(flonum, n+1)$
(%i17) Q:make_array(flonum, n+1)$
(%i18) R:make_array(flonum, n+1)$
(%i19) X:make_array(flonum, n+1)$
(%i20) Y:make_array(flonum, n+1)$
(%i24) X[0]:a$ X[n]:b$ Y[0]:A$ Y[n]:B$
```

Заповнюємо масив `X` вузловими точками, масив `Y` невідомими значеннями y_i та формуємо відповідні масиви для `P`, `Q`, `R`. Кожен з масивів виведемо на екран щоб переконатись, що все правильно задалось.

```
(%i25) for i:1 thru n do
       X[i]:X[i-1]+h$
(%i26) for i:0 thru n do
       display(X[i]);
```

$X_0 = 1 \quad X_1 = 1.2 \quad X_2 = 1.4 \quad X_3 = 1.6 \quad X_4 = 1.8 \quad X_5 = 2.0$

(%o26) *done*

```
(%i27) for i:1 thru n-1 do
       Y[i]:concat(y,i)$
(%i28) for i:0 thru n do
       display(Y[i]);
```

$Y_0 = 1 \quad Y_1 = y1 \quad Y_2 = y2 \quad Y_3 = y3 \quad Y_4 = y4 \quad Y_5 = 2$

(%o28) *done*

```
(%i29) for i:0 thru n do
       P[i]:p(X[i])$
(%i30) for i:0 thru n do
       Q[i]:q(X[i])$
(%i31) for i:0 thru n do
       R[i]:r(X[i])$
(%i32) for i:0 thru n do
       display([P[i],Q[i],R[i]]);
```

$[P_0, Q_0, R_0] = [1, 1, 0]$

$[P_1, Q_1, R_1] = [1, 0.8333333333333334, 0]$

$[P_2, Q_2, R_2] = [1, 0.7142857142857143, 0]$

$[P_3, Q_3, R_3] = [1, 0.625, 0]$

$[P_4, Q_4, R_4] = [1, 0.5555555555555556, 0]$

$[P_5, Q_5, R_5] = [1, 0.5000000000000001, 0]$

(%o32) *done*

Формуємо ще один масив, котрий буде містити рівняння для y_i та заповнюємо його формулами для дискретизованої системи.

```
(%i33) sys1:make_array(any, n+1)$
(%i34) for i:1 thru n-1 do
       sys1[i]: (1/h^2)*(Y[i+1]-2*Y[i]+Y[i-1])+
                (P[i]/h)*(Y[i+1]-Y[i])+Q[i]*Y[i]=R[i]$
(%i35) for i:1 thru n-1 do
       display(sys1[i]);
```

$5.0 \cdot (y2 - y1) + 25.0 \cdot (y2 - 2 \cdot y1 + 1) + 0.8333333333333334 \cdot y1 = 0$

$5.0 \cdot (y3 - y2) + 25.0 \cdot (y3 - 2 \cdot y2 + y1) + 0.7142857142857143 \cdot y2 = 0$

$5.0 \cdot (y4 - y3) + 25.0 \cdot (y4 - 2 \cdot y3 + y2) + 0.625 \cdot y3 = 0$

$0.5555555555555556 \cdot y4 + 5.0 \cdot (2 - y4) + 25.0 \cdot (-2 \cdot y4 + y3 + 2) = 0$

(%o35) *done*

Маємо необхідні чотири рівняння для визначення невідомих чотирьох величин y_i (y_0 та y_5 вже встановлені). Перетворимо цей масив у список для формування системи та розв'яжемо її.

```
(%i36) sys1a:makelist(sys1[i],i,1,n-1)$
(%i37) ratprint:false$
(%i38) sol1a:flatten(solve(sys1a,[y1,y2,y3,y4]));
```

(%o38) $[y1 = \frac{120927}{88370}, y2 = \frac{115759}{70696}, y3 = \frac{48319}{26511}, y4 = \frac{34269}{17674}]$

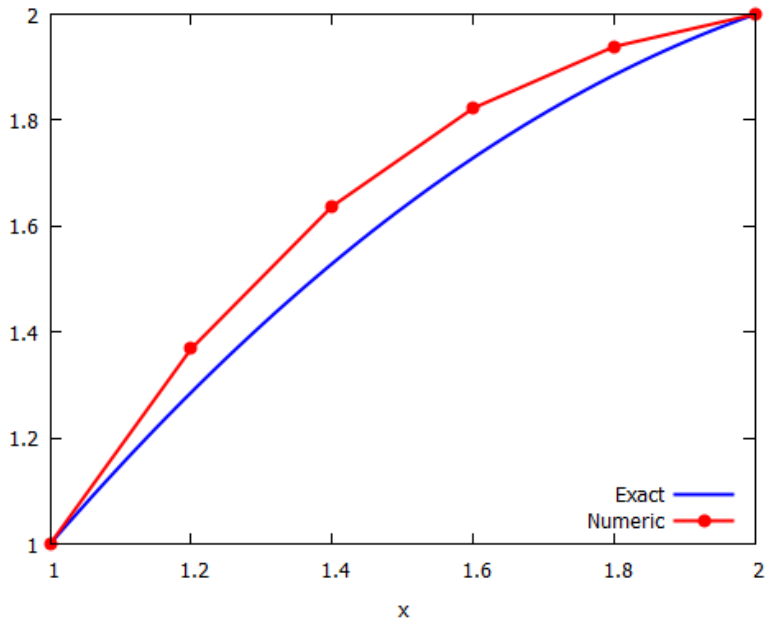


Рис. 13.5: Точний та числовий розв'язки крайової задачі першого роду.

Розв'язок отриманий. Залишилось здійнити певні фінальні дії для створення графіку: заповнити допоміжний масив `Ya` всіма знайденими точками y_i та перетворити `X` та `Ya` у списки. Зображення — на Рис.13.5.

```
(%i39) Ya:make_array(flonum, n+1)$
(%i40) Ya[0]:Y[0]$ Ya[n]:Y[n]$
(%i42) for i:1 thru n-1 do
       (Ya[i]:rhs(sol1a[i]))$
(%i43) x_list:listarray(X);
```

$$(\%o43)\quad [1, 1.2, 1.4, 1.6, 1.8, 2.0]$$

```
(%i44) y_list:listarray(Ya);
```

$$(\%o44)\quad [1, \frac{120927}{88370}, \frac{115759}{70696}, \frac{48319}{26511}, \frac{34269}{17674}, 2]$$

```
(%i45) wxplot2d([rhs(sol1),[discrete,x_list,y_list]],[x,1,2],
       [style,[lines,3],[linespoints,3,3,2,1]],[legend,"Exact","Numeric"],
       [gnuplot_preamble,"set key right bottom"]);
```

13.2.2 Крайові умови другого роду

Нехай задане рівняння

$$(x^2+1)y'' + 4xy' + 2y = 2; \quad y'(0) = 1, y'(2) = 1.$$

Задамо його розв'язок у **Maxima**.

```
(%i1)  deqn2:(1+x^2)*'diff(y,x,2)+4*x*'diff(y,x)+2*y=2;
```

$$(\%o1)\quad \left(x^2+1\right)\cdot\left(\frac{d^2}{d\,x^2}\cdot y\right)+4\cdot x\cdot\left(\frac{d}{d\,x}\cdot y\right)+2\cdot y=2$$

```
(%i2)  sol2:ode2(deqn2,y,x);
```

$$(\%o2)\quad y=\frac{x^2}{x^2+1}+\frac{\%k1\cdot x}{x^2+1}+\frac{\%k2}{x^2+1}$$

Щоб зобразити графік розв'язку, необхідно знайти сталі $\%k1$ та $\%k2$. Для цього скористаємось командою `bc2` для похідної знайденої функції.

```
(%i3)  dsol2:y=diff(rhs(sol2),x);
```

$$(\%o3)\quad y=\frac{2\cdot x}{x^2+1}+\frac{\%k1}{x^2+1}-\frac{2\cdot x^3}{\left(x^2+1\right)^2}-\frac{2\cdot \%k1\cdot x^2}{\left(x^2+1\right)^2}-\frac{2\cdot \%k2\cdot x}{\left(x^2+1\right)^2}$$

```
(%i4)  bc2(dsol2,x=0,y=1,x=2,y=1);
```

$$(\%o4)\quad y=\frac{2\cdot x}{x^2+1}+\frac{1}{x^2+1}-\frac{2\cdot x^3}{\left(x^2+1\right)^2}-\frac{2\cdot x^2}{\left(x^2+1\right)^2}+\frac{12\cdot x}{\left(x^2+1\right)^2}$$

Як видно із порівняння двох останніх виразів, $\%k1 = 1$, $\%k2 = -6$. Підставимо їх у вихідну функцію та накреслимо графік (Рис.13.6).

```
(%i5)  sol2a:subst([%k1=1,%k2=-6],sol2);
```

$$(\%o5)\quad y=\frac{x^2}{x^2+1}+\frac{x}{x^2+1}-\frac{6}{x^2+1}$$

```
(%i6)  wxplot2d(rhs(sol2a),[x,0,4],[style,[lines,3]]);
```

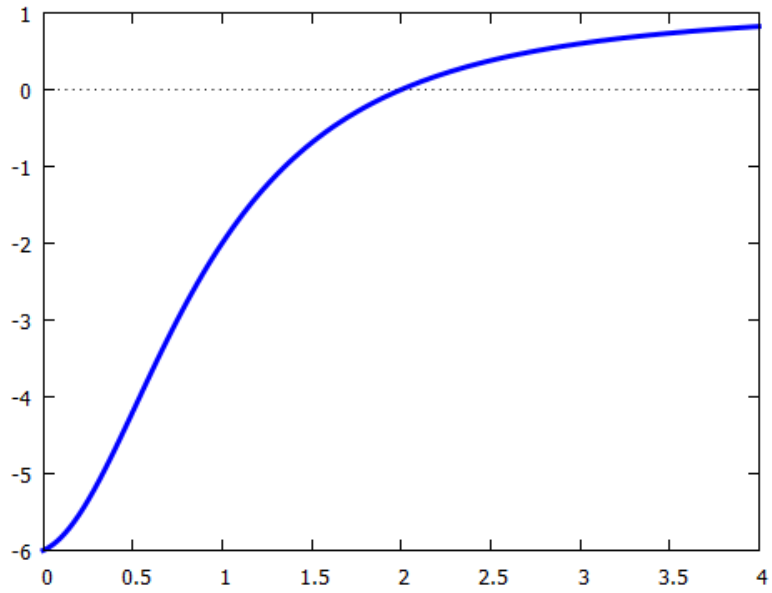


Рис. 13.6: Точний графік крайової задачі.

Наступні дії будемо робити за вже відомою схемою.

```
(%i9)  p(x):=4*x/(x^2+1)$ q(x):=2/(x^2+1)$ r(x):=2/(x^2+1)$
(%i12) a:0$ b:2$ h:0.2$
(%i14) A:1$ B:1$
(%i15) n:floor((b-a)/h)$
(%i16) P:make_array(flonum, n+1)$
(%i17) Q:make_array(flonum, n+1)$
(%i18) R:make_array(flonum, n+1)$
(%i19) X:make_array(flonum, n+1)$
(%i20) Y:make_array(flonum, n+1)$
(%i22) X[0]:a$ X[n]:b$
(%i23) for i:1 thru n do
       X[i]:X[i-1]+h$
(%i24) for i:0 thru n do
       display(X[i]);
```

$X_0 = 0 \quad X_1 = 0.2 \quad X_2 = 0.4 \quad X_3 = 0.6 \quad X_4 = 0.8 \quad X_5 = 1.0$

$X_6 = 1.2 \quad X_7 = 1.4 \quad X_8 = 1.6 \quad X_9 = 1.8 \quad X_{10} = 2.0$

(%o24) *done*

```
(%i25) for i:0 thru n do
       Y[i]:concat(y,i)$
(%i26) for i:0 thru n do
       display(Y[i]);
```

$Y_0 = y0 \quad Y_1 = y1 \quad Y_2 = y2 \quad Y_3 = y3 \quad Y_4 = y4 \quad Y_5 = y5$

$Y_6 = y6 \quad Y_7 = y7 \quad Y_8 = y8 \quad Y_9 = y9 \quad Y_{10} = y10$

(%o26) *done*

```
(%i27) for i:0 thru n do
       P[i]:p(X[i])$
(%i28) for i:0 thru n do
       Q[i]:q(X[i])$
(%i29) for i:0 thru n do
       R[i]:r(X[i])$
(%i30) for i:0 thru n do
       display([P[i],Q[i],R[i]]);
```

$[P_0, Q_0, R_0] = [0, 2, 2]$

$[P_1, Q_1, R_1] = [0.7692307692307693, 1.923076923076923, 1.923076923076923]$

$[P_2, Q_2, R_2] = [1.379310344827586, 1.724137931034483, 1.724137931034483]$

$[P_3, Q_3, R_3] = [1.764705882352941, 1.470588235294118, 1.470588235294118]$

$[P_4, Q_4, R_4] = [1.951219512195122, 1.219512195121951, 1.219512195121951]$

$[P_5, Q_5, R_5] = [2.0, 1.0, 1.0]$

$[P_6, Q_6, R_6] = [1.967213114754098, 0.819672131147541, 0.819672131147541]$

$[P_7, Q_7, R_7] = [1.891891891891892, 0.6756756756756757, 0.6756756756756757]$

$[P_8, Q_8, R_8] = [1.797752808988764, 0.5617977528089888, 0.5617977528089888]$

$[P_9, Q_9, R_9] = [1.69811320754717, 0.4716981132075472, 0.4716981132075472]$

$[P_{10}, Q_{10}, R_{10}] = [1.6, 0.4000000000000001, 0.4000000000000001]$

(%o30) *done*

Задаємо ще один масив для розміщення в ньому системи рівнянь, та заповнюємо його основною формулою.

```
(%i31) sys2:make_array(any, n+1)$
(%i32) for i:1 thru n-1 do
       sys2[i]: (1/h^2)*(Y[i+1]-2*Y[i]+Y[i-1])+
                (P[i]/h)*(Y[i+1]-Y[i])+Q[i]*Y[i]=R[i]$
(%i33) for i:1 thru n-1 do
       display(sys2[i]);
```

Виведені на дисплей дані свідчать, що 9 рівнянь сформовані. У масиві `sys2` ще дві порожніх клітинки, перша та остання. Ми заповнюємо їх двома умовами на похідні в крайових вузлах.

```
(%i34) sys2[0]:(Y[1]-Y[0])/h=A;
```

(%o34) $5.0 \cdot (y1 - y0) = 1$

```
(%i35) sys2[n]:(Y[n]-Y[n-1])/h=B;
```

(%o35) $5.0 \cdot (y10 - y9) = 1$

Перетворюємо нарешті масив у список та розв'язуємо систему з 11 рівнянь.

```
(%i36) sys2a:makelist(sys2[i],i,0,n)$
(%i37) ratprint:false;
(%i38) sol2b:flatten(solve(sys2a,[y0,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10]));
```

(%o38) $[y0 = -\frac{53857894862}{13677128041}, y1 = -\frac{255612346269}{68385640205}, y2 = -\frac{1110794847676}{341928201025},$

$$y3 = -\frac{33343223640209}{12651343437925}, y4 = -\frac{25610626927914}{12651343437925}, y5 = -\frac{748242336109}{506053737517},$$
$$y6 = -\frac{515131919000}{506053737517}, y7 = -\frac{43758515673}{68385640205}, y8 = -\frac{387648542782}{1162555883485},$$
$$y9 = -\frac{99992982797}{1162555883485}, y10 = \frac{26503638780}{232511176697}]$$

Отримали розв'язки, які перетворюємо на числовий список для формування графіка.

```
(%i39) y_list:makelist(rhs(sol2b[i]),i,1,n+1)$
(%i40) x_list:listarray(X)$
(%i41) plot2d([rhs(sol2a),[discrete,x_list,y_list]],[x,0,2],
       [style,[lines,2],[linespoints,2,2,2,1]],
       [legend,"Exact","Numeric"],
       [gnuplot_preamble,"set key right bottom"]);
```

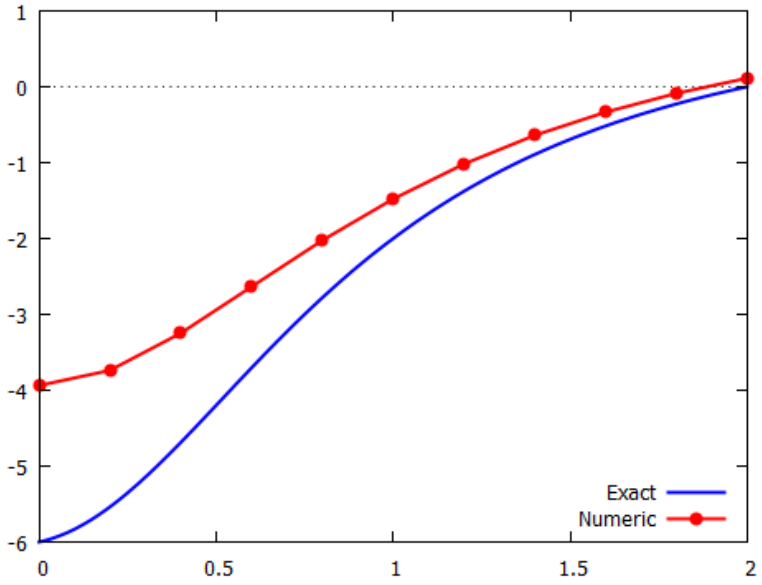


Рис. 13.7: Точний та числовий розв'язки крайової задачі другого роду.

Побудована числова крива з лівого боку досить далеко відходить від точної, хоча профіль її такий самий. Це недолік крайових умов другого роду, рівність похідних двох функцій означає лише що нахил їх буде таким же у відповідних x-координатах точок, тоді як y-координати можуть бути зовсім різними.

13.2.3 Крайові умови третього роду

Вирішимо наступну крайову задачу:

$$x^2y'' = 2y - x, \quad \begin{cases} 2y(1) + y'(1) = 4; \\ y(4) - 2y'(4) = 0. \end{cases}$$

Спочатку задамо це рівняння у **Maxima** та знайдемо точний розв'язок.

```
(%i1)  deqn3:'diff(y,x,2)*x^2=2*y-x;
```

$$(\%o1) \quad x^2 \cdot \left(\frac{d^2}{dx^2} \cdot y\right) = 2 \cdot y - x$$

```
(%i2)  sol3:ode2(deqn3,y,x);
```

$$(\%o2) \quad y = \%k2 \cdot x^2 + \frac{x}{2} - \frac{\%k1}{3 \cdot x}$$

Тепер для побудови графіка необхідно знайти невідомі $\%k1$ та $\%k2$. Для цього обчислимо похідну y' та розв'яжемо систему двох рівнянь, утворену додатковими умовами.

```
(%i3)  dsol3:dy=diff(rhs(sol3),x);
```

$$(\%o3) \quad dy = 2 \cdot \%k2 \cdot x + \frac{\%k1}{3 \cdot x^2} + \frac{1}{2}$$

```
(%i4)  eq1:2*subst(x=1,rhs(sol3))+subst(x=1,rhs(dsol3))=4;
```

$$(\%o4) \quad 2 \cdot \left(\%k2 - \frac{\%k1}{3} + \frac{1}{2}\right) + 2 \cdot \%k2 + \frac{\%k1}{3} + \frac{1}{2} = 4$$

```
(%i5)  eq2:subst(x=4,rhs(sol3))-2*subst(x=4,rhs(dsol3))=0;
```

$$(\%o5) \quad -2 \cdot \left(8 \cdot \%k2 + \frac{\%k1}{48} + \frac{1}{2}\right) + 16 \cdot \%k2 - \frac{\%k1}{12} + 2 = 0$$

```
(%i6)  solve([eq1,eq2],[%k1,%k2]);
```

$$(\%o6) \quad [[\%k1 = 8, \%k2 = \frac{31}{24}]]$$

```
(%i7)  sol3a:subst([%k1=8,%k2=31/24],sol3);
```

$$(\%o7) \quad y = \frac{31 \cdot x^2}{24} + \frac{x}{2} - \frac{8}{3 \cdot x}$$

```
(%i10) plot2d(rhs(sol3a),[x,0.1,5],[style,[lines,3]]);
```

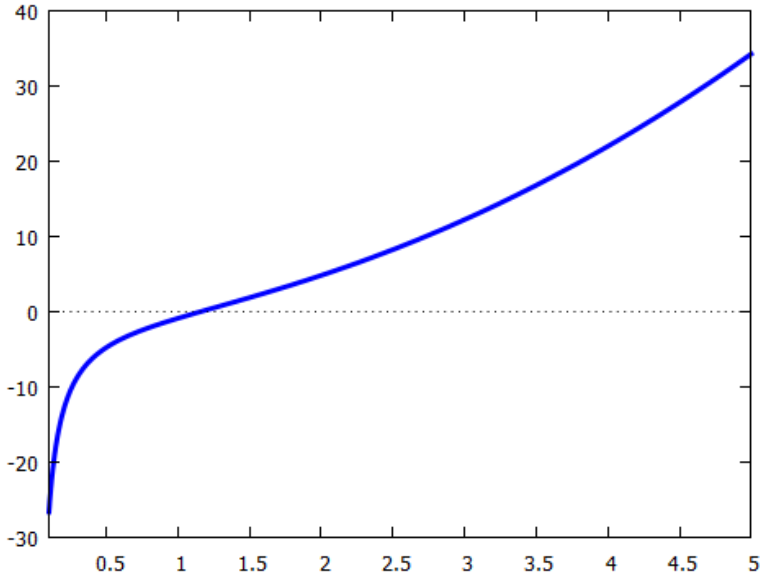


Рис. 13.8: Точний розв'язок крайової задачі.

У попередніх двох випадках при числовому розв'язку ми формували системи рівнянь та безпосередньо вирішували їх вбудованою командою `solve`. Цього разу скористаємось матричним методом. Нагадаємо, що рівняння $M \cdot X = N$, де M — матриця коефіцієнтів біля невідомих, має розв'язок $X = M^{-1} \cdot N$. Отже, необхідно фактично знайти матриці M та N щоб розв'язати систему.

Перш ніж почати числовий розв'язок, перетворимо вихідну систему відносно y_i таким чином, щоб виділити окремо множники при однакових i.

$$\begin{cases} [hp(x_i)+1]\,y_{i+1} + [h^2q(x_i) - hp(x_i) - 2]\,y_i + y_{i-1} = h^2r(x_i); \\ [h\alpha_1 - \alpha_2]\,y_0 + \alpha_2 y_1 = hA; \\ -\beta_2 y_{n-1} + [h\beta_1 + \beta_2]\,y_n = hB. \end{cases}$$

Тепер зробимо наступне зауваження. Попередньо для роз'язання систем рівнянь ми використовували заповнення відповідних масивів. У **Maxima** та **Lisp** індекси масивів можуть приймати значення 0, але для списків та матриць це виявляється неможливим: нумерація елементів починається з 1. Тому необхідно зсунути номери списка функцій на одиницю, отож ми запроваджуємо новий індекс k, який нумерує змінні; він буде набувати значень від 2 до n.

$$\begin{cases} [hp(x_k)+1]\,y_{k+1} + [h^2q(x_k) - hp(x_k) - 2]\,y_k + y_{k-1} = h^2r(x_k); \\ [h\alpha_1 - \alpha_2]\,y_1 + \alpha_2 y_2 = hA; \\ -\beta_2 y_n + [h\beta_1 + \beta_2]\,y_{n+1} = hB. \end{cases}$$

Задаємо вихідні дані нашої задачі у **Maxima**.

```
(%i1)  ratprint:false$
(%i4)  p(x):=0$ q(x):=-2/x^2$ r(x):=-1/x$
(%i9)  a:1$ b:4$ A:4$ B:0$ h:0.5$
(%i13) %alpha_1:2$ %alpha_2:1$ %beta_1:1$ %beta_2:-2$
(%i14) n:floor((b-a)/h);
(%o14)  6
```

Відрізків утворилось 6, отже вузлових точок та рівнянь у системі буде 7. Створюємо тепер список вузлових точок:

```
(%i15) X:makelist(a+(k-1)*h,k,1,n+1);
(%o15)  [1, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0]
```

Для заповнення матриці M коефіцієнтами при невідомих y_k спочатку сформуємо нульову матрицю розмірності $(n+1)\times(n+1)$.

```
(%i16) M:zeromatrix(n+1,n+1)$
```

Тепер заповнимо перший та останній рядки матриці двома додатковими умовами. Перший індекс у $M[j,k]$ це номер рівняння, другий це номер коефіцієнта при відповідному y_k. Отже ми формуємо зараз рядки 1 та $n+1$.

```
(%i18) M[1,1]:h*%alpha_1-%alpha_2$ M[1,2]:%alpha_2$
(%i20) M[n+1,n]:-%beta_2$ M[n+1,n+1]:h*%beta_1+%beta_2$
```

Щоб заповнити всі інші рядки від 2 до n, скористаємось циклом. При цьому треба мати на увазі, що у цих рядках фігуруватимуть функції від змінних x_k. Ці змінні будуть братись із утвореного списку `X[k]`, тому для того, щоб він починався із першого значення (а у нас початковий номер 2), необхідно у рівняннях записувати `X[k-1]`.

```
(%i22) for k:2 thru n do
       for j:1 thru n+1 do
       ( if j=k then M[k,j]: (h^2*q(X[k-1])-h*p(X[k-1])-2)
           else if j=k+1 then M[k,j]: (h*p(X[k-1])+1)
           else if j=k-1 then M[k,j]: 1
           else 0
       )$
```

Поглянемо на сформовану матрицю.

```
(%i23) M;
```

$$(\%o23)\quad \begin{pmatrix} 0.0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2.5 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2.222222222222222 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2.125 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2.08 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2.055555555555555 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & -1.5 \end{pmatrix}$$

Перейдемо до створення матриці-стовпчика правої частини системи. Знову задамо нульову матрицю розмірності $(n+1)\times 1$.

```
(%i24) N:zeromatrix(n+1,1)$
```

Заповнюємо перший та останній рядки.

```
(%i25) N[1,1]:h*A;
```

(%o25) 2.0

```
(%i26) N[n+1,1]:h*B;
```

(%o26) 0

Для заповнення рядків з 2 по n запишемо цикл:

```
(%i27) for k:2 thru n do
       N[k,1]:h^2*r(X[k-1]);
```

(%o27) *done*

Виведемо матрицю N на екран.

```
(%i28) N;
```

$$(\%o28)\quad \begin{pmatrix} 2.0 \\ -0.25 \\ -0.1666666666666666 \\ -0.125 \\ -0.1 \\ -0.08333333333333333 \\ 0 \end{pmatrix}$$

Нарешті знаходимо розв'язок системи, і перетворюємо його на список для створення графіку (Рис.13.9).

```
(%i29) Ya:invert(M).N;
```

$$(\%o29)\quad \begin{pmatrix} 1.09768907563025 \\ 2.0 \\ 3.652310924369749 \\ 5.949579831932778 \\ 8.865546218487402 \\ 12.39075630252102 \\ 16.52100840336136 \end{pmatrix}$$

```
(%i30) y_list:flatten(args(Ya))$
(%i31) plot2d([(31*x^2)/24+x/2-8/(3*x),[discrete,X,y_list]],[x,1,4],
       [style,[lines,2],[linespoints,2,2,2,1]],[legend,"Exact","Numeric"],
       [gnuplot_preamble,"set key right bottom"])$
```

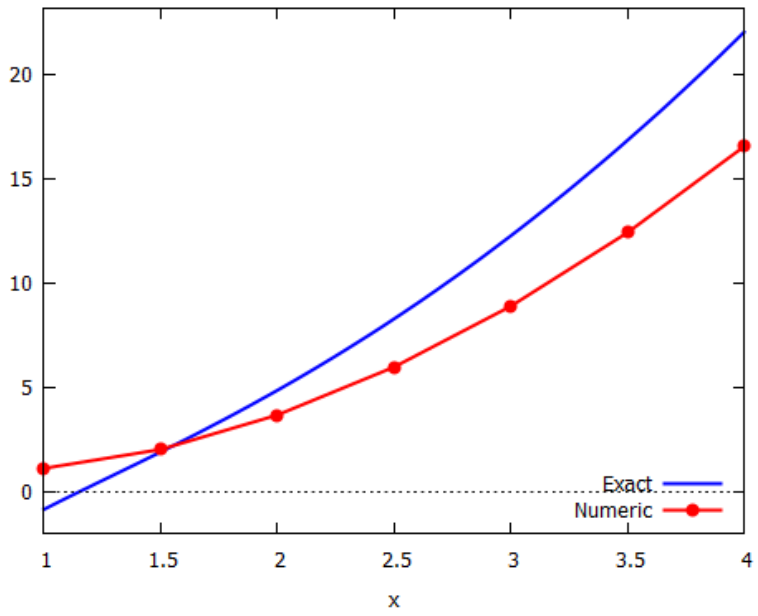


Рис. 13.9: Точний та числовий розв'язки крайової задачі третього роду.

Як видно з порівняння графіків, числовий розв'язок крайових задач даного типу зберігає форму кривої та має досить близьке розміщення обчислених точок і точної кривої.

13.3 Прямий двокроковий метод у задачі Коші для ЗДР другого порядку

У фізичних процесах часто зустрічається ситуація, коли для даної динамічної системи відомі початкове значення якоїсь величини у часі та початкове значення швидкості цієї величини. Для прикладу, саме такий випадок ми розглядали при вивченні коливань маятника. З точки зору математики це еквівалентно заданню для ЗДР другого порядку значення $y(x_0)$ та $y'(x_0)$. Теорія диференціальних рівнянь говорить, що цих величин достатньо для однозначного вирішення ЗДР, тому ця задача є підвидом задачі Коші.

Пригадаємо метод Ейлера при розв'язанні ЗДР першого порядку: у ньому похідна в точці замінялась різницевим відношенням, яке пов'язувало між собою y_{i+1} та y_i, звідси кожне наступне значення функції виражалось через попереднє. При розгляді ЗДР другого порядку дискретизоване рівняння пов'язує вже три точки: y_{i+1}, y_i та y_{i-1}. Отже, для задання функції y у наступній точці, необхідно визначити її величину у двох попередніх.

Значення $y(x_0)$ дає нам функцію у першому вузлі сітки, значення $y'(x_0)$ (представлене у вигляді різницевого відношення) пов'язує перший та другий вузол сітки, а отже дає нам функцію у другому вузлі. Ця обставина дозволяє нам застосувати прямий метод Ейлера у випадку ЗДР другого порядку, який тепер став двокроковим.

Загальний план дій полягає в наступному:

1) записати вихідне диференціальне рівняння у дискретизованому вигляді, викори-

стовуючи вирази для похідних

$$y_i' = \frac{1}{h}(y_{i+1} - y_i); \quad y_i'' = \frac{1}{h^2}\left(y_{i+1} - 2y_i + y_{i-1}\right),$$

2) визначити з нього величину y_{i+1},

3) використовуючи цикл знайти значення функції у всіх точках сітки.

Реалізуємо цю схему на прикладі рівняння коливань маятника із тертям та вимушуючою силою.

Нехай задане рівняння

$$y'' + 2y' + y = x^3; \quad y(0) = 4, \quad y'(0) = -1.$$

Знайдемо його розв'язок у **Maxima**, застосувавши команду `ic2` для визначення невідомих коефіцієнтів.

```
(%i1) deqn4:'diff(y,x,2)+2*'diff(y,x)+y=x^3;
```

$$(\%o1) \quad \frac{d^2}{dx^2}\cdot y + 2\cdot\left(\frac{d}{dx}\cdot y\right) + y = x^3$$

```
(%i2) sol4:ode2(deqn4,y,x);
```

$$(\%o2) \quad y = (\%k2\cdot x + \%k1)\cdot e^{-x} + x^3 - 6\cdot x^2 + 18\cdot x - 24$$

```
(%i3) sol4a:ic2(sol4,x=0,y=4,'diff(y,x)=-1);
```

$$(\%o3) \quad y = (9\cdot x + 28)\cdot e^{-x} + x^3 - 6\cdot x^2 + 18\cdot x - 24$$

```
(%i4) wxplot2d(rhs(sol4a),[x,0,3],[style,[lines,2,1]],[ylabel,"y"]);
```

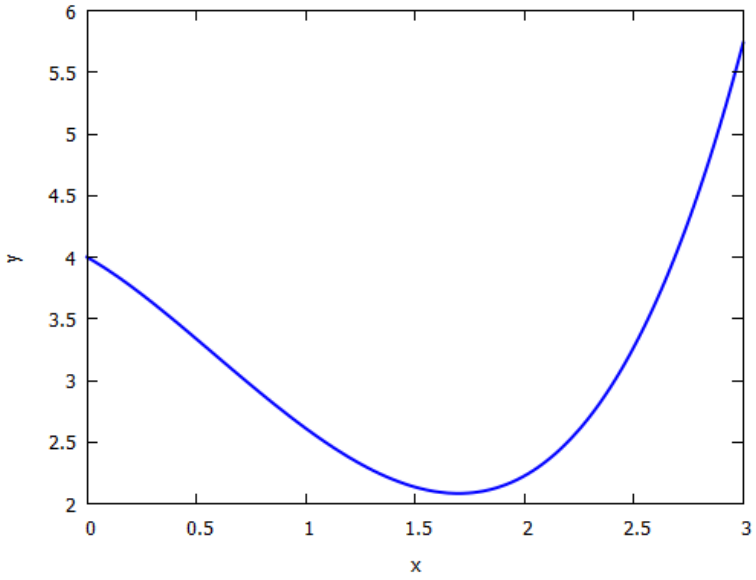


Рис. 13.10: Точний розв'язок задачі Коші для ЗДР другого порядку.

Тепер здійснимо дискретизацію задачі на проміжку $[0; 3]$ з кроком $h = 0.05$.

```
(%i7) a:0$ b:3$ h:0.05$
(%i8) n:floor((b-a)/h);
```

$$(\%o8) \quad 60$$

Створюємо масив `X` вузлових точок.

```
(%i9)  for i:0 thru n do
       (arraymake(X, [i]), X[0]:a,
       X[i]:X[i-1]+h);
(%o9)    done
```

Вносимо початкові дані у масив `Y`. При цьому маємо на увазі, що оскільки

$$y'(x_0) = \frac{y_1 - y_0}{h} = -1, \quad \text{то} \quad y_1 = -h + y_0.$$

```
(%i11) Y[0]:4$ Y[1]:(-1)*h+Y[0]$
```

Формуємо дискретизоване вихідне рівняння та розв'язуємо його відносно `Y[i+1]`.

```
(%i12) deqn4b:(1/h^2)*(Y[i+1]-2*Y[i]+Y[i-1])+
                2*(1/h)*(Y[i+1]-Y[i])+Y[i]=(X[i])^3;
```

$$(\%o12) \quad 40.0 \cdot (Y_{i+1} - Y_i) + 399.9999999999999 \cdot (Y_{i+1} - 2 \cdot Y_i + Y_{i-1}) + Y_i = X_i^3$$

```
(%i13) solve(deqn4b,Y[i+1]);
```

$$(\%o13) \quad [Y_{i+1} = \frac{839 \cdot Y_i + X_i^3 - 400 \cdot Y_{i-1}}{440}]$$

Записуємо цикл для знаходження усіх точок масиву `Y`. Отримані точки перетворюємо у списки та формуємо графік.

```
(%i14) for i:1 thru n-1 do
       Y[i+1]:(839*Y[i]+X[i]^3-400*Y[i-1])/440;
(%o14)   done

(%i15) x_list:listarray(X)$
(%i16) y_list:listarray(Y)$
(%i18) plot2d([rhs(sol4a),[discrete,x_list,y_list]],[x,0,3],
       [style,[lines,1],[points,1,2,1]],[legend,"Exact","Numeric"])$
```

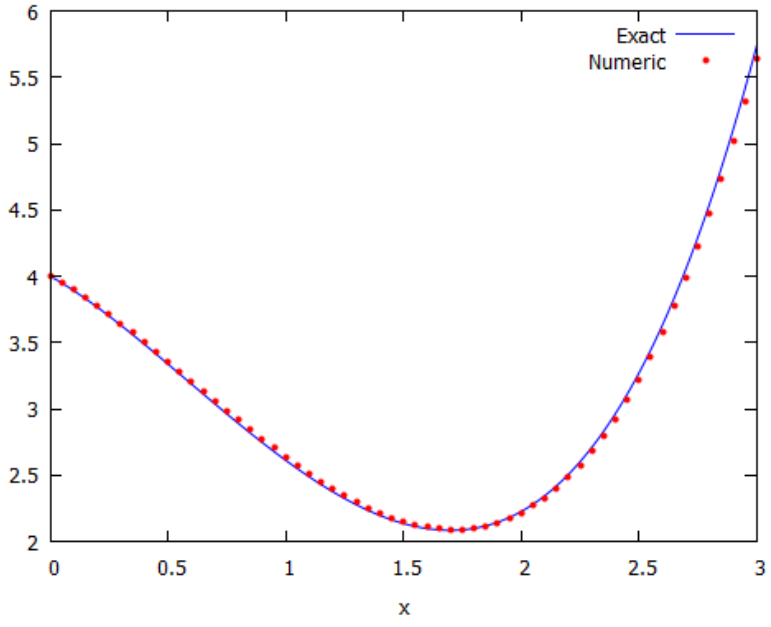


Рис. 13.11: Точний розв'язок ЗДР та числовий за прямим двокроковим методом.

Як видно, обчислені точки чудово співпадають із точним значенням знайденої аналітично функції. Може здатися, що цей метод надає найкращий результат порівняно зі всіма, розглянутими раніше. Це пов'язано, однак, лише з тим, що тут використана набагато більша кількість вузлових точок. Якби така сама кількість їх була обрана у попередніх випадках, результат був би приблизно подібний. Ця ситуація підкреслює важливість підбору щонайменшого дискретного проміжку при числовому інтегруванні. Для прикладу, якби у цій задачі обрати кількість проміжків 6 (а точок відповідно 7), то картина була б отримана така, як на Рис.13.12.

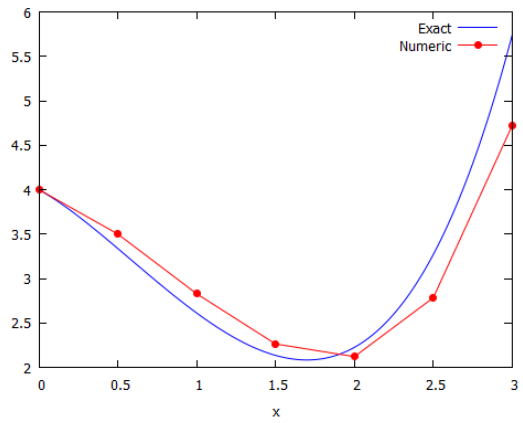


Рис. 13.12: Точний розв'язок ЗДР та числовий при $n = 6$.

Із графіка можна відмітити характерні особливості, що властиві методу Ейлера: на проміжках, де функція опукла, числові точки знаходяться вище неї, тоді як на проміжках, де функція вгнута — нижче. Це наслідок заміни функції на інтервалі дискретності її дотичною. Обчислена нами крива має як проміжки вгнутості, так і опуклості, і ця щаслива обставина трохи скомпенсувала розбіжності. При інтегруванні монотонних функцій помилка весь час набігає, і в результаті обчислена крива на кінцевих відрізках досить далеко відходить від справжнього розв'язку.

Втім, поряд з цими недоліками розглянутий двокроковий метод має досить суттєві переваги порівняно з іншими: по-перше, таким чином можна розглядати не лише лінійні ЗДР другого порядку, але й нелінійні, головне щоб з дискретизованого рівняння можна було окремо виразити величину y_{i+1}; по-друге, кількість вузлових точок можна значно збільшити (із десятків до тисяч), оскільки для обчислювального застосунку (не лише **Maxima**, а й будь-якого іншого, для прикладу **Maple** чи **Python**) набагато легше маніпулювати числовим масивом із n значень, аніж розв'язувати систему із n рівнянь з n невідомими. Такі переваги обумовлюють широке використання цього методу у вирішенні різних задач математики та математичної фізики.

13.4 Завдання до Розділу 13

ЗДР першого порядку

Розв'язати ЗДР першого порядку чисельно за методами Ейлера, Ейлера-Коші та Рунге-Кутта 4. Знайти аналітичний розв'язок та зобразити всі криві на одному графіку.

13.1. $2y' + 3y\cos x = y^{-1}e^{2x}(2 + 3\cos x); \quad y(0) = 1; \quad x \in [0; 3].$

13.2. $y' = \dfrac{x - 2y - 3}{-2x - 2}; \quad y(0) = 3; \quad x \in [0; 4].$

13.3. $y' = \dfrac{x + 2y - 3}{x - 1}; \quad y(2) = 4; \quad x \in [2; 5].$

13.4. $y' = 2xe^{x^2-y}; \quad y(0) = \ln 2; \quad x \in [1; 4].$

13.5. $y' + (2y + 1)\operatorname{ctg} x = 0; \quad y(\pi/2) = 1; \quad x \in [0.5; 2.5].$

13.6. $y \ln y + xy' = 0; \quad y(1) = e; \quad x \in [1; 4].$

13.7. $y - xy' = 3(1 + x^2 y'); \quad y(1) = 1; \quad x \in [1; 5].$

13.8. $xy' = y - xe^{y/x}; \quad y(1) = 0; \quad x \in [1; 6].$

13.9. $xy' - y = x \operatorname{tg} \frac{y}{x}; \quad y(2/\pi) = 1; \quad x \in [-0.5; 0.5].$

13.10. $y' = \dfrac{5x^2 - xy + y^2}{x^2}; \quad y(1) = 1; \quad x \in [0; 2].$

Прямий двокроковий метод для ЗДР другого порядку

Розв'язати ЗДР другого порядку за допомогою прямого двокрокового методу з кроком $h = 0.05$ на проміжку $[a; b]$. Зобразити точний розв'язок та числовий на одному графіку.

13.11. $y'' - y' + x^2 = \sin x; \quad y(0) = 1, y'(0) = 4; \quad x \in [0; 3].$

13.12. $y'' - y' \cdot x^2 = 0; \quad y(0) = 1, y'(0) = 2; \quad x \in [-3; 0].$

13.13. $xy'' - 0.3y' + x = 0; \quad y(0) = 2, y'(0) = -2; \quad x \in [0; 4].$

13.14. $y'' - 0.1y' + e^x + y = 0; \quad y(0) = 4, y'(0) = 1; \quad x \in [0; 5].$

13.15. $y'' + 3y' - 4y = e^{-4x}; \quad y(0) = 6, y'(0) = 2; \quad x \in [0; 3].$

13.16. $y'' + y = x^3; \quad y(0) = 4, y'(0) = -1; \quad x \in [-1; 3].$

13.17. $y'' - y + \cos x = x^2; \quad y(0) = 1, y'(0) = 3; \quad x \in [-1; 2].$

13.18. $y'' + y' + y = \sin^2 x; \quad y(0) = 0, y'(0) = 6; \quad x \in [0; 6].$

13.19. $y'' + 5y' - y = e^x; \quad y(0) = 1, y'(0) = 1; \quad x \in [0; 4].$

13.20. $y'' + (2x + 3)y' = 0; \quad y(0) = 2, y'(0) = 1; \quad x \in [-2; 1].$

Розділ 14

Числові методи для ДР у частинних похідних

Більшість математичних моделей фізичних процесів породжують системи лінійних або нелінійних рівнянь у частинних похідних. Оскільки аналітичні засоби не завжди придатні для вирішення цих рівнянь, їх необхідно розв'язувати числовими методами. Тут розглядатимемо методи скінченних різниць (FDM — Finite Difference Method).

14.1 Класифікація ДР у частинних похідних

Будь-яке лінійне диференціальне рівняння у частинних похідних другого порядку може бути представлене у вигляді

$$A(x)u_{xx} + B(x)u_{xy} + C(x)u_{yy} + D(x)u_x + E(x)u_y + F(x)u + G(x) = 0.$$

В залежності від вигляду множників $A(x), B(x), C(x)$ диференціальні рівняння поділяються на:

- *еліптичного типу*, при
$$B^2 - 4AC < 0;$$
- *параболічного типу*, при
$$B^2 - 4AC = 0;$$
- *гіперболічного типу*, при
$$B^2 - 4AC > 0.$$

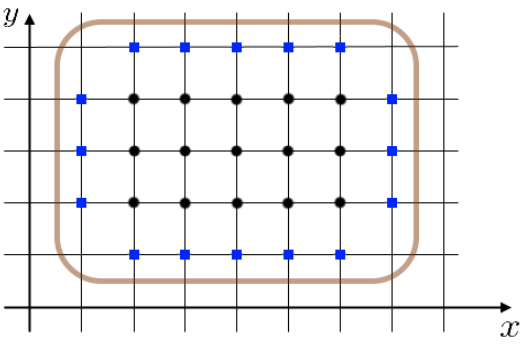


Рис. 14.1: Область інтегрування із внутрішніми (•) та граничними (■) вузлами.

Для вирішення диференціальних рівнянь методом скінченних різниць (сіток) та область, на якій шукається розв'язок, замінюється дискретною множиною точок. В цьому методі, як правило, використовуються регулярні сітки, крок яких або сталий, або змінюється за нескладним законом.

Нехай в якості області зміни функції задано прямокутник. Осі Ox та Oy розбиваються на відрізки, які є кроками сітки за відповідними напрямками. Через точки поділу проводяться прямі, паралельні осям

координат. Сукупність точок перетину (вузлів) цих прямих утворює сітку в заданій двовимірній області (Рис.14.1). Вузли, відстані між якими рівні кроку сітки по одній із осей, називаються сусідніми. Спосіб побудови сітки не змінюється і в тому випадку, коли задана область довільної форми. Вузли сітки, які потрапили всередину області, називаються внутрішніми. Точки перетину прямих, що творять сітку, з межами області, називаються граничними вузлами.

Наступний крок полягає у дискретизації вихідного рівняння. Нехай крок сітки по осі Ox буде рівним h. Розкладемо функцію $f(x+h)$ у ряд Тейлора поблизу x:

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + ...$$

Утримуючи перші два члени розкладу, можемо записати

$$f'(x) = \frac{1}{h}(f(x+h) - f(x)).$$

Цей вираз називається правою різницевою похідною (з якою ми вже мали справу раніше при розгляді методу Ейлера). Вона апроксимує похідну $f'(x)$ в точці x.

Замінивши в розкладі Тейлора h на $-h$, отримаємо ліву різницеву похідну

$$f'(x) = \frac{1}{h}(f(x) - f(x-h)).$$

Нарешті, додавши ці дві рівності та розділивши на 2, отримаємо центральну різницеву похідну

$$f'(x) = \frac{1}{2h}(f(x+h) - f(x-h)).$$

Останню використовують для визначення другої похідної $f''(x)$. Підставивши її у ряд Тейлора та утримавши перші три члени розкладу, маємо

$$f''(x) = \frac{1}{h^2}(f(x+h) - 2f(x) + f(x-h)).$$

Розглянемо тепер функцію від двох змінних $u(x,y)$. Розіб'ємо сіткою вісь Oy прямими з кроком s. Ряди Тейлора такої функції матимуть вигляд

$$u(x+h,y) = u(x,y) + u_x(x,y)h + u_{xx}(x,y)\frac{h^2}{2!} + ...;$$

$$u(x-h,y) = u(x,y) - u_x(x,y)h + u_{xx}(x,y)\frac{h^2}{2!} + ...;$$

$$u(x,y+s) = u(x,y) + u_y(x,y)s + u_{yy}(x,y)\frac{s^2}{2!} + ...;$$

$$u(x,y-s) = u(x,y) - u_y(x,y)s + u_{yy}(x,y)\frac{s^2}{2!} + ...$$

Звідси маємо наступні апроксимації частинних похідних:

$$u_x(x,y) = \frac{1}{h}(u(x+h,y) - u(x,y));$$

$$u_y(x,y) = \frac{1}{s}(u(x,y+s) - u(x,y));$$

$$u_{xx}(x,y) = \frac{1}{h^2}(u(x+h,y) - 2u(x,y) + u(x-h,y));$$

$$u_{yy}(x,y) = \frac{1}{s^2}(u(x,y+s) - 2u(x,y) + u(x,y-s)).$$

Різницеві вирази, що відповідають диференціальному рівнянню, записуються для внутрішніх вузлів сітки, а такі, що відповідають крайовим умовам, записуються для граничних вузлів сітки. В результаті отримується система алгебричних рівнянь, кількість яких рівна кількості вузлів. Для отримання числового розв'язку її необхідно вирішити яким-небудь методом.

Роль однієї із змінних може відігравати час t, тоді часовий проміжок теж дискретизується та розбивається на скінченну кількість точок. Отримані розв'язки будуть відображати часову еволюцію системи.

Додаткові умови, що накладаються на ДР, повинні призводити до однозначного розв'язку. Це означає, що завжди їхня кількість буде рівна сумарній кількості ступенів похідних у рівнянні. Для кожного типу рівнянь, в залежності від його вигляду і значень змінних, додаткові умови носять назви відповідних вчених, що їх розглядали. Загалом ця проблема в англомовній літературі іменується IBVP — Initial Boundary Value Problem, оскільки вона є комбінацією задачі Коші та крайової задачі.

Нехай функція $u(x,y)$ приймає значення у відповідних вузлових точках (x_i, y_j). Після здійснення операції дискретизації вирази для похідних перетворюються на наступні:

$$u(x_i,y_j) = u_{i,j}; \quad u(x_i+h,y_j) = u_{i+1,j}; \quad u(x_i-h,y_j) = u_{i-1,j};$$

$$u(x_i,y_j+s) = u_{i,j+1}; \quad u(x_i,y_j-s) = u_{i,j-1};$$

$$u_x(x_i,y_j) = \frac{1}{h}(u_{i+1,j} - u_{i,j}); \quad u_y(x_i,y_j) = \frac{1}{s}(u_{i,j+1} - u_{i,j});$$

$$u_{xx}(x_i,y_j) = \frac{1}{h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}); \quad u_{yy}(x_i,y_j) = \frac{1}{s^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}).$$

14.2 Рівняння параболічного типу

Такі рівняння описують процес теплопровідності та процес дифузії у речовинах. Загальний вигляд їх наступний:

$$\frac{\partial u(\vec{r},t)}{\partial t} = \sigma \nabla^2 u(\vec{r},t).$$

Коефіцієнт пропорційності σ для кожної задачі має своє фізичне значення. Наприклад, для рівняння теплопровідності u — температура точки тіла у заданий час, $\sigma = k/(c\rho)$, де k — коефіцієнт теплопровідності, c — питома теплоємність, ρ — густина маси речовини. Для рівняння дифузії u — концентрація речовини у просторі в заданий момент часу, $\sigma = D/c$, де D — коефіцієнт дифузивного потоку, c — коефіцієнт пористості тіла. Загалом кожні з цих величин можуть бути не сталими, а залежати від точки простору.

14.2.1 Задача Діріхле одновимірного рівняння теплопровідності

Нехай маємо стрижень довжиною L, в якому відбуваються певні теплові процеси. Це можуть бути як нагрівання якихось частин зовнішніми джерелами, так і внутрішнє виділення тепла внаслідок електричного опору.

Розглянемо одновимірне рівняння теплопровідності

$$u_t(x,t) = \sigma u_{xx}(x,t).$$

Присутня одна похідна по t та дві по x, отже для однозначного вирішення необхідна одна початкова умова — розподіл температури в нульовий момент часу, та дві крайові умови. Із Розділу 13 ми вже знаємо, що крайових умов можуть бути три види: задані значення функції на кінцях відрізка, значення похідних функції на кінцях відрізка, або їх лінійна комбінація. Крайовим умовам першого роду відповідає задання температурної залежності від часу на кінцях стрижня. Отже, розв'язання рівняння теплопровідності з умовами

$$\begin{cases} u(x,0) = A(x); \\ u(0,t) = B_1(t); \\ u(L,t) = B_2(t) \end{cases}$$

називається *задачею Діріхле*.

Розв'яжемо цю задачу числово за допомогою прямого методу Ейлера. Розіб'ємо довжину стрижня L на m частин, проміжок часу T на n частин. Тоді вузлових точок буде відповідно $m+1$ та $n+1$, а довжина кожного проміжка $h = L/m$, $s = T/n$. Запишемо дискретизоване вихідне рівняння, прийнявши за i — просторовий індекс, k — часовий індекс:

$$\frac{1}{s}(u_{i,k+1} - u_{i,k}) = \frac{\sigma}{h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}).$$

Граничні умови тепер матимуть вигляд

$$u_{i,0} = A_i; \quad u_{0,k} = B_{1;k}; \quad u_{m,k} = B_{2;k}.$$

Виразивши з цього рівняння $u_{i,k+1}$, матимемо

$$u_{i,k+1} = \frac{\sigma s}{h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}) + u_{i,k}.$$

Отримана рівність виражає функцію u в наступний момент часу $k+1$ через її значення в попередній момент часу k в різних точках. Задання значення функції при $k = 0$ дозволяє обрахувати її для $k = 1$, $k = 2$ і т.д., аж поки часовий інтервал не вичерпається.

Як бачимо, в правій частині присутній множник $\sigma s/h^2$. Для того, щоб розв'язок був стійким та не розбігався на нескінченність, необхідною умовою є меншість цього множника за $1/2$, тому цю обставину потрібно перевіряти.

► Вирішимо наступну задачу Діріхле: розв'язати рівняння теплопровідності для стрижня довжиною 1 м, час спостереження 0.2 с, при $\sigma = 1$, з граничними умовами

$$u(x,0) = x(1-x); \quad u(0,t) = 0; \quad u(1,t) = 0.$$

Переведемо цю задачу на мову **Maxima**. Приймемо довжину стрижня $L = 1$, проміжок часу $T = 0.2$ та $\sigma = 1$. Кількість проміжків оберемо $m = 30$, $n = 600$.

```
(%i3)  L:1$ T:0.2$ %sigma:1$
(%i5)  m:30$ n:600$
(%i7)  h:L/m$ s:T/n$
```

Перевіряємо необхідну умову стійкості:

```
(%i8)  %sigma*s/h^2,numer;
(%o8)     0.30000000000000004
```

Записуємо граничні умови. Функціональну залежність $A(x)$ відразу перетворюємо на числа, оскільки числовий масив набагато легше опрацьовується системою:

```
(%i9)  A(x):=float(x*(1-x))$
(%i11) B1(t):=0$ B2(t):=0$
```

Створюємо масив `X` та заповнюємо його вузловими точками x-координат.

```
(%i12) for i:1 thru m do
       (arraymake(X, [i]), X[0]:0,
       X[i]:X[i-1]+h);
(%o12)  done
```

Створюємо масив `T1` (символ `T` вже зайнятий) та заповнюємо його вузловими точками t-координат.

```
(%i13) for k:1 thru n do
       (arraymake(T1, [k]), T[0]:0,
       T1[k]:T1[k-1]+s);
(%o13)  done
```

Тепер записуємо у масив `u` початкові та крайові умови.

```
(%i14) for i:0 thru m do
       u[i,1]:A(X[i]);
(%o14)  done

(%i15) for k:0 thru n do
       (u[0,k]:B1(T1[k]),
        u[m,k]:B2(T1[k]));
(%o15)  done
```

Записуємо базове рівняння для $u_{i,k+1}$ та обчислюємо його, створивши відповідний цикл.

```
(%i16) for k:1 thru n do
       for i:1 thru m-1 do
       u[i,k+1]:(%sigma^2·s/h^2)*(u[i+1,k]-2·u[i,k]+u[i-1,k])+u[i,k];
(%o15)  done
```

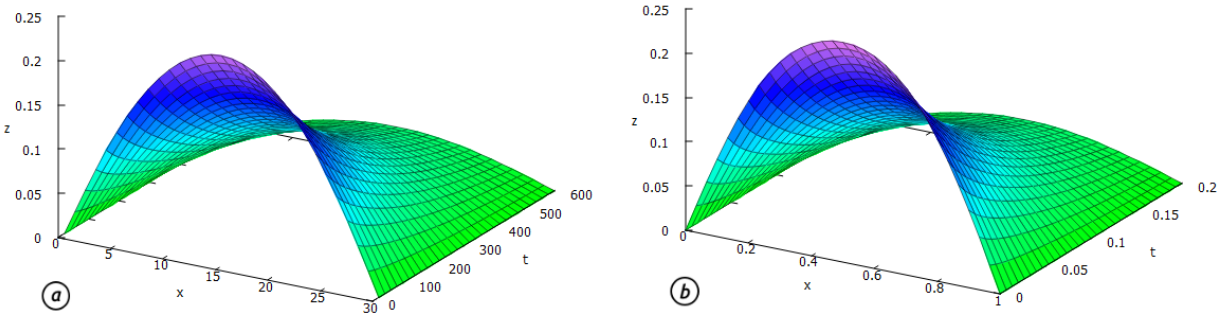


Рис. 14.2: Числовий (а) та точний (б) розрахунок задачі Діріхле для одновимірного рівняння теплопровідності.

Отримали набір відповідних значень для $u(x,t)$. Щоб побудувати графік, небхідно перетворити масив у функцію. «Змінними» елементами масива є цілі числа, тому округлюємо до цілого будь-яке число x та t командою `round`.

```
(%i17) G(x,t):=float(u[round(x),round(t)])$
```

Нарешті створюємо графік обчисленого розподілу температур (Рис.14.2(а)):

```
(%i18) plot3d(G(x,t),[x,0,m],[t,0,n]);
```

Щоб оцінити правильність отриманого розв'язку, скористаємось точним виразом для $u(x,t)$. Курс математичної фізики дає наступний результат:

$$u(x,t) = \sum_{1}^{\infty} D_n e^{-kn^2\pi^2 t/L^2} \sin\frac{\pi n x}{L}, \quad \text{де} \quad D_n = \frac{2}{L}\int_0^L A(x)\sin\frac{\pi n x}{L}dx.$$

Задавши ці співвідношення у **Maxima** та побудувавши графік (обмежившись $N = 3$ у сумі ряду), маємо Рис.14.2(б).

Бачимо досить гарний збіг обчислених даних та точної поверхні. Це пояснюється «гладкістю» початкової функції та виконанням усіх необхідних умов для стійкості моделі. Ця обставина вселяє впевненість у тому, що модель буде достатньо точно описувати і ті випадки, коли аналітичний обрахунок затруднений.

Створимо тепер анімацію еволюції розв'язку в часі. Оскільки у нас є 600 часових вузлових точок, зменшуємо кількість кадрів до 40, обравши кожен 15-ий вузол.

```
(%i19) with_slider(k,makelist(15*i,i,1,n/15),G(x,k),
       [x,0,m],[y,0,0.25],[style,[lines,2]]);
```

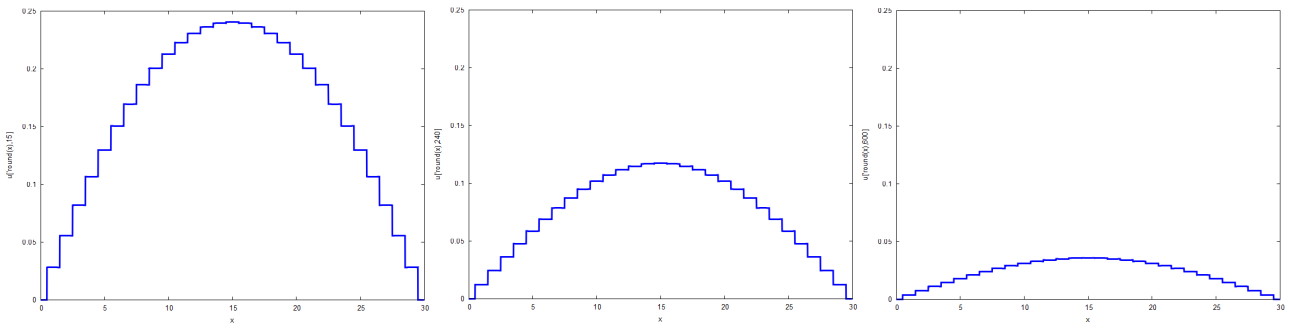


Рис. 14.3: Кадри анімації залежності обрахованої температури стрижня від часу.

Для наочності розподілу температури скористаємось можливістю розфарбовування елементів матриці із пакету `draw`. Для цього створимо спочатку нульову матрицю розмірністю $m \times n$ та заповнимо її елементами двовимірного масиву `u`. Отримана таким чином матриця переводиться в кольорове представлення згідно з її числовими значеннями в кожній комірці.

Найбільше значення чисел у матриці становить 0.25, оскільки це максимум функції $A(x) = x(1-x)$. Постає питання, яка ж реальна температура буде на стрижні. Звернемося до початкового розподілу $A(x)$. Зрозуміло, що ця величина повинна мати розмірність температури, тобто фактично наведена функція у справжніх розрахунках прийматиме вигляд $A(x) = A_0\frac{x}{L}\left(1-\frac{x}{L}\right)$, де A_0 — певне задане характеристичне значення, $\frac{x}{L}$ — безрозмірна координата. При такій постановці задачі обчислене значення u буде безрозмірною температурою, і для переходу до реальної її необхідно домножити на A_0.

З'ясуємо ще значення коефіцієнта σ в розрахунках. Для кожного металу він має свою величину. Наприклад, для заліза:

$k = 80.4$ Вт/м·К; $c = 640.57$ Дж/кг·К; $\rho = 7874$ кг/м^3.

Звідси $\sigma = 1.6 \cdot 10^{-5}$ м^2/с. Фізична роль σ наступна: це «температурна пропускна здатність» стрижня, тобто властивість матеріалу змінювати свою температуру за одиницю часу. Обране модельне значення $\sigma = 1$ у розглянутому прикладі є надзвичайно великим.

```
(%i20) U:zeromatrix(m,n)$
(%i21) for i:1 thru m do
       for k:1 thru n do
       U[i,k]:u[i,k];
(%o21)   done

(%i22) load(draw)$
(%i23) wxdraw2d(image(U,0,0,0.2,1),
       font  = "Ubuntu Medium",
       font_size = 18,
       xlabel = "час, с",
       ylabel = "довжина, м");
(%o23)   [gr2d(image)]
```

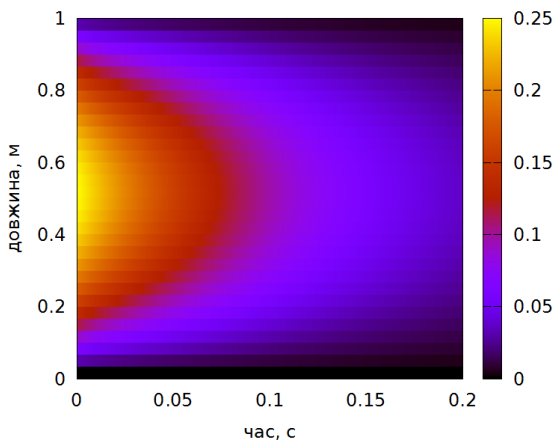


Рис. 14.4: Температурна діаграма стрижня у часі.

Задача Діріхле, яку ми щойно розв'язали, називається задачею з однорідними крайовими умовами, оскільки температура на кінцях стрижня однакова. Якщо ж ця температура буде різною, наприклад T_1 та T_2, умови стають неоднорідними. Теорія процесів теплопровідності у такому випадку дає наступний результат:

$$u(x,t) = u_1(x) + u_2(x,t),$$

де $u_1(x) = \left(\dfrac{T_2 - T_1}{L}\right) x + T_1$,
$u_2(x,t)$ — розв'язок однорідної задачі з умовами

$$u_2(x,0) = A(x) - u_1(x); \quad u_2(0,t) = 0; \quad u_2(L,t) = 0.$$

З фізичної точки зору сталість температури на кінцях стрижня може забезпечуватись поміщенням їх у резервуари із нескінченною теплоємністю (температура резервуара не змінюється, скільки б тепла він не поглинув), та нескінченною теплопровідністю (тепло миттєво забирається з кінців стрижня). Звісно, такі умови є абстракцією та лише наближено можуть реалізуватись на практиці. Взяте нами велике значення σ пояснює практично миттєве спадання загальної температури стрижня до нуля (за 0.2 секунди), все тепло якого відійшло у резервуари.

14.2.2 Задача Неймана одновимірного рівняння теплопровідності

Для крайових умов третього роду розв'язку диференціального рівняння другого порядку необхідне задання комбінації значення функції та значення похідної функції на краях відрізка. Отже, розв'язання рівняння теплопровідності з умовами

$$\begin{cases} u(x,0) = A(x); \\ u(0,t) = B(t); \\ u_x(L,t) = C(t) \end{cases}$$

називається *задачею Неймана*.

Ця задача описує стрижень, один кінець якого міняє свою температуру згідно з законом $B(t)$, через інший кінець відбувається теплообмін з навколишнім середовищем, тобто до стрижня або надходить тепло ($C(t) > 0$), або відходить ($C(t) < 0$). Фізично це означає поміщення лівого кінця в ідеальний резервуар, та нагрівання або охолодження правого кінця.

Здійснимо операцію дискретизації системи за попереднім сценарієм. Базове рівняння матиме вигляд

$$\frac{1}{s}(u_{i,k+1} - u_{i,k}) = \frac{\sigma}{h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}),$$

а граничні умови будуть наступні:

$$u_{i,0} = A_i; \quad u_{0,k} = B_k; \quad \frac{1}{h}(u_{m,k} - u_{m-1,k}) = C_k.$$

Нехай температура зовнішнього середовища змінюється за законом $g(t)$. В досить широкому колі явищ тепловий потік прямо пропорційний різниці температур середовища та кінця стрижня, тобто

$$C(t) = c(g(t) - u(L,t)), \quad \text{або дискретизовано} \quad C_k = c(g_k - u_{m,k}).$$

Підставивши це співвідношення в граничну умову, маємо

$$\frac{1}{h}(u_{m,k} - u_{m-1,k}) = c(g_k - u_{m,k}), \quad \text{тоді} \quad u_{m,k} = \frac{u_{m-1,k} + chg_k}{1 + ch}.$$

Цей вираз ми візьмемо для визначення значення функції $u(L,t)$ в наступний момент часу $k+1$. Дискретизована система в такому випадку складатиметься з двох рівнянь:

$$\begin{cases} u_{i,k+1} = \dfrac{\sigma s}{h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}) + u_{i,k}, & i = 1...m-1; \\ u_{m,k+1} = \dfrac{u_{m-1,k} + chg_k}{1 + ch}, & i = m. \end{cases}$$

► Вирішимо наступну задачу Неймана: розв'язати рівняння теплопровідності для стрижня довжиною 1 м, час спостереження 0.2 с, при $\sigma = 1$, $c = 1$ з граничними умовами

$$u(x,0) = \sin \pi x; \quad u(0,t) = 0; \quad u_x(L,t) = c(10t - u(L,t)).$$

Запрограмуємо цю задачу у **Maxima**. Приймемо довжину стрижня $L = 1$, проміжок часу $T = 0.2$, $\sigma = 1$ та $c = 1$. Кількість проміжків оберемо $m = 30$, $n = 600$.

```
(%i4)  L:1$ T:0.2$ a:1$ c:1$
(%i6)  m:30$ n:600$
(%i8)  h:L/m$ s:T/n$
```

Перевірка умови стійкості:

```
(%i9)  a*s/h^2,numer;
(%o9)    0.3
```

Записуємо граничні умови.

```
(%i10) A(x):=float(sin(%pi*x))$
(%i12) B(t):=0$ g(t):=10*t$
```

Створюємо масиви `X1` та `T1` і заповнюємо його вузловими точками x-координат, t-координат.

```
(%i13) for i:1 thru m do
       (arraymake(X1, [i]), X1[0]:0,
       X1[i]:X1[i-1]+h);
(%o13)   done

(%i14) for k:1 thru n do
       (arraymake(T1, [k]), T1[0]:0,
       T1[k]:T1[k-1]+s);
(%o14)   done

(%i15) for i:0 thru m do
       for k:0 thru n do
       (arraymake(u,[i,k]));
(%o15)   done
```

Вносимо у масиви граничні умови.

```
(%i16) for i:0 thru m do
       (u[i,1]:A(X1[i]),u[i,0]:A(X1[i]));
(%o16)   done

(%i17) for k:0 thru n do
       u[0,k]:B(T1[k]);
(%o17)   done
```

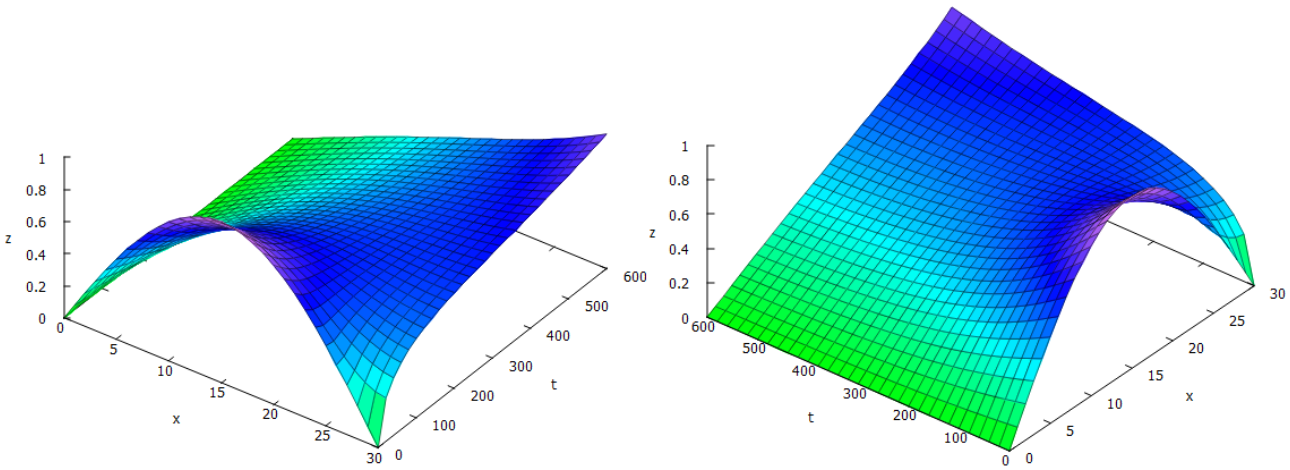


Рис. 14.5: Числовий розрахунок задачі Неймана, поверхня подана з різних ракурсів.

Записуємо систему базових рівнянь.

```
(%i18) for k:1 thru n do
       for i:1 thru m-1 do
       (u[i,k+1]:(a^2*s/h^2)*(u[i+1,k]-2*u[i,k]+u[i-1,k])+u[i,k],
        u[m,k+1]:(u[m-1,k]+c*h*g(T1[k]))/(c*h+1)
        );
(%o18)   done
```

Перетворюємо отриманий числовий масив у функцію округленням змінних.

```
(%i19) G(x,t):=float(u[round(x),round(t)])$
```

Створюємо графік обчисленого розподілу температур (Рис.14.5):

```
(%i20) plot3d(G(x,t),[x,0,m],[t,0,n])$
```

Із графіка видно наступну обставину: лівий кінець стрижня завжди має фіксовану температуру, як і встановлено початковими умовами; правий кінець хоч на початку і мав нульову температуру, однак постійний потік тепла з часом підвищив її до максимуму. У кінці спостереження маємо правдоподібний рівноважний розподіл температури від лівого кінця до правого у вигляді прямої. Загалом можна стверджувати, що при теплообміні якогось з кінців при достатньо довгому часі спостережень початковий розподіл температури не відіграє ніякої ролі, і при $t \to \infty$ функція розподілу стає рівною $u(x,t) \approx \dfrac{g(t)-T_1}{L}x$.

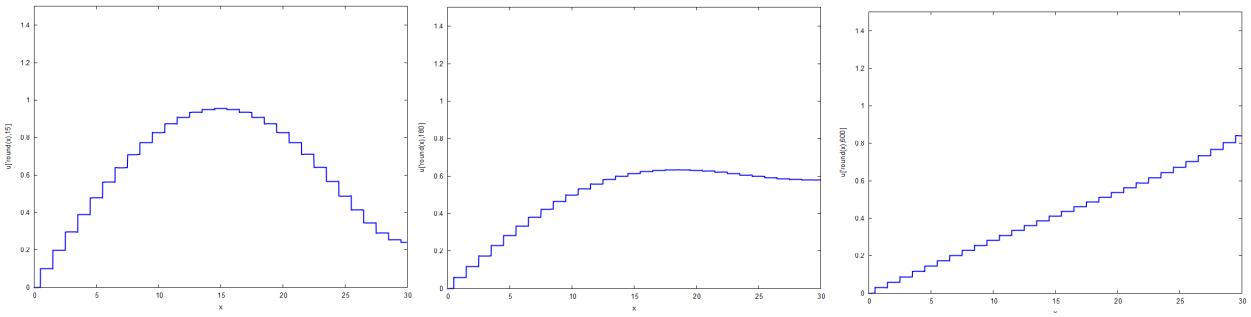


Рис. 14.6: Кадри анімації залежності обрахованої температури стрижня від часу.

Сформуємо тепер анімацію еволюції розв'язку в часі (Рис.14.6).

```
(%i21) with_slider(k,makelist(15*i,i,1,n/15),G(x,k),
       [x,0,m],[y,0,0.25],[style,[lines,2]]);
```

Створюємо також температурну діаграму стрижня.

```
(%i22) U:zeromatrix(m,n)$
(%i23) for i:1 thru m do
       for k:1 thru n do
       U[i,k]:u[i,k];
(%o23)  done

(%i24) load(draw)$
(%i25) wxdraw2d(image(U,0,0,0.2,1),
       font  = "Ubuntu Medium",
       font_size = 18,
       xlabel = "час, с",
       ylabel = "довжина, м");
(%o25)  [gr2d(image)]
```

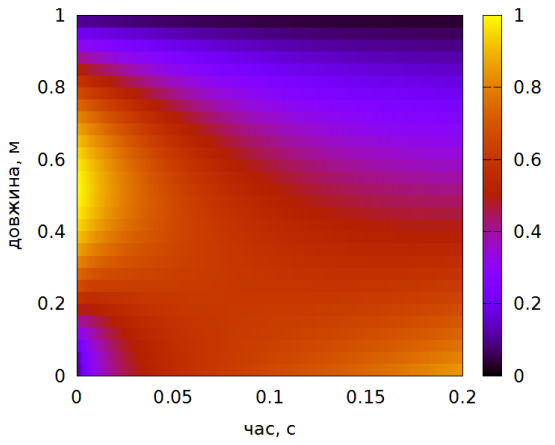


Рис. 14.7: Температурна діаграма стрижня.

14.2.3 Двовимірне рівняння теплопровідності

Якщо розглядати поширення тепла вздовж двовимірної прямокутної плити, то рівняння теплопровідності буде мати наступний вигляд:

$$u_t(x,y,t) = \sigma(u_{xx}(x,y,t) + u_{yy}(x,y,t)).$$

Функція u буде залежати вже від трьох змінних. В даному рівнянні сумарна кількість похідних рівна 5, тому необхідно задати 5 граничних умов для його однозначного вирішення. Одна з умов буде початковою — задає розподіл температури на плиті в нульовий момент часу, та чотири крайові — задають значення температури на чотирьох межах плити. Нехай плита має лінійні розміри $L_1 \times L_2$. Тоді задача Діріхле розв'язання двовимірного рівняння теплопровідності формулюється так:

$$\begin{cases} u(x,y,0) = A(x,y); \quad u(0,y,t) = B_1(y,t); \quad u(L_1,y,t) = B_2(y,t); \\ u(x,0,t) = C_1(x,t); \quad u(x,L_2,t) = C_2(x,t). \end{cases}$$

Розіб'ємо осі x, y, t на плиті паралельними лініями з кроками h_1, h_2 та s відповідно. Дискретизована система матиме вигляд

$$\frac{1}{s}(u_{i,j,k+1} - u_{i,j,k}) = \frac{\sigma}{h_1^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}) + \frac{\sigma}{h_2^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}).$$

Відповідні граничні умови:

$$u_{i,j,0} = A_{i,j}; \quad u_{0,j,k} = B_{1;j,k}; \quad u_{m_1,j,k} = B_{2;j,k}; \quad u_{i,0,k} = C_{1;i,k}; \quad u_{i,m_2,k} = C_{2;i,k}.$$

Умова стійкості даної системи: $\sigma s \left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right) < \frac{1}{2}$. Як бачимо, щоб задовільнити цим умовам, необхідно або збільшувати розміри плити, або зменшувати час розгляду задачі.

► Вирішимо наступну задачу Діріхле: розв'язати рівняння теплопровідності для плити зі сторонами 1 м×1 м, час спостереження 0.1 с, при $\sigma = 1$, з початковою температурою плити 20 °C, бічними температурами 10 °C, 10 °C по стороні x та 60 °C, 100 °C по стороні y, тобто граничними умовами

$$u(x,y,0) = 20; \quad u(0,y,t) = 10; \quad u(L_1,y,t) = 10; \quad u(x,0,t) = 60; \quad u(x,L_2,t) = 100.$$

Запрограмуємо задачу на мові **Maxima**. Приймемо довжини сторін плити $L_1 = 1$, $L_2 = 1$, проміжок часу $T = 0.1$ та $\sigma = 1$. Кількість проміжків оберемо $m_1 = 30$, $m_2 = 30$, $n = 500$.

```
(%i4)  L1:1$ L2:1$ S:0.1$ a:1$
(%i7)  m1:30$ m2:30$ n:500$
(%i10) h1:L1/m1$ h2:L2/m2$ s:S/n$
```

Перевіряємо умову стійкості.

```
(%i11) a*s/(h1^2)+a*s/(h2^2),numer;
(%o11)  0.36
```

Вносимо граничні умови задачі.

```
(%i15) B1:10$ B2:10$ C1:60$ C2:100$
```

Створюємо три необхідні масиви `X1`, `Y1`, `T1` та заповнюємо їх вузловими точками.

```
(%i16) for i:1 thru m1 do
       (arraymake(X1, [i]), X1[0]:0,
       X1[i]:X1[i-1]+h1);
(%o16)  done
```

```
(%i17) for j:1 thru m2 do
       (arraymake(Y1, [j]), Y1[0]:0,
       Y1[j]:Y1[j-1]+h2);
(%o17)  done

(%i18) for k:1 thru n do
       (arraymake(T1, [k]), T1[0]:0,
       T1[k]:T1[k-1]+s);
(%o18)  done
```

Записуємо граничні умови: початкову температуру плити та температуру на її межах.

```
(%i19) for i:0 thru m1 do
       for j:0 thru m2 do
       u[i,j,1]:20;
(%o19)  done

(%i20) for i:0 thru m1 do
       for j:0 thru m2 do
       for k:0 thru n do
       (u[0,j,k]:B1,
        u[m1,j,k]:B2,
        u[i,0,k]:C1,
        u[i,m2,k]:C2);
(%o20)  done
```

Записуємо цикл для базового рівняння.

```
(%i21) for k:1 thru n-1 do
       for i:1 thru m1-1 do
       for j:1 thru m2-1 do
       u[i,j,k+1]:((a^2*h2^2*u[i+1,j,k]+a^2*h1^2*u[i,j+1,k]+
                   (-(2*a^2*h2^2)-2*a^2*h1^2)*u[i,j,k]+
                    a^2*h1^2*u[i,j-1,k]+a^2*h2^2*u[i-1,j,k])*s+
                    h1^2*h2^2*u[i,j,k])/(h1^2*h2^2);
(%o21)  done
```

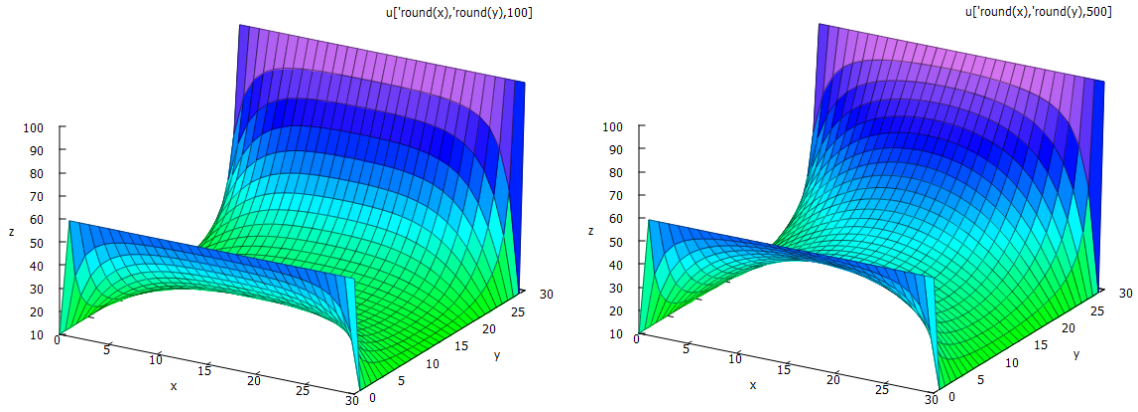


Рис. 14.8: Знімки обчисленої функції в різні моменти часу.

Обчислений масив `u` перетворюємо у функцію трьох змінних.

```
(%i22) G(x,y,t):=float(u[round(x),round(y),round(t)]);
```

Зобразимо кілька знімків розподілу температури для моментів часу $k = 100$ та $k = 500$ (Рис.14.8).

```
(%i23) plot3d(G(x,y,100),[x,0,m1],[y,0,m2]);
(%i24) plot3d(G(x,y,500),[x,0,m1],[y,0,m2]);
```

Для наочності еволюції розподілу температури на плиті скористаємось анімацією температурної діаграми в часі. Для цього створимо n матриць розмірності $m_1 \times m_2$ та заповнимо їх відповідними елементами масиву `u`, які пов'язані з просторовими індексами i, j.

```
(%i25) U:makelist(zeromatrix(m1,m2),k,1,n)$
```

Отримали список нульових матриць `U`, який містить 500 членів. Тепер заповнюємо їх елементами масиву `u`.

```
(%i26) for i:1 thru m1 do
       for j:1 thru m2 do
       for k:1 thru n do
       U[k][i,j]:u[i,j,k];
(%o26)  done
```

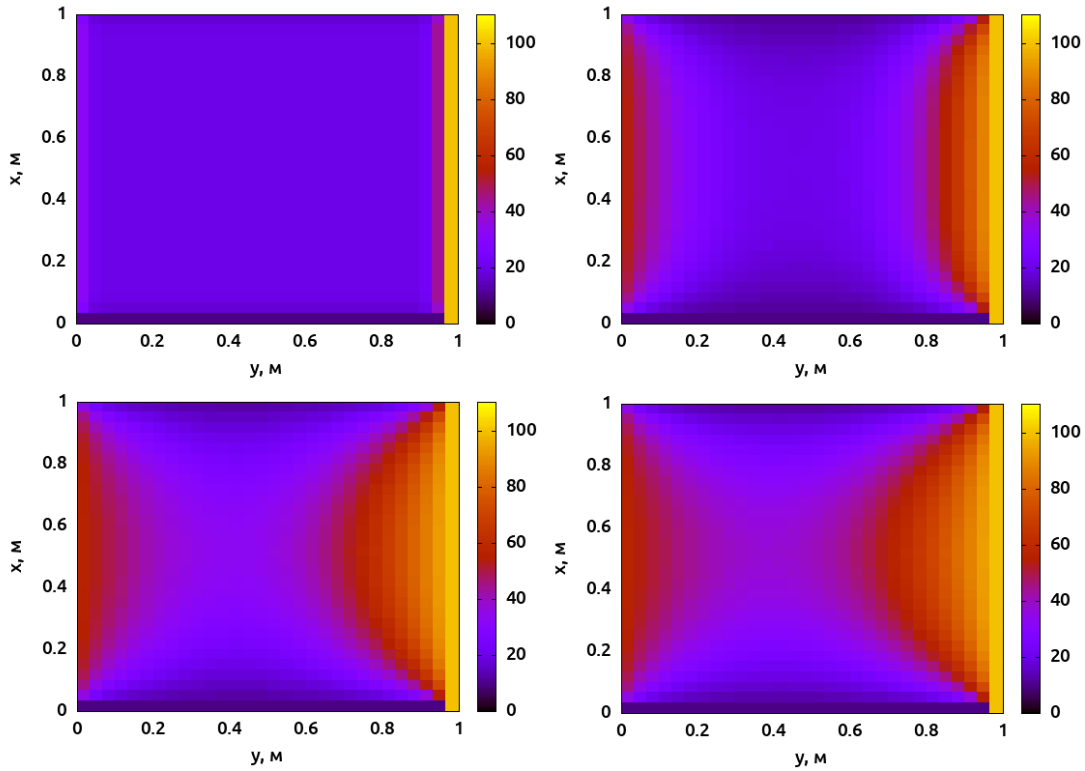


Рис. 14.9: Кадри еволюції розподілу температури на плиті з часом.

На кожну з цих матриць запроваджуємо кольоровий розподіл командою `image`. При цьому ми міняємо місцями назви осей x та y, оскільки це зробити легше ніж транспонувати кожну з 500 матриць.

```
(%i27) U_list:
         makelist(
             gr2d(
               font = "Ubuntu Medium",
               font_size = 14,
               xlabel = "y, м",
               ylabel = "x, м",
               cbrange = [0, 110],
               image((U[k]),0,0,1,1)
                 ),
         k,1,n )$
```

Нарешті створюємо .gif файл (він буде досить великий), який буде міститись у директорії, де збережена сесія **Maxima**, або у директорії **wxMaxima**, якщо цього не було зроблено (Рис.14.9).

```
(%i28) draw(
         file_name  = "Heat2D_1",
         dimensions = [600, 400],
         terminal   = animated_gif,
         U_list  );
```

Скористаємось ще можливістю анімації безпосереднього розподілу температури засобами команди `elevation_grid`. Вона будує поверхневу сітку з матриці U, використовуючи як координати набір точок (i, j, U_{ij}). Створимо поверхневу сітку для кожної з n матриць U_k та сформуємо з них один .gif-файл (Рис.14.10).

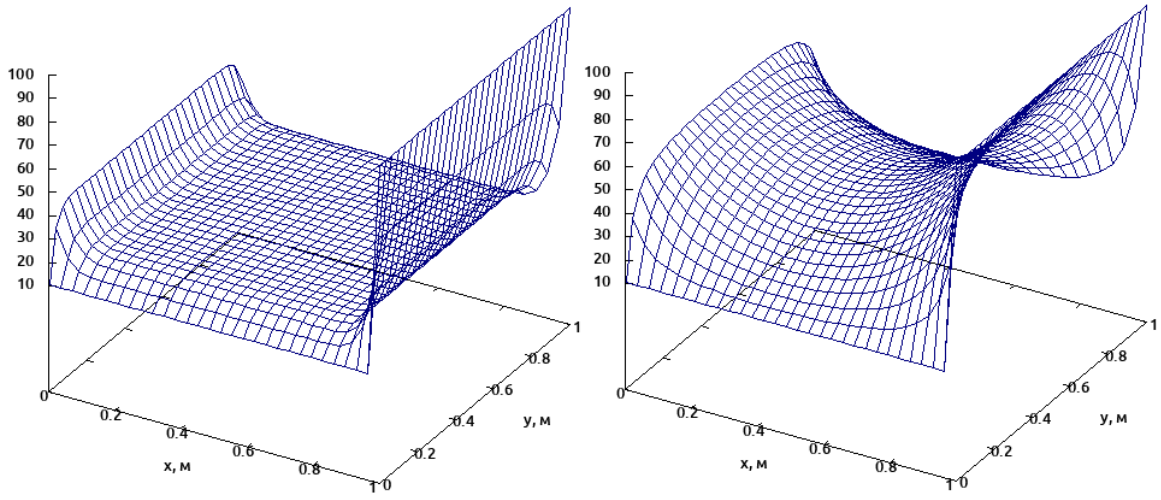


Рис. 14.10: Кадри еволюції поверхні температури у часі.

```
(%i29) U_grid_list:
         makelist(
             gr3d(
               font = "Ubuntu Medium", font_size = 10,
               xlabel = "x, м", ylabel = "y, м",
               color = navy,
               elevation_grid((U[k]),0,0,1,1)),
           k,1,n)$
```

```
(%i30) draw(
         file_name  = "Heat2D_grid_1",
         dimensions = [600, 600],
         terminal   = animated_gif,
         U_grid_list  )$
```

Наостанок зробимо наступне зауваження. При знаходженні розв'язку рівняння теплопровідності ми користувались прямою схемою Ейлера, в якому вираз часової еволюції повністю залежав від стану системи в попередній момент часу. Недоліком такої схеми є набігання помилок при кожному наступному кроці, яке за достатньо довгий час інтегрування починає зрівнюватись за порядком із самим розв'язком; та обов'язкове виконання умови стійкості, яка накладає обмеження на густину сіток розбиття. Ми не можемо як завгодно дрібно ділити просторові проміжки, оскільки кількість часових вузлових точок при цьому збільшується квадратично. Тому прямим методом можна знайти надійний розв'язок лише для невеликого часу спостереження.

Таких недоліків позбавлені непрямі методи розв'язання, наприклад Кранка-Ніколсона. Однак при цьому виникає необхідність мати справу з матрицями із $m \times n$ елементів, що робить їхню схему дуже ресурсозатратною при достатніх кількостях вузлових точок. Побудова схеми Кранка-Ніколсона пов'язує між собою три часових кроки, створюючи систему багатьох рівнянь, яку потрібно розв'язувати відповідними методами.

14.3 Рівняння гіперболічного типу

Такі рівняння описують хвильові та коливальні процеси. Загальний вигляд їх наступний:

$$\frac{\partial^2 u(\vec{r},t)}{\partial t^2} = \frac{1}{v_p^2}\nabla^2 u(\vec{r},t).$$

Тут $u(\vec{r},t)$ — поперечне зміщення відносно напрямку фазової швидкості хвилі, величина v_p — фазова швидкість хвилі. Надалі позначатимемо її буквою v.

14.3.1 Коливання одновимірної струни з закріпленими кінцями

Нехай маємо хвильовий процес у пружному середовищі довжини L (яке назвемо струною) вздовж одного напрямку. Прийнявши цей напрям за вісь x, можемо записати

$$u_{tt}(x,t) = \frac{1}{v^2}u_{xx}(x,t).$$

Всього в рівнянні присутні 4 похідні, тому необхідно задати 4 граничні умови: дві для часової координати та дві для просторової. Зазвичай розглядають два випадки: струна з закріпленими кінцями, та струна з одним вільним кінцем.

У першому випадку задаються умови фіксованості на краях відрізка (крайові умови першого роду), та умови розподілу зміщень в комбінації з розподілом швидкостей в початковий момент часу (крайові умови третього роду). Отже, задача розв'язання хвильового рівняння з умовами

$$\begin{cases} u(0,t) = 0; \quad u(L,t) = 0; \\ u(x,0) = A(x); \quad u_t(x,0) = B(x) \end{cases}$$

називається *задачею струни з закріпленими кінцями.*

Розбивши вісь x на m інтервалів з кроком h, та вісь t на n інтевалів з кроком s, переходимо до дискретизованого рівняння.

$$\frac{1}{s^2}(u_{i,k+1} - 2u_{i,k} + u_{i,k-1}) = \frac{1}{v^2h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}).$$

Виділимо залежність функції в наступний момент часу $k+1$ через попередні.

$$u_{i,k+1} = \frac{s^2}{v^2h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}) + 2u_{i,k} - u_{i,k-1}.$$

Можемо вже відмітити відмінність цього виразу від розглянутого в рівнянні теплопровідності: раніше момент часу $k+1$ залежав лише від попереднього моменту k, тоді як зараз від двох попередніх, k та $k-1$.

Граничні умови мають вигляд

$$\begin{cases} u_{0,k} = 0; \quad u_{m,k} = 0; \\ u_{i,0} = A_i; \quad (1/s)(u_{i,1} - u_{i,0}) = B_i. \end{cases}$$

Умова стійкості розв'язку: $\dfrac{s^2}{v^2h^2} < \dfrac{1}{2}$.

► Вирішимо наступну задачу: розв'язати хвильове рівняння для струни з закріпленими кінцями довжини $L = 2$ м, час спостереження $T = 2$ с, при $v = 1$, з початковими умовами

$$u(0,t) = 0; \quad u(L,t) = 0; \quad u(x,0) = \sin(\pi x); \quad u_t(x,0) = 0.$$

Переведемо цю задачу на мову **Maxima**. Задамо початкові умови та приймемо кількість відрізків розбиття $m = 100$, $n = 800$. Для зручності будемо проводити обчислення починаючи з першого вузла, а не з нульового.

```
(%i3)  L:2$ T:2$ v:1$
(%i5)  m:100$ n:800$
(%i7)  h:L/m$ s:T/n$
```

Перевіряємо умову стійкості.

```
(%i8)  s^2/(v^2*h^2);
```

$$(\%o8) \quad \frac{1}{64}$$

Створюємо граничні умови.

```
(%i9)  A(x):=float(sin(%pi*x));
```

$$(\%o9) \quad \mathrm{A}\,(x) := \mathrm{float}\,(\sin\,(\pi \cdot x))$$

```
(%i10) B(x):=0;
```

$$(\%o10) \quad \mathrm{B}\,(x) := 0$$

Створюємо необхідні масиви та заповнюємо їх вузловими точками.

```
(%i11) for i:1 thru m do
       for k:1 thru n do
       (arraymake(u,[i,k]));
```

$$(\%o11) \quad \textit{done}$$

```
(%i12) for i:1 thru m do
       (arraymake(X1, [i]), X1[0]:0,
       X1[i]:X1[i-1]+h);
(%o12)  done

(%i13) for k:1 thru n do
       (arraymake(T1, [k]), T1[0]:0,
       T1[k]:T1[k-1]+s);
(%o13)  done
```

Вносимо граничні умови у масив `u`.

```
(%i14) for i:1 thru m do
       (u[i,1]:A(X1[i]), u[i,2]:s*B(X1[i])+u[i,1]);
(%o14)  done

(%i15) for k:1 thru n do
       (u[1,k]:0, u[m,k]:0);
(%o15)  done
```

Записуємо базове рівняння.

```
(%i16) for k:2 thru n-1 do
       for i:2 thru m-1 do
       u[i,k+1]:float((s/(v*h))^2*(u[i+1,k]-2*u[i,k]+
                      u[i-1,k])+2*u[i,k]-u[i,k-1]);
(%o16)  done
```

Отримали масив значень координат зміщення. Перетворимо масив у функцію округленням чисел та зобразимо графік (Рис.14.11).

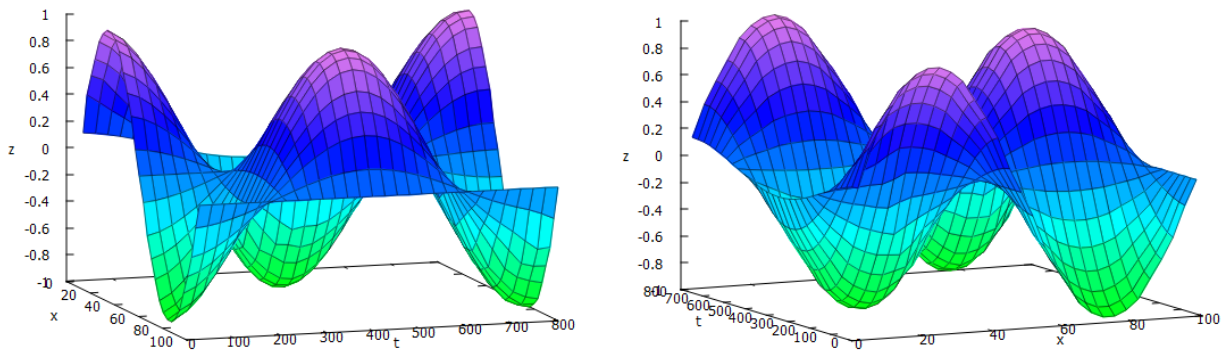


Рис. 14.11: Поверхня числового розрахунку хвильової задачі, подана з різних ракурсів.

```
(%i17) G(x,t):=float(u[round(x),round(t)])$
(%i18) plot3d(G(x,t),[x,0,m],[t,0,n]);
```

Створимо тепер анімацію еволюції розв'язку в часі. Із 800 часових вузлових точок ми залишаємо 40, обравши кожен 20-ий вузол (Рис.14.12). Як бачимо, маємо сталі коливання стоячої хвилі.

```
(%i19) with_slider(k,makelist(20*i,i,1,n/20),G(x,k),
       [x,0,m],[y,-1.1,1.1],[style,[lines,2]]);
```

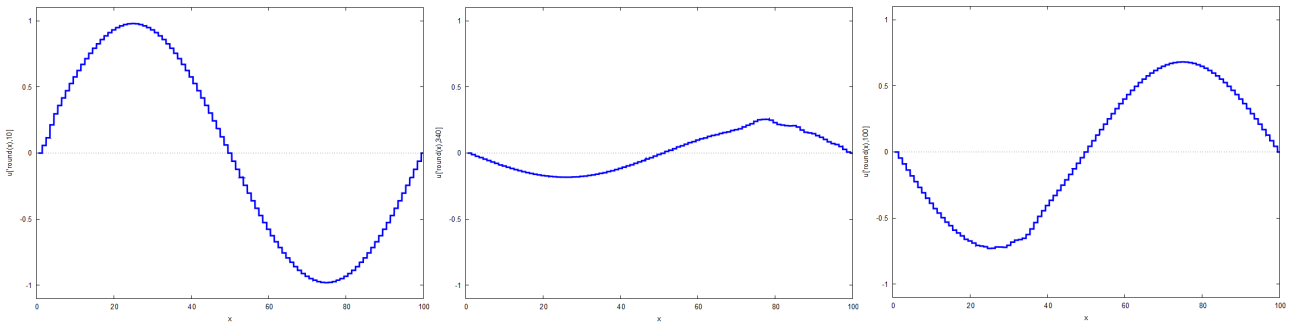


Рис. 14.12: Кадри анімації відхилення струни від положення рівноваги у часі.

14.3.2 Коливання одновимірної струни з вільним кінцем

Розглянемо другий випадок коливань струни. Нехай лівий край буде фіксований, а правий вільний, і на нього діє певна сила, що зміщує в поперечному напрямку (крайові умови третього роду). Додаються ще умови розподілу зміщень в комбінації з розподілом швидкостей в початковий момент часу (крайові умови третього роду). Отже, задача розв'язання хвильового рівняння з умовами

$$\begin{cases} u(0,t) = 0; \quad u_x(L,t) = C(t); \\ u(x,0) = A(x); \quad u_t(x,0) = B(x) \end{cases}$$

називається *задачею струни з вільним кінцем.*

Провівши процедуру дискретизації, маємо вираз

$$u_{i,k+1} = \frac{s^2}{v^2h^2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}) + 2u_{i,k} - u_{i,k-1}$$

із граничними умовами

$$\begin{cases} u_{0,k} = 0; \quad (1/h)(u_{m,k} - u_{m-1,k}) = C_k; \\ u_{i,0} = A_i; \quad (1/s)(u_{i,1} - u_{i,0}) = B_i. \end{cases}$$

► Вирішимо наступну задачу: розв'язати хвильове рівняння для струни з закріпленим лівим кінцем, правий кінець вільний і на нього діє поперечна сила $F = e^{-t}$, довжина струни $L = 2$ м, час спостереження $T = 6$ с, при $v = 1$, з початковими умовами

$$u(0,t) = 0; \quad u_x(L,t) = e^{-t}; \quad u(x,0) = \sin(\pi x); \quad u_t(x,0) = 0.$$

Запрограмуємо задачу у **Maxima**. Оскільки час спостереження збільшився порівняно з попереднім випадком, збільшимо також і n. Задамо початкові умови та приймемо кількість відрізків розбиття $m = 100$, $n = 1000$. Для зручності будемо проводити обчислення починаючи з першого вузла, а не з нульового.

Задаємо початкові дані та перевіряємо умову стійкості.

```
(%i3)  L:2$ T:6$ v:1$
(%i5)  m:100$ n:1000$
(%i7)  h:L/m$ s:T/n$
(%i8)  v^2*s^2/(h^2);
(%o8)  9/100
```

Записуємо функції граничних умов.

```
(%i9)  A(x):=float(sin(%pi*x));
(%i10) B(x):=0;
(%i11) C(t):=float(exp(-t));
```

Створюємо три масиви `u`, `X1`, `T1`.

```
(%i12) for i:1 thru m do
       for k:1 thru n do
       (arraymake(u,[i,k]));
(%o12)  done

(%i13) for i:1 thru m do
       (arraymake(X1, [i]), X1[0]:0,
       X1[i]:X1[i-1]+h);
(%o13)  done

(%i14) for k:1 thru n do
       (arraymake(T1, [k]), T1[0]:0,
       T1[k]:T1[k-1]+s);
(%o14)  done
```

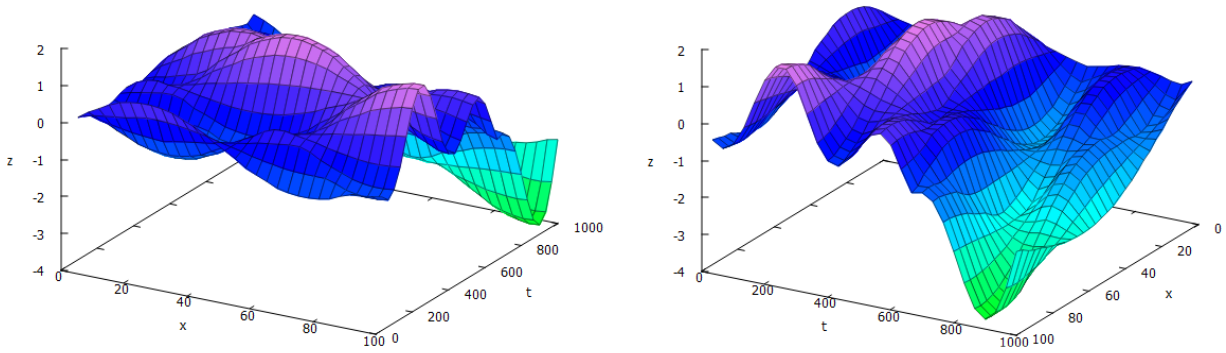


Рис. 14.13: Поверхня задачі з закріпленим лівим кінцем, подана з різних ракурсів.

Заповнюємо масиви відповідними граничними умовами. Умова для силової дії на правий кінець дописується до базового рівняння.

```
(%i15) for i:1 thru m do
       (u[i,1]:A(X1[i]), u[i,2]:s*B(X1[i])+u[i,1]);
(%o15)  done

(%i16) for k:1 thru n do
       (u[1,k]:0);
(%o16)  done

(%i17) for k:2 thru n do
       for i:2 thru m-1 do
       (u[i,k+1]:(s/(v*h))^2*(u[i+1,k]-2*u[i,k]+u[i-1,k])+
           2*u[i,k]-u[i,k-1],
        u[m,k]:h*C(T1[k])+u[m-1,k]);
```

```
(%o17)  done
```

Отримані значення переводимо у функцію та будуємо графік (Рис.14.13).

```
(%i18) G(x,t):=float(u[round(x),round(t)])$
(%i19) plot3d(G(x,t),[x,0,m],[t,0,n]);
```

Із графіка видно, що правий кінець струни відразу зміщується вгору, це є проявом дії заданої сили. Але оскільки функція e^{-t} швидко йде до нуля, із часом вона фактично перестає діяти та правий кінець починає коливатись як повністю вільний.

Сформуємо тепер анімацію розв'язку в часі, обравши кожне 25-те значення часових вузлових точок (Рис.14.14).

```
(%i20) with_slider(k,makelist(25*i,i,1,n/25),G(x,k),
       [x,0,m],[y,-4,4],[style,[lines,2]]);
```

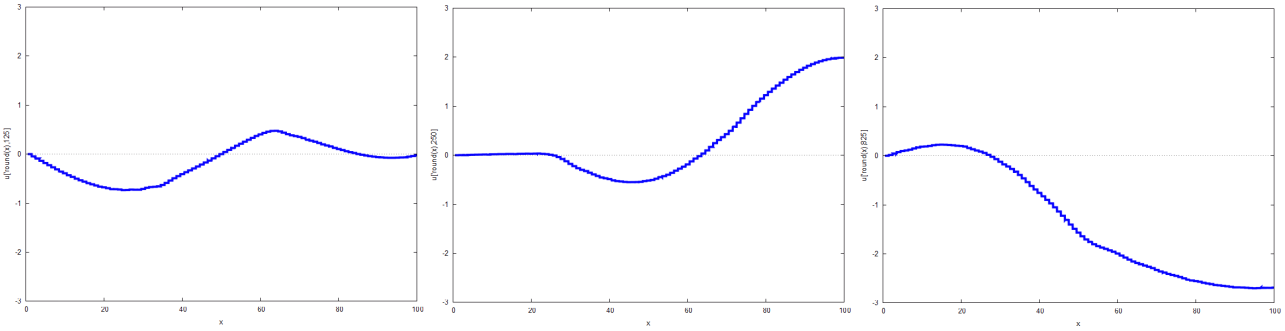


Рис. 14.14: Кадри анімації коливань струни із закріпленим лівим кінцем.

14.3.3 Коливання двовимірної прямокутної мембрани

Розглянемо хвильове рівняння у двох вимірах, на прямокутній мембрані розмірами $L_1 \times L_2$. Частинки мембрани можуть вільно зміщуватись у поперечному напрямку вздовж осі z, величиною цього зміщення буде функція u:

$$u_{tt}(x,y,t) = \frac{1}{v^2}(u_{xx}(x,y,t) + u_{yy}(x,y,t)).$$

Сумарна кількість похідних рівна 6, тому необхідно задати 6 граничних умов. Зазвичай задаються розподіл координат та швидкостей точок мембрани у початковий момент часу та умови на її межах. Задача розв'язання хвильового рівняння з граничними умовами

$$\begin{cases} u(x,y,0) = A_1(x,y); \quad u_t(x,y,0) = A_2(x,y); \\ u(0,y,t) = B_1(y,t); \quad u(L_1,y,t) = B_2(y,t); \\ u(x,0,t) = C_1(x,t); \quad u(x,L_2,t) = C_2(x,t) \end{cases}$$

називається *задачею коливань двовимірної мембрани.*

Перейдемо до дискретизованої системи, розбивши осі x, y, t на мембрані паралельними лініями з кроками h_1, h_2 та s відповідно.

$$\frac{1}{s^2}(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}) = \frac{1}{v^2 h_1^2}(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}) + \frac{1}{v^2 h_2^2}(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}).$$

Граничні умови:

$$u_{i,j,0} = A_{1;i,j}; \quad (1/s)(u_{i,1} - u_{i,0}) = A_{2;i,j};$$

$$u_{0,j,k} = B_{1;j,k}; \quad u_{m_1,j,k} = B_{2;j,k};$$
$$u_{i,0,k} = C_{1;i,k}; \quad u_{i,m_2,k} = C_{2;i,k}.$$

Умова стійкості розв'язку: $\dfrac{s^2}{v^2(h_1^2+h_2^2)} < \dfrac{1}{4}$.

► Вирішимо наступну задачу: розв'язати хвильове рівняння для закріпленої мембрани зі сторонами 2 м×2 м, час спостереження 2 с, при $v = 1$, з початковими умовами

$$u(x,y,0) = 2\sin(\pi x)\sin(\pi y/2); \quad u_t(x,y,0) = 0;$$

$$u(0,y,t) = 0; \quad u(L_1,y,t) = 0; \quad u(x,0,t) = 0; \quad u(x,L_2,t) = 0.$$

Запрограмуємо **Maxima** для розв'язання задачі. Вносимо початкові дані, обравши кількість проміжків розбиття $m_1 = 100$, $m_2 = 100$, $n = 800$. Для зручності переведення у матриці всі масиви будемо нумерувати індексами, починаючи від 1.

```
(%i4)  L1:2$ L2:2$ T:2$ v:1$
(%i7)  m1:100$ m2:100$ n:800$
(%i10) h1:L1/m1$ h2:L2/m2$ s:T/n$
```

Перевіряємо умову стійкості.

```
(%i11) s^2/(v^2*(h1^2+h2^2)),numer;
(%o11)  0.013888888888888892
```

Записуємо граничні умови.

```
(%i12) A1(x,y):=float(2*sin(%pi*x)*sin(%pi*y/2));
```
$$(\%o12) \quad \mathrm{A1}\,(x,y) := \mathrm{float}\left(2\cdot\sin\left(\pi\cdot x\right)\cdot\sin\left(\frac{\pi\cdot y}{2}\right)\right)$$

```
(%i13) A2(x,y):=0;
```
$$(\%o13) \quad \mathrm{A2}\,(x,y) := 0$$

```
(%i17) B1:0$ B2:0$ C1:0$ C2:0$
```

Створюємо три масиви для заповнення вузловими точками `X1`, `Y1`, `T1`.

```
(%i18) for i:1 thru m1 do
       (arraymake(X1, [i]), X1[1]:0,
       X1[i]:X1[i-1]+h1);
(%o18)  done

(%i19) for j:1 thru m2 do
       (arraymake(Y1, [j]), Y1[1]:0,
       Y1[j]:Y1[j-1]+h2);
(%o19)  done

(%i20) for k:1 thru n do
       (arraymake(T1, [k]), T1[1]:0,
       T1[k]:T1[k-1]+s);
(%o20)  done
```

Вносимо граничні умови у масив `u`.

```
(%i21) for i:1 thru m1 do
       for j:1 thru m2 do
       (u[i,j,1]:A1(X1[i],Y1[j]),
        u[i,j,2]:s*A2(X1[i],Y1[j])+u[i,j,1]);
(%o21)   done

(%i22) for i:1 thru m1 do
       for j:1 thru m2 do
       for k:1 thru n do
       (u[1,j,k]:B1,
        u[m1,j,k]:B2,
        u[i,1,k]:C1,
        u[i,m2,k]:C2);
(%o22)   done
```

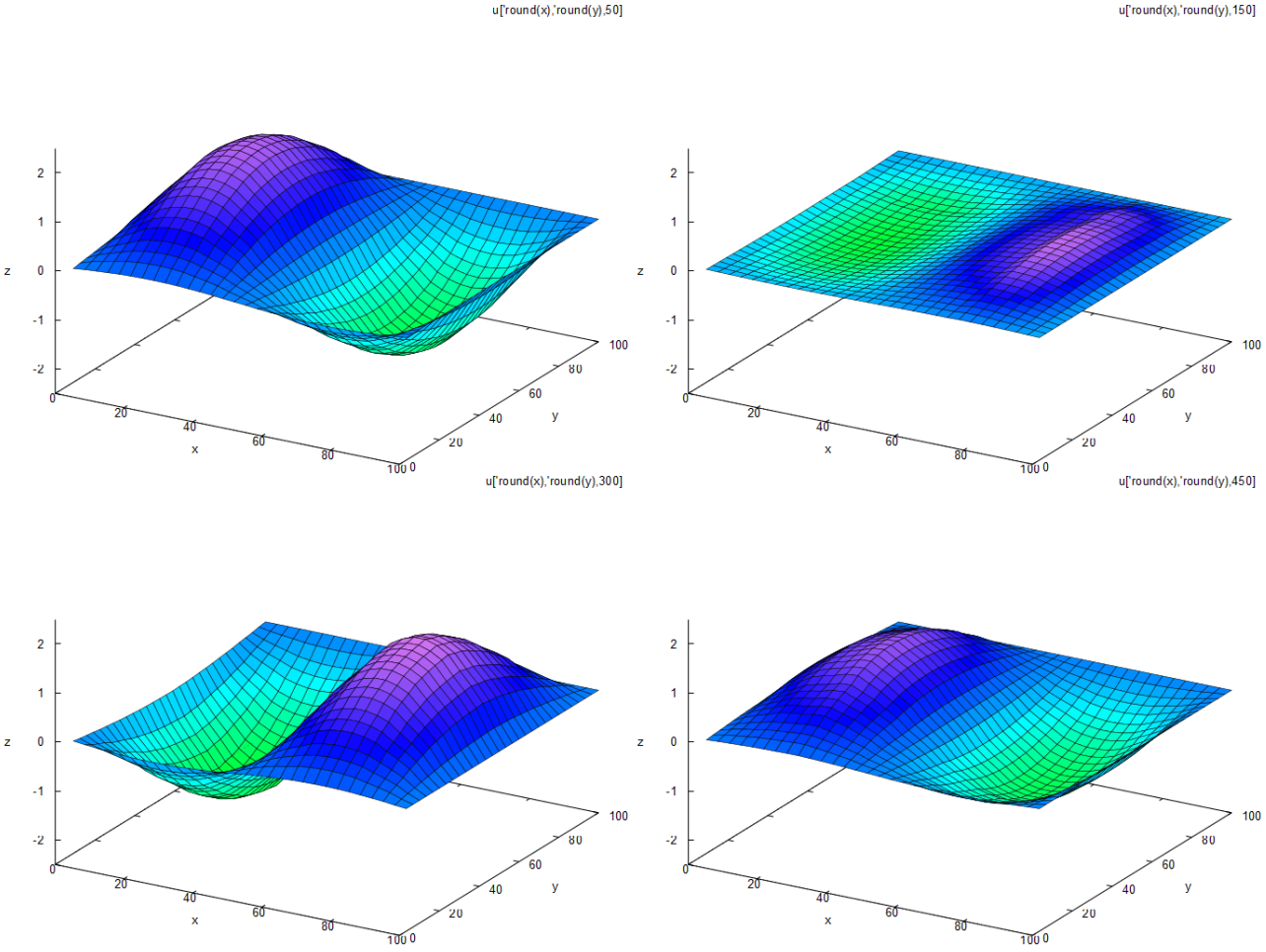


Рис. 14.15: Часові зрізи поверхні розв'язку двовимірного хвильового рівняння.

Записуємо базове рівняння для обчислення елементів $u_{i,j,k+1}$.

```
(%i23) for k:2 thru n do
       for i:2 thru m1-1 do
       for j:2 thru m2-1 do
       u[i,j,k+1]:(s/(v*h1))^2*(u[i+1,j,k]-2*u[i,j,k]+u[i-1,j,k])+
                  (s/(v*h2))^2*(u[i,j+1,k]-2*u[i,j,k]+u[i,j-1,k])+
                   2*u[i,j,k]-u[i,j,k-1];
(%o23)  done
```

Отриманий масив переводимо у функцію округленням чисел до цілого та будуємо кілька поверхонь часових зрізів (Рис.14.15).

```
(%i24) G(x,y,t):=float(u[round(x),round(y),round(t)])$
(%i25) plot3d(G(x,y,50),[x,0,m1],[y,0,m2],[z,-2.5,2.5])$
(%i26) plot3d(G(x,y,150),[x,0,m1],[y,0,m2],[z,-2.5,2.5])$
```

Із побудованих графіків видно, що довгострокова часова еволюція призводить до того, що хвильова поверхня на останніх зрізах часу стає трохи деформованою. Це наслідок накопичення помилок при обчисленнях, така поведінка властива для прямих однокрокових методів.

Сформуємо тепер анімацію коливань у просторі, скориставшись командою побудови матричної сітки. Подібно до того, як це було у випадку двовимірного рівняння теплопровідності, створимо n матриць, кожна з яких містить відповідне зміщення на просторових координатах (i, j).

```
(%i27) load(draw)$
(%i28) U:makelist(zeromatrix(m1,m2),k,1,n)$
(%i29) for i:1 thru m1 do
       for j:1 thru m2 do
       for k:1 thru n do
       U[k][i,j]:u[i,j,k];
(%o29)  done
```

Оберемо кожен 10-ий кадр для анімації.

```
(%i30) U_grid_list:
          makelist(
         gr3d(
            font = "Ubuntu Medium", font_size = 10,
            xlabel = "x, м", ylabel = "y, м",
            color = royalblue,
            elevation_grid((U[10*k]),0,0,2,2)),
          k,1,n/10
       )$
(%i31) draw(
       file_name = "Wave2D_grid_1",
       dimensions = [600, 600],
       terminal = animated_gif,
       U_grid_list )$
```

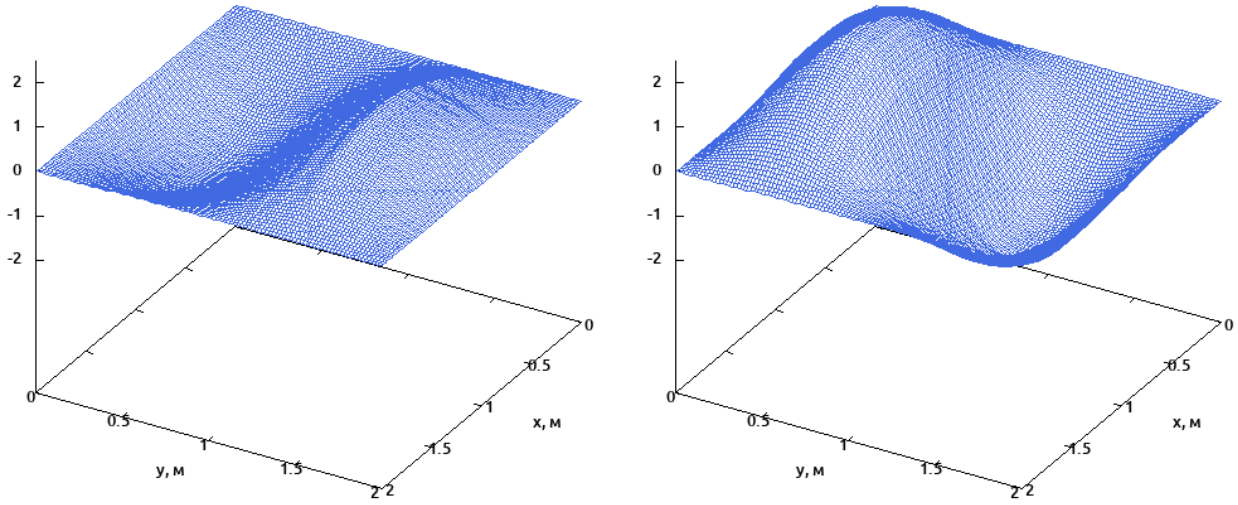


Рис. 14.16: Кадри анімації коливань двовимірної мембрани.

Як і у випадку рівняння теплопровідності, для розв'язання хвильового рівняння можна використовувати непрямі багатокрокові методи, що пов'язують три часові кроки при переході до дискретизації системи. При цому артефактів (деформації поверхні) пізньої еволюції не виникає, щоправда ціною такого методу є зменшення кількості вузлових точок.

14.4 Рівняння еліптичного типу

Рівняння цього типу описують усталений розподіл температури в певній області або розподіл потенціалу електричного поля за присутності зарядів або їх відсутності. Залежність від часу в таких рівняннях зазвичай не розглядається.

Широке коло таких явищ описується за допомогою рівняння Гельмгольца

$$\nabla^2 u(\vec{r}) + g(\vec{r}) \cdot u(\vec{r}) = f(\vec{r}).$$

При $g(\vec{r}) = 0$ воно перетворюється на рівняння Пуассона, а при $g(\vec{r}) = 0$ та $f(\vec{r}) = 0$ на рівняння Лапласа.

Перш ніж перейти до реалізації вирішення конкретних задач, проведемо процедуру дискретизації вихідного рівняння. Будемо розглядати його у двовимірному просторі:

$$u_{xx}(x, y) + u_{yy}(x, y) + g(x, y)u(x, y) = f(x, y).$$

Розділимо область інтегрування сіткою з кроком h_1 по осі Ox та кроком h_2 по осі Oy. Задамо відразу різницеве рівняння у **Maxima** та визначимо вираз для $u_{i,j}$.

```
(%i1) eqn1:(1/h1)^2*(u[i+1,j]-2*u[i,j]+u[i-1,j]);
```

$$(\%o1) \quad \frac{u_{i+1,j} - 2 \cdot u_{i,j} + u_{i-1,j}}{h1^2}$$

```
(%i2) eqn2:(1/h2)^2*(u[i,j+1]-2*u[i,j]+u[i,j-1]);
```

$$(\%o2) \quad \frac{u_{i,j+1} - 2 \cdot u_{i,j} + u_{i,j-1}}{h2^2}$$

Записуємо рівняння Гельмгольца.

```
(%i3)  Helm:eqn1+eqn2+g[i,j]*u[i,j]=f[i,j];
```

$$(\%o3)\quad \frac{u_{i+1,j}-2\cdot u_{i,j}+u_{i-1,j}}{h1^2}+\frac{u_{i,j+1}-2\cdot u_{i,j}+u_{i,j-1}}{h2^2}+g_{i,j}\cdot u_{i,j}=f_{i,j}$$

```
(%i4)  solve(Helm,u[i,j]);
```

$$(\%o4)\quad [u_{i,j}=-\frac{h2^2\cdot u_{i+1,j}+h1^2\cdot u_{i,j+1}-h1^2\cdot h2^2\cdot f_{i,j}+h1^2\cdot u_{i,j-1}+h2^2\cdot u_{i-1,j}}{h1^2\cdot h2^2\cdot g_{i,j}-2\cdot h2^2-2\cdot h1^2}]$$

Отримали співвідношення, за яким будемо шукати невідому функцію u. Здійснимо ще наступне спрощення: покладемо $h_1 = h_2$, $f_{i,j} = 0$, $g_{i,j} = 0$.

```
(%i5)  ratsimp(%),g[i,j]=0,f[i,j]=0,h1=h2;
```

$$(\%o5)\quad [u_{i,j}=\frac{u_{i+1,j}+u_{i,j+1}+u_{i,j-1}+u_{i-1,j}}{4}]$$

Як бачимо, маємо досить простий вираз, який пов'язує значення функції $u_{i,j}$ із її значеннями у чотирьох сусідніх вузлах на осях Ox та Oy. Він називається *стандартною п'ятиточковою формулою.*

14.4.1 Двовимірне рівняння Лапласа

Це рівняння описує сталий у часі розподіл температури без джерел тепла, або розподіл скалярного потенціалу без джерел електричного поля. У двовимірному просторі воно має вигляд

$$u_{xx}(x,y)+u_{yy}(x,y)=0.$$

У нас присутні 4 похідних, тому потрібно задати 4 крайові умови для однозначного вирішення. Нехай областю інтегрування буде прямокутник зі сторонами $L_1 \times L_2$. Зазвичай задаються значення функції на межах області інтегрування (умови Діріхле):

$$\begin{cases} u(0,y)=A_1(y); & u(L_1,y)=A_2(y);\\ u(x,0)=B_1(x); & u(x,L_2)=B_2(x). \end{cases}$$

▶ Вирішимо наступну задачу: розв'язати рівняння Лапласа на прямокутнику 4 м×4 м із крайовими умовами

$$\begin{cases} u(0,y)=e^y-\cos y; & u(4,y)=e^y\cos 4-e^4\cos y;\\ u(x,0)=\cos x-e^x; & u(x,4)=e^4\cos x-e^x\cos 4. \end{cases}$$

Вносимо початкові дані у **Maxima** та створюємо відповідні масиви з вузловими точками. Кожну сторону прямокутника розіб'ємо на 10 частин, тоді всіх вузлових точок буде $11 \times 11 = 121$.

```
(%i6)  ratprint:false$ fpprintprec:5$
(%i10) L1:4$ L2:4$ m1:10$ m2:10$
(%i12) h1:L1/m1$ h2:L2/m2$
(%i14) g(x,y):=0$ f(x,y):=0$
(%i15) for i:2 thru m1+1 do
       (arraymake(X1, [i]), X1[1]:0,
       X1[i]:X1[i-1]+h1);
(%o15)  done
```

```
(%i16) for j:2 thru m2+1 do
       (arraymake(Y1, [j]), Y1[1]:0,
       Y1[j]:Y1[j-1]+h2);
```

(%o16) *done*

Нам потрібно буде розв'язати систему з 121 рівняння, тому відразу формуємо список лівих частин цих рівнянь, який буде складатись з невідомої функції $u_{i,j}$ для всіх можливих комбінацій i та j. Список формується наступним чином: спочатку виписується $u_{1,j}$ для впорядкованого j, потім $u_{2,j}$, і т.д., поки індекс i не вичерпається. Ця послідовність впорядкування буде важливою нам в подальшому.

```
(%i17) u_list:flatten(makelist(makelist(u[i,j],i,1,m1+1),j,1,m2+1))$
```

Створюємо задані крайові функції та заповнюємо масив `u` вузловими точками на межах області.

```
(%i19) A1(y):=float(exp(y) - cos(y))$
       A2(y):=float(exp(y)*cos(4) - exp(4)*cos(y))$
(%i20) for j:1 thru m2+1 do
       (u[1,j]:A1(Y1[j]),
        u[m1+1,j]:A2(Y1[j]));
```

(%o20) *done*

```
(%i22) B1(x):=float(cos(x) - exp(x))$
       B2(x):=float(exp(4)*cos(x) - exp(x)*cos(4))$
(%i23) for i:1 thru m1+1 do
       (u[i,1]:B1(X1[i]),
        u[i,m2+1]:B2(X1[i]));
```

(%o23) *done*

Поглянемо на заповнений масив, чи все правильно задалося.

```
(%i24) for i:1 thru m1+1 do
       for j:1 thru m2+1 do
       display(u[i,j]);
```

$u_{1,1} = 0.0; u_{1,2} = 0.57076; u_{1,3} = 1.5288; \quad \ldots \quad u_{1,10} = 37.495; u_{1,11} = 55.252$
$u_{2,1} = -0.57076; u_{2,2} = u_{2,2}; u_{2,3} = u_{2,3}; \quad \ldots \quad u_{2,10} = u_{2,10}; u_{2,11} = 51.263$
$u_{3,1} = -1.5288; u_{3,2} = u_{3,2}; u_{3,3} = u_{3,3}; \quad \ldots \quad u_{3,10} = u_{3,10}; u_{3,11} = 39.494$
$u_{4,1} = -2.9578; u_{4,2} = u_{4,2}; u_{4,3} = u_{4,3}; \quad \ldots \quad u_{4,10} = u_{4,10}; u_{4,11} = 21.954$
$u_{5,1} = -4.9822; u_{5,2} = u_{5,2}; u_{5,3} = u_{5,3}; \quad \ldots \quad u_{5,10} = u_{5,10}; u_{5,11} = 1.6433$
$u_{6,1} = -7.8052; u_{6,2} = u_{6,2}; u_{6,3} = u_{6,3}; \quad \ldots \quad u_{6,10} = u_{6,10}; u_{6,11} = -17.891$
$u_{7,1} = -11.761; u_{7,2} = u_{7,2}; u_{7,3} = u_{7,3}; \quad \ldots \quad u_{7,10} = u_{7,10}; u_{7,11} = -33.055$
$u_{8,1} = -17.387; u_{8,2} = u_{8,2}; u_{8,3} = u_{8,3}; \quad \ldots \quad u_{8,10} = u_{8,10}; u_{8,11} = -40.695$
$u_{9,1} = -25.531; u_{9,2} = u_{9,2}; u_{9,3} = u_{9,3}; \quad \ldots \quad u_{9,10} = u_{9,10}; u_{9,11} = -38.47$
$u_{10,1} = -37.495; u_{10,2} = u_{10,2}; u_{10,3} = u_{10,3}; \quad \ldots \quad u_{10,10} = u_{10,10}; u_{10,11} = -25.039$
$u_{11,1} = -55.252; u_{11,2} = -51.263; u_{11,3} = -39.494; \quad \ldots \quad u_{11,10} = 25.039; u_{11,11} = 0.0$

(%o24) *done*

Як бачимо, усі дані внесені вірно: масив має задані числові значення на всіх своїх межах, тобто лівий, правий, верхній та нижній рядки заповнені. Невизначеними залишились внутрішні вузли сітки, їхня кількість буде $9 \times 9 = 81$. Отож необхідно розв'язати 81 рівняння з такою ж кількістю невідомих. Задаємо базове рівняння на $u_{i,j}$.

```
(%i25) for i:2 thru m1 do
       for j:2 thru m2 do
       u[i,j]:-((h2^2·u[i+1,j]+h1^2·u[i,j+1]-
              h1^2·h2^2·f(X1[i],Y1[j])+h1^2·u[i,j-1]+
              h2^2·u[i-1,j])/(h1^2·h2^2·g(X1[i],Y1[j])-
              2·h2^2-2·h1^2));
(%o25)  done
```

Це співвідношення пов'язало кожен із внутрішніх вузлів зі своїми сусідами за допомогою певного рівняння. Щоб скласти систему рівнянь, перетворимо масив `u` у список.

```
(%i26) U:listarray(u)$
```

Поглянувши на структуру цього списку, можна бачити, що він іде порядково: спочатку вирази для $u_{1,j}$ для всіх j, потім $u_{2,j}$ і т.д. Тобто список `U` точно відповідає списку `u_list`, який ми створили раніше. Отож створюємо систему рівнянь прирівнявши їх, та розв'язуємо її.

```
(%i27) sys2:makelist(u_list[k]=U[k],k,1,(m1+1)*(m2+1))$
(%i28) sol2:linsolve(sys2,u_list)$
```

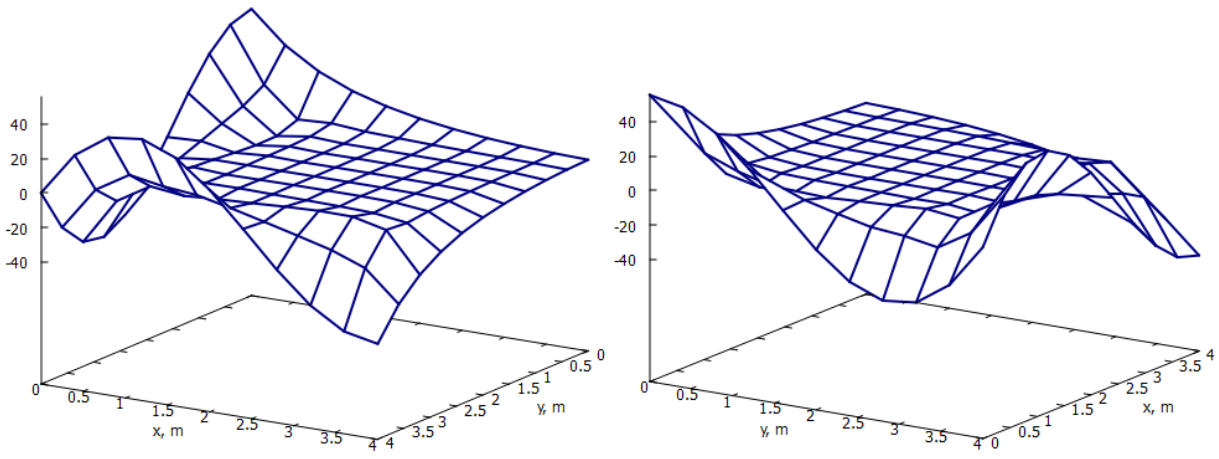


Рис. 14.17: Поверхня числового розв'язку рівняння Лапласа, подана з різних ракурсів.

Список `sol2` дає нам всі невідомі числові значення масиву `u`. Щоб побудувати поверхню розв'язку, необхідно сформувати матрицю, де на перетині вузла (i, j) стоятиме значення $u_{i,j}$. Впорядкування списку `sol2` обумовлює вигляд елементів матриці M.

```
(%i29) M:zeromatrix(m1+1,m2+1)$
(%i30) for i:1 thru m1+1 do
       for j:1 thru m2+1 do
       M[j,i]:rhs(sol2[j+(i-1)*m1+i-1])$
```

Нарешті створюємо поверхню розв'язку командою побудови поверхневої сітки (Рис.14.17). Оскільки в матриці M змінені місцями індекси i та j, осі Ox та Oy теж необхідно перейменувати.

```
(%i31) load(draw)$
```

```
(%i32) draw3d(
           color = navy,
           line_width = 2,
           surface_hide = true,
           xlabel = "y, m",
           ylabel = "x, m",
       elevation_grid(M,0,0,4,4)
       );
(%o32)  [gr3d (elevation_grid)]
```

Із здійсненого розв'язку можна одразу відмітити, що густина обраної сітки тут є набагато меншою, аніж у попередніх випадках. Якщо раніше вона була приблизно 30×600, то зараз стала 10×10. Кількість вузлових точок кардинально зменшилась, оскільки розв'язати систему з $(m_1-1)\times(m_2-1) = 81$ рівняння зовсім непросто, це вимагає значних обчислювальних ресурсів.

14.4.2 Двовимірне рівняння Пуассона

Рівняння цього типу описує розподіл скалярного потенціалу $\varphi(\vec{r})$ у присутності густини заряду $\rho(\vec{r})$. У двовимірному випадку воно набуває вигляду

$$u_{xx}(x,y) + u_{yy}(x,y) = f(x,y).$$

Процедура дискретизації призводить до вже наведеного базового рівняння, у якому в правій частині присутня ненульова функція $f_{i,j}$. Тому хід розв'язку фактично буде таким же.

► Вирішимо наступну задачу: розв'язати рівняння Пуассона на прямокутнику 1 м×1 м із функцією густини заряду $f(x,y) = 100(x^2+y^2)$ та крайовими умовами

$$\begin{cases} u(0,y) = 2\sin(2\pi y); \quad u(4,y) = 2\sin(2\pi y); \\ u(x,0) = \sin(2\pi x); \quad u(x,4) = \sin(2\pi x). \end{cases}$$

Запишемо розв'язок задачі без коментарів, він буде аналогічний до попереднього.

```
(%i2)  ratprint:false$ fpprintprec:5$
(%i6)  L1:1$ L2:1$ m1:10$ m2:10$
(%i8)  h1:L1/m1$ h2:L2/m2$
(%i10) g(x,y):=0$ f(x,y):=float(100*(x^2+y^2))$
(%i11) for i:2 thru m1+1 do
       (arraymake(X1, [i]), X1[1]:0,
       X1[i]:X1[i-1]+h1);
(%o11)  done

(%i12) for j:2 thru m2+1 do
       (arraymake(Y1, [j]), Y1[1]:0,
       Y1[j]:Y1[j-1]+h2);
(%o12)  done

(%i13) u_list:flatten(makelist(makelist(u[i,j],i,1,m1+1),j,1,m2+1))$
```

```
(%i15) A1(y):=float(2*sin(2*%pi*y))$ A2(y):=float(2*sin(2*%pi*y))$
(%i16) for j:1 thru m2+1 do
       (u[1,j]:A1(Y1[j]),
        u[m1+1,j]:A2(Y1[j]));
(%o16)  done

(%i18) B1(x):=float(sin(2*%pi*x))$ B2(x):=float(sin(2*%pi*x))$
(%i19) for i:1 thru m1+1 do
       (u[i,1]:B1(X1[i]),
        u[i,m2+1]:B2(X1[i]));
(%o19)  done

(%i20) for i:2 thru m1 do
       for j:2 thru m2 do
       u[i,j]:-((h2^2·u[i+1,j]+h1^2·u[i,j+1]-
                 h1^2·h2^2·f(X1[i],Y1[j])+h1^2·u[i,j-1]+
                 h2^2·u[i-1,j])/(h1^2·h2^2·g(X1[i],Y1[j])-
                 2·h2^2-2·h1^2));
(%o20)  done

(%i21) U:listarray(u)$
(%i22) sys2:makelist(u_list[k]=U[k],k,1,(m1+1)*(m2+1))$
(%i23) sol2:linsolve(sys2,u_list)$
(%i24) M:zeromatrix(m1+1,m2+1)$
```

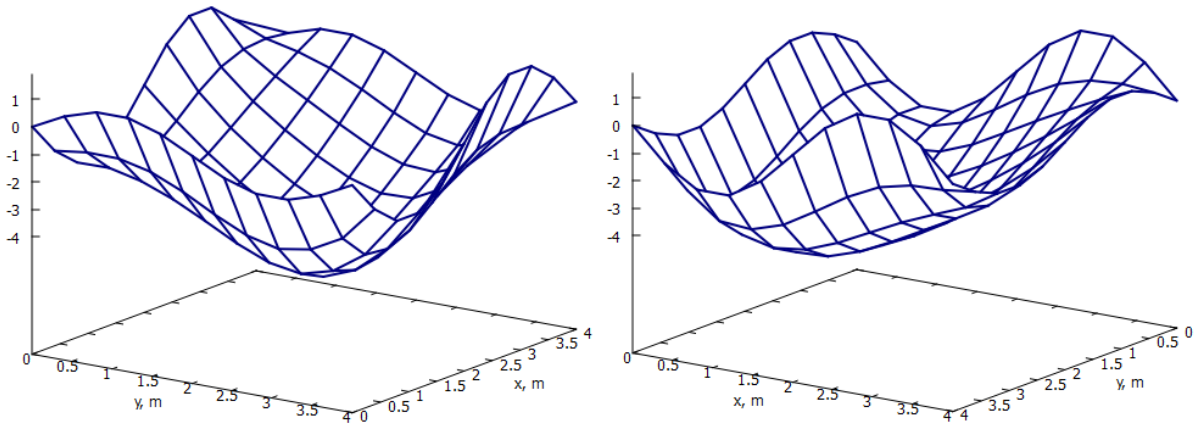


Рис. 14.18: Поверхня числового розв'язку рівняння Пуассона, подана з різних ракурсів.

```
(%i25) for i:1 thru m1+1 do
       for j:1 thru m2+1 do
       M[j,i]:rhs(sol2[j+(i-1)*m1+i-1]);
(%o25)  done

(%i26) load(draw)$
```

```
(%i27) draw3d(
           color = navy,
           line_width = 2,
           surface_hide = true,
           enhanced3d = false,
           xlabel = "y, m",
           ylabel = "x, m",
       elevation_grid(M,0,0,4,4)
       );
```

(%o27) [gr3d (*elevation_grid*)]

Побудована поверхня — на Рис.14.18.

14.5 Завдання до Розділу 14

Задачі теплопровідності

Розв'язати задачу Діріхле для одновимірного рівняння теплопровідності з початковою умовою $u(x,0) = A(x)$ та крайовими умовами $u(0,t) = B_1(t)$, $u(L,t) = B_2(t)$. Стрижень має довжину 0.5 м, час спостереження 0.3 с, $\sigma = 1$. Кількість проміжків розбиття обрати таким чином, щоб виконувалась умова стійкості. Зобразити результуючу поверхню зміни температури у часі, анімацію зміни температури, температурну діаграму.

14.1. $A(x) = \cos 2x$; $B_1(t) = 1 - 6t$; $B_2(t) = 0.4$.

14.2. $A(x) = x(x+1)$; $B_1(t) = 0$; $B_2(t) = 2t + 1$.

14.3. $A(x) = 3x(2-x)$; $B_1(t) = 0.1t$; $B_2(t) = 1$.

14.4. $A(x) = 1 - \ln(x + 0.4)$; $B_1(t) = 1.4$; $B_2(t) = t + 1$.

14.5. $A(x) = \sin(0.55x + 0.1)$; $B_1(t) = 3t + 1$; $B_2(t) = 0.5$.

14.6. $A(x) = 2x(1-x)$; $B_1(t) = 0.2$; $B_2(t) = t + 0.68$.

14.7. $A(x) = x(x+4) + 0.3$; $B_1(t) = 0.3$; $B_2(t) = 6t + 0.9$.

14.8. $A(x) = \cos(2x + 0.19)$; $B_1(t) = 0.9$; $B_2(t) = 0.18$.

14.9. $A(x) = (x - 0.2)(x+1) + 0.2$; $B_1(t) = 6t$; $B_2(t) = 0.84$.

14.10. $A(x) = \sin(\pi x/2)$; $B_1(t) = t$; $B_2(t) = 0.8$.

Розв'язати задачу Неймана для одновимірного рівняння теплопровідності з початковою умовою $u(x,0) = A(x)$, крайовою умовою $u(0,t) = B(t)$, функцією теплообміну $u_x(L,t) = C(t) = c(g(t) - u(L,t))$. Стрижень має довжину 1.2 м, час спостереження 0.3 с, $\sigma = 0.8$, $c = 0.5$. Кількість проміжків розбиття обрати таким чином, щоб виконувалась умова стійкості. Зобразити результуючу поверхню зміни температури у часі, анімацію зміни температури, температурну діаграму.

14.11. $A(x) = x(2x + 0.3)$; $B(t) = 0$; $g(t) = 2t - 1$.

14.12. $A(x) = \sin(x + 0.6)$; $B(t) = 0.45t$; $g(t) = -t$.

14.13. $A(x) = \cos(x + 0.5)$; $B(t) = 6t + 0.9$; $g(t) = 3$.

14.14. $A(x) = 1.5 - x(1 - x)$; $B(t) = 3(0.5 - t)$; $g(t) = 1.26t$.

14.15. $A(x) = \ln(2.42 + x)$; $B(t) = 0.38$; $g(t) = 6(0.1 - t)$.

14.16. $A(x) = 1.4 + \ln(x + 0.2)$; $B(t) = t + 0.6$; $g(t) = 1$.

14.17. $A(x) = x(1.2 - x) + 0.4$; $B(t) = 0$; $g(t) = t + 0.2$.

14.18. $A(x) = \sin x + \cos x$; $B(t) = 2(t + 0.1)$; $g(t) = 0.6 - 2t$.

14.19. $A(x) = \operatorname{tg}(x - 0.1)$; $B(t) = 0.1$; $g(t) = 0.2t$.

14.20. $A(x) = 2(2x - 1)$; $B(t) = 0.2t$; $g(t) = -t - 6$.

Хвильові задачі

Знайти числовий розв'язок одновимірного хвильового рівняння для струни із закріпленими кінцями, з початковими умовами $u(x, 0) = A(x)$, $u_t(x, 0) = B(x)$, довжина струни $L = 0.5$ м, час спостереження $T = 2$ с, швидкість $v = 1.5$. Кількість проміжків розбиття обрати таким чином, щоб виконувалась умова стійкості. Зобразити результуючу поверхню зміщення елементів струни у часі, анімацію коливань.

14.21. $A(x) = 2x(2x - 1)$; $B(x) = 0$.

14.22. $A(x) = x(0.5 - x)$; $B(x) = 0$.

14.23. $A(x) = \sin 2\pi x$; $B(x) = 0$.

14.24. $A(x) = 1 - \cos 4\pi x$; $B(x) = 0$.

14.25. $A(x) = x^2(0.5 - x)$; $B(x) = 0$.

14.26. $A(x) = 0$; $B(x) = \frac{1}{3}\sin\frac{5\pi x}{2}$.

14.27. $A(x) = 0$; $B(x) = \cos\frac{\pi}{2}(x - 1)$.

14.28. $A(x) = 0$; $B(x) = \frac{1}{2}\sin\frac{3\pi x}{2}$.

14.29. $A(x) = x(0.5 - x)$; $B(x) = \cos x$.

14.30. $A(x) = \operatorname{tg} x(0.5 - x)$; $B(x) = x + 0.1$.

Знайти числовий розв'язок одновимірного хвильового рівняння для струни із вільним кінцем, з початковими умовами $u(x, 0) = A(x)$, $u_t(x, 0) = B(x)$, $u_x(L, t) = C(t)$, довжина струни $L = 1$ м, час спостереження $T = 5$ с, швидкість $v = 0.5$. Кількість проміжків розбиття обрати таким чином, щоб виконувалась умова стійкості. Зобразити результуючу поверхню зміщення елементів струни у часі, анімацію коливань.

14.31. $A(x) = x\sin\pi x$; $B(x) = 0$; $C(t) = \cos t$.

14.32. $A(x) = 1 - \cos\pi x$; $B(x) = 0$; $C(t) = \exp(-2t)$.

14.33. $A(x) = \pi/4 - 2x$; $B(x) = 0$; $C(t) = \sin t$.

14.34. $A(x) = (x - 1)\sin\pi x$; $B(x) = 0$; $C(t) = 0.34t$.

14.35. $A(x) = 1 - \cos(x/2)$; $B(x) = 0$; $C(t) = t^2 - 1$.

14.36. $A(x) = 2 - \operatorname{arctg} x$; $B(x) = 0$; $C(t) = \exp(0.1t)$.

14.37. $A(x) = (2 - x)/2$; $B(x) = 0$; $C(t) = t(1 - t)$.

14.38. $A(x) = x(2 - x)$; $B(x) = 0$; $C(t) = 0.3 + 0.2t$.

14.39. $A(x) = \pi - x$; $B(x) = 0$; $C(t) = \operatorname{tg} t$.

14.40. $A(x) = \exp(-x)$; $B(x) = 0$; $C(t) = \ln(3 + t)$.

Рівняння Лапласа та Пуассона

Знайти числовий розв'язок двовимірного рівняння Лапласа із крайовими умовами $u(0,y)=A_1(y)$, $u(L_1,y)=A_2(y)$, $u(x,0)=B_1(x)$, $u(x,L_2)=B_2(x)$. Розмір області 1 м×1 м, кількість проміжків розбиття обрати $m_1=10$, $m_2=10$. Отриманий розв'язок зобразити у вигляді поверхні.

14.41. $A_1(y)=y-\sin y$; $A_2(y)=y^3-\cos y$; $B_1(x)=\cos x-x^3$; $B_2(x)=\sin x-x$.

14.42. $A_1(y)=e^{2y}$; $A_2(y)=y^2$; $B_1(x)=x^2$; $B_2(x)=e^{-2x}$.

14.43. $A_1(y)=\operatorname{tg} y$; $A_2(y)=-e^y$; $B_1(x)=e^x$; $B_2(x)=e^{-x}$.

14.44. $A_1(y)=y/(1+y)$; $A_2(y)=\operatorname{tg} y$; $B_1(x)=\operatorname{arctg} x$; $B_2(x)=x/(1+x)$.

14.45. $A_1(y)=\cos(y-1)$; $A_2(y)=e^{2y}$; $B_1(x)=e^{-2x}$; $B_2(x)=\sin(1-x)$.

14.46. $A_1(y)=2\sin 4\pi y$; $A_2(y)=4\sin 2\pi y$; $B_1(x)=\cos 3\pi x$; $B_2(x)=\frac{1}{3}\cos\pi x$.

14.47. $A_1(y)=10e^{-y^2}$; $A_2(y)=\frac{1}{4}\operatorname{tg} 4\pi y$; $B_1(x)=e^{1-x^2}$; $B_2(x)=10e^{-x^2}$.

14.48. $A_1(y)=y(y^2+1)$; $A_2(y)=-e^{y+1}$; $B_1(x)=-e^{x+1}$; $B_2(x)=x(x^2+1)$.

14.49. $A_1(y)=y\cos y$; $A_2(y)=\sin y+y^2$; $B_1(x)=x^2+\cos x$; $B_2(x)=x\cos x$.

14.50. $A_1(y)=-\cos(y^2+1)$; $A_2(y)=-\sin(y^2-1)$; $B_1(x)=\sin(x^2)$; $B_2(x)=\cos(x^2)$.

Знайти числовий розв'язок двовимірного рівняння Пуассона із крайовими умовами $u(0,y)=A_1(y)$, $u(L_1,y)=A_2(y)$, $u(x,0)=B_1(x)$, $u(x,L_2)=B_2(x)$ та заданою правою частиною $f(x,y)$. Розмір області 1 м×1 м, кількість проміжків розбиття обрати $m_1=10$, $m_2=10$. Величини $A_1(y)$, $A_2(y)$, $B_1(x)$, $B_2(x)$ взяти із відповідного номера попередньої задачі (рівняння Лапласа). Отриманий розв'язок зобразити у вигляді поверхні.

14.51. $f(x,y)=100e^{x^2+y^2}$.

14.52. $f(x,y)=50\operatorname{tg}(x-y)$.

14.53. $f(x,y)=40\operatorname{arctg}(x+y)$.

14.54. $f(x,y)=60(x+y)$.

14.55. $f(x,y)=80xy$.

14.56. $f(x,y)=50x-10y^2$.

14.57. $f(x,y)=10x^2+50y^2$.

14.58. $f(x,y)=100\ln(x+y+1)$.

14.59. $f(x,y)=70\sin x-20\cos y$.

14.60. $f(x,y)=90(x^2-y^2)$.

Бібліоґрафія

[1] Zachary Hannan. wxMaxima for Calculus I, II. — Solano Community College, 2022.

[2] Jose A Vallejo, Antonio Morante. Matematicas Basicas (con Software Libre). —Facultad de Ciencias, Universidad Autonoma de San Luis Potosi, 2020.

[3] Wilhelm Haager. Graphics with MAXIMA.
[http://www.austromath.at/daten/maxima/zusatz/Graphics_with_Maxima.pdf]

[4] Jan Awrejcewicz. Mathematical and Physical Pendulum. — Chapter, May 2012
[https://www.researchgate.net/publication/302218544].

[5] М.В. Ваврух, С.В. Смеречинський, О.М. Стельмах, Н.Л. Тишко. Збірник задач з механіки. — Львів: ЛНУ імені Івана Франка, 2017.

[6] Q. J. A. Khan, E. Balakrishnan, G. C. Wake. Analysis of a Predator–Prey System with Predator Switching. — Bulletin of Mathematical Biology (2004) 66, 109–123.

[7] Guillermo Davila, Antonio Morante and Jose A. Vallejo. Synchronization of dynamical systems: an approach using a Computer Algebra System. [arXiv:1809.05271v1 [nlin.CD] 14 Sep 2018].

[8] Свідзинський А. В. Математичні методи теоретичної фізики. У 2-х т. — Вид. 4-е, доповн. і переробл. — К.: Ін-т теорет. фізики ім. М. М. Боголюбова, 2009.

[9] Guillermo Davila. Synchronization of Chaos with the CAS Maxima. [https://atcm.mathandtech.org/EP2018/invited/4382018_21632.pdf]

[10] Hairer, E., et al. Solving Ordinary Differential Equations I: Nonstiff Problems. 2nd rev. ed, Springer, 2009.

[11] Yang, Cao, Chung, and Morris. Applied Numerical Methods Using MATLAB. — John Wiley and Sons, 2005.

[12] Abdelwahab Kharab, Ronald B. Guenther. An introduction to numerical methods: a MATLAB approach. — Boca Raton, Florida: CRC Press, 2018.

[13] S. R. K. Iyengar, R. K. Jain. Numerical Methods. — New Age International Limited Publisher, 2009.

[14] David M. Cook. Computation and problem solving in undergraduate physics. Second Edition. — Department of Physics, Lawrence University, 2023.

[15] E. Roque, J. A. Vallejo. Automated solving of constant-coefficients second-order linear PDEs using Fourier analysis, — [arXiv:2004.02604v1 [math.NA] 2 Apr 2020].