І. О. Князь

КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

Розділ
"Основи комп'ютерного моделювання"

Навчальний посібник

Рекомендовано вченою радою
Сумського державного університету

Суми
Сумський державний університет
2011

УДК 004.94 (075.8)
ББК 32.973.2–018я73
К 54

Рецензенти:

О.С. Мазманішвілі — доктор фізико-математичних наук, професор Сумського державного університету;
О.В. Хоменко — доктор фізико-математичних наук, професор Сумського державного університету;

Рекомендовано вченою радою Сумського державного університету як навчальний посібник
(протокол № 12 від 12.05.2011р.)

Князь І. О.

К 54 Комп'ютерне моделювання динамічних систем. Розділ "Основи комп'ютерного моделювання" : навч. посіб. / І. О. Князь. — Суми : Сумський державний університет, 2011.— 102 с.

У посібнику на наочних прикладах розглянуто методику та основні підходи до комп'ютерного моделювання. Теоретичний матеріал підкріплюється прикладами програм на мові С та MATLAB, які можуть бути модифіковані під будь-яку власну модель.

Посібник розрахований насамперед на студентів фізичних та математичних спеціальностей, а також аспірантів та науковців, які займаються комп'ютерним моделюванням.

УДК 004.94 (075.8)
ББК 32.973.2–018я73

Зміст

Вступ

Одне з перспективних застосувань ЕОМ у сучасній науці — це чисельне моделювання різноманітних природних явищ і процесів. У багатьох випадках комп'ютерне моделювання дозволяє істотно заощадити час та витрати при розв'язанні конкретних науково-технічних задач. Часто комп'ютерні моделі простіше і зручніше досліджувати, вони дозволяють проводити обчислювальні експерименти, реальне поставлення яких пов'язане зі значними затратами часу та коштів або може дати непередбачений результат. Логічність і формалізованість комп'ютерних моделей дозволяє виявити основні фактори, що визначають властивості досліджуваних об'єктів (а отже, глибше зрозуміти фундаментальні закони природи), досліджувати відгук фізичної системи на зміни її параметрів і початкових умов.

Зрозуміло, використання комп'ютера при розв'язанні фізичних та інших задач не може скасувати необхідність володіння теорією та уміння вирішувати проблеми аналітичними методами. Так само, як володіння звичайним калькулятором даремно без знань правил обчислень, так і наявність навіть найпотужнішої ЕОМ не допоможе без уміння сформулювати задачу і вибрати правильний алгоритм розрахунків; розуміння, як перевірити фізичну правдоподібність і оцінити похибки одержуваних результатів. З цього погляду комп'ютер — лише потужний інструмент, і чи будуть одержувати за його допомогою цінні наукові результати — визначається виняткова знаннями та уміннями дослідника. Тому наведений матеріал у першу чергу розрахований на студентів, які володіють основами математичного аналізу, базовими знаннями з курсів загальної фізики та програмування.

Матеріал даного посібника охоплює курс "Методика комп'ютерного експерименту" та частково курс "Моделювання синергетичних систем", які викладаються студентам старших курсів спеціальності "Прикладна математика" СумДУ. Частина розділів даного посібника розглядається у рамках дисципліни "Обчислювальна техніка, алгоритмічні мови та програмне забезпечення", що викладається студентам спеціальності "Електронні прилади та пристрої".

Головна мета даного курсу — набуття студентами базових теоретичних і практичних навичок моделювання фізичних явищ на ЕОМ. Передбачається, що всі програми для розв'язання запропонованих задач пишуться самими студентами. Успішне засвоєння курсу свідчить про те, що студент здатний самостійно моделювати на ЕОМ досить складні фізичні явища.

Мовами програмування для виконання задач курсу обрано мову С та спеціалізовану мову технічних розрахунків MATLAB. На це є декілька поважних причин. З одного боку, мову С більшість студентів технічних спеціальностей вузів вивчають у курсі "Програмування"; написання програми на традиційній мові (у нашому випадку С) дозволяє досліднику "відчути" окремі дрібні деталі програмного алгоритму. З іншого боку, програмна реалізація окремих "підробиць" вимагає значного часу, високої кваліфікації програміста, а для багатьох задач і математичної підготовки, достатнього досвіду. MATLAB дозволяє дещо автоматизувати процес написання програм, оскільки у своїх бібліотеках містить масу готових відлагоджених високоефективних алгоритмів (процедур) для вирішення задач з багатьох областей природознавства. Крім того, ця система є матрично-орієнтованою (MATLAB — MATrix LABoratory), що дозволяє вирішувати багато обчислювальних задач, пов'язаних з матричним формулюванням, істотно скорочуючи час, необхідний для програмування на стандартних мовах типу С або Pascal. Зазначимо також, що вибір мови програмування у даному випадку не є принциповим і основні розрахункові процедури програм на С можуть бути легко модифіковані під будь-який інший компілятор. Процедури, що реалізують математичні розрахунки, із незначними змінами синтаксису природним чином можна включити у програми на Delphi, Visual C++, Visual Basic і т.п. Що стосується MATLAB, ця система, на нашу думку, є найбільш ефективною серед програмних пакетів обчислювальної математики, недаремно у західних країнах MATLAB фактично є стандартом при проведенні наукових досліджень.

Основний акцент при розв'язанні задач робиться на поданні одержуваних результатів у наочному графічному вигляді (візуалізація процесів поширення тепла, побудова траєкторій руху частинок, графічне

зображення силових ліній тощо). Графічна інтерпретація рішень дозволяє легше виявити закономірності, наочно побачити динаміку процесів. Варто також сказати, що значна частина запропонованих задач або не має аналітичного розв'язку, або розв'язується досить складними наближеними аналітичними методами.

І останнє зауваження. Незважаючи на те що курс побудований в основному на задачах фізики, більшість з використовуваних чисельних методів та підходів є досить загальними і застосовуються для розв'язання задач з інших галузей знань: хімії, економіки, біології тощо.

Розділ 1

Складові процесу моделювання

У даному розділі розглядаються основні етапи проведення повноцінного наукового дослідження з використанням комп'ютерної техніки: від поставлення задачі до аналізу отриманих результатів.

1.1 Основні етапи моделювання

Процес побудови та дослідження комп'ютерних моделей прийнято називати комп'ютерним експериментом (за наявності випадкових чинників, що впливають на динаміку системи) або комп'ютерним моделюванням (у випадку цілком детерміністичних систем). Виділимо основні складові цього процесу:

а) виділення суттєвих для даного дослідження властивостей вихідного об'єкта та законів його поведінки;

б) побудова математичної моделі;

в) програмна реалізація отриманої моделі;

г) оцінка адекватності отриманої комп'ютерної моделі; як правило, оцінка адекватності приводить до підвищення вимог до моделі і повернення на етап 1;

д) дослідження моделі та аналіз отриманих результатів.

Розглянемо основні етапи проведення комп'ютерного моделювання більш детально.

1.2 Побудова математичної моделі

Розглянемо процес створення математичної (комп'ютерної) моделі. На цьому етапі можливими є два випадки: перший — закон руху та властивості об'єкта апріорно невідомі, другий — існує базова модель, яка має бути модифікована з урахуванням мети моделювання.

У першому випадку необхідно провести серію експериментів з об'єктом, опрацювати отримані експериментальні дані (часовий ряд) і на їх основі, у найкращому разі, реконструювати модельні рівняння системи (див., наприклад, роботу [1] та посилання в ній).

Розглянемо більш детально другий випадок, що, як правило, частіше зустрічається на практиці. У цьому випадку для опису поведінки об'єкта, що досліджується, можуть бути застосовані вже відомі закони фізики, хімії, економіки тощо, — залежно від природи об'єкта що розглядається. На цьому етапі вибирається структура моделі — придатний математичнимй апарат, вигляд та число рівнянь, вигляд функції, задаються певні розумні наближення для коефіцієнтів рівняння, які будуть уточнюватися в процесі проведенні комп'ютерного моделювання і перевірки адекватності моделі.

Як приклад розглянемо, наприклад, моделювання коливань вантажу на пружині. Цей об'єкт можна розглядати як фізичну модель, наприклад, підвіски автомобіля, атома у кристалічній гратці та багатьох інших систем, на інертні елементи яких при їх відхилинні від положення рівноваги діє пружна сила. З другого закону Ньютона випливає рівняння руху

$$m\frac{d^2\overrightarrow{r}}{dt^2} = \overrightarrow{F},$$

де $\overrightarrow{F}$ — рівнодійна сил, $\overrightarrow{r}$ — радіус-вектор центра мас. Дане рівняння є базовим, на основі якого з урахуванням зроблених припущень утворюються різноманітні моделі. Наприклад, якщо сила пружності прямо пропорційна величині деформації пружини ($F_{\text{упр}} = -kx$), отримуємо рівняння консервативного лінійного осцилятора (одновимірний випадок):

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0.$$

За наявності сили тертя, $F_{\text{тр}} = -r(dx/dt)$, утворюється рівняння дисипативного осцилятора:

$$\frac{d^2x}{dt^2} + r\frac{dx}{dt} + \frac{k}{m}x = 0.$$

З урахуванням залежності сили пружності від деформації, $k = k_1 + k_2 x$, або залежності коефіцієнта тертя від швидкості, $r = r_1 - r_2(dx/dt)$, отримуємо нові моделі з суттєво збагаченими властивостями.

На початковому етапі побудови моделі доцільним є намагання спростити модель, ураховуючи лише суттєві фактори. Чи є необхідним урахування того чи іншого фактора, залежить від діапазону зміни параметрів. Крім того, відмітимо, що апарат диференціальних рівнянь не завжди є найбільш адекватним для опису руху об'єкта. Так, наприклад, якщо б метою моделювання був би якісний опис згасаючих коливань, можна використовувати різницеві рівняння ($x_{n+1} = f(x_n)$, де n — дискретний час). Для загасаючого маятника різницеву модель можна отримати припустивши, що закон загасання є експоненціальним. У такому разі $x_{n+1} = e^{-rT} x_n$, де T — квазіперіод коливань.

Відзначимо, що, як правило, модель поступово розвивається у напрямку її ускладнення, а її формування закінчується, коли вона із придатною точністю описує об'єкт (явище) у необхідному діапазоні зміни параметрів.

1.3 Програмна реалізація математичної моделі

На даному етапі незалежно від вибору мови програмування важливим є підбір придатного чисельного методу для розв'язання рівнянь руху об'єкта, що досліджується. Некоректний вибір чисельного методу може стати джерелом багатьох проблем, починаючи від неефективності розрахунків і закінчуючи одержанням некоректних результатів. Справа в тому, що одній і тій самій математичній моделі можна поставити у відповідність безліч дискретних моделей і обчислювальних алгоритмів (чисельних методів). Тому при виборі чисельного методу необхідно враховувати дві групи вимог: 1) дискретна модель повинна бути адекватна до математичної моделі; 2) чисельний метод має бути коректним і придатним для реалізації на комп'ютері.

Для забезпечення першої вимоги дискретна модель повинна мати властивості збіжності чисельного методу, виконання дискретних аналогів збереження і якісно правильного поводження розв'язку. Збіжність чисельного методу означає, що при зменшенні кроку інтегрування то-

чність чисельного інтегрування зростає. Крім того, різні математичні моделі є вираженням фізичних законів збереження, тому для дискретної моделі закони збереження також повинні виконуватися. Якісно правильне поводження розв'язку означає, що при переході до дискретного аналогу моделі не втрачаються важливі деталі поводження реальної системи.

Коректність чисельного методу означає, що дискретна задача повинна мати однозначний розв'язок і бути стійкою до похибок вихідних даних та похибкам обчислень. Придатність алгоритму до реалізації визначається об'ємом пам'яті і швидкодією комп'ютера. Обчислювальний алгоритм має ставити розумні вимоги до ресурсів комп'ютера. Наприклад, математично коректний метод Крамера розв'язання систем лінійних алгебраїчних рівнянь є абсолютно непридатним для розв'язання реальних задач: якщо прийняти, що кожна арифметична операція виконується за 10^{-6}с , то для розв'язання системи з 20 невідомими методом Крамера потрібно більше мільйона років. У той самий час найпростішим методом Гаусса ця система буде розв'язана за декілька секунд.

Таким чином, процедура підбору чисельного методу для конкретної системи є достатньо складною і потребує певних теоретичних знань. Невдало підібраний чисельний метод може призвести до якісно неправильних результатів, і дуже часто важко визначити, чи дійсно дана поведінка притаманна моделі, чи вона є результатом помилок.

1.4 Оцінка адекватності моделі

Оцінка адекватності моделі — це проведення спеціальних чисельних експериментів, результати яких є апріорно відомими. Для перевірки правильності моделі можуть застосовуватися вже відомі експериментальні залежності, існуючі оцінки розв'язку, аналітичні розрахунки у граничних випадках. Для моделі маятника, наприклад, корисним буде простежити за поводженням візуалізації коливань — це і буде грубою перевіркою правдоподібності руху, що моделюється. Недаремно останнім часом дуже багато уваги приділяється питанням візуалізації еволюції складних систем. Ще одним тестом може бути перевірка закону збереження енергії, наприклад, у випадку маятника без тертя повна енергія

повинна зберігатися. Однак слід зауважити, що збереження будь-яких відомих інваріантів або закономірностей тільки збільшує ступінь довіри до моделі, але ще не свідчить про відсутність помилок. Нарешті, отримані дані чисельного експерименту можна порівняти з експериментальними.

У результаті проведення комп'ютерних експериментів виявляються помилки та неточності моделі. Наприклад, якщо амплітуда коливань маятника за відсутності зовнішньої сили починає збільшуватися, то швидше за все існує помилка у рівняннях руху, наприклад, неправильно вибрані знаки у відповідних коефіцієнтах. Або, наприклад, не врахований певний суттєвий фактор. Тому необхідно виправити рівняння і повторити експерименти з самого початку.

1.5 Запитання для самоконтролю

1 Назвіть основні етапи проведення комп'ютерного моделювання.

2 Яким чином будуються математичні моделі? Наведіть приклади.

3 Якими критеріями необхідно користуватися при виборі чисельного методу для розв'язання задач моделювання на комп'ютері?

4 Яким чином можна оцінити результати моделювання?

Розділ 2

Елементи чисельних методів

У розділі розглядаються окремі питання теорії чисельних методів, що стосуються практичного розв'язання задач даного посібника, та особливості їх практичної програмної реалізації. Робиться акцент на деяких "підводних камінцях", що часто зустрічаються при створенні власних програм та пов'язані зі специфікою обчислень на ПК. Оскільки у багатьох випадках пряме моделювання фізичних систем потребує значних витрат машинного часу, особлива увага приділяється підвищенню точності розрахунків із одночасною оптимізацією програмних алгоритмів за швидкістю.

2.1 Теоретичний матеріал

Дуже часто при розв'язанні практичних задач виникає необхідність у виконанні операцій чисельного диференціювання та інтегрування функцій. Це обумовлено декількома причинами. По-перше, функції, що описують динаміку системи, не завжди мають аналітичне представлення. Вони можуть обчислюватися, наприклад, у процесі чисельного розв'язання задачі. По-друге, досить часто у практичних випадках інтеграл або похідну не можна виразити у вигляді елементарних функцій або навіть коли останні можуть бути отримані аналітично, часто більш простим і швидким способом одержання результатів є саме чисельні методи.

Розглянемо основні підходи до чисельного диференціювання та інтегрування.

2.1.1 Дискретне представлення неперервної змінної

Нехай задана функція $y(x)$. Розіб'ємо інтервал значень $x \in [a, b]$ на $J-1$ елементарних відрізків довжиною $h = \triangle x$ і побудуємо J-вимірний вектор $\{x_j\}$, визначаючи змінну тільки в точках з номером j: $x_j = a + (j-1) * h, j \in [1, J]$.

У такому випадку довільну функцію $y(x)$ можна наблизити вектором $\{y(x_j)\}$, визначеним у вузлах гратки x_j. Незважаючи на те що $\{y(x_j)\}$ є неповним описом функції $y(x)$, її можна наблизити значеннями $\{y(x_j)\}$ для будь-якої точки x' ($x_j \le x' \le x_{j+1}$) за допомогою інтерполяції векторних компонентів $y(x_j)$ та $y(x_{j+1})$ між сусідніми точками. Нехай

$$\varepsilon = \frac{x' - x_j}{x_{j+1} - x_j}, \tag{2.1}$$

тоді наближене значення функції y у точці x' розраховується за формулою

$$y(x') = \varepsilon y(x_{j+1}) + (1 - \varepsilon) y(x_j). \tag{2.2}$$

Зазвичай, якість такого наближення значно знижується, якщо функція $y(x)$ сильно змінюється на інтервалі $[x_j, x_{j+1}]$, що потребує додаткового розвинення процедури апроксимації або збільшення кількості інтервалів J.

2.1.2 Різницеві похідні

Розглянувши подання неперервної функції $y(x)$ у дискретному вигляді, зупинимося тепер на різницевій апроксимації її похідних. Виходячи з рис. 2.1а, бачимо, що апроксимація лівої (-), правої (+) і центральної різницевих похідних першого порядку $y'(x_j) \equiv dy/dx\,|_{x_j}$ у точці x_j може бути подана у вигляді :

$$y'(x_j)_- = \frac{y(x_j) - y(x_{j-1})}{h}, \tag{2.3}$$

$$y'(x_j)_+ = \frac{y(x_{j+1}) - y(x_j)}{h}, \tag{2.4}$$

$$y'(x_j) = \frac{y(x_{j+1}) - y(x_{j-1})}{2h}. \tag{2.5}$$

Доведемо, що остання формула є більш точною, що забезпечує її широке практичне застосування. Розвинемо функцію $y(x)$ у ряд Тейлора

в околі точки x_j для $y(x_{j+1})$ та $y(x_{j-1})$:

$$y(x_{j+1}) = y(x_j) + y'(x_j)h + y''(x_j)h^2/2 + y'''(x_j)h^3/6 + O(h^4). \quad (2.6)$$

Звідси

$$\begin{aligned} y'(x_j) &= \frac{y(x_{j+1}) - y(x_j)}{h} - y''(x_j)h/2 - y'''(x_j)h^2/6 + O(h^3) = \\ &= \frac{y(x_{j+1}) - y(x_j)}{h} + O(h). \end{aligned} \quad (2.7)$$

Порівнюючи останнє співвідношення із (2.4), бачимо, що порядок похибки становить $O(h)$. Аналогічно

$$y(x_{j-1}) = y(x_j) - y'(x_j)h + y''(x_j)h^2/2 - y'''(x_j)h^3/6 + O(h^4). \quad (2.8)$$

Віднімаючи (2.8) від (2.6), маємо

$$\begin{aligned} y'(x_j) &= \frac{y(x_{j+1}) - y(x_{j-1})}{2h} - y'''(x_j)h^2/6 + O(h^3) = \\ &= \frac{y(x_{j+1}) - y(x_{j-1})}{2h} + O(h^2). \end{aligned} \quad (2.9)$$

Отже, похибка має більш високий порядок $O(h^2)$, що і необхідно було довести. За необхідності можна провести інтерполювання за чотирма точками і отримати інтерполяційну формулу для наближення $y'(x_j)$, порядок похибки якої становить $O(h^4)$ (див., наприклад, роботу [2]).

Аналогічно можуть бути отримані різницеві похідні більш високих порядків. Наприклад, формули центральних похідних з точністю порядку $O(h^2)$ мають вигляд:

$$y''(x_j) = \frac{y(x_{j+1}) - 2y(x_j) + y(x_{j-1})}{h^2}. \quad (2.10)$$

$$y'''(x_j) = \frac{y(x_{j+2}) - 2y(x_{j+1}) + 2y(x_{j-1}) - y(x_{j-2})}{2h^3}. \quad (2.11)$$

$$y''''(x_j) = \frac{y(x_{j+2}) - 4y(x_{j+1}) + 6y(x_j) - 4y(x_{j-1}) + y(x_{j-2})}{h^4}. \quad (2.12)$$

Для будь-якої похідної рекурентна функція знаходження її чисельного значення є достатньо простою і може бути реалізована програмно, наприклад, так:

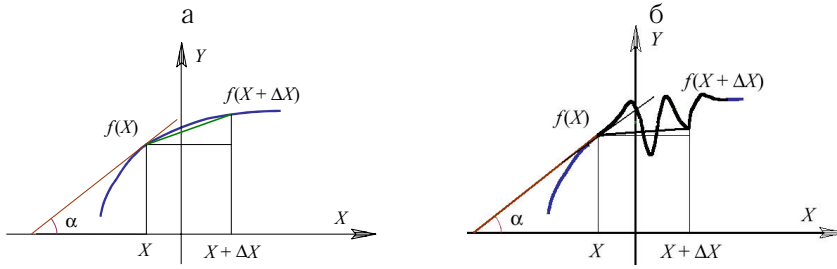

Рисунок 2.1 — Приклад "гарного" (а) та "поганого" (б) наближення першої похідної від функції $Y = f(X)$.

```
float dx=0.001;
float F(float x, int n) {
  if (n<=0)
    return x*x/(1+x*x);
  else
    return (F(x+dx,n-1)-F(x,n-1))/dx;
};
```

У даному фрагменті програми наведена реалізація функції знаходження n-ї правої похідної від функції $F = x^2/(1 + x^2)$ у точці x (крок 0.001).

Природно, що різницеві співвідношення (2.3)-(2.5) та (2.10)-(2.12) є придатними апроксимаціями похідних, якщо сама функція $y(x)$ не дуже швидко змінюється на відрізку $[x_{j-1}, x_{j+1}]$ (рис. 2.1б).

Значення кроку $h > 0$, що за своїм змістом повинне бути досить малим у формулах наближеного обчислення похідних, на практиці вже для другої різницевої похідної не можна брати занадто малим. Це пов'язане з тим, що комп'ютер обчислює значення функції у точках сітки з деякою похибкою (округляє до певної кількості знаків залежно від типу змінної (див. наступний розділ), і може виникнути ситуація, коли в наближеній формулі похибка чисельника стає величиною того самого порядку, що сам знаменник (особливо, якщо на проміжку $[x, x + h]$ функція змінюється монотонно), і тому результат обчислення може бути дуже далеким від шуканого точного результату. Обчислюючи з дуже малим h значення $y''(x)$ при різних x і спостерігаючи за поводженням цих значень, ми мо-

жемо одержати "пульсації" графіка функції $y''(x)$, навіть якщо ця функція насправді є монотонною і гладкою. Для усунення цього недоліку необхідно робити вибір: або збільшувати точність обчислення функції (наприклад, змінюючи тип змінних), або задовольнятися великими, скажімо, $h = 0.01$ замість $h = 0.001$ (але не занадто великими) значеннями кроку h.

Як демонстраційний приклад розглянемо процес наближення похідної від функції $y(x) = e^x$ у точці $x = 1$ при різних значеннях кроку h [2]. Використовуючи визначення похідної

$$y'(x) = \lim_{h \to 0} \frac{y(x+h) - y(x)}{h}, \tag{2.13}$$

виберемо послідовність $\{h_k\}$ у такий спосіб, що $h_k = 10^{-k}$, і розрахуємо величину

$$G_k = \frac{y(x+h_k) - y(x)}{h_k} \quad \text{для} \quad k = 1, 2, .., 8. \tag{2.14}$$

Результати обчислень наведені у табл. 2.1.

Таблиця 2.1 — Дискретна похідна від функції $y(x) = e^x$ у точці $x = 1$

h_k	$y_k = y(1 + h_k)$	$y_k - e$	$G_k = (y_k - e)/h_k$
$h_1 = 0.1$	3.0041661	0.2858843	2.8588428
$h_2 = 0.01$	2.7456009	0.0273191	2.7319112
$h_3 = 0.001$	2.7210014	0.0027196	2.7195582
$h_4 = 0.0001$	2.7185538	0.0002720	2.7195308
$h_5 = 0.00001$	2.7183089	0.0000271	2.7097173
$h_6 = 10^{-6}$	2.7182846	0.0000028	2.7784748
$h_7 = 10^{-7}$	2.7182820	0.0000002	1.5587020
$h_8 = 10^{-8}$	2.7182817	-0.0000001	-8.2548409

Як відомо, математичне значення похідної для функції $y(x) = e^x$ у точці $x = 1$ дорівнє ≈ 2.7182818. Отже, аналізуючи таблицю, бачимо, що досить великі значення h не дають гарного наближення похідної.

Аналогічна ситуація простежується у випадку, якщо h_k є достатньо малим. При цьому розраховані значення $y(x + h_k)$ та $y(x)$ дуже близькі, а їх різниця є демонстрацією втрати точності (за рахунок машинних округлень (див. розділ 3) при відніманні практично рівних значень, наприклад, величина $h > h_7$ настільки мала, що відношення приростів дорівнює нулю. З таблиці бачимо, що найкраще наближення похідної $G_4 = 2.7195308$ забезпечується вибором кроку $h_4 = 10^{-4}$.

Зазначимо, що наведені розрахунки проведені з використанням чисел одинарної точності (тип float у C++), що забезпечують точність 7-8 десяткових розрядів. При використанні чисел подвійної точності найбільш оптимальне значення кроку $h = 10^{-7}$.

Зрозуміло, що оптимальне значення кроку залежить не лише від типу змінних, але і від вигляду самої функції. Очевидно, якщо функція є монотонною та гладкою, різниця $y(x + h_k) - y(x)$ фактично дорівнює нулю вже при досить великих h, наприклад $h = 0.1$.

2.1.3 Наближені методи розв'язання систем диференціальних рівнянь

У багатьох випадках потрібно досліджувати динамічні системи — системи, рух яких визначається диференціальними рівняннями. Лише для небагатьох диференціальних рівнянь розв'язки можуть бути знайдені аналітично (тобто виражені через відомі функції). У тих випадках, коли аналітичний розв'язок не може бути знайденим, використовуються наближені чисельні методи. У даному посібнику наводиться декілька чисельних схем, які, незважаючи на свою простоту, можуть бути застосовані для розв'язання багатьох задач.

Розглянемо простий приклад — рух частинки у полі сил, причому обмежимося рухом уздовж однієї прямої. Координата частинки x та її швидкість v визначаються рівняннями:

$$\begin{aligned} \frac{dx}{dt} &= v, \\ \frac{dv}{dt} &= a(x), \end{aligned} \tag{2.15}$$

де $a(x) = F(x)/m$, $F(x)$ — сила, що діє на частинку; m — маса частинки. Завдання полягає в обчисленні залежностей $x(t)$, $v(t)$ за умови, що задано початкові значення координати і швидкості $x(0)$ і $v(0)$.

Вигляд цих рівнянь являє собою очевидний натяк на можливий спосіб обчислень (**метод Ейлера**): вибрати досить мале значення величини $\triangle t$, а потім скористатися співвідношеннями:

$$x(t+\triangle t) \approx x(t) + \frac{dx}{dt}\triangle t = x(t) + v(t)\triangle t, \qquad (2.16)$$

$$v(t+\triangle t) \approx v(t) + \frac{dv}{dt}\triangle t = v(t) + a(t)\triangle t. \qquad (2.17)$$

Багаторазово застосовуючи ці співвідношення, можна розрахувати значення x і v у ряді дискретних, але досить близьких одна до одної точок. Щоб оцінити величину відхилення від дійсного закону руху, запишемо, наприклад для x, більш точну рівність:

$$x(t+\triangle t) \approx x(t) + \frac{dx}{dt}\triangle t + \frac{1}{2}\frac{d^2x}{dt^2}(\triangle t)^2 = x(t) + v(t)\triangle t + \frac{1}{2}a(t)(\triangle t)^2. \qquad (2.18)$$

З цього виразу бачимо, що неточність на кожному кроці пропорційна величині $(\triangle t)^2$; для просування на час t необхідне число кроків дорівнює $t/\triangle t$, так що неточність виявиться пропорційною величині $t \cdot \triangle t$. Якщо необхідно (при заданому інтервалі t) поліпшити точність у 10 разів, потрібно в 10 разів зменшити крок $\triangle t$. Е стільки ж разів збільшиться число кроків і час, необхідний для розрахунку.

Метод Ейлера приваблює своєю простотою і легкістю програмної реалізації, однак його істотним недоліком є нестійкість. Нестійкість методу Ейлера приводить до незбереження енергії в тих задачах, де ця величина повинна залишатися сталою, наприклад, у задачі про вільні коливання маятника або про рух частинки у центральному полі. Наведена обставина веде до якісно неправильного з погляду законів фізики поводження чисельного розв'язку задачі при великій кількості ітерацій. Наприклад, амплітуда коливань маятника, що вільно коливається у полі

сили тяжіння, не буде залишатися незмінною, а буде поступово збільшуватися так, неначе на маятник діє сила, що розгойдує його! Незважаючи на це, метод Ейлера може бути застосований до неосциляторних рівнянь (рівнянь із "загасанням") за умови достатньо малих $\triangle t$.

Зміст рівності (2.18) полягає у розрахунку прискорення на інтервалі $\triangle t$. Можна досягти приблизно тієї ж точності, що й у (2.18), якщо взяти середнє значення швидкості на тому самому інтервалі або, що зручніше за все, значення швидкості у середині інтервалу, $v(t + \triangle t/2)$ (**метод з переступом**). Прискорення ж, необхідне для виконання зсуву на крок за часом для швидкості, потрібно буде обчислювати у середині інтервалу $(t+\triangle t/2, t+\triangle t/2+\triangle t)$, тобто в момент $t+\triangle t$, що нас теж улаштовує, тому що це дозволить знайти $x(t + \triangle t)$. Таким чином, для збільшення точності досить обчислювати значення координат у моменти часу t, $t + \triangle t$, $t + 2\triangle t$, $t + 3\triangle t$,..., а значення швидкості — у моменти $t + \triangle t/2$, $t + 3\triangle t/2$, $t + 5\triangle t/2$, $t + 7\triangle t/2$,... Для оцінки неточності розрахунку запишемо (знову тільки для x):

$$x(t + \triangle t) \approx x(t) + v(t + \triangle t/2)\triangle t \approx x(t) + \left(v(t) + \frac{dv}{dt}\triangle t/2\right)\triangle t,$$

що збігається з (2.18). Неточність має порядок $(\triangle t)^3$.

Отже, схема розрахунків буде мати вигляд: перший крок — просуваємося на крок $\triangle t/2$:

$$v(\triangle t/2) = v(0) + a(0)\triangle t/2, \tag{2.19}$$

другий крок — ітераційна процедура за часом (починаємо з $t = 0$ та збільшуємо на кожній ітерації час на $\triangle t$):

$$\begin{aligned} x(t + \triangle t) &= x(t) + v(t + \triangle t/2)\triangle t, \\ v(t + 3\triangle t/2) &= v(t + \triangle t/2) + a(t + \triangle t)\triangle t, \\ x(t + 2\triangle t) &= x(t + \triangle t) + v(t + 3\triangle t/2)\triangle t, \\ v(t + 5\triangle t/2) &= v(t + 3\triangle t/2) + a(t + 2\triangle t)\triangle t, \\ &\cdot \quad \cdot \quad \cdot \end{aligned} \tag{2.20}$$

Очевидно, такий спосіб розрахунків можна застосувати і для векторних величин. Однак цей прийом не спрацьовує, якщо сила, що діє на частинку, залежить не тільки від координати, але й від швидкості.

Розглянемо ще один корисний метод, який має просту комп'ютерну реалізацію та може бути застосований для розв'язання багатьох фізичних задач (у тому числі і для систем, права частина яких залежить як від координати, так і від швидкості) — **модифікований метод Ейлера**. Зазначений метод є модифікацією метода Ейлера: координата частинки $x(t + \triangle t)$ визначається через прискорення $a(t + \triangle t)$ у кінцевій точці інтервалу.

$$\begin{aligned} v(t + \triangle t) &= v(t) + a(t)\triangle t, \\ x(t + \triangle t) &= x(t) + v(t + \triangle t)\triangle t. \end{aligned} \tag{2.21}$$

2.1.4 Чисельне розв'язання звичайних диференціальних рівнянь у системі MATLAB

У системі MATLAB існує цілий пакет процедур, призначених для розв'язання систем звичайних диференціальних рівнянь (ЗДУ). Даний пакет дає можливість застосувати найбільш відомі чисельні методи, причому для економії зусиль і полегшення роботи написання необхідного додаткового коду (опис конкретної системи рівнянь) однакове і не залежить від обраного алгоритму. Отже, у даному підрозділі розглядається метод підготовки M-файлів для систем ЗДУ незалежно від алгоритму, що застосовується для розв'язання задачі [1]. З наведених вище прикладів зрозуміло, що великий клас ЗДУ зводиться до системи диференці-

[1] MATLAB може виконувати послідовність операторів, записаних у файл на диску. Такі файли називаються m-файлами, тому що імена цих файлів мають вигляд <ім'я>.m. Велика частина роботи в MATLAB полягає у створенні, редагуванні і виконанні таких m-файлів. Існує два типи m-файлів: файли-програми, або сценарії, та файли-функції. Файл-програма складається з послідовності звичайних операторів MATLAB. Якщо файл із таким сценарієм має ім'я, наприклад, solve.m, то команда solve, введена в командному рядку, викликає виконання відповідної послідовності операторів. Функції фактично дають можливість розширювати MATLAB, оскільки визначені нові функції, специфічні для розв'язання конкретних задач, мають той самий статус, що й інші функції MATLAB. **Ім'я функції повинне збігатися з іменем файла на диску!**

альних рівнянь першого порядку виду $\overrightarrow{y}'(t) = \overrightarrow{F}(t, \overrightarrow{y}(t))$ з початковими умовами $\overrightarrow{y}(t_0) = \overrightarrow{y}_0$.

Розглянемо приклад підготовки M-файла для розв'язання рівняння Ван-дер-Поля

$$y'' - \mu(1 - y^2)y' + y = 0. \tag{2.22}$$

Виконуючи заміну змінних $y_1 = y$, $y_2 = y'$, можна переписати це рівняння у вигляді системи рівнянь:

$$\begin{aligned} y_1' &= y_2, \\ y_2' &= \mu(1 - y_1^2)y_2 - y_1. \end{aligned} \tag{2.23}$$

Створимо M-файл (*File* → *New* → *MFile*), який буде описувати праву частину даної системи рівнянь. Для нашого випадку ця функція (μ= 1, а y_1 і y_2 стають елементами `y(1)` та `y(2)` двовимірного вектора `y`) повинна мати такий вигляд:

```
function dy =vdp1(t,y)
dy =[y(2); (1-y(1)^2)*y(2)-y(1)];
```

Хоча у розглянутому випадку права частина системи не залежить явно від t, а для деяких систем рівнянь може не залежати від y, функція повинна мати не менше двох формальних параметрів `t` і `y`.

Для розв'язання системи рівнянь на часовому інтервалі [0..30] і з початковими значеннями $y_1(0) = 1$, $y_2(0) = 0$ у вікні команд необхідно написати

```
[T,Y]=ode45('vdp1',[0 30],[1 0]);
```

У результаті одержимо вектор-стовпчик `T`, що містить моменти часу (крок вибрано за замовчуванням) на заданому часовому інтервалі [0..30], та матрицю `Y`, що складається з двох стовпчиків (y_1 та y_2), кожен рядок якої відповідає рядку (моменту часу) з масиву `T`. Результат можна подати графічно, використовуючи таку послідовність команд

```
.............
plot(T,Y(:,1),'-',T,Y(:,2),'--');
```

```
% Команда plot створює графік за набором точок відповідних
% масивів. Y(:,1) - усі рядки першого стовпця матриці Y,
% '-' - суцільна крива. Y(:,2) - усі рядки другого стовпця
% матриці Y, '--' - пунктирна крива.
% Таким чином на графіку маємо дві криві y_1(t) та y_2(t).
title(' Розв'язок рівняння Ван-дер-Поля для mu=1 ');
xlabel('Час T '); ylabel(' Y ');
legend('y1','y2 ');
...............
```

Для розв'язання тих чи інших задач можна вибирати різні процедури чисельного розв'язку ЗДУ. Так, у наведеному вище прикладі використовувалася функція `ode45`, що базується на явному методі Рунге-Кутта. Це однокроковий алгоритм — для обчислення $y(t_n)$ необхідно знати розв'язок в одній попередній точці $y(t_{n-1})$. Ця функція найбільш зручна для першого, "пристрілочного" розв'язку більшості задач. Крім `ode45`, можуть бути використані функції:

- `ode23` — теж базується на явному методі Рунге-Кутта, але меншого порядку, і використовується для одержання більш грубого розв'язку (з меншою точністю).
- `ode113` — використовує метод Адамса-Башфорта-Мілтона. Він може виявитися більш ефективним порівняно з методом ode45, особливо при високих точностях і при складності обчислення правих частин рівнянь. Метод багатокроковий, тому для початку розв'язання необхідно знати розв'язок у декількох початкових точках.
- `ode15s` — базується на методі чисельного диференціювання назад, відомого як метод Гіра. Метод є багатокроковим і у багатьох випадках є ефективнішим, ніж метод ode45.
- `ode23s` — використовує метод Розенброка другого порядку. Оскільки це однокроковий метод, він може бути більш ефективним порівняно з методом `ode15s` для випадків невисокої точності.

Усі перелічені вище функції викликаються однаковим чином. У найпростішому випадку це має такий вигляд:

```
[T,Y ]=odeXX('FileName',tspan,y0),
```

де `odeXX` — одна з функцій, перелічених вище; `'FileName'` — рядок, що містить ім'я файла (ім'я процедури) з описом правих частин системи; `tspan` — вектор, що визначає інтервал інтегрування `tspan =[t0 tfinal]`. Якщо вектор `tspan` має більше двох елементів, то функція `odeXX` видає розв'язок в усіх точках, перелічених у векторі `tspan`; `y0` — вектор початкових умов задачі; `T` — вектор-стовпець моментів часу; `Y` — матриця розв'язків. Кожен рядок матриці містить вектор розв'язків (усі y_i) для відповідного моменту часу. Кожна з функцій `odeXX` може містити четвертий і наступні аргументи:

`[T,Y ]=odeXX('F ',tspan,y0,options,p1,p2,...)`,

які задають додаткові параметри розв'язку (крок, похибка і т.п.)[2].

При написанні функції, що описує праву частину системи рівнянь, необхідно керуватися наступними правилами. Функція повинна містити не менше двох вхідних аргументів `t` і `y`, навіть якщо якийсь з них не використовується явно при обчисленні правої частини системи. Права частина системи, що обчислюється цією функцією, повинна утворювати вектор-стовпець. Будь-які додаткові параметри, які необхідно передавати функції, повинні бути наприкінці списку параметрів самої функції (після спеціального параметра `flag`).

2.1.5 Екстраполяція Річардсона. Метод Ромберга

Відомо, що інтеграл від функції чисельно дорівнює площі криволінійної трапеції, обмеженої графіком цієї функції і межами інтегрування $[a, b]$. Кожна i-та смужка розглядається як трапеція висотою h з довжинами основ $y_i \equiv y(a + ih)$ та $y_{i+1} \equiv y(a + (i + 1)h)$. Площа однієї такої смужки дорівнює $\triangle S = (y_i + y_{i+1})h/2$. Інтеграл функції дорівнює сумі всіх елементарних площин цих трапеціеподібних смужок

$$S \approx \sum_{i=0}^{N-1} \frac{y_i + y_{i+1}}{2} h. \tag{2.24}$$

Використання екстраполяції Річардсона при інтегруванні відомими методами дозволяє значно скоротити машинний час при незмінній то-

[2] див. документацію до MATLAB

чності результату (оскільки уточнення результату інтегрування не потребує додаткових обчислень функції). Застосування наведеної нижче методики до ітераційної формули трапецій складає відомий метод Ромберга[3].

Суть екстраполяції Річардсона полягає у такому. Виберемо деякий крок h і розрахуємо значення інтеграла S_h^0, наприклад, за формулою трапеції. Далі зменшимо крок h удвічі та отримаємо нове значення інтеграла $S_{h/2}^0$. Згідно з екстраполяцією Річардсона розраховане значення інтеграла може бути уточнене за формулою

$$S_{h/2}^1 = \frac{2^2 * S_{h/2}^0 - S_h^0}{2^2 - 1}. \tag{2.25}$$

Надрядковий індекс при S визначає порядок уточнення (номер ітерації). Далі крок знову зменшуємо удвічі, розраховуємо нове значення інтеграла $S_{h/4}^0$ і за отриманими даними уточнюємо інтеграл вже до другого порядку:

$$S_{h/4}^1 = \frac{2^2 * S_{h/4}^0 - S_{h/2}^0}{2^2 - 1}, \tag{2.26}$$

$$S_{h/4}^2 = \frac{2^4 * S_{h/4}^1 - S_{h/2}^1}{2^4 - 1}. \tag{2.27}$$

Точність на даній стадії обчислювальної процедури перевіряється умовою $|S_{h/4}^2 - S_{h/4}^1| < \varepsilon$, де ε — задана точність інтегрування. Якщо задана точність не досягнута, зменшуємо крок удвічі і виконуємо подальші розрахунки за рекурентною формулою

$$S_{n_k}^i = S_{n_k}^{i-1} + \frac{1}{2^{2i} - 1}\left(S_{n_k}^{i-1} - S_{n_{k-1}}^{i-1}\right) = \frac{2^{2i} S_{n_k}^{i-1} - S_{n_{k-1}}^{i-1}}{2^{2i} - 1}, \tag{2.28}$$

де $n_k = 2^{k+1}$ — число відрізків, на яке розбивається інтервал інтегрування $[a, b]$, $k = 0, 1...$

[3] Саме цей метод є одним з найбільш рекомендованих у бібліотеках стандартних програм багатьох фірм-розробників програмного забезпечення.

Згідно з методом Ромберга наближене значення інтеграла на кожній ітерації розраховується за ітераційною формулою трапецій (поділене на $(b-a)$):

$$S_{n_k} = \frac{1}{2} S_{n_{k-1}} + \frac{1}{n_k} \sum_{j=1}^{n_{k-1}} F\left(a + \frac{2j-1}{n_k}(b-a)\right). \qquad (2.29)$$

Отриманий результат уточнюється за формулою (2.28). Потім число n_k подвоюється. На кожній ітерації значення $S^i_{n_k}$ запам'ятовується в масиві `NewEst[i]` (див. програму), а $S^{i-1}_{n_k}$ — у масиві `OldEst[i]`.

Досягнення заданої точності при обчисленні інтеграла може контролюватися перевіркою виконання нерівності

$$|S^i_{n_k} - S^{i-1}_{n_k}| < \varepsilon, \qquad (2.30)$$

де ε — задана точність інтегрування.

Фрагмент програмної реалізації методу Ромберга наводиться у додатку.

2.2 Запитання для самоконтролю

1 Складіть рекурсивну процедуру знаходження лівої та центральної похідної будь-якого порядку.

2 Отримайте різницеву схему першого порядку точності для розв'язання диференціального рівняння.

3 Чому метод Ейлера не може бути застосований для розв'язання осциляторних рівнянь?

4 Яким чином можна збільшити точність різницевої схеми?

5 Складіть процедуру розв'язання диференціального рівняння за модифікованим методом Ейлера.

6 Складіть процедуру розв'язання диференціального рівняння першого порядку за методом з переступом.

7 Назвіть придатні схеми чисельного диференціювання для розв'язання осциляторних рівнянь.

8 Опишіть послідовність команд MATLAB для розв'язання систем ЗДУ.

9 У чому полягає суть екстраполяції Річардсона?

10 Чи можна застосувати екстраполяцію Річардсона до методу прямокутників?

2.3 Завдання для самостійної роботи

1 Матеріальна точка рухається із швидкістю $v(t) = 5 + 5t + 3t^2$. Чисельними методами визначте прискорення точки у момент часу $t = 3$ с і пройдений шлях за час від $t_1 = 3$ с до $t_2 = 5$ с, якщо рух є прямолінійним. Порівняйте результати з аналітичними розрахунками. Повторіть розрахунки при різних значеннях кроку за часом і порівняйте отримані результати [4].

2 Тіло масою m кидають під кутом α до горизонту із початковою швидкістю v_0. За наявності сили тертя траєкторія руху тіла описується формулами:

$$x(t) = \frac{mv_0 \cos\alpha}{r}\left(1 - e^{-rt/m}\right),$$

$$y(t) = \frac{m}{r}\left(\frac{mg}{r} + v_0 \sin\alpha\right)\left(1 - e^{-rt/m}\right) - \frac{mgt}{r},$$

де r — коефіцієнт тертя; g — прискорення вільного падіння.

Задаючи m, v_0, r та α, чисельними методами виконайте такі розрахунки:

• побудуйте залежність довжини траєкторії руху

$$L = \int_0^T \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt$$

[4] Задачі даного підрозділу частково взяті з робіт [3], [4]

від кута α;

• побудуйте залежність дальності S польоту від кута α;

• побудуйте залежність координати найвищої точки траєкторії від кута α.

3 Є пластинка товщиною h, яка обмежена кривою $y = x^2$ та прямою $y = 1$. Її густина є функцією координати y:

$$\rho(y) = \rho_0(1 + \alpha y),$$

де α — довільний коефіцієнт пропорційності, ρ_0 —константа. Визначте площу пластини та її масу.

4 Визначте координати центра мас пластини товщиною h та густиною ρ. Пластина обмежена прямими $x = 0$, $y = 0$, $y = 4 - x^2$.

5 Пластина товщиною h має форму кола радіусом R. Її густина зі збільшенням відстані до центра зменшується за законом

$$\rho(r) = \rho_0(1.5 - r/R).$$

Використовуючи чисельне інтегрування, визначте момент інерції пластини щодо осі, яка проходить через її центр і лежить в її площині.

6 Побудуйте криву залежності випромінювальної здатності абсолютно чорного тіла від частоти ω та температури T, яка описується формулою Планка: $f(\omega, T) = A\omega^3/(e^{B\omega/T} - 1)$, де A та B — сталі коефіцієнти. Побудуйте графік при різних T.

7 Напишіть програму, що обчислює частоту випромінювання, на яку припадає максимум випромінювальної здатності абсолютно чорного тіла. Виконавши розрахунки при різних T, переконайтесь у справедливості закону зсуву Віна: добуток абсолютної температури тіла T на довжину хвилі λ_m, яка відповідає максимуму випромінювальної здатності, є постійною величиною: $\lambda_m T = b$.

8 Методом чисельного інтегрування знайдіть інтегральну світність абсолютно чорного тіла R^*:

$$R^* = \int_0^{+\infty} \frac{A\omega^3}{e^{B\omega/T} - 1} d\omega.$$

Виконайте розрахунки для різних температур T і переконайтеся у справедливості закону Стефана-Больцмана: інтегральна світність абсолютно чорного тіла пропорційна четвертій степені його температури $R^* = \sigma T^4$.

9 За допомогою ідеальної безабераційної лінзи з поперечним розміром D одержують у її фокальній площині зображення точки, що знаходиться від лінзи на великій відстані (вилученої на нескінченність). Оскільки розмір D обмежений, дифракція світла на лінзі приводить до того, що у фокальній площині замість точкового зображення виходить розмита пляма. Розподіл інтенсивності світла у плямі описується виразом

$$I(\varphi) = I_0 \left(\frac{2J_1(x)}{x} \right)^2,$$

де I_0 — інтенсивність у центрі плями; $\varphi = \xi/F$ — кут між осьовою лінією лінзи і лінією, яка проведена із центра лінзи у точку ξ спостереження у фокальній площині (припускають, що точка-об'єкт лежить на осьовій лінії); F — фокусна відстань лінзи; $x = \frac{\pi D}{\lambda}\varphi$, λ — довжина хвилі світла. Функція $J_1(x)$ називається функцією Бесселя 1-го порядку і може бути виражена через своє інтегральне подання так:

$$J_1(x) = \frac{2}{\pi} \int_0^{\pi/2} \sin(x \sin t) \sin t dt.$$

Чисельними методами виконайте такі розрахунки:

• побудуйте графіки розподілу інтенсивності світла у фокальній площині лінзи;

• знайдіть значення кутів φ_{min} і φ_{max}, при яких спостерігаються мінімуми і максимуми розподілу світла. Відкладіть значення цих кутів на графіку, де по осі ординат — кут, а по осі абсцис — номер порядку дифракції;

• для обох випадків знайдіть розподіл енергії світла за порядками дифракції і побудуйте гістограми розподілу енергії для перших десяти порядків.

10 У фокальній площині круглої лінзи з діаметром D спостерігається зображення двох точок, що віддалені одна від одної на кутову відстань φ_0. Відповідно до попередньої задачі розподіл інтенсивності світла може бути записаний у вигляді

$$I(\varphi) = I_0 \left[\left(\frac{2J_1(x)}{x} \right)^2 + \left(\frac{2J_1(x - x_0)}{x - x_0} \right)^2 \right],$$

де $x_0 = \frac{\pi D}{\lambda} \varphi_0$,

Чисельними методами виконайте такі розрахунки:

• нарисуйте функцію $I(\varphi)$ і проаналізуйте її поводження залежно від кута φ_0. При яких значеннях φ_0 зображення точок зливаються і перестають спостерігатися роздільно?

• побудуйте функцію $V = (I_{max} - I_{min})/I_{max}$ залежно від кутової відстані φ_0 між точками. За критерієм Релея точки спостерігаються ще як роздільні, якщо максимум основної дифракційної плями від першої точки збігається з першим мінімумом інтенсивності для другої точки. Якому значенню V це відповідає?

• використовуючи отримані результати, оцініть мінімальні розміри об'єктів на землі, які ще можна спостерігати із супутника з висоти 200 км.

Розділ 3

Моделювання динамічних систем

У даному розділі розглядаються основні підходи до моделювання динамічних систем, рух яких описується звичайними диференціальними рівняннями. Матеріал даного розділу може бути застосований при дослідженні широкого кола явищ: рух заряджених частинок в електричних і магнітних полях, рух багатьох взаємодіючих тіл, перехідні процеси в електричних ланцюгах, кінетика хімічних реакцій, моделі розвитку економіки, динаміка біологічних популяцій тощо. Оскільки у більшості зазначених випадків отримати аналітичний розв'язок неможливо, на допомогу приходять наближені чисельні методи.

3.1 Постановка задачі

Є фізична система з одним ступенем вільності, що складається із інерційного елемента масою m, пружного елемента жорсткістю k та дисипативного елемента з коефіцієнтом опору r. Визначте відгук системи $x(t)$, а також її першу та другу похідні $\dot{x}$, $\ddot{x}$ на зовнішній вплив $F_x(t)$, якщо відомі початкові умови $x_0 = x(t = 0)$, $v_0 = v(t = 0)$.

3.2 Теоретичний матеріал

3.2.1 Побудова моделі

Величезна кількість процесів, які відбуваються у природі, описується диференціальними рівняннями, побудованими на основі класичних законів Ньютона. Наприклад, із другого закону Ньютона випливає лінійне неоднорідне диференціальне рівняння другого порядку

$$m\ddot{x} + r\dot{x} + kx = F_x(t).$$

Дане диференціальне рівняння описує широкий клас задач, що досліджуються у хімії, економіці, фізиці: кінетику хімічних реакцій, рух зв'я-

заного з пружиною тіла у в'язкому середовищі, проходження струму в електричному ланцюжку тощо.

Розглянемо, наприклад, лінійний осцилятор масою m, який здійснює коливання у в'язкому середовищі під дією зовнішньої сили $F_x(t)$. У цьому випадку рівняння руху має вигляд

$$\ddot{x} + \gamma\dot{x} + \omega^2 x = \frac{F_x(t)}{m},$$

де γ — коефіцієнт тертя; ω — власна частота коливань.

Процеси, які відбуваються в послідовному коливальному контурі, що складається з послідовно з'єднаних резистора R, конденсатора C та котушки індуктивності L, на який подана напруга $U(t)$, описуються рівнянням

$$L\ddot{q} + R\dot{q} + \frac{q}{C} = U(t),$$

де q — заряд, що проходить по ланцюгу; $i = \dot{q}$ — сила струму.

У паралельному коливальному контурі, що складається з резистора R, конденсатора C та котушки індуктивності L, з'єднаних паралельно і підключених до джерела струму $I = I(t)$, відбуваються процеси, описувані рівнянням

$$C\ddot{u} + \frac{\dot{u}}{R} + \frac{u}{L} = \dot{I}(t),$$

де u — напруга на резисторі, конденсаторі та індуктивності; $\dot{I}(t)$ — похідна від сили струму за часом.

Характер руху механічної системи залежить від її параметрів, початкових умов, до яких належать координати та швидкість у початковий момент часу, типу зовнішньої сили.

Більшість рівнянь руху не мають аналітичного розв'язку, а отже, виникає задача підбору відповідного чисельного методу для розв'язання досліджуваного рівняння. Отримане рівняння звичайно замінюють його різницевим аналогом, яке інтегрується за допомогою придатної різницевої схеми.

3.2.2 Спрощення моделі

У багатьох задачах буває корисно спростити модель шляхом введення знерозмірених змінних. Розглянемо задачу про рух математичного маятника — важеля маси m, підвішеного у полі сили тяжіння на невагомому стрижні довжини l [5]. Будемо вважати, що маятник рухається в одній площині. Нехай на маятник діють сила тертя $\overrightarrow{F} = r\overrightarrow{v}$ (r — коефіцієнт тертя) і зовнішня періодична сила $F(t) = F\cos\omega t$, спрямована горизонтально. Наглядним прикладом даної системи є маятник (із зарядженим важелем), поміщений у великий плоский конденсатор з вертикальними пластинами, до яких прикладена змінна напруга. Позначимо кут відхилення маятника від вертикального напрямку через x. Для кута відхилення нитки від вертикалі можна записати рівняння

$$ml\frac{d^2x}{dt^2} = -mg\sin x - rl\frac{dx}{dt} + F\cos x\cos\omega t. \tag{3.1}$$

За відсутності сили тертя і зовнішньої сили період малих коливань маятника дорівнює, як відомо, $2\pi\sqrt{l/g}$.

Уведемо замість часу t нову змінну τ. Відповідно до співвідношення $t = \tau\sqrt{l/g}$ змінна τ виявляється знерозміреною. Тоді рівняння (3.1) приводять до вигляду

$$\frac{d^2x}{d\tau^2} = -\sin x - R\frac{dx}{d\tau} + f\cos x\cos\Omega\tau, \tag{3.2}$$

де

$$R = \frac{r}{m}\sqrt{\frac{l}{g}}, \qquad f = \frac{F}{mg}, \qquad \Omega = \omega\sqrt{\frac{l}{g}} \tag{3.3}$$

— знерозмірені величини. Таке перетворення виявляє деякі закони подібності: залежність $x(\tau)$ (при заданих $x(0)$, $v(0)$ виявляється тією самою при різних значеннях m, l, g, r, F, ω, якщо однакові складені з них знерозмірені комбінації R, f, Ω. Цей факт дозволяє істотно скоротити об'єм повного дослідження задачі, оскільки досить розглядати різні значення трьох параметрів замість шести. Інакше кажучи, результати дослідження одного маятника можна перенести на інші простою зміною масштабів. Крім того, при чисельному визначенні розв'язку рівняння (3.2) ми не

будемо, як правило, мати справу з величинами, що відрізняються одна від іншої на багато порядків, у той час як для рівняння (3.1) це могло б статися при невдалому виборі одиниць виміру.

Перехід від рівняння (3.1) до (3.2) можна оформити й інакше. За одиницю довжини візьмемо довжину маятника, за одиницю маси — його масу, а одиницю часу виберемо так, щоб $g = 1$ (тобто такою, що дорівнює $\sqrt{l/g}$). Підставивши $l = m = g = 1$ у (3.1), ми одержимо рівняння вигляду (3.2). Щоб отримати співвідношення (3.3), потрібно побудувати з l, m, g множники необхідної розмірності. Скажімо, сила має розмірність mg, тому запишемо $F = mgf$. Ця рівність справедлива в прийнятих одиницях і справедлива при переході до будь-яких інших одиниць, якщо вважати f знерозміреною величиною, оскільки розмірності його лівої та правої частин тоді однакові. Аналогічно можуть бути отримані інші співвідношення (3.3).

У задачі про рух частинки маси m у полі $U = -k/R^2$ мова може йти як про рух планети навколо Сонця, так і про рух електрона навколо атомного ядра [5]. У першому випадку $k = \gamma mM$, де γ — гравітаційна стала, m — маса планети, M — маса Сонця; у другому m — маса електрона, $k = |qQ|$, де q — заряд електрона, Q — заряд ядра. Рівняння руху має вигляд

$$m\frac{d^2\overrightarrow{R}}{dt^2} = -k\frac{\overrightarrow{R}}{R^3}.$$

Уведемо як одиницю довжини характерну довжину R_0 (для астрономічної задачі, наприклад, $R_0 = 10^8$ км, для атомної $R_0 = 10^{-8}$ см). Тоді природно як одиницю часу вибрати $t_0 = \sqrt{R_0^3 m/k}$ (для руху планети t_0 виявиться порядку року, для руху електрона — порядку періоду обертання електрона в атомі). Знерозмірена довжина r і час τ визначаються рівностями $t = t_0\tau$, $R = R_0 r$, а рівняння руху набуває вигляду

$$\frac{d^2\overrightarrow{r}}{d\tau^2} = -\frac{\overrightarrow{r}}{r^3}.$$

Ще один приклад — задачі релятивістської фізики частинок, де звичайно беруть швидкість світла $c = 1$. При цьому швидкість частинки стає

[5] У рамках Борівської теорії.

знерозміреною величиною v/c. Маса ж залишається розмірною величиною, але її розмірність не відрізняється від розмірності енергії. Наприклад, маса електрона $m = 511$ кеВ.

3.2.3 Оцінка результатів моделювання. Можливі помилки

Відправним пунктом чисельного моделювання є розроблення ідеалізованої математичної моделі фізичної системи та підбір придатного алгоритму для реалізації даної моделі на комп'ютері. При цьому виникають такі запитання:

- наскільки реально математична модель описує фізичне явище?
- як переконатися у правильності чисельного розв'язку рівнянь, які описують модель?

Щодо відповіді на перше запитання необхідно зазначити: більшість задач даного посібника являють собою сильно ідеалізовані моделі фізичних процесів, що, однак, дозволяє скласти досить повне уявлення про основні фундаментальні фізичні закони, які лежать в їх основі. Правильно застосовуючи ці закони та вводячи нові параметри, можна деталізувати будь-яку модель реального фізичного процесу. Наприклад, у другій частині посібника ми розглянемо модель руху тіла у центрально-симетричному полі, що дозволить, наприклад, побудувати цілком розумні траєкторії обертання супутника навколо планети. Однак при цьому вважається, що центр притягання (планета) є матеріальною точкою, а рух відбувається у площині. З іншого боку, відомо, що орбіта штучного супутника Землі є істотно тривимірною і при її розрахунках необхідно (для отримання високої точності) враховувати форму земної кулі.

Відповідь на друге запитання частково розібрана у різних розділах даного посібника, де обговорюються питання придатності того чи іншого чисельного методу. Взагалі універсального алгоритму, що дозволяє стовідсотково переконатися в тому, що отриманий розв'язок є коректним, не існує. Перелічимо кілька способів перевірки коректності обчислень:

Оберненість задачі у часі. Для цього, розрахувавши поводження моделі на певному часовому відрізку, варто взяти за початкові умови поточні значення змінних і змінити знаки векторів швидкостей. При цьому автоматично зміняться знаки в усіх силах, що залежать від швидкості.

Якщо в розрахунках не відбувається накопичення помилки, система повинна повернутися у вихідний стан через той самий часовий інтервал.

Перевірка законів збереження (енергії, імпульсу) у задачах, де сили, що діють на тіло, консервативні, тобто не викликають зміни повної енергії системи. Наприклад, рух тіла у полі сили тяжіння (за відсутності тертя), рух тіла у центральному полі, рух заряду в стаціонарному магнітному полі тощо.

Перевірка розв'язку на граничний випадок. Для цього певним параметрам задачі надають таких значень, при яких поводження всієї системи стає цілком передбачуваним. Природно, що чисельний алгоритм повинен описувати таку ситуацію. Наприклад, можна виключити силу тертя в задачі про політ снаряда. Можна також спростити модель, одержати аналітичний розв'язок і порівняти його з чисельним розрахунком.

Перевірка збіжності розв'язку до певної границі при зменшенні кроку за часом у декілька разів. Обчислення повинні збігатися з точністю до певної кількості перших десяткових знаків залежно від застосовуваного методу.

Невиконання будь-якої з цих умов може бути наслідком погрішностей, що виникають на кожному етапі комп'ютерного експерименту. Вони обумовлені такими причинами:

а) математичний опис задачі є спрощеним;

б) недостатньо точно задані вихідні дані, що є, як правило, результатом проведених експериментів;

в) будь-який чисельний метод є наближеним;

г) у процесі розв'язання задач на ЕОМ при введенні вихідних даних або виконанні арифметичних операцій виконуються округлення.

Похибки, обумовлені пунктами (а) і (б), називаються такими, що не можуть бути усуненими. При переході від математичної моделі до чисельного методу (пункт (в)) виникає похибка методу. Побудова чисельного методу для математичної моделі складається з двох етапів — формулювання дискретної моделі і розроблення обчислювального алгоритму, що дозволяє відшукати розв'язок дискретної задачі. Відповідно похибка методу підрозділяється на похибку дискретизації, або похибку апро-

ксимації, і на обчислювальну похибку, або похибку округлення, яка виникає у процесі розв'язання (пункт (г)). Зазначимо, що **аналіз похибки чисельного результату повинен бути обов'язковим етапом розв'язання будь-якої практичної задачі**. Повна похибка є результатом складної взаємодії всіх видів похибок і відповідь на запитання, яка з трьох похибок є домінуючою, не може бути однозначною. Типовою є ситуація, коли похибка, що не може бути усуненою, значно перевищує похибку методу, а обчислювальною похибкою можна знехтувати порівняно з похибкою методу. У загальному випадку треба намагатися, щоб похибки мали один і той самий порядок. У зв'язку з цим при аналізі похибок виникають дві задачі: пряма — знаючи точність вихідних даних, оцінити точність отриманого результату, і зворотна — визначити точність, з якою необхідно задавати вихідну інформацію, щоб забезпечити необхідну точність розв'язку. У будь-якому разі аналіз похибок вимагає від дослідника певних знань про представлення чисел в ЕОМ. Розглянемо це питання більш детально.

Двійкові числа. Будь-яке число в ЕОМ подається у вигляді двійкового коду: набору цифр із нулів та одиниць. При цьому говорять, що ЕОМ використовує арифметику з двійковою системою числення. Таким чином будь-яке вхідне число (як правило, з основою 10) ЕОМ приводить до основи 2, далі, з використанням двійкової арифметики, виконує з ним арифметичні операції, а при виведенні на екран результат знову перетворює до основи 10. Наведемо простий приклад [2]. Комп'ютер, що виконує обчислення з точністю 8 десяткових знаків, дасть відповідь

$$\sum_{k=1}^{100000} 0.1 = 9998.5566, \tag{3.4}$$

хоча точна відповідь дорівнює 10000. Маємо очевидну похибку. Для пояснення результату розглянемо машинну арифметику більш детально.

Нагадаємо найпростіший алгоритм переведення десяткового числа у двійкове (точніше, найпростіший для людини, комп'ютер виконує переведення інакше) — ділимо число на 2 та беремо остачу. Результат ділимо знову на 2 і т.д. Остачі у зворотному порядку формують двійкове представлення числа (рис. 3.1). Для переведення дробової частини зазвичай

використовують такий алгоритм: множимо число на 2 та виділяємо цілу частину результату. Потім дробову частину результату знову множимо на 2 і т.д. Цілі частини формують двійкову форму. Як наочний приклад представимо число 19.375_{10} у двійковій формі (нижній індекс позначає систему числення):

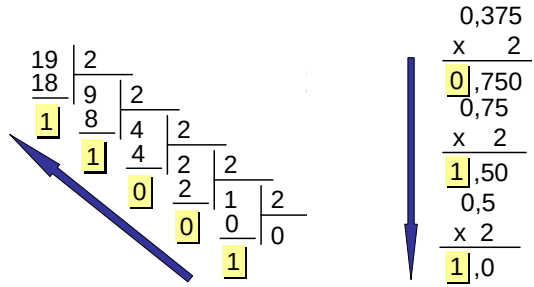

Рисунок 3.1 — Переведення дійсного числа у двійкову форму

Таким чином, $19.375_{10} = 10011.011_2$. Зворотне переведення виконується шляхом помноження кожної цифри двійкового числа на 2^n (n — порядковий номер цифри у числі):

$$10011.011_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + \\ + 1 \times 2^{-2} + 1 \times 2^{-3} = 16 + 2 + 1 + 0.25 + 0.125 = 19.375_{10}.$$

Відмітимо, що існує величезна кількість дійсних чисел, для двійкового представлення яких необхідна нескінченна послідовність цифр. Наприклад, $0.7_{10} = 0.1011001100110...{}_2 \equiv 0.1(0110)_2$. Даний двійковий дріб є періодичним, у ньому групи цифр 0110 повторюються (такі групи зазвичай записують у круглих дужках). Якщо раціональне число є еквівалентним нескінченному двійковому виразу, для більш зручного його представлення можна використовувати зсув цифр. Наприклад, помноження числа $0.000(11000)_2$ на 2^3 є еквівалентним зсуву двійкової точки на 3 позиції ліворуч, тобто $0.000(11000)_2 \times 2^3 \equiv 0.(11000)_2$.

Машинні числа. У комп'ютері для дійсних чисел використовується нормоване (нормалізоване) двійкове представлення з плаваючою точкою.

Це означає, що будь-яке число зберігається у формі $X = M \times 2^p$, де $1/2 \le M < 1$ — мантиса, яка складається із обмеженої кількості цифр (залежно від типу змінної згідно з прийнятими стандартами), p — показник степеня. Звідси випливає, що M завжди матиме вигляд $M = (0.1...)_2$. На простому прикладі розглянемо втрату точності обчислень у випадку, коли під мантису M у пам'яті ЕОМ відводиться лише чотири розряди, а показник степеня $p \in \{-3, -2, -1, 0, 1, 2, 3, 4\}$ [2]. У даному випадку існує 8×8 комбінацій цифр мантиси та показника степеня; кожна з цих комбінацій відповідає десятковому числу (табл. 3.1).

Таблиця 3.2 — Десятковий еквівалент набору двійкових чисел з мантисою із 4 розрядів та відповідним показником степеня

Мантиса	Показник степеня							
	p=−3	p=−2	p=−1	p=0	p=1	p=2	p=3	p=4
M=0.1000$_2$	0.0625	0.125	0.25	0.5	1	2	4	8
M=0.1001$_2$	0.0703125	0.140625	0.28125	0.5625	1.125	2.25	4.5	9
M=0.1010$_2$	0.078125	0.15625	0.3125	0.625	1.25	2.5	5	10
M=0.1011$_2$	0.0859375	0.171875	0.34375	0.6875	1.375	2.75	5.5	11
M=0.1100$_2$	0.09375	0.1875	0.375	0.75	1.5	3	6	12
M=0.1101$_2$	0.1015625	0.203125	0.40625	0.8125	1.625	3.25	6.5	13
M=0.1110$_2$	0.109375	0.21875	0.4375	0.875	1.75	3.5	7	14
M=0.1111$_2$	0.1171875	0.234375	0.46875	0.9375	1.875	3.75	7.5	15

Використовуючи дану таблицю, розрахуємо величину $(1/10+1/5)+1/6$. У двійковій системі числення кожне з цих чисел (відповідно до табл. 3.1) матиме вигляд:

$$\frac{1}{10} \approx 0.1101_2 \times 2^{-3}, \qquad \frac{1}{5} \approx 0.1101_2 \times 2^{-2}, \qquad \frac{1}{6} \approx 0.1011_2 \times 2^{-2}.$$

Представляючи перше двійкове число у вигляді 0.01101×2^{-2} та використовуючи двійкову арифметику, знаходимо суму перших двох чисел:

$$0.01101_2 \times 2^{-2} + 0.1101_2 \times 2^{-2} = 1.00111_2 \times 2^{-2}.$$

Нормалізуючи результат, маємо $0.100111_2 \times 2^{-1}$. Оскільки під мантису відводиться лише 4 розряди, ЕОМ округляє результат до $0.1010_2 \times 2^{-1}$. Далі виконується додавання

$$0.1010_2 \times 2^{-1} + 0.1011_2 \times 2^{-2} = 0.1010_2 \times 2^{-1} + 0.01011_2 \times 2^{-1} = \\ = 0.11111 \times 2^{-1}.$$

Далі результат округляється до $0.1000_2 \times 2^0$.

Останній результат є комп'ютерним розв'язком задачі. Порівнюючи отриманий розв'язок з точним значенням, отримаємо похибку обчислень

$$\frac{7}{15} - 0.1000_2 \times 2^0 \approx 0.466667 - 0.500000 \approx 0.033333.$$

Стандарт IEEE 754. Згідно зі стандартом IEEE 754 (найбільш поширений формат представлення даних) двійкове число з точкою у пам'яті (та регістрах процесора) ЕОМ представляється набором бітів. Перший біт ліворуч визначає знак — плюс (0) або мінус (1), далі йде група бітів, що визначає показник степеня, потім йдуть біти, що відводяться для зберігання мантиси. Різні типи даних відрізняються розрядністю (кількістю біт у пам'яті) мантиси та показника степеня. Числам одинарної точності (single precision)[6] відповідає показник степеня із 8 біт, мантисі — із 23 біт[7]. Разом зі знаковим бітом — 32 біти під кожне число. Зазначимо, що показник степеня (ціле число) може бути як додатним, так і від'ємним. Для спрощення роботи з від'ємними числами він зазвичай зберігається у вигляді беззнакового цілого із зсувом. А для одержання реального значення показника степеня від записаного значення віднімається деяка константа (127 для одинарної точності).

Знайдемо точність, яку забезпечує даний тип даних у десяткових знаках. Оскільки для точності "у знаках" масштаб не має значення, помножимо мантису на 2^{24}. Показник степеня на точність не впливає, так що

[6]Числам одинарної точності звичайно відповідає тип даних float (компілятор C/C++).

[7]Точніше 24 біти, оскільки перший біт мантиси за замовчуванням завжди дорівнює 1 і під його зберігання пам'ять спеціально не відводиться.

його можна залишити як є. Ціле число, яке вийшло у результаті, мантиса представляє точно. Для його представлення в десятковому вигляді потрібно: $\log_{10} 2^{24} = 24 * \log_{10} 2 = 7.2$ десяткових знака. Це означає, що похибка двійкового представлення числа не перевищує половини сьомого десяткового знака.

Як приклад, представимо в цьому форматі число $1234,5$ [6]. Переводимо його у двійкову форму:

$$1234,5 = 2^{10} + 128 + 64 + 16 + 2 + 1/2 = 10011010010,1_2.$$

Результат нормалізуємо і представляємо таким чином, щоб у цілій частині опинилася 1 (особливість даного формату):

$$10011010010,1_2 = 1,00110100101_2 \times 2^{10}.$$

Оскільки показник степеня зберігається зі зсувом 127_{10}, маємо

$$1,00110100101_2 \times 2^{137_{10}-127_{10}} = 1,00110100101_2 \times 2^{10001001_2-01111111_2}$$

Отже, число $1234,5$ буде представлено у пам'яті ЕОМ таким чином:

Знак	Показник	Мантиса
0	1000 1001	(1)001 1010 0101 0000 0000 0000

Повертаючись до вихідної задачі (3.4), на основі представленого вище матеріалу пояснимо результат. Число 0.1 у двійковій формі представляється нескінченною кількістю цифр:

$$\frac{1}{10} = 0.0(0011)_2. \tag{3.5}$$

Коли мантиса має 24 двійкових розряди, відбувається усікання і комп'ютер використовує внутрішнє наближення:

$$\frac{1}{10} \approx 0.11001100110011001100110_2 \times 2^{-3}. \tag{3.6}$$

У наближенні (3.6) похибка дорівнює різниці між (3.5) та (3.6):

$$0.(1100)_2 \times 2^{-27} \approx 5.96 \times 10^{-9}. \tag{3.7}$$

Отже, кожне машинне сумування у (3.4) виконується із похибкою (3.7). Відповідно загальна похибка повинна становити $100000 \times (5.96 \times 10^{-9}) = 5.96 \times 10^{-4}$. Згідно з (3.4) реальна похибка $10000 - 9998.5566 \approx 1.45$. Отже, існують інші, більш серйозні похибки. Як буде показано нижче, вони пов'язані з округленням часткової суми у процесі додавання чергового числа 0.1.

Діапазон та точність. Найменше нормалізоване число типу single precision буде зберігатися у пам'яті у вигляді

Знак	Показник	Мантиса
0	0000 0001	(1)000 0000 0000 0000 0000 0000

Послідовно виконуючи зворотну процедуру переведення у десяткову форму, маємо $1.0_2 \times 2^{00000001_2 - 01111111_2} = 1.0_2 \times 2^{-126_{10}} = 1.1755 \times 10^{-38}$.

Відповідно до даного стандарту, найбільше нормалізоване число представляється у вигляді

Знак	Показник	Мантиса
0	1111 1110	(1)111 1111 1111 1111 1111 1111

Десяткове представлення числа матиме вигляд 3.4028×10^{38}.

Підсумовуючи вищенаведене, відмітимо, що **не слід плутати мінімальне значення числа певного типу та його точність**. Як було показано, одинарна точність представляється не більше ніж 8 десятковими розрядами, а показник степеня забезпечує лише зсув відносно десяткової точки на відповідну кількість розрядів ліворуч або праворуч. Наприклад, намагаючись зберегти число 123456789 у змінній типу float (реалізація на C++), отримуємо результат 1.234567×10^8. Отже, система відкинула два останні розряди, замінивши їх нулями. При реалізації на MATLAB, отримаємо відповідь 123456792, тобто відповідь з округленням. Спроба зберегти, наприклад, число 0.123456789 приводить до результату 0.123457 (C++) та 0.1234568 (MATLAB), а зберігаючи $123456789 \times 10^{-10}$ та 123456789×10^{10}, маємо відповідно 0.012346 та 1.234568×10^{18} (C++). Зазначимо, що 7-8-й десятковий розряд для чисел одинарної точності вже є результатом округлення. Наприклад, результатом виконання команди `a=2/3` є $a = 0.6666667$ (MATLAB) та $a = 0.666667$ (C++).

Виконання математичних операцій. При виконанні додавання двох додатних чисел з плаваючою точкою відбуваються такі дії:

1. Вирівнювання порядків.

Визначається число з меншим порядком. Потім послідовно його порядок збільшується на одиницю, а мантиса поділяється на 2, поки порядки двох чисел не зрівняються. Апаратно ділення на 2 відповідає зсуву двійкового коду мантиси праворуч (ця операція виконується набагато швидше, ніж операція ділення). При зсуві праві розряди губляться, через це може відбутися втрата точності.

2. Додавання мантис.

3. Нормалізація.

Якщо мантиса результату набуває значення 2 або більше, порядок збільшується на одиницю, а мантиса поділяється на 2. У результаті мантиса попадає в інтервал [1, 2). При цьому можлива втрата точності, а також переповнення, коли порядок перевищує максимально можливу величину.

Віднімання виконується аналогічним чином.

При множенні порядки додаються, а мантиси перемножуються як цілі числа, після чого праві розряди результату відкидаються.

Висновок: дії з числами (з плаваючою точкою) через похибки округлення лише приблизно відображають арифметику реальних дійсних чисел. Так, якщо до великого числа додати дуже маленьке, то воно не зміниться, оскільки при вирівнюванні порядків усі значущі біти мантиси меншого числа можуть вийти за межі розрядної сітки. У результаті воно буде дорівнювати 0. Наприклад, якщо до числа 0.1234567 додати число, що є меншим за модулем половини останнього розряду (тобто менше за 0.00000005), у результаті знову отримаємо те ж саме число 0.1234567. Тобто **величина похибки округлення визначається кількістю розрядів мантиси**. Розглянемо ще один корисний приклад (фрагмент програми на C++):

```
...
 float f = 200000000.0f; // 200 мільйонів
 printf("%f\n", f);
 for (int i = 0; i < 1000; i++)
```

```
   f += 1;
 printf("%f\n", f);
 f += 1000;
 printf("%f\n", f);
...
```

Результат роботи:

```
200000000.0
200000000.0
200001000.0
```

Результат легко пояснити. Для 200 мільйонів восьмий знак — це вже не одиниці, а десятки. Тому додаючи 1, одержуємо 200000001, округляючи до восьми знаків, одержуємо знову 200 мільйонів. А якщо відразу додати 1000 — результат є коректним. Якщо в прикладі замінити `float` на `double`, така ситуація не виникне — точність double 15-16 десяткових знаків.

Висновок: виконання операцій над дійсними числами починається і закінчується вирівнюванням порядків. Якщо порядки є різними — погрішність зростає, що приводить до втрати точності. **По можливості необхідно уникати працювати з числами, порядки яких відрізняються на величину, що є близькою до довжини розрядної сітки, а також віднімання близьких за значенням чисел. Додавати великі і маленькі числа потрібно обережно. Наприклад, якщо потрібно з великою точністю скласти багато чисел, то починати слід із найменших, і додавати не всі підряд, а групувати числа так, щоб увесь час підсумовувати значення приблизно одного порядку**. Наприклад, розглянемо послідовне сумування чисел одинарної точності:

$$((((12345678 + 0.1) + 0.2) + 0.3) + 0.4) + 0.5.$$

Результатом знову буде число 12345678. Змінимо порядок сумування:

$$12345678 + (0.1 + 0.2 + 0.3 + 0.4 + 0.5).$$

Результат — 12345680.

Округлення при розрахунках часто можуть призвести до помилок при виконанні порівнянь чисел, що є теоретично (математично) однаковими. Для початку ще раз повторимо висновок про різні шляхи і різні результати: математично-однакові числа, отримані двома різними способами, на практиці можуть виявитися різними — через відмінності в послідовності виконання округлень [6]. По-перше, потрібно чітко розуміти причину: різні округлення, викликані різною послідовністю операцій. Ніякого випадкового шуму, ймовірностей або невизначеності. Числа, отримані одним способом, гарантовано будуть однаковими: в однакових умовах детермінований алгоритм (додавання, множення і т.п.) повертає однакові результати. Наприклад, у випадку

```
...
float a = 1.0/3.0;
float b = a; // ніяких округлень
if (a == b)
 {
  ... // команда 1
 }
else
 {
  ...  // команда 2
 }
...
```

завжди виконується набір команд, позначених `команда 1`. Розглянемо інший випадок:

```
...
float a = sin(pi); // pi - константа "пі"
float b = sin(2*pi);
...
```

У даному випадку виконається набір команд, позначених `команда 2`, хоча, як відомо, $\sin(\pi) = \sin(2\pi)$. Таким чином, якщо числа отримані різними шляхами, більш коректною з точки зору програмування є така реалізація:

```
...
eps = 0,000001;
...
if (abs(a-b) < eps*max(abs(a),abs(b)))
{
...
```

Висновок. При створенні власних програм слід уникати типу float (C++) і завжди користуватися типом double. До того ж процесор сконструйований для роботи з восьмибайтовими дійсними числами, а при роботі з чотирибайтовими він у будь-якому разі спочатку приводить їх до восьмибайтового типу.

Погано обумовлені задачі. Погана обумовленість задачі означає, що навіть незначні похибки у вихідних даних приводять до великої похибки у результаті або взагалі до неправильного результату. Наприклад, розглянемо систему

$$\begin{cases} 3x - 7.0001y = 0.9998, \\ 3x - 7y = 1. \end{cases} \tag{3.8}$$

Розв'язком є $x = 5$, $y = 2$. Розглянемо іншу систему, що є близькою до системи (3.9):

$$\begin{cases} 3x - 7.0001y = 1, \\ 3x - 7y = 1. \end{cases} \tag{3.9}$$

Можна переконатися, що ця система має розв'язок $x = 1/3$, $y = 0$. Отже, зміна вихідних даних на 0.0002 приводить до сильної зміни розв'язку системи.

Задачі теорії хаосу. Необхідно мати на увазі, що в деяких класах задач (наприклад, задачі теорії динамічного хаосу) накопичення машинних похибок округлення при обчисленнях здатне приводити до сильних змін розв'язку. Це є принциповою особливістю таких задач, що не залежить від вибору чисельного методу. Як приклад розглянемо просту ітераційну модель $x^{n+1} = a - (x^n)^2$, де n — момент часу. Задамо початкові умови $a = 2$, $x^0 = 1.5$, кількість ітерацій — 200. Складемо простішу програму

```
float a;
float x;
double y;
...
a=2;
x=1.5;
y=1.5;
for(n=0,n<200,n++)
 {
  x=a-x*x;
  y=a-y*y;
 }
...
```

У результаті роботи програми після 200 ітерацій ми отримаємо неочікуваний результат: $x^{199} \approx -0.395$, $y^{199} \approx -1.819$. Результат легко пояснити: змінимо початкову умову $x^0 = 1.50001$. Знову запускаємо програму і отримуємо $x^{199} \approx -1.603$. Відповідь очевидна: розбіжність у 5-му знаку після точки привела до значних змін результату. Аналогічно, при використанні типу `float` машина, відкинувши (точніше округливши) остачу, врахує лише 7-8 знаків після точки. Тип `double` дозволяє врахувати 15-16 знаків. Тобто відкинутий "хвіст" для даної задачі є принципово важливим! Змінюючи тип на `long double`, отримуємо інший ("більш точний") результат [8]. Але і це не буде точною відповіддю. Точний результат отримати (чисельно) взагалі неможливо!

Відповідно перевірка за допомогою обернення часу тут не може бути застосована і варто скористатися іншими прийомами. До таких задач відносять, наприклад, розрахунок руху планети у системі подвійної зірки. Легко переконатися, що дуже малі відхилення у початкових умовах приводять до радикальної розбіжності траєкторій, хоча повна енергія системи при цьому може зберігатися з великою точністю.

Таким чином, у випадку погано обумовленої задачі найкраще буде подумати над іншим способом представлення моделі, вибрати інший метод або змінити алгоритм (часто це можливо).

[8] Тип long double дозволяє оперувати числами з точністю приблизно 19-20 розрядів.

Як висновок до цього підрозділу наведемо програмну заяву в передмові до своєї роботи відомих спеціалістів у сфері математичного моделювання О. М. Білоцерківського та В. В. Щеннікова [7]: "Бурное развитие вычислительной техники, которое особенно ярко проявилось в последние 10-15 лет, с особой остротой поставило проблему создания принципиально новой технологии решения задач на ЭВМ... Исторически сложилось так, что проблемы численного моделирования (в это понятие мы включаем собственно математическое моделирование, сопряженное с численным экспериментом), будучи заметно продвинуты еще в "домашинный" период и развиваясь опережающими темпами в последующие периоды, оказались наиболее консервативной компонентой современной математической технологии решения задач на ЭВМ. Прибегая, может быть, к излишней с точки зрения математиков образности изложения, можно охарактеризовать сложившуюся ситуацию двумя устойчивыми тенденциями:

- увеличение сложности математических моделей;

- построение очень изощренных математических методов.

И та и другая тенденция неизбежно приводят к технологическому тупику, поскольку создают практически трудно преодолимые сложности в решении задачи создания программно-аппаратных средств поддержки функционирования всей технологической цепочки... Не претендуя на глубину и значимость аналогии, мы отваживаемся утверждать, что ситуация, складывающаяся в современном численном моделировании, схожа с ситуацией, наблюдавшейся в механике перед появлением основных идей и концепций квантовой механики".

У даній роботі автори також акцентують увагу на феномені накопичення похибок округлень при чисельній реалізації алгоритмів, що включають до 10^{12} операцій, а також відсутності реальних засобів оцінки похибки розв'язків, зокрема, еволюційних задач. За їх думкою: "... вполне обоснованным является следующее заключение: априори любая эволюционная задача на больших временах является численно (или вычислительно) некорректной в смысле отсутствия практически значимого решения. ... В случае же, если отсутствует априорная или апостериорная информация о погрешности приближенного решения, нельзя говорить

о существовании решения. Этот вывод вполне согласуется с теоремой А. Н. Тихонова, утверждающей, что задача с данными об операторе и правой части не имеет решения во множестве приближенных чисел".

Відомо, яку велику увагу приділяв методології математичного моделювання Н. Н. Яненко (див. [8]). Висунуту ним концепцію подолання зазначеної кризи пояснює О. М. Білоцерківський [9]: "Исследование разностных схем, аппроксимирующих различные классы уравнений математической физики, приводит Н. Н. Яненко к расширению понятия схемы. Впервые он начинает рассматривать разностную схему как самостоятельный объект исследования, как математическую модель, адекватную той или иной физической модели. Это фундаментальное положение основано на глубоком понимании основ дифференциального и интегрального исчисления. Действительно, физико-математические модели, описываемые дифференциальными или интегральными уравнениями, получают из дискретных моделей путем осреднения и предельного перехода по тем или иным параметрам. Это имеет место, например, в модели сплошной среды, где для достаточно большого числа элементов в единице объема путем осреднения и предельного перехода по объему приходят к понятию сплошной среды. В этой связи можно трактовать разностную схему как самостоятельную математическую модель, обладающую теми или иными свойствами".

3.3 Моделювання динамічних систем з використанням пакета SIMULINK

У цьому підрозділі на простому прикладі наводяться основні принципи роботи зі спеціальними програмами, що орієнтовані на спрощення процесу моделювання динамічних та інженерних систем. Як приклад такої програми вибрано модуль SIMULINK із пакета MATLAB. Більш детальна інформація про можливості даної програми наводиться, наприклад, у [10]- [12].

Завантажити програму можна шляхом уведення команди `simulink` у вікні **Command Window**. При цьому відкривається вікно **Simulink Library**. Робота у програмі починається зі створення моделі $File \rightarrow$ $\rightarrow New \rightarrow Model$. При цьому відкриється пусте вікно моделі. Модель у

програмі SIMULINK створюється шляхом копіювання елементів-блоків із різних бібліотек у вікно моделі.

Розглянемо дану процедуру на прикладі моделі, що описується простим диференціальним рівнянням $\ddot{x} + 2\dot{x} + 7x = 0$. Припустимо, що система знаходиться під впливом випадкової зовнішньої сили $\xi(t)$. Результат конструювання зазначеної моделі у вікні **Model** показаний на рис. 3.2. Для конструювання зазначеної схеми виконаємо наступну по-

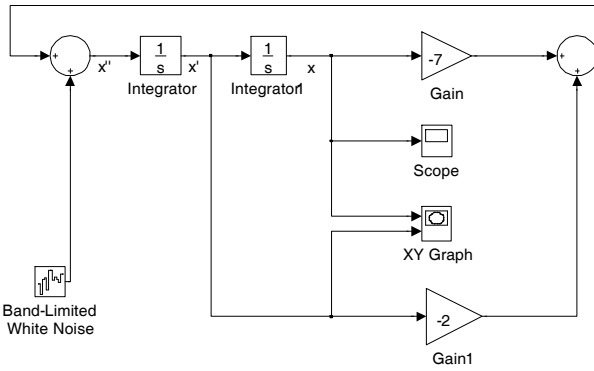

Рисунок 3.2 — Модель для рівняння $\ddot{x} = -2\dot{x} - 7x + \xi(t)$ у програмі SIMULINK.

слідовність дій. Оскільки змінна x є неперервною, спочатку ми відкриємо бібліотеку **Continious** і перетягнемо за допомогою миші дві копії блока INTEGRATOR у вікно моделі. Використовуючи ліву клавішу миші, з'єднаємо вихідний порт блока INTEGRATOR із вхідним портом блока INTEGRATOR1. Подвійним клацанням миші на стрілці можна активізувати текстове поле, що належить стрілці, та ввести необхідні пояснення, наприклад *x'*. Аналогічним чином додаємо стрілку, що входить у блок INTEGRATOR (представляє $\ddot{x}$), та стрілку, що виходить із блока INTEGRATOR1 (представляє x). Ідея полягає у тому, що x можна отримати із $\dot{x}$ (шляхом інтегрування), а $\dot{x}$ із $\ddot{x}$. Далі необхідно додати інші блоки, так щоб пов'язати $\ddot{x}$ (вхід блока INTEGRATOR) з x та $\dot{x}$ відповідно до рівняння $\ddot{x} = -2\dot{x} - 7x$. З цією метою включаємо два блока

GAIN (збільшення), що виконують множення на константу, та один блок SUM (сума). Зазначені блоки знаходяться у бібліотеці **Math Operations**. Далі подвійним клацанням миші на кожному з блоків GAIN змінимо їх властивості (уводимо константи множення відповідно -7 та -2). Далі з'єднуємо блоки INTEGRATOR1 та GAIN. З'єднання блоків зручно виконувати таким чином: виділяємо блок INTEGRATOR1, утримуючи клавішу *Ctrl*, клацаємо мишею на блоці GAIN. Аналогічним чином з'єднуємо вихід блока INTEGRATOR із блоком GAIN1. Тут виникає проблема: оскільки вихід блока INTEGRATOR приєднаний до наступного блока, потрібно створити розгалужену лінію. Для цього необхідно помістити вказівник миші на середину стрілки, що з'єднує два блока Integrator, та, утримуючи клавішу *Ctrl* та ліву клавішу миші, перемістити вказівник миші до входу блока GAIN1. Для перегляду результатів застосуємо два блоки: SCOPE (із бібліотеки **Sinks**) — для перегляду залежності $x(t)$, XY GRAPH — для перегляду фазового портрета $\dot{x}(x)$. Розташуємо та приєднаємо зазначені блоки, як показано на рис. 3.2. Далі

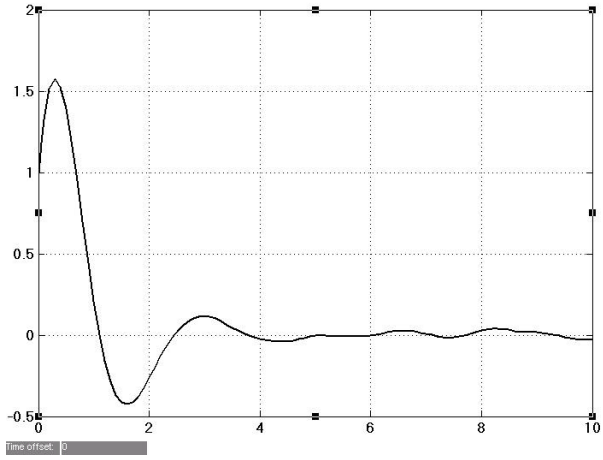

Рисунок 3.3 — Результат симуляції моделі для рівняння $\ddot{x} = -2\dot{x} - 7x + \xi(t)$ у програмі SIMULINK. Графік залежності $x(t)$

приєднаємо виходи блоків GAIN та GAIN1 до блока SUM, вихід якого, у свою чергу, подамо на вхід блока INTEGRATOR. Перемістимо блок BAND-LIMITED WHITE NOISE із бібліотеки **Sources** у вікно моделі і за допомогою блока SUM з'єднаємо із блоком INTEGRATOR. Початкові умови задаються шляхом зміни властивостей блоків INTEGRATOR та INTEGRATOR1. Для цього подвійним кліком на блоці INTEGRATOR викликаємо вікно властивостей та вводимо значення 4 (тобто $\dot{x}(0) = 4$). Аналогічно змінюємо властивості блоків INTEGRATOR1 ($x(0) = 1$) та BAND-LIMITED WHITE NOISE. Далі подвійним кліком миші активізуємо вікно SCOPE і запускаємо процес симуляції (*Start* → *Simulation*). Отримані результати подано на рис. 3.3. З рисунка бачимо, що при даних параметрах у системі реалізується швидке загасання коливань; невеликі збурення при $t > 4$ обумовлені дією шуму.

3.4 Приклади

3.4.1 Осцилятор Уєди

Розглянемо відому модель "осцилятор Уєди":

$$\ddot{x} + \dot{x} + x^3 = A \sin \omega t.$$

Чисельний аналіз моделі виконано за допомогою модифікованого методу Ейлера, для застосування якого дана система зводиться до двох рівнянь першого порядку:

$$\begin{aligned} \frac{dx}{dt} &= v, \\ \frac{dv}{dt} &= -v - x^3 + A \sin \omega t. \end{aligned} \tag{3.10}$$

На рис. 3.4 показана частина результатів, що були отримані за допомогою розробленої програми (приклад програми, яка виводить на екран залежність $v(x)$, наводиться у додатку), що дають уявлення про динаміку системи при різних значеннях A та ω. У результаті проведеного моделювання встановлено, що при малих амплітудах (рис. 3.4(а)) єдиним

атрактором є замкнута крива — симетричний цикл періоду 1. При збільшенні амплітуди спостерігається біфуркація втрати симетрії. Вона полягає в тому, що симетричний цикл стає нестійким і виникає два цикли-атрактори. Далі на базі кожного з асиметричних циклів спостерігається каскад біфуркацій подвоєння періоду, що завершується переходом до хаосу (рис. 3.4(б)).

Увесь спектр динамічних режимів, притаманних даній моделі, наведено на рис. 3.5. Як бачимо, результати моделювання збігаються з відомими результатами. Однак слід звернути увагу на рис. 3.4(в)-(г). Використання змінних одинарної точності приводить до якісно неправильних

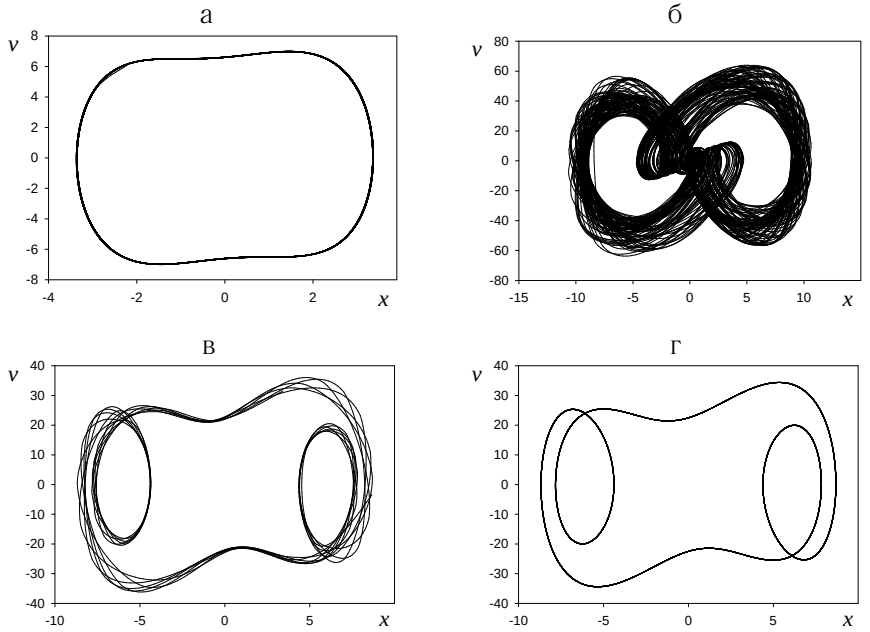

Рисунок 3.4 — Фазові портрети для системи Уєди: (а) $A = 10, \omega = 2.5$ (тип даних — double); (б) $A = 300, \omega = 3$ (тип даних — double); (в) $A = 300, \omega = 2.5$ (тип даних — float); (б) $A = 300, \omega = 2.5$ (тип даних — double). Перехідний режим $t \in [0..100]$ опущено

результатів. Так, при $A = 300, \omega = 2.5$ замість граничного циклу періоду 3 ми отримали траєкторію, що нагадує хаотичний атрактор[9].

3.4.2 Маятник

Як другий приклад розглянемо задачу про коливання маятника, модель якого представлена рівнянням (3.1). Результати дослідження руху маятника зручно представити у вигляді набору кривих на площині (x, p),

[9] Як зазначалося раніше, це пов'язано з нелінійним характером функціональної частини системи і є наслідком машинних округлень.

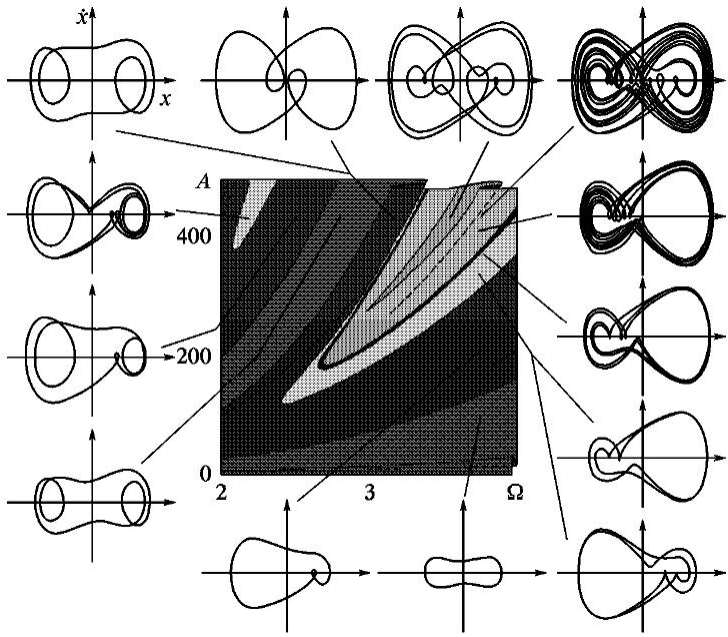

Рисунок 3.5 — Карта динамічних режимів для осцилятора Уєди у площині параметрів (A, ω). По периферії рисунка відповідно показано вигляд фазового портрета для різних підобластей (A, ω). Рисунок узято з роботи [13]

де $p = \dot{x}$ — швидкість зміни кута. Площина (x, p) називається фазовою площиною, змінна p — імпульсом, а криві $x(t)$, $p(t)$ — фазовими траєкторіями.

Фазові траєкторії лінійного осцилятора являють собою еліпси, що справедливо для математичного маятника лише при малих кутах відхилення. У загальному ж випадку рух маятника буде більш складним.

Кут відхилення маятника досить задавати в деяких кінцевих границях, наприклад, $-\pi \leq x < \pi$. При цьому необхідно розуміти, що точки фазової площини $(-\pi, p)$ і (π, p) є ототожненими, інакше кажучи, прямі $x = \pi$ та $x = -\pi$ склеєні одна з одною так, що зі смуги $-\pi \leq x < \pi$ та $-\infty < p < \infty$ вийшов циліндр. Якщо маятник робить один або декілька обертів, то точка, яка визначає його стан, рухається по кривій, що обвиває цей циліндр.

Відмінність коливань з великою амплітудою від малих коливань зводиться до того, що закон зміни кута з часом стає відмінним від гармонічного, а їх частота залежить від амплітуди.

Ми будемо використовувати знерозмірене рівняння вільного руху маятника без тертя

$$\ddot{x} + \sin x = 0.$$

Запишемо це диференціальне рівняння у вигляді системи рівнянь першого порядку:

$$\begin{aligned} \dot{x} &= p, \\ \dot{p} &= -\sin x. \end{aligned} \tag{3.11}$$

Для чисельного розв'язку даної системи рівнянь застосуємо метод з переступом.

Незважаючи на достатньо малий крок за часом, необхідним є контроль правильності розрахунків. Одним із методів контролю розрахунків є перевірка сталості повної енергії системи (зрозуміло, якщо є підстави вважати, що енергія зберігається). Можуть бути й інші величини, що зберігаються, але часто взагалі немає ніяких інтегралів руху (наприклад, за наявності сили тертя). У подібних випадках надійність розрахунків можна перевірити, застосовуючи метод повторного розрахунку

зі зменшеним кроком t. Якщо при цьому розв'язок не змінився, можна вважати, що результат отримано правильно.

Приклад програми на MATLAB, яка є базовою для роботи з даною системою, наводиться у додатку.

3.5 Запитання для самоконтролю

1 На основі яких принципів будуються математичні моделі фізичних систем? Наведіть приклади.

2 Як пояснити той факт, що фізичні системи різної природи описуються аналогічними рівняннями?

3 З якою метою проводиться знерозмірювання моделей динамічних систем? На яких принципах будується ця процедура?

4 Які фактори визначають коректність отриманих результатів моделювання фізичних систем?

5 Складіть алгоритми перевірки придатності чисельних схем для дослідження моделей фізичних систем (збереження енергії, оберненість часу і т.д).

6 Назвіть основні джерела похибок.

7 Що таке нормалізована форма запису числа?

8 Які особливості проведення математичних операцій з наближеними числами з використанням комп'ютера?

9 Які задачі називають погано обумовленими?

10 Чи можливе коректне моделювання систем, динаміка яких є хаотичною? Чому?

11 Наведіть алгоритм розв'язання диференціального рівняння другого порядку (наприклад, рівняння гармонічного осцилятора) придатним для цього методом?

3.6 Завдання для самостійної роботи

1 Знайдіть похибку наближень для $1/10$ та $1/7$ у випадку, коли під мантису відводиться шість двійкових розрядів [10].

2 Використовуючи табл.3.1, розрахуйте похибку обчислень:
а) (1/3+1/5)+1/6;
б) (1/10+1/3)+1/5;
в) (7/10+1/9)+1/7.

3 Промоделюйте рух матеріальної точки масою m на пружині жорсткістю k у в'язкому середовищі з коефіцієнтом в'язкості r. Рівняння руху має вигляд

$$m\ddot{x} = -r\dot{x} - kx.$$

Проаналізуйте графіки залежностей $x(t), v(t), a(t)$ при різних значеннях параметрів системи. Доведіть, що у разі відсутності тертя енергія системи $E = mv^2/2 + kx^2/2$ є сталою величиною.

4 Виконайте попередню задачу для випадку слабкого загасання, коли $r/2m < \omega_0 = \sqrt{k/m}$. Проведіть серію обчислювальних експериментів при різних початкових умовах системи і переконайтеся в тому, що прискорення змінюється у протифазі з координатою, а швидкість випереджає координату на $\pi/2$. Покажіть, що амплітуди координати та швидкості зменшуються за експонентою.

5 Промоделюйте рух осцилятора у випадку сильного загасання (при $r/2m > \omega_0 = \sqrt{k/m}$). Переконайтеся в тому, що в цьому випадку рух є аперіодичним.

6 Промоделюйте одновимірний рух матеріальної точки масою m, на яку діє сила $F(t)$:

$$\begin{aligned} F(t) &= A, \quad t \le t_1; \\ F(t) &= -A, \quad t_1 < t \le t_2; \\ F(t) &= 2 * A, \quad t_2 < t \le t_3. \end{aligned}$$

[10] Задачі даного підрозділу частково взяті з робіт [3], [14]

Значення A, m, r, k задайте самостійно. Проаналізуйте графіки залежностей $x(t)$, $v(t)$, $a(t)$.

7 Промоделюйте одновимірний рух тіла у в'язкому середовищі під дією зовнішньої гармонічної сили $F = A\sin\omega t$ (A — константа). $m \neq 0$, $k = 0$, $r \neq 0$.

Ця ситуація відповідає перехідному процесу, що відбувається при підключенні активно-індуктивного навантаження до джерела змінної напруги. При $t \to \infty$ перехідний струм наближається до вимушувального струму, що змінюється з тією самою частотою, що і прикладена ЕРС.

8 Створіть програму, яка моделює процеси у коливальній системі із зовнішньою періодичною силою, частота якої є пропорційною часу $F(t) = A\sin(\omega(1 + \alpha t)t)$, де $\alpha > 0$, A — константа. Значення ω та α підберіть такими, щоб резонансна частота коливальної системи знаходилася у центрі робочого діапазону частот.

Оскільки частота коливань прямо пропорційна часу, то обвідна графіка $x(t)$ є амплітудно-частотною характеристикою коливальної системи і називається резонансною кривою.

9 Тіло кидають під кутом α_0 до горизонту з початковою швидкістю v_0. На тіло діють сила тяжіння та сила тертя, яка є пропорційною швидкості польоту. Двовимірний рух тіла описується рівняннями:

$$\begin{aligned} m\frac{d^2x}{dt^2} &+ r\frac{dx}{dt} = 0, \\ m\frac{d^2y}{dt^2} &+ r\frac{dy}{dt} + mg = 0, \end{aligned}$$

де r — коефіцієнт тертя. Компоненти швидкості можуть бути розраховані за формулами:

$$\begin{aligned} \frac{d^2x}{dt^2} = \frac{dv_x}{dt} &= v\cos\alpha, \\ \frac{d^2y}{dt^2} = \frac{dv_y}{dt} &= v\sin\alpha, \end{aligned}$$

де $v = \sqrt{v_x^2 + v_y^2}$ — поточна швидкість; α — поточний кут між напрямком руху тіла та віссю x.

Побудуйте траєкторію руху тіла.

10 Промоделюйте роботу електричного фільтра: ланцюжок, що складається із активного опору R та конденсатора ємністю C, підключений до джерела змінної напруги $U_{inp} = U_0 \sin \omega t$. Вихідна напруга, яка знімається з опору, розраховується за формулою

$$U_{out} = IR,$$

де $I = dq/dt$ — сила струму у ланцюжку; q — заряд на пластинах конденсатора. Динамічна поведінка ланцюжка описується рівнянням

$$\frac{dq}{dt} = \frac{U_{inp}}{R} - \frac{q}{RC}.$$

Отримайте часову залежність зниження напруги на опорі (U_{out}). Які частоти пропускаються фільтром? Чи функціонує даний ланцюжок як фільтр високих (низьких) частот? Виконайте обчислення при різних параметрах вхідного сигналу. Константи R та C задайте самостійно.

11 Виконайте попередню задачу у випадку, коли напруга знімається з конденсатора ($U_{out} = q/C$). Фільтром яких частот є даний ланцюжок?

12 Промоделюйте роботу електричного ланцюжка, що складається із послідовно з'єднаних котушки індуктивності L, конденсатора C, резистора R та джерела змінної напруги $U_{inp} = U_0 \sin \omega t$. Підбираючи параметри L, R, C, U_0, ω, розрахуйте зниження напруги на всіх елементах ланцюжка. Динаміка системи описується рівнянням

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{q}{C} = U_{inp}.$$

Зниження напруги на котушці $U = L(d/dt)I$.

13 Створіть модель перехідного процесу у ланцюжку, що містить резистор R та котушку індуктивності L, які підключені до джерела постійної напруги. Проаналізуйте аналогічний перехідний процес у ланцюжку, що містить послідовно з'єднані резистор і конденсатор.

14 Промоделюйте роботу згладжуючого RL-фільтра при подачі на нього пульсуючої напруги, отриманої у результаті однонапівперіодного випрямлення. Для цього можна задати такі параметри коливальної системи: $L = 1$ [мГн], $R = 20$ [Ом], $C = 0$ [Ф], $\omega = 40$ [1/рад] і закон зміни зовнішньої напруги:

$$\begin{aligned} U_{inp}(t) &= U_0 \sin \omega t, \quad 0 < t \leq \frac{T}{2}; \\ U_{inp}(t) &= 0, \quad \frac{T}{2} < t \leq T, \end{aligned}$$

де $T = 2\pi/\omega$ — період коливань. Переконайтеся в тому, що із зростанням індуктивності зменшується коефіцієнт пульсацій струму і напруги на резисторі. Вивчіть залежність амплітуди пульсацій від індуктивності L, опору R та частоти імпульсів ω.

15 Вивчіть роботу електричного ланцюжка, що складається з послідовно з'єднаних резистора та конденсатора, з якого знімається вихідна напруга. Доведіть, що при подачі на ланцюжок прямокутних імпульсів

$$\begin{aligned} U_{inp}(t) &= A, \quad 0 < t \leq \frac{T}{2}; \\ U_{inp}(t) &= 0, \quad \frac{T}{2} < t \leq T, \end{aligned}$$

заряд конденсатора і відповідно напруга на ньому зростають пропорційно інтегралу від вхідної напруги (A — константа).

16 Розглянемо модель руху автомобіля масою m: вантаж на пружині жорсткістю k, що рухається з горизонтальною швидкістю v по шляху, який має складний профіль $y(x)$ (y — висота дороги у точці

з координатою x). Профіль ділянки шляху почергово описується рівняннями:

$$\begin{aligned} y(x) &= h(1 - e^{-\gamma x}); \\ y(x) &= A_1 \cos(x); \\ y(x) &= A_2 \sin(x), \end{aligned}$$

де h, A_1, A_2 — сталі; γ — параметр, що характеризує кривизну профілю. Беручи другу похідну від функції $y(x)$, отримуємо переносне вертикальне прискорення $w(t)$. Помножуючи $w(t)$ на m, отримуємо силу інерції вантажу $F(t)$. Використовуючи другий закон Ньютона та враховуючи, що $x = vt$, запишемо диференціальні рівняння, які описують вертикальні коливання на трьох ділянках шляху відповідно:

$$\begin{aligned} \frac{d^2\xi}{dt^2} &= -\frac{k}{m}\xi + h\gamma^2 v^2 e^{-\gamma vt}; \\ \frac{d^2\xi}{dt^2} &= -\frac{k}{m}\xi + A_1 v^2 \cos(vt); \\ \frac{d^2\xi}{dt^2} &= -\frac{k}{m}\xi + A_2 v^2 \sin(vt), \end{aligned}$$

де ξ — відхилення вантажу від положення рівноваги ($\xi(0) = 0$).

Самостійно задаючи параметри системи, проведіть серію експериментів та побудуйте графік залежності відхилення ξ від часу t. Додатково припустіть, що над дорогою на висоті H є навіс. Визначте при яких швидкостях автомобіля рух по дорозі є безпечним (амплітуда коливання не перевищує H).

17 Розглянемо дві популяції хижаків, які змагаються за одну і ту саму їжу. Нехай чисельність однієї популяції, наприклад ведмедів, N_1, а чисельність іншої, наприклад вовків, N_2. Нехай існує природний приріст кожної популяції з коефіцієнтами ε_1 та ε_2 відповідно. Нехай задані коефіцієнти споживання їжі γ_1 та γ_2 кожної з популяцій. Нехай $F(N_1, N_2)$ — кількість їжі, що поїдається обома популяціями за одиницю часу. Нехай у випадку, коли обидві популяції активні, $F(N_1, N_2) = \lambda_1 N_1 + \lambda_2 N_2$. Припустимо, що кожні три місяці

на рік ведмеді впадають у сплячку і $F(N_1, N_2) = \lambda_2 N_2$, де λ_1 та λ_2 — додатні константи.

Тоді еволюція популяцій описується такими рівняннями:

$$\begin{aligned} \frac{dN_1}{dt} &= [\varepsilon_1 - \gamma_1 F(N_1, N_2)]N_1; \\ \frac{dN_2}{dt} &= [\varepsilon_2 - \gamma_2 F(N_1, N_2)]N_2. \end{aligned}$$

Вибираючи коефіцієнти ε, γ, λ та початкову кількість популяцій, побудуйте фазовий портрет $N_1(N_2)$ та залежність чисельності кожної з популяцій від часу.

18 У вертикальній циліндричній трубці, закритій з нижнього кінця, без тертя рухається поршень масою m. У положенні рівноваги відстань між поршнем та дном трубки l_0. Площа поперечного перерізу трубки S. На поршень діє нормальний атмосферний тиск p_0. У початковий момент часу поршень відхиляють від положення рівноваги на відстань $x << l_0$ і відпускають. У результаті поршень починає здійснювати коливання, які описуються рівнянням

$$m\frac{d^2x}{dt^2} + \frac{mg + p_0 S}{l_0}x = 0,$$

де g — прискорення вільного падіння. Атмосферний тиск кожні $10\ c$ раптово змінює своє значення на менше ніж на 80% і через $10\ c$ знову набуває значення p_0. Самостійно вибираючи параметри системи, побудуйте фазовий портрет $(\dot{x}(x))$ та візуальну картину процесу коливань поршня.

19 У воді плаває пробка у формі паралелепіпіда з площею основи S і висотою h. Пробку занурюють у воду на незначну глибину x_0, а потім відпускають (початкова швидкість дорівнює 0). У результаті вона починає здійснювати коливання. Опір води не враховуємо. Зміна глибини занурення пробки у воду описується рівнянням

$$m\frac{d^2x}{dt^2} + \frac{\rho_{H_2O} g}{\rho_c h}x = 0,$$

де $\rho_{H_2O} = 1000$ — густина води; $\rho_c = 200$ — густина матеріалу пробки; g — прискорення вільного падіння. Припустимо, що кожні 20 c вода перетворюється у ртуть, а ртуть з тією самою періодичністю — у воду. Густина ртуті $\rho_r = 1360$. Побудуйте залежність віхилення пробки від часу при різних початкових значеннях координати та швидкості.

20 Розгляньте попередню задачу у випадку, коли на пробку діє сила опору води $F = -r(dx/dt)$, де r — коефіцієнт опору.

Розділ 4

Моделювання статичних систем

У розділі наведено основні підходи до розрахунку електричних та магнітних полів, утворених стаціонарно розподіленими зарядами або струмами. Розглядається метод релаксації та пропонується його застосування для отримання чисельного розв'язку рівнянь Лапласа та Пуассона.

4.1 Постановка задачі

Побудуйте еквіпотенціальні та силові лінії електростатичного поля, створеного окремими зарядами або провідниками.

4.2 Теоретичний матеріал

4.2.1 Силові лінії електричного поля

Дія зарядженого тіла на оточуючі тіла проявляється у вигляді сил притягання та відштовхування, які намагаються перевертати та переміщати ці тіла відносно зарядженого тіла. Залежно від заряду та форми тіла його дія у різних точках простору буде різною. Тому для повної характеристики заряду необхідно знати, яку дію він виконує у точках простору, який його оточує, тобто знати електричне поле, яке виникає навколо заряду [15]. Таким чином, під терміном "електричне поле" розуміється простір, в якому проявляється дія електричного заряду.

Якщо існує не один, а декілька зарядів, розміщених у різних місцях, то у будь-якій точці простору проявляється сумісна дія цих зарядів, електричне поле, що створене даними зарядами (рис. 4.1 (а)).

Зазначимо, що спроба пояснити електричне поле на основі вже вивчених законів зазнає невдачі. Тому слід вважати, що електричне поле є самостійною фізичною реальністю, яка не зводиться ні до теплових, ні до механічних явищ. Електричні явища являють собою новий клас явищ

природи, з властивостями яких можна познайомитися в експерименті, у тому числі комп'ютерному.

Згідно із принципом суперпозиції величина напруженості електричного поля $\vec{E}(\vec{r})$ у деякій точці $\vec{r}$ простору знаходиться за формулою

$$\vec{E}(\vec{r}) = k\sum_{i=1}^{N} q_i \frac{\vec{r} - \vec{r}_i}{|\vec{r} - \vec{r}_i|^3}, \qquad (4.1)$$

де N — кількість точкових зарядів $(q_1 ... q_N)$, які утворюють поле, $\vec{r}_i$ — координата нерухомого i- го заряду; k — електрична стала. Оскільки напруженість електричного поля $\vec{E}$ є векторною величиною, то, зрозуміло, в кожній точці простору воно характеризується напрямком та абсолютним значенням. Для наочного уявлення такого поля зручним є його зображення за допомогою силових ліній [11]. Для побудови силової лінії розрахуємо компоненти вектора напруженості.

а б

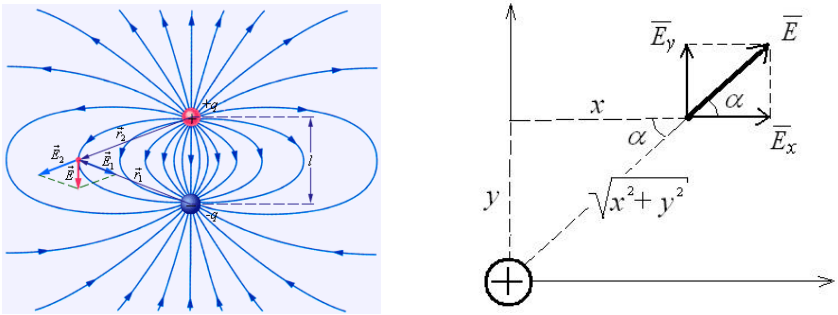

Рисунок 4.1 — (а) Принцип суперпозиції; (б) розрахунок компонентів вектора електричного поля.

Як видно з рис. 4.1(б):

$$E_x = |\vec{E}|\cos\alpha, \qquad E_y = |\vec{E}|\sin\alpha, \qquad (4.2)$$

[11] Властивості силових ліній: кожна лінія спрямована таким чином, що дотична до неї збігається з напрямком електричного поля; силова лінія є неперервною; повна кількість ліній, що виходять з даного заряду, пропорційна його величині.

$$\cos\alpha = x/\sqrt{x^2+y^2}, \qquad \sin\alpha = y/\sqrt{x^2+y^2}. \tag{4.3}$$

У результаті маємо:

$$E_x = |\overrightarrow{E}|\frac{x}{\sqrt{x^2+y^2}}, \tag{4.4}$$

$$E_y = |\overrightarrow{E}|\frac{y}{\sqrt{x^2+y^2}}. \tag{4.5}$$

Оскільки

$$|\overrightarrow{E}| = \frac{q}{(\overrightarrow{r})^2} = \frac{q}{x^2+y^2}, \tag{4.6}$$

остаточно отримуємо

$$E_x = q\frac{x}{(x^2+y^2)^{3/2}}, \tag{4.7}$$

$$E_y = q\frac{y}{(x^2+y^2)^{3/2}}. \tag{4.8}$$

Алгоритм побудови силової лінії наводиться у підрозділі "Алгоритми".

4.2.2 Силові лінії магнітного поля

Напруженість магнітного поля $\triangle\overrightarrow{B}(\overrightarrow{r})$, створена ділянкою $\triangle\overrightarrow{L}$ нескінченно довгого (або колового провідника) у точці $\overrightarrow{r}$ простору, розраховується за законом Біо-Савара:

$$\triangle\overrightarrow{B}(\overrightarrow{r}) = \frac{\mu_0 I}{4\pi}\left(\frac{\triangle\overrightarrow{L}\times\overrightarrow{r}}{|\overrightarrow{r}|^3}\right), \tag{4.9}$$

де μ_0 — магнітна стала; I — струм, який проходить по провіднику.

4.2.3 Розрахунок потенціалу електричного поля методом релаксації

На практиці досить часто виникає ситуація, коли важко визначити розподіл окремих зарядів, наприклад на зарядженому тілі складної форми, а отже, і розрахувати поля окремих зарядів теж неможливо. У

такому випадку корисно ввести у розгляд потенціал електричного поля $\varphi(\overrightarrow{r})$, виміряти який не викликає особливих труднощів [16]. Нагадаємо, що напруженість поля пов'язана з потенціалом так:

$$\overrightarrow{E}(\overrightarrow{r}) = -\nabla\varphi(\overrightarrow{r}) = -\left(\frac{\partial\varphi(\overrightarrow{r})}{\partial x}\overrightarrow{i} + \frac{\partial\varphi(\overrightarrow{r})}{\partial y}\overrightarrow{j} + \frac{\partial\varphi(\overrightarrow{r})}{\partial z}\overrightarrow{k}\right). \tag{4.10}$$

Сформулюємо наступну двовимірну задачу. Нехай потенціал $\varphi(\overrightarrow{r})$ заданий на деякій системі границь і потрібно знайти потенціал у будь-якій точці області, де немає зарядів (частинок). Така задача називається крайовою. Прямий метод знаходження $\varphi(x, y)$ грунтується на рівнянні Лапласа, яке у декартових координатах має вигляд

$$\nabla^2\varphi(x, y) \equiv \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} = 0. \tag{4.11}$$

Подане рівняння розв'яжемо чисельно. Для цього розбиваємо область на окремі комірки і переходимо від неперервної моделі до дискретної. Переписуючи часткові похідні у різницевому вигляді, перепишемо наведене рівняння у вигляді

$$\varphi(x+\triangle x, y)+\varphi(x-\triangle x, y)+\varphi(x, y+\triangle y)+\varphi(x, y-\triangle y)-4\varphi(x, y) = 0, \tag{4.12}$$

звідки

$$\varphi(x, y) = \frac{1}{4}\left[\varphi(x+\triangle x, y) + \varphi(x-\triangle x, y) + \varphi(x, y+\triangle y) + \varphi(x, y-\triangle y)\right]. \tag{4.13}$$

Інакше кажучи, $\varphi(x, y)$ дорівнює середньому за сусідніми комірками праворуч, ліворуч, зверху та знизу. Співвідношення (4.13) і є основною розрахунковою формулою методу релаксації.

Рівняння Лапласа справедливе тільки в областях, де немає зарядів. Якщо ж усередині області розподілені заряди з густиною $\rho(x, y)$, то потрібно використовувати рівняння Пуассона, яке у диференціальній формі можна запасати у вигляді

$$\nabla^2\varphi(\overrightarrow{r}) = -\frac{\rho(\overrightarrow{r})}{\varepsilon_0}. \tag{4.14}$$

Різницеве рівняння Пуассона у двовимірному випадку має вигляд

$$\varphi(x,y) = \frac{1}{4}\left[\varphi(x+\triangle x, y) + \varphi(x-\triangle x, y) + \right.$$
$$\left. + \varphi(x, y+\triangle y) + \varphi(x, y-\triangle y)\right] + \frac{1}{4}\triangle x \triangle y \frac{\rho(x,y)}{\varepsilon_0}, \qquad (4.15)$$

де $\rho(x,y)\triangle x\triangle y$ — повний заряд у комірці з центром у точці x, y.

4.3 Алгоритми

4.3.1 Силові лінії електричного поля

Наведемо простий алгоритм побудови окремої силової лінії поля електричного заряду, розміщеного у точці (0,0) координатної плоскості:

1 Вибираємо точку (x, y) та обчислюємо компоненти E_x, E_y за формулами (4.7)-(4.8).

2 Проводимо відрізок довжиною l (починаючи з цієї точки), компоненти якого розраховуються за формулами:

$$\triangle x = l\frac{E_x}{\sqrt{E_x^2 + E_y^2}}, \qquad (4.16)$$

$$\triangle y = l\frac{E_y}{\sqrt{E_x^2 + E_y^2}}. \qquad (4.17)$$

3 Переходимо у нову точку $(x+\triangle x, y+\triangle y)$ та повторимо процедуру 1-3.

4.3.2 Силові лінії магнітного поля

Для побудови силових ліній застосуємо такий алгоритм. Розбиваємо провідник на окремі ділянки довжиною $\triangle \overrightarrow{L}_j$. Припускаємо, що j-та ділянка провідника знаходиться у точці $\overrightarrow{r}_j = (x_j, y_j, z_j)$ простору. Розраховуємо компоненти напруженості магнітного поля, створеного кожною

з цих ділянок у вибраній точці $\overrightarrow{r}$ простору, за формулою (4.9):

$$\triangle B_x(\overrightarrow{r}) = \frac{\mu_0 I}{4\pi}\left(\frac{\triangle L_y(z-z_j) - \triangle L_z(y-y_j)}{|\overrightarrow{r} - \overrightarrow{r}_j|^3}\right), \tag{4.18}$$

$$\triangle B_y(\overrightarrow{r}) = \frac{\mu_0 I}{4\pi}\left(\frac{\triangle L_z(x-x_j) - \triangle L_x(z-z_j)}{|\overrightarrow{r} - \overrightarrow{r}_j|^3}\right), \tag{4.19}$$

$$\triangle B_z(\overrightarrow{r}) = \frac{\mu_0 I}{4\pi}\left(\frac{\triangle L_x(y-y_j) - \triangle L_y(x-x_j)}{|\overrightarrow{r} - \overrightarrow{r}_j|^3}\right), \tag{4.20}$$

де

$$|\overrightarrow{r} - \overrightarrow{r}_j|^3 = \left((x-x_j)^2 + (y-y_j)^2 + (z-z_j)^2\right)^{3/2}. \tag{4.21}$$

4.3.3 Розрахунок потенціалу електричного поля методом релаксації

Підхід, називаний методом релаксації, базується на такому алгоритмі:

1 Розбиваємо розглянуту область сіткою комірок.

2 Комірки поділяються на граничні і внутрішні. Приписуємо кожній граничній комірці значення потенціалу на межі цієї області.

3 Приписуємо усім внутрішнім коміркам довільний потенціал (краще яке-небудь розумне початкове наближення).

4 На першому кроці для усіх внутрішніх комірок обчислюємо нові значення φ за формулою (4.13). Це перша ітерація процесу релаксації.

5 Повторюємо описану у п. 4 процедуру, використовуючи значення φ, отримані на попередній ітерації. Даний ітераційний процес продовжується до того часу, поки потенціал кожної внутрішньої комірки не буде змінюватися в межах необхідної точності.

Представлений алгоритм реалізований у програмі `potent`, текст якої наводиться у додатку.

4.4 Приклади

4.4.1 Побудова силових ліній електричного поля, утвореного точковими зарядами

Подана у додатку програма дозволяє побудувати силові лінії електричного поля, утвореного двома зарядами. Координати точок початку силових ліній вводяться з клавіатури. Програма може бути легко модифікована під будь-яку кількість зарядів та силових ліній.

Результат роботи програми поданий на рис. 4.2(а). Як бачимо, отримана картина цілком відповідає теоретичним уявленням та експериментальним даним відносно електричного поля диполя: силові лінії напрямлені від одного заряду до іншого, причому густина ліній між зарядами більша, ніж густина силових ліній, що лежать ззовні відносно диполя. Оскільки напруженість електричного поля пропорційна кількості ліній поля, що проходять через одиничну площадку, що є перпендикулярною до цих ліній, можна зробити висновок про те, що напруженість поля посередині між зарядами більша, ніж ззовні.

а б

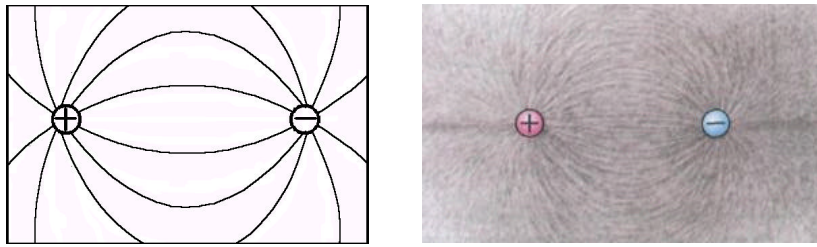

Рисунок 4.2 — (а) Результати комп'ютерних розрахунків — силові лінії електричного поля, утвореного диполем; (б) візуалізація ліній електричного поля у реальному експерименті

Не слід думати, що лінії напруженості — це існуючі у дійсності утворення начебто розтягнутих пружних ниток або шнурів, як припускав сам Фарадей. Лінії напруженості лише допомагають представити розподіл поля у просторі. Однак, з іншого боку, слід зазначити, що силові лінії

можна зробити "видимими". Для цього потрібно два електроди з'єднати з полюсами електростатичної машини і занурити у в'язкий діелектрик (наприклад, у касторову олію). У цю рідину треба насипати і добре перемішати довгасті частинки ізолятора (наприклад, віскози). При цьому у рідині створюється досить сильне електричне поле. Під впливом електричного поля частинки діелектрика поляризуються: на їхніх кінцях з'являються заряди протилежного знака. Частинки повертаються в зовнішньому полі вздовж ліній напруженості, і заряди на їх кінцях взаємодіють один з одним. У результаті частинки діелектрика орієнтуються вздовж силових ліній (рис. 4.2(б)).

4.4.2 Побудова силових ліній та еквіпотенціальних поверхонь у MATLAB

Результати роботи програми (подана у додатку) наведені на рис. 4.3. Як бачимо з рисунка, еквіпотенціальні лінії (потенціал уздовж такої лінії є сталим) біля заряду, величина якого є більшою за модулем, розміщені більш щільно. Як відомо [15], напруженість пов'язана з потенціалом таким співвідношенням:

$$E = \frac{U_{12}}{l}, \tag{4.22}$$

де l — відстань між еквіпотенціальними лініями 1 та 2; U_{12} — різниця потенціалів між лініями 1 та 2. Звідси випливає, що чим тісніше розміщені еквіпотенціальні лінії, тим більша напруженість поля у даному місці. Виходячи з рисунка, бачимо, що напруженість електричного поля в околі правого заряду більша, ніж в околі лівого.

Зазначимо, що за допомогою різниці потенціалів можна охарактеризувати електричне поле тією самою мірою, що й за допомогою напруженості. Графік еквіпотенціальних ліній являє собою таку ж саму "електричну карту", як і графік ліній поля.

4.4.3 Розрахунок потенціалу методом релаксації.

Результат роботи програми поданий на рис. 4.4.

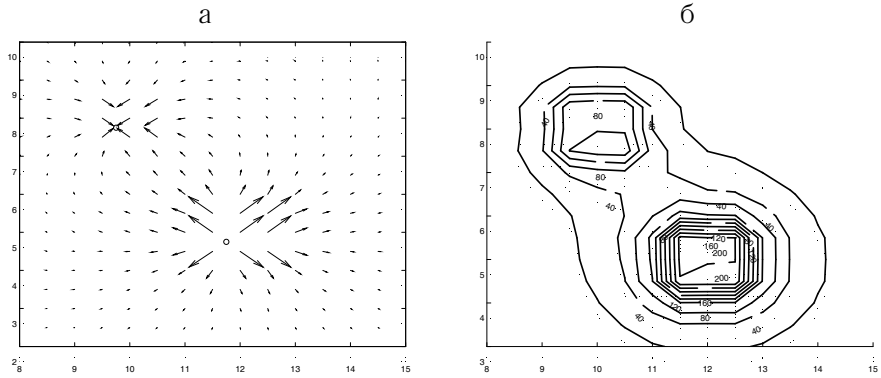

Рисунок 4.3 — Силові (а) та еквіпотенціальні (б) лінії електричного поля двох зарядів

4.5 Запитання для самоконтролю

1 Назвіть основні кроки алгоритму розрахунку напруженості електричного поля. На яких фізичних ідеях грунтується цей алгоритм?

2 Назвіть основні кроки алгоритму розрахунку напруженості магнітного поля. На яких фізичних ідеях грунтується цей алгоритм?

3 У чому полягає суть методу релаксації? В яких випадках він застосовується?

4.6 Завдання для самостійної роботи

1 Побудуйте силові лінії електричного поля, утвореного багатьма різнойменними зарядами, розподіленими випадковим чином.

2 Побудуйте траєкторію руху частинки, яка несе заряд q і рухається у полі декількох нерухомих зарядів. Для знаходження координати та швидкості частинки у різні моменти часу використайте алгоритми, наведені у розділі 3. Нагадаємо, що прискорення частинки масою m дорівнює $(q/m)\overrightarrow{E}$, де $\overrightarrow{E}$ — електричне поле, створене нерухомими зарядами.

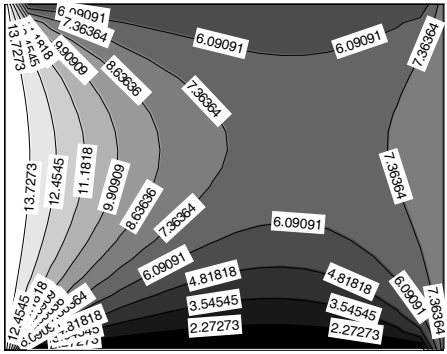

Рисунок 4.4 — Розподіл потенціалу в області, на границях якої потенціал є сталим. Потенціал лівої границі — 15, правої — 8, верхньої — 5, нижньої — 1

3 Розрахуйте електричне поле тонкої однорідної пластини кінцевих розмірів. Розгляньте випадок, коли пластина лежить у площині $x - z$, і відповідно лінії поля напрямлені перпендикулярно до неї вздовж осі y. Для проведення розрахунків пропонується наступний алгоритм. Нехай Q — заряд пластини. Розіб'ємо пластину сіткою із $N \times N$ комірок так, що кожну комірку можна вважати точковим зарядом $q = Q/N^2$. Застосовуючи формулу для потенціалу точкового заряду $\varphi(r) = q/4\pi\varepsilon_0 r$ (r — відстань від заряду до точки, де визначається потенціал), розраховуємо повний потенціал у певній точці y. Використовуючи співвідношення (4.10), знайдемо напруженість електричного поля E_y у даній точці простору на осі, перпендикулярній до пластини. Кількість комірок $N \times N$ вибирається так, щоб подальше збільшення N не впливало суттєво на результати розрахунків. Отримайте залежність $E_y(y)$ та побудуйте силові лінії поля.

4 Розрахуйте магнітне поле, створене коловим провідником зі струмом I. Припустіть, що коловий провідник із центром у початку координат розміщений у площині $x - z$, причому вісь z напрямлена перпендикулярно до екрана (на спостерігача). Тоді площина $x - y$

є площиною екрана. Побудуйте силові лінії магнітного поля.

5 Розрахуйте магнітне поле соленоїда. Припустіть, що соленоїд — це набір із N колових витків (задача 4). Побудуйте силові лінії магнітного поля.

6 Промоделюйте рух зарядженої частинки (наприклад, електрона) у магнітному полі соленоїда. Нагадаємо, що на заряджену частинку з боку магнітного поля діє сила Лоренца. Припустіть, що частинка має певну початкову швидкість та влітає у соленоїд паралельно його осі.

7 Розрахуйте потенціал $V(x, y)$ усередині квадратної області розмірами $L \times L$, якщо потенціал на її межі становить 20 [В].

8 Розгляньте попередню задачу у випадку, коли потенціали кожної із чотирьох границь різні: 5, 10, 15 та 20 [В]. Проведіть розрахунки для випадку, коли всередині області розподілені заряди з щільністю 5 [В/см^2].

Розділ 5

Моделювання синергетичних процесів

У розділі наводиться методика моделювання процесів поширення хвиль збудження в активних середовищах типу нейронних сіток. Подаються основні теоретичні відомості; показано можливі випадки та відповідні алгоритми реалізації різних типів хвиль.

5.1 Постановка задачі

Промоделюйте процеси, що відбуваються у двовимірному активному середовищі, утвореному клітинними автоматами. Розгляньте поширення хвиль збудження при різних параметрах середовища і випадковому початковому розподілі збуджених елементів.

5.2 Теоретичний матеріал

5.2.1 Побудова моделі активного середовища

Активні середовища характеризуються наявністю потоку енергії від зовнішнього джерела та її дисипацією. Кожний елемент середовища, через який проходить потік енергії, виводиться зі стану теплової рівноваги і набуває здатності здійснювати автоколивання. У випадку, коли такі пов'язані між собою елементи формують розподілене активне середовище, у ньому стає можливим поява різноманітних просторових структур (самоорганізація).

У більшості випадків активне середовище будують на основі одного з трьох типів активних елементів: бістабільних, мультивібраторних або автоколивальних. **Бістабільний елемент** має два стаціонарні стани, в кожному з яких він може перебувати нескінченно довго. Зовнішній вплив переводить елемент із одного стану в інший та навпаки. Щоб викликати такий перехід, інтенсивність зовнішнього впливу має перевищувати певний рівень. **Мультивібраторний елемент** перебуває у стані спокою при малих зовнішніх впливах. При перевищенні порогового

рівня інтенсивності впливу елемент переходить у збуджений стан і через певну послідовність активних переходів знову повертається у стан спокою. **Автоколивальний елемент** постійно здійснює циклічну послідовність переходів незалежно від енергії зовнішнього впливу. Останній може лише прискорити або уповільнити цей процес. Активне середовище можна представити у вигляді ланцюжка (або сітки) таких елементів, що є придатним наближенням до реальних систем[12]. Добре відомо, наприклад, що саме моделі дискретних сіток найбільш адекватно описують роботу нейронних сіток або процеси у живих тканинах із переплетеними м'язовими волокнами.

На основі мультивібраторних елементів побудуємо модель, що дозволяє всебічно дослідити поширення хвиль збудження однорідною нейронною сіткою [17]- [19]. Припустимо, що вплив елементів сітки один на одного здійснюється лише між сусідніми елементами. У випадку, коли елементи перебувають в однаковому стані, вони не впливають один на одного. Якщо ж їх стани є різними, вони взаємодіють. Елемент, який перебуває у менш стійкому стані, переходить у більш стійкий стан, в якому перебуває його сусід. Природно покласти, що елемент в активному стані є несприйнятливим до зовнішнього впливу до того часу, поки він не здійснить увесь ланцюжок переходів. У такому випадку сіткою циклічно будуть поширюватися хвилі збудження. Кожний з елементів середовища може перебувати у трьох станах: спокою, збудження та рефрактерності. Елемент може перейти зі стану спокою у збуджений стан за рахунок зовнішнього впливу або наявності збуджених сусідів. У цьому стані елемент набуває здатності збуджувати сусідні елементи. Через певний час елемент переходить у стан рефрактерності (у цьому стані він втрачає здатність збуджувати сусідні елементи), а потім у стан спокою.

Розглянемо двовимірний випадок: сітка утворена елементами, які нумеруються індексами i та j. Нехай стан кожного елемента описується фазою y_{ij}^n і концентрацією активатора u_{ij}^n, де n — позначає дискретний момент часу t^n. Припускаємо, що τ_s — дискретний час (тривалість) збудженого стану, τ_r — тривалість стану рефрактерності. Значення фази $y_{ij}^n = 0$ — відповідає стану спокою; $0 < y_{ij}^n \leq \tau_s$ — стан збудження;

[12] Такі моделі мають загальну назву **клітинних автоматів**.

$\tau_s < y_{ij}^n < \tau_s + \tau_r$ — стан рефрактерності. Перехід елемента у вузлі (i, j) зі стану спокою у стан збудження відбувається при перевищенні концентрації активатора u_{ij}^n у цьому вузлі певного порогового значення h: $u_{ij}^n \geq h$. З часом елемент здійснює фіксовану послідовність переходів, при цьому в кожний наступний момент часу значення фази може зростати на одиницю, і при досягненні фазою значення $\tau_s + \tau_r$ елемент переходить у стан спокою. Будемо вважати, що при переході зі стану рефрактерності у стан спокою концентрація активатора стає нульовою. За наявності сусіднього елемента, що перебуває у збудженому стані, вона збільшується на одиницю. Якщо m найближчих сусідів є збудженими, то на відповідному кроці до попереднього значення концентрації активатора додається число збуджених сусідів

$$u_{ij}^{n+1} = u_{ij}^n + m. \tag{5.1}$$

Кількість сусідів (величина кореляції за сіткою) вибирається самостійно. Можна обмежитися урахуванням найближчих восьми сусідніх елементів.

Таким чином, переходи між станами відбуваються відповідно до схеми:

$$y_{ij}^{n+1} = \begin{cases} y_{ij}^n + 1, & \text{якщо } 0 < y_{ij}^n < 1\tau_s + \tau_r, \\ 0, & \text{якщо } y_{ij}^n = \tau_s + \tau_r, \\ 0, & \text{якщо } y_{ij}^n = 0, u_{ij}^{n+1} < h, \\ 1, & \text{якщо } y_{ij}^n = 0, u_{ij}^{n+1} \geq h. \end{cases} \tag{5.2}$$

Щоб отримати локально діюче періодичне джерело (**пейсмекер**), достатньо виділити певні вузли, в яких фаза змінюється за простим законом

$$y_{ij}^{n+1} = \begin{cases} y_{ij}^n + 1, & \text{якщо } y_{ij}^n < T - 1, \\ 0, & \text{якщо } y_{ij}^n \geq T - 1. \end{cases} \tag{5.3}$$

Зрозуміло, що незалежно від оточення такий елемент буде здійснювати циклічні коливання з періодом T.

5.2.2 Гра "Життя"

Як зазначалося вище, клітинні автомати є придатним наближенням до моделювання простих динамічних систем. Можна переконатися, що

просторові візерунки клітинних автоматів нагадують картини, які простежуються у реальних природних явищах організації та самоорганізації, типу коагуляції або росту кристалів. Яскравим прикладом такої організації є гра "Життя"[13]. Правила гри є доволі простими. На квадратній гратці випадковим чином розподіляємо "живі" клітини. Для кожної клітини визначаємо кількість "живих" сусідів серед восьми клітин, що її оточують. "Жива" клітина "виживає" тільки у випадку, коли ця кількість дорівнює 2 або 3. Якщо кількість менша 2, то клітина "вмирає від самотності", якщо більша 3 — "вмирає від перенаселення". "Мертва" клітина "оживає" на наступному кроці за часом, тільки якщо кількість "живих" сусідів дорівнює 3. Приклад програми на MATLAB, що реалізує даний алгоритм, наводиться у кінці розділу.

5.3 Алгоритми

Розглянемо алгоритм моделювання поширення хвиль збудження однорідною сіткою:

1 Задаємо число елементів активного середовища N x M, початковий розподіл збуджених елементів (ненульові значення y у певних вузлах побудованої гратки), параметри активних елементів: час збудження τ_s, час рефрактерності τ_r та порогове значення концентрації активатора h.

2 Запускаємо цикл по t.

2.1 Змінній t присвоюємо значення $t + \triangle t$.

2.2 Перебираємо всі елементи активного середовища, визначаючи їх фази $y_{ij}(t+\triangle t)$ та концентрацію активатора $u_{ij}(t+\triangle t)$ у момент часу $t + \triangle t$ згідно із наведеними формулами (5.1) — (5.3).

2.3 Очищаємо екран і будуємо збуджені елементи активного середовища. У найпростішому випадку можна покласти, що збуджені елементи мають один колір (темно-сірий у програмі, яка наведена нижче), елементи у стані рефрактерності та спокою — інший колір (наприклад, світло-сірий та білий відповідно).

[13] Була розроблена Джоном Конвеєм у 1970 році.

(2.4) Повертаємося до операції 2.1. Якщо цикл по t закінчився — вихід з циклу.

5.4 Приклади

5.4.1 Автохвильові процеси на однорідній сітці

Програма, подана у додатку, моделює активне середовище і процеси, які у ньому відбуваються. Початкові значення фаз елементів активного середовища задаються у функціях `Init1()` (один збуджений елемент у центрі), `Init2()` (дві смужки елементів, одна з яких являє собою ділянку збуджених елементів, інша — ділянку елементів у стані рефрактерності) та `Init3()` (чотири асиметричні смужки збуджених та рефрактерних елементів). Для активації відповідних функцій зніміть коментар. Збуджені елементи виводяться на екран темно-сірим кольором, елементи у стані рефрактерності — світло-сірим, у стані спокою — білим кольором. Програма дозволяє промоделювати огинання хвилями перепони (смужка стаціонарних елементів у стані спокою) та генерацію автохвиль за допомогою пейсмекера. Параметри середовища `tau_s=6` (шість станів збудження), `tau_r=6` (шість станів рефрактерності) та `h=4` (границя накопичення концентрації активатора, перетинаючи яку, елемент у даній комірці переходить у стан збудження).

Результат роботи програми поданий на рис. 5.1 — 5.4.

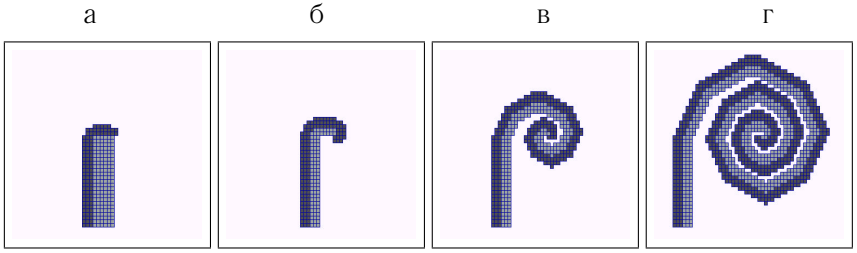

Рисунок 5.1 — Утворення однорукавної хвилі

У результаті проведеного комп'ютерного моделювання було з'ясовано, що у побудованому активному середовищі можуть поширюватися

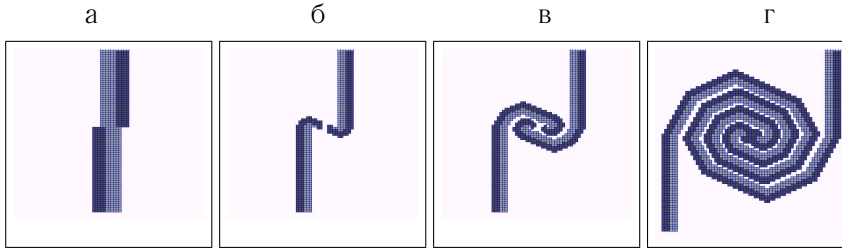

Рисунок 5.2 — Утворення та еволюція дворукавної хвилі у послідовні моменти часу (рис. а — г)

плоскі хвилі збудження, що рухаються з певною швидкістю, яка залежить від порогу активації h. При цьому хвиля має сумарну ширину, що дорівнює сумі $\tau_s + \tau_r$. При збільшенні порогу активації швидкість поширення хвилі зменшується. Вияснено, що поширення хвилі можливе лише у випадку, коли розпад активатора не є занадто швидким.

Оскільки стан середовища до і після проходження хвилі збудження не міняється, ніщо не перешкоджає створенню початкового розподілу у вигляді обірваної напівхвилі (рис. 5.1). Наступна еволюція такого початкового розподілу залежить від збудливості середовища, яка обумовлена значеннями порогу h та тривалості збудженого стану. Якщо збудливість середовища низька, напівхвиля скорочується з часом, і початкове

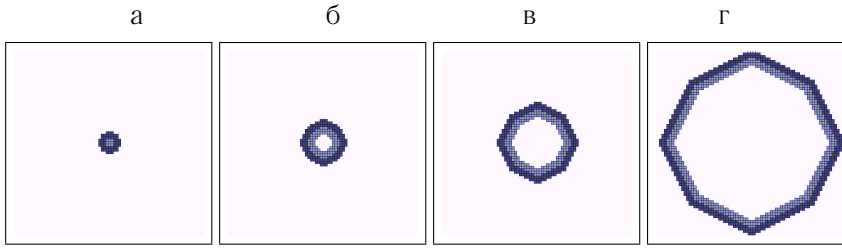

Рисунок 5.3 — Еволюція поодинокої хвилі у часі (рис. (а)-(г)). Джерело хвиль — один збуджений елемент у центрі. Темні ділянки — область збудження, сірі — рефрактерності, білі — область спокою

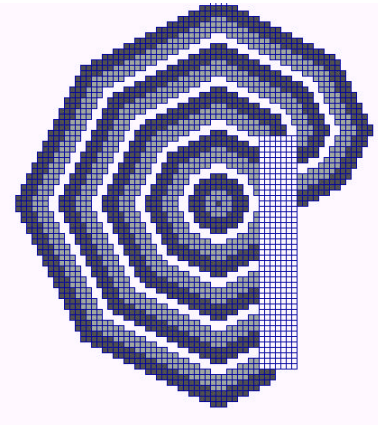

Рисунок 5.4 — Генерація хвиль за допомогою пейсмекера. Праворуч від центра — перепона зы стаціонарних неактивних елементів

збурювання загасає.

Якщо задати початковий розподіл у вигляді двох напівхвиль, орієнтованих назустріч одна одній, з нього розвивається дворукавна спіральна хвиля (рис. 5.2).

Крім того, у результаті проведених експериментів було з'ясовано, що хвилі збудження, які зіштовхуються, анигілюють, тобто гасять одна одну. Це досить загальна властивість хвиль в активних середовищах. Дійсно, при зіткненні двох таких хвиль збуджені елементи виявляються затиснутими з обох боків елементами, що перебувають у стані рефрактерності та не здатні передати збудження іншим елементам середовища. Анигіляція хвиль при їх зіткненні приводить до одного дуже важливого ефекту. Якщо в такому середовищі діє кілька періодичних джерел хвиль, то з часом джерело, що генерує хвилі з максимальною частотою, пригнічує всі інші джерела.

Щоб одержати локальне періодично діюче джерело хвиль, або пейсмекер, досить узяти групу автоколивальних елементів, фаза яких міняється згідно з (5.3). Якщо помістити досить велику групу таких елементів в активне середовище, останнє буде відігравати роль джерела концентричних хвиль збудження (рис. 5.3-5.4). При цьому фактичний період

генерації хвиль збігається з T, якщо $T > \tau_s + \tau_r + 1$. У протилежному випадку середовище "пропускає" деякі такти активації, оскільки автоколивальні елементи, що перебувають у стані збудження, виявляються оточеними елементами, що перебувають у стані рефрактерності та не здатні породити хвилю збудження. У результаті пейсмекер починає генерувати хвилі з більш високим періодом.

На відміну від пейсмекерів, утворення яких обумовлено неоднорідністю активного середовища, спіральні хвилі є локальними джерелами автохвиль, не прив'язаними до будь-якої неоднорідності. Положення центра спіральної хвилі визначається лише початковими умовами, що склалися при її зародженні. Усі спіральні хвилі у даному середовищі мають однакову частоту циркуляції. Тому дві спіральні хвилі не пригнічують з часом одна одну. Своєрідні ефекти спостерігаються при взаємодії спіральної хвилі з пейсмекером. Якщо частота генерації хвиль пейсмекером нижче частоти циркуляції спіральної хвилі, остання пригнічує пейсмекер. При зворотному співвідношенні частот спіральна хвиля вироджується у додаткову напівхвилю, що, випробовуючи послідовні перезамикання, поступово витісняється на периферію пейсмекера.

Як висновок зазначимо, що до використання моделей клітинних автоматів звертаються у декількох випадках. Навіть якщо відомі точні диференціальні рівняння, що описують середовище, як правило, не вдається знайти аналітичний розв'язок цих рівнянь і необхідно проводити чисельні розрахунки. Якщо задача полягає в дослідженні хвильових процесів, зв'язаних з утворенням і розвитком структур у таких двовимірних (а тим більше тривимірних) середовищах, розрахунки виявляються надзвичайно трудомісткими [17]. У цій ситуації (особливо коли нас цікавлять у першу чергу якісні результати) можна відмовитися від чисельного інтегрування відповідних диференціальних рівнянь і звернутися до аналізу набагато простіших систем, що являють собою сітки клітинних автоматів. Як показують розрахунки, вони достатньо якісно відтворюють динаміку реальних систем. Наприклад, розвиток хвильових структур у відомій реакції Білоусова-Жаботинського [20] дуже добре описується наведеною у даному розділі моделлю (рис. 5.5).

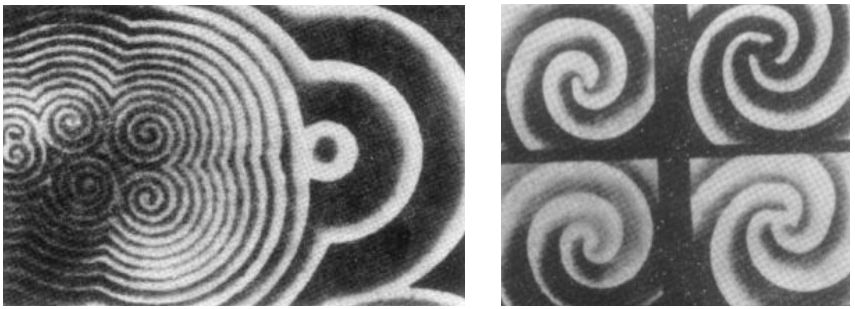

Рисунок 5.5 — Експеримент. Спіральні хвилі у реакції Бєлоусова-Жаботинського. Із роботи [17]

5.4.2 Гра "Життя"

Програма подана у додатку.

5.5 Запитання для самоконтролю

1 Дайте визначення активного середовища.

2 Які типи активних елементів ви знаєте?

3 Що таке пейсмекер? Яким чином можна промоделювати його роботу?

4 Які стани активних елементів ви знаєте? Яка між ними різниця?

5 Чи може елемент мати декілька активних станів?

6 Що визначає концентрація активатора?

7 Яким чином можна згенерувати однорукавну та дворукавну хвилю?

8 Яким чином в актиному середовищі генерується перепона?

9 Модифікуйте програму "Життя", вводячи власні правила гри. Проаналізуйте отримані результати.

5.6 Завдання для самостійної роботи

1 Проаналізуйте роботу двох та більше пейсмекерів. Візуально доведіть, що високочастотні пейсмекери заглушують низькочастотні.

2 Промоделюйте роботу пейсмекера, період якого змінюється з часом за деяким законом.

3 Промоделюйте дифракцію хвиль. Для цього необхідно створити хвилю, на шляху якої розміщена перешкода із щілиною, наприклад, непрозорий екран, що складається з елементів, які перебувають у стаціонарному стані спокою.

4 Проаналізуйте ефекти, що відбуваються при взаємодії спіральної дворукавної хвилі з пейсмекером. Покажіть, що у випадку високої частоти циркуляції спіральної хвилі остання заглушує низькочастотний пейсмекер.

5 Промоделюйте анігіляцію хвиль, що поширюються назустріч одна одній.

6 Вивчіть поширення автохвиль у середовищі, яке містить екран з отвором.

7 Промоделюйте ефект синхронізації, який полягає у тому, що за наявності двох або більше джерел автохвиль відбувається їх взаємодія, у результаті якої високочастотні джерела заглушують низькочастотні. Зрештою настає синхронізація коливань елементів середовища: коливання відбуваються з частотою, що дорівнює частоті високочастотного джерела.

8 Промоделюйте утворення однорукавних спіральних хвиль. Оскільки спіральні хвилі утворюються на краях фронту хвилі, то спочатку необхідно задати плоску хвилю, фронт якої обривається у середині екрана.

9 Промоделюйте утворення дворукавних спіральних хвиль.

10 Візуально визначте зв'язок між частотою обертання однорукавної спіральної хвилі та параметрами середовища. Розгляньте випадок дворукавної хвилі.

11 Промоделюйте взаємодію спіральних автохвиль з автохвилями, що генеруються за допомогою низькочастотного пейсмекера.

12 Проаналізуйте поширення та анігіляцію одиночного імпульсу в активному середовищі.

13 Вивчіть поширення автохвиль в одновимірному активному середовищі за наявності пейсмекера.

14 Промоделюйте поширення одиночного імпульсу в одновимірному активному середовищі, останній елемент якого контактує з першим.

Список літератури

1. Безручко Б.П. Математическое моделирование и хаотические временные ряды / Б.П. Безручко, Д.А. Смирнов. — Саратов: ГосУНЦ «Колледж», 2005. — 320 с.

2. Мэтьюз Д. Численные методы. Использование MATLAB, 3-е издание. : пер. с англ / Д. Мэтьюз, К. Финк. — М.: Издательский дом "Вильямс", 2001. — 720 с.

3. Майер Р.В. Основы компьютерного моделирования: учебное пособие. — Глазов: ГГПИ, 2005. — 25 с.

4. Моделирование физических явлений на ЭВМ: методическое пособие в 5 частях / И.В. Кандауров, Н.А. Мезенцев, О.И. Мешков, Н.Ю. Мучной, В.Ф. Пиндюрин, Е.А. Симонов. — Новосибирск: Изд-во НГУ, 2000.

5. Коткин Г.Л. Компьютерное моделирование физических процессов с использованием MATLAB / Г.Л. Коткин, В.С. Черкасский. — Новосибирск: Изд-во НГУ, 2001. — 173 с.

6. Сергей Холодилов. Плавающая запятая // RSDN Magazine. — 2007. — №4. — С. 19–29.

7. Белоцерковский О.М. Рациональное численное моделирование в нелинейной механике / О.М. Белоцерковский, В.В. Щенников. – М.: Наука, 1990. — 123 с.

8. Яненко Н.Н. Методологические проблемы математической физики / Н.Г. Преображенский, О.С. Разумовский. -– Новосибирск: Наука, 1986. — 296 с.

9. Бородина Н.Н. Николай Николаевич Яненко. Очерки. Статьи. Воспоминания — Новосибирск: Наука, 1988. — 303 с.

10. Терёхин В.В. Моделирование в системе MATLAB: учебное пособие. — Новокузнецк: Кузбассвузиздат, 2004. — 376 с.

11. Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс. — СПб.: Питер, 2000. — 364 с.

12. Мироновский Л.А. Введение в MATLAB: учебное пособие / Л.А. Мироновский, К.Ю. Петрова. – СПб.: ГУАП., 2006. — 164 с.

13. Кузнецов С. П. Динамический хаос. — М.: Физматлит, 2001. – 296 с.

14. Колесов Ю.Б. Визуальное моделирование сложных динамических систем / Ю.Б. Колесов, Ю.Б. Сениченков. — СПб.: Изд-во Мир и Семья & Интерлайн, 2000. — 242 с.

15. Элементарный учебник физики: в 3 т. / под ред. Г.С. Ландсберга. — М.: АОЗТ "Шрайк", 1995.

16. Гулд Х. Компьютерное моделирование в физике: в 2 частях / Х. Гулд, Я. Тобочник. — М.: Мир, 1990. — Ч.1.— 400 с.

17. Лоскутов А.Ю. Введение в синергетику: учеб. руководство / А.Ю. Лоскутов, А.С. Михайлов. — М.: Наука, 1990. — 272 с.

18. Хакен Г. Синергетика и иерархии неустойчивостей в самоорганизующихся системах и устройствах. — М.: Мир, 1985. – 423 с.

19. Николис Дж. Самоорганизация в неравновесных системах / Дж. Николис, И. Пригожин — М.: Мир, 1979. — 512 с.

20. Белоусов Б.П. Периодически действующая реакция и ее механизм. // Автоволновые процессы в системах с диффузией. — Горький, 1981.

Покажчик

Додаток А
(довідковий)

Програмна реалізація методу Ромберга (до розділу 2)

Підінтегральна функція — `UserF(x)`, відрізок інтегрування — $[a, b]$, точність `Toler`. Максимальна кількість ітерацій `MaxIter=50` може бути збільшена, якщо розрахунки не забезпечують задану точність. Програма може бути модифікована таким чином, щоб кількість ітерацій вибиралася автоматично (згідно із заданою точністю), починаючи з деякого малого числа, наприклад, `MaxIter=2`.

```
double Romberg2( double a, double b, double Toler, double \
                                                UserF(double))
{
 double S;
 double NewEst[MaxIter], OldEst[MaxIter];
 double Spacing = b-a;
 OldEst[1] = Spacing * (UserF(a) + UserF(b)) / 2;
 long int Iter = 1;
 long int IterMinus2 = 1;
 do
  {
   Iter++;
         // Інтегруємо за методом трапецій.
   S = 0;
   for (int D = 1; D<=IterMinus2; D ++)
     S += UserF(a + (D - 0.5) * Spacing);
   NewEst[1] = (OldEst[1] + Spacing * S)/2;
   IterMinus2 *= 2;
         // Уточнюємо результат згідно із екстраполяцією
         // Річардсона.
   double ExtrapMinus1=1;
```

Продовження додатка А

```
   for (int Extrap = 2; Extrap<= Iter; Extrap++)
    {
     ExtrapMinus1 *= 4;
     NewEst[Extrap] = (ExtrapMinus1 * NewEst[Extrap - 1] \
              - OldEst[Extrap - 1]) / (ExtrapMinus1 - 1);
    }
   Spacing = Spacing / 2;
         // Копіюємо значення елементів масиву NewEst
         // у масив OldEst.
   memcpy(OldEst,NewEst,sizeof(NewEst));
   if (Iter>=MaxIter)
    {
      ...
         // Виводимо на екран err та Iter.
     break;
    }
 }
 while (fabs(NewEst[Iter - 1] - NewEst[Iter]) >=
                           fabs(Toler* NewEst[Iter]));
 return (NewEst[Iter]);
}
```

Осцилятор Уєди (до розділу 3)

```
double x0, v0, dt, x, v, t, A, Omega;
 ...
void Euler_K() {
       // Реалізація методу Ейлера.
 double x_n, v_n;
 t=0;
 x=x0;
 v=v0;
  ...
       // Точку з координатами (x,y) виводимо на екран.
```

Продовження додатка А

```
 while (!kbhit())
 {
  v_n=v+(-v-pow(x,3)+A*sin(Omega*t))*dt;
  x_n=x+v_n*dt;
   ...
      // Точку з координатами (x_n,v_n) виводимо на екран.
  x=x_n;
  v=v_n;
  t+=dt;
 };
};
void main() {
  x0=2; v0=3; dt=0.005; A=300; Omega=3;
  ...
  Euler_K();
};
```

Маятник (до розділу 3)

```
x=2;
p=1;
dt=0.01;
t=0;
   % Початкові умови (координата, імпульс, крок за часом).
xb=x; pb=p;
axis([-pi pi -pi pi]);  % Встановлюємо діапазон за осями.
hl=line(x,p);           % Задаємо дескриптор лінії.
set(hl,'EraseMode','none','LineStyle',':','Color','r');
      %  Команда set - встановлення параметрів лінії, що
      %  виводиться на екран. 'EraseMode','none' - режим
      %  виведення  без  видалення  попередніх  точок;
      % 'LineStyle ',':' - встановлюємо тип лінії
```

Продовження додатка А

```
      % (пунктирна); 'Color ','r' - встановлюємо   колір
      % лінії (червоний).
grid on;            % Виводимо координатну сітку.
p=p-sin(x)*dt/2;    % Просуваємося на півкроку за часом
                    % згідно з методом з переступом.
while t<100         % Цикл за часом.
  xb=x+dt*p;        % Перераховуємо координати.
  pb=p-sin(xb)*dt; % Перераховуємо імпульс.
                    % Склеювання граничних умов.
  if xb>pi
    xb=xb-2*pi;
  end;
  if xb<-pi
    xb=xb+2*pi;
  end;
  set(hl,'XData',xb,'YData',pb); % Виводимо на екран нові
                                 % точки фазової траєкторії.
  x=xb;
  p=pb;
  t=t+dt;
end;
```

Силові лінії електричного поля (до розділу 4)

```
int xbegin, ybegin;
int x[2], y[2], q[2];
int i, x_min=1, x_max=635, y_min=1, y_max=480, r_c=7;
void Input_Data();
void Calculation();
...
void main()
{
  ...
```

Продовження додатка А

```
  for(int k=0; k<5; k++) // Кількість силових ліній (5).
   {
    Input_Data();
    Calculation();
   }
}
void Input_Data()
     // Функція виводить зображення диполя на екран.
{
 x[0]=105; y[0]=115;
 x[1]=205; y[1]=115;
 q[0]=1.0; q[1]=-1.0;
 ...
 // Задаємо координати (xbegin,ybegin) початку силової лінії.
 for( int i=0; i<2; i++)
  {
   ...
     // Рисуємо заряди на екрані.
     // Координати зарядів - (x[i], y[i]).
  }
}
void Calculation()
     // Розрахунок координат точок лінії електричного поля
{
 int stop_plot=0;
 int xtmp, ytmp;
 float delta=0.1, E, Ei, Ex, Ey, dx, dy, xline, yline, r, \
       xx, yy;
 xline=xbegin;
 yline=ybegin;
 do
  {
```

Продовження додатка А

```
Ex=0.0;
Ey=0.0;
for(int i=0; i<2; i++)
 {
  dx=xline-x[i];
  dy=yline-y[i] ;
  r=sqrt(dx*dx+dy*dy);
  xx=fabs(dx);
  yy=fabs(dy);
  if(xx*xx+yy*yy>=(r_c+2)*(r_c+2))
   {
     Ei=q[i]/(r*r*r);
     Ex+=Ei*dx;
     Ey+=Ei*dy;
   }
  else
    stop_plot=1;
 }
if (stop_plot==0)
 {
  E=sqrt(Ex*Ex+Ey*Ey);
  xline+=delta*Ex/E;
  yline+=delta*Ey/E;
  xtmp=xline;
  ytmp=yline;
  if (xtmp>x_max || xtmp<x_min || ytmp>y_max || ytmp<y_min)
     stop_plot=1;
 }
if(stop_plot==0)
  {
   ...
     // Точку з координатами (xtmp,ytmp) рисуємо на екрані.
  }
```

Продовження додатка А

```
 }while(stop_plot==0);
}
```

Побудова еквіпотенціальних ліній (до розділу 4)

```
M=20;         % Розмір області M x M.
N=2;          % Кількість зарядів.
q=[20 -10];   % Величина заряду.
x=[12 10];    % x- координати зарядів.
y=[5 8];      % y- координати зарядів.
step=0.5;
[koord_x, koord_y]=meshgrid(0:step:M,0:step:M);
     % Область покриваємо сіткою із кроком step.
Ex=0;
Ey=0;
for i=1:N,
  Rx=koord_x-x(i);
  Ry=koord_y-y(i);
  j=find(abs(Rx)<step)
  Rx(j)=step;
  j=find(abs(Ry)<step)
  Ry(j)=step;
  R=(Rx.*Rx+Ry.*Ry).^(3/2);
  Ex=Ex+q(i).*Rx./R;
  Ey=Ey+q(i).*Ry./R;
end;
quiver(koord_x,koord_y,Ex,Ey)
     % Виводимо на екран поле векторів.
hold on
contour(koord_x,koord_y,sqrt(Ex.^2+Ey.^2))
     % Будуємо еквіпотенціальні лінії.
colormap hsv
plot(x-step/2,y-step/2,'o','Color','black') % Рисуємо заряди.
```

Продовження додатка А

Розрахунок потенціалу методом релаксації (до розділу 4)

```
%%%%%%%%%%%%%%%%%%
% програма potent %
%%%%%%%%%%%%%%%%%%
eps=1e-5;    % Точність розрахунків.
Nx=20;
Ny=20;       % Розмір області.
   % Задаємо розподіл потенціалу на границях області.
u=15*ones(Nx,Ny);
u(2:Nx-1,2:Ny-1)=0;
u(1,2:Ny-1)=1;
u(end,2:Ny-1)=5;
u(1:Nx,end)=8;
  % Запускаємо ітераційний процес.  Розраховуємо початковий
  % розподіл (отримуємо грубу оцінку) потенціалу у внутрішніх
  % комірках області.
for i=1:Nx
  bak=u;
  u(2:Nx-1,2:Ny-1)=(u(3:Nx,2:Ny-1)+u(1:Nx-2,2:Ny-1)+...
       u(2:Nx-1,3:Ny)+u(2:Nx-1,1:Ny-2))/4;
end;
  % Уточнюємо потенціал згідно з заданою точністю.
while abs(u(Nx/2,Ny/2)-bak(Nx/2,Ny/2))>eps
    bak=u;
    u(2:Nx-1,2:Ny-1)=(u(3:Nx,2:Ny-1)+u(1:Nx-2,2:Ny-1)+...
        u(2:Nx-1,3:Ny)+u(2:Nx-1,1:Ny-2))/4;
end;
[X,Y]=meshgrid(1:1:Nx,1:1:Ny);
[C,h]=contourf(X,Y,u,10)
text_handle = clabel(C,h);
set(text_handle,'BackgroundColor','white',...
     'FontSize',14)
```

Продовження додатка А

```
colormap gray
axis off
```

Автохвильові процеси на однорідній сітці (до розділу 5)

```
// Розмір системи N x M
#define N 100
#define M 80
  // Час збудженого (tau_s) та рефракторного (tau_r) стану.
int tau_s=6, tau_r=6, h=4;
int y[N][M], u[N][M], y_new[N][M];
void Init1()
     // Поодинока хвиля.
{
  y[N/2][M/2]=1;
};
void Init2()
     // Одноруκавна хвиля.
{
  for(int j=1; j<=25; j++)
   for(int i=1; i<=tau_s+tau_r; i++)
     y[N/2+i][j]=i;
};
void Init3()
     // Дворукавна хвиля.
{
  for(int j=1; j<=M; j++)
   for(int i=1; i<=tau_s+tau_r; i++)
     {
       y[N/2+i][j]=i;
       if (j>40)
          y[N/2+i][j]=1+tau_s+tau_r-i;
     };
```

Продовження додатка А

```
};
void main()
{
  Init1();    // Генерація поодинокої хвилі.
//  Init2();  // Генерація однорукавної хвилі.
//  Init3();  // Генерація дворукавної хвилі.
  while(!kbhit())
   {
     y[N/2][M/2]=1; //Пейсмекер.
        // Розраховуємо  фазу  елементів середовища у
        // наступний момент  часу згідно із наведеним
        // алгоритмом.
     for (int i=1; i<N-1; i++)
      for(int j=1; j<M-1; j++)
        {
          if (y[i][j]>0 && y[i][j]<tau_s+tau_r)
            y_new[i][j]=y[i][j]+1;
          if (y[i][j]==tau_s+tau_r)
            y_new[i][j]=u[i][j]=0;
          if (!y[i][j])
            for(int l=i-1; l<=i+1; l++)
             for(int r=j-1; r<=j+1; r++)
              {
                if (y[l][r]>0 && y[l][r]<=tau_s)
                  u[i][j]=u[i][j]+1;
                if (u[i][j]>=h)
                  y_new[i][j]=1;
              };
        };
   // Встановлюємо затримку для покращання візуального
   // сприйняття. Виводимо стан системи на екран.
   // Рисуємо перепону  на  екрані. Усі елементи
   // перепони у стаціонарному стані спокою.
```

Продовження додатка А

```
...
for(i=58;i<=64;i++)
 for(int j=10;j<=52;j++)
  {
   y_new[i][j]=0;
      // Рисуємо прямокутник, ліва верхня та права
      // нижня вершини якого мають координати:
      // (6*i,460-6*j) та (6*i+6,460-6*j+6).
   ...
  };
for(i=0; i<N; i++)
 for(int j=0; j<M; j++)
   {
     y[i][j]=y_new[i][j];
        // Темно-сірим кольором виводимо на екран
        // елементи, які перебувають у збудженому
        // стані.
     if (y[i][j]>=1 && y[i][j]<=tau_r)
      {
       ...
          // Рисуємо прямокутник, ліва верхня та права
          // нижня вершини якого мають координати:
          // (6*i,460-6*j) та (6*i+6,460-6*j+6).
      };
          // Світло-сірим кольором виводимо на екран
          // елементи, які перебувають у рефракторному
          // стані.
     if (y[i][j]>tau_r && y[i][j]<=tau_s+tau_r)
      {
       ...
          // Рисуємо прямокутник, ліва верхня та права
          // нижня вершини якого мають координати:
          // (6*i,460-6*j) та (6*i+6,460-6*j+6).
```

Продовження додатка А

```
          };
      };
  };
};
```

Гра "Життя" (до розділу 5)

```
m = 101;      % розмір квадратної гратки
X = sparse(m,m);
    % Визначаємо двовимірний масив, елементи якого
    % лише 0 та 1.
    % 1 - якщо клітинка є "живою", 0 - у протилежному
    % випадку. Для економії пам'яті та підвищення
    % ефективності алгоритму масив X ми визначаємо
    % як розріджену матрицю: у пам'яті PC зберігаються
    % лише ненульові елементи.
b = -1:1;
    % Оскільки поодинокі "живі" клітини "вмирають" вже
    % на другому кроці, для отримання динамічної
    % картини еволюції системи необхідно сформувати
    % групи "живих" клітин з різними комбінаціями
    % розміщення сусідів.
for ic=1:50, % Формуємо 50 груп живих клітинок.
  x=floor(rand*(m-4))+2;
  y=floor(rand*(m-4))+2;
    % Випадковим чином вибираємо координати.
  X(x+b,y+b)=(rand(3)>0.5);
    % (rand(3)>0.5) - формує випадкову матрицю розміром 3x3.
    % Якщо елемент більший за 0.5, відповідному елементу
    % матриці X з коорд. (x+b,y+b) присвоюємо 1, інакше 0.
end;
    % Знаходимо координати ненульових елементів матриці X.
[i,j] = find(X);
```

Продовження додатка А

```
    % Виводимо на екран початкову конфігурацію клітин у
    % вигляді точок з координатами (i,j).
plothandle = plot(i,j,'.', ...
      'Color','black', ...
      'MarkerSize',12);
axis([0 m+1 0 m+1]);
n = [m 1:m-1];   % north
e = [2:m 1];     % east
s = [2:m 1];     % south
w = [m 1:m-1];   % west
pause;
while 1
  N = X(n,:) + X(s,:) + X(:,e) + X(:,w) + ...
         X(n,e) + X(n,w) + X(s,e) + X(s,w);
    % Матриця N містить кількість живих сусідів кожної
    % клітинки.
  X = (X & (N == 2)) | (N == 3);
    % Якщо поточна ''жива'' клітина має дві  ''живі''
    % сусідні клітини, вона  залишається ''живою''.
    % Крім того, якщо три ''живі'' клітини оточують
    % ''мертву'' клітину, вона  стає ''живою''.
  [i,j] = find(X);
  set(plothandle,'xdata',i,'ydata',j)
  drawnow    % Перерисовуємо конфігурацію "живих" клітин.
end;
```

Навчальне видання

Князь Ігор Олександрович

Комп'ютерне моделювання динамічних систем

Розділ

"ОСНОВИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ"

Навчальний посібник

Дизайн обкладинки А. М. Вітренко, І. О. Князя
Редактор Т. Г. Чернишова
Комп'ютерне верстання І. О. Князя

Формат 60×84/16. Ум.друк. арк. 6,04. Обл.-вид.арк. 4,67. Тираж 40 прим. Зам. №

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.