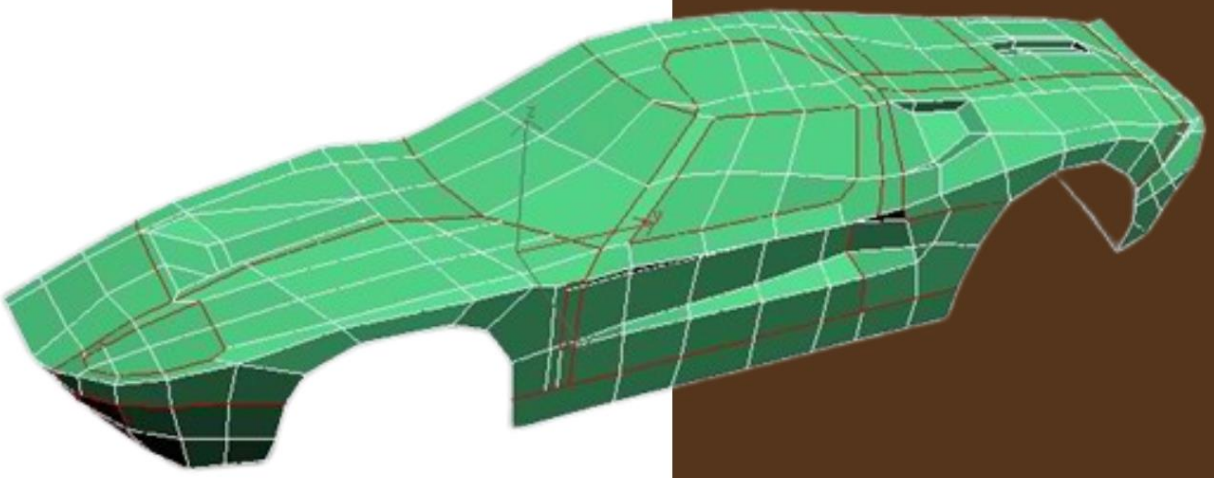
ХНТУ



МАТЕМАТИЧНІ ТА КОМП'ЮТЕРНІ ЗАСОБИ РОЗРАХУНКІВ

Навчально-методичний посібник (лабораторний практикум) для спеціальності «Автомобільний транспорт»

Сергій Русанов
Олег Клюєв
Владислав Проценко

Сергій РУСАНОВ
Олег КЛЮЄВ
Владислав ПРОЦЕНКО

МАТЕМАТИЧНІ ТА КОМП'ЮТЕРНІ ЗАСОБИ РОЗРАХУНКІВ

Навчально-методичний посібник (лабораторний практикум)
для спеціальності «Автомобільний транспорт»

Хмельницький – Херсон 2025

УДК 004.896
Р 88

Русанов С. А., Клюєв О. І., Проценко В. О.

Р 88 Математичні та комп’ютерні засоби розрахунків. Навчально-методичний посібник (лабораторний практикум) для спеціальності «Автомобільний транспорт» / С. А. Русанов, О. І. Клюєв, В. О. Проценко – Херсон : Книжкове вид-во ФОП Вишемирський В.С., 2025. – 101 с.

ISBN 978-617-8187-56-9 (електронне видання)

Лабораторний практикум призначений для студентів технічних ЗВО, зокрема для зокрема для спеціальності J8 (274) – Автомобільний транспорт.

В практикумі наведено методики та приклади виконання лабораторних робіт, які починаються з лінійної апроксимації вхідних даних за методикою обробки багатофакторного експерименту. Далі наведено роботи з інтерполяції поліномами та сплайнами, апроксимації за методом найменших квадратів, рішення систем нелінійних рівнянь за методом Ньютона-Рафсона, застосування методу скінченних елементів та прикладів роботи в сучасних CAE-системах з розрахунками конструктивних елементів автомобільної техніки. Приклади виконання надаються, окрім іншого, у вільновживаних комп’ютерних системах.

Рецензенти:

Смирнов І.В. – д.т.н., проф., проф. кафедри зварювального виробництва Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».
Носов П. С. – к.т.н., доцент, зав. каф. суднових комп’ютерних систем та мереж Херсонської державної морської академії.

Затверджено на засіданні кафедри транспортних систем і технічного сервісу, протокол № 11 від 18 червня 2025 року.

Затверджено на засіданні Вченої ради факультету інженерії та транспорту Херсонського національного технічного університету, протокол № 7 від 19 червня 2025 року.

Затверджено на засіданні Вченої ради Херсонського національного технічного університету, протокол № 1 від 28 серпня 2025 року.

УДК 004.896

ISBN 978-617-8187-55-2 (електронне видання)

Вступ

Для ефективної роботи з існуючим програмним забезпеченням, що автоматизує процеси розрахунків, користувач повинен мати уяву не тільки про середовище, у якому він працює, але й про принципи, що лежать у основі. Тобто фундаментальне знання допомагає швидко вивчити будь-яку систему з конкретним середовищем і використовувати її максимально ефективно.

Як відомо в більшості випадків не представляється можливим описати реальні об'єкти аналітично, тобто алгебраїчними рівняннями. Але ж з появою та потужним розвитком комп'ютерної техніки дуже широко було впроваджено математичні методи, за допомогою яких ми можемо вести широкий діапазон проєктних, наукових, дизайнерських, статистичних, ергономічних та інших розрахунків. Наприклад, до впровадження комп'ютерних технологій, геометрична форма об'єкта задавалася вручну за допомогою довільно побудованих ліній, а їх з'єднання забезпечували висококваліфіковані фахівці — модельники, лекальники та інші. Інакше кажучи, розв'язання інженерних завдань починалося з поетапного виконання інтерполяції та апроксимації, які виконувалися вручну. Кожен етап базувався або на результатах практичних досліджень, або на вказівках розробника — дизайнера чи конструктора. У реальному проєктуванні це особливо стосувалося формотворення складних технічних об'єктів, таких як кузови автомобілів, корпуси суден, крила літаків чи лопаті авіадвигунів, геометрія яких визначалась набором точно виготовлених перерізів.

З появою обчислювальної техніки та розвитком математичних методів стало можливим автоматизоване завдання кривих і поверхонь. Зокрема, значного поширення набули сплайн-інтерполяція та інші алгоритми, які лягли в основу сучасних CAD-систем. Застосування ж методу скінченних елементів (МСЕ) дозволило моделювати поведінку об'єкта в реальних умовах безпосередньо на комп'ютері, що було реалізовано в CAE-системах. Суть методу полягає в тому, що складні диференціальні рівняння, які описують фізичні процеси, зводяться до системи алгебраїчних (лінійних чи нелінійних) рівнянь, для яких існують ефективні чисельні методи розв'язання, що успішно застосовуються у практичних розрахунках за допомогою ЕОМ.

У сучасній автомобільній галузі математичні методи моделювання, зокрема сплайн-інтерполяція, апроксимація, методи чисельного вирішення систем лінійних і нелінійних рівнянь та метод скінченних елементів, відіграють ключову роль на всіх етапах рішення задач транспортної галузі. Ці підходи дозволяють точно описувати складні геометричні форми деталей кузова, шасі, аеродинамічних елементів і внутрішніх компонентів. Сплайн-функції особливо ефективні для побудови гладких поверхонь, які повинні не лише мати привабливий зовнішній вигляд, але й забезпечувати потрібні аеродинамічні властивості. Методи чисельного моделювання поведінки автомобіля та його

вузлів в реальних умовах дозволяють значно скоротити вартість натурних випробувань, технологічних операцій. Також вони застосовуються в ремонті, експертному аналізі, прогнозуванні.

Даний перелік лабораторних робіт починається з теми про лінійну апроксимацію вхідних даних за методикою обробки багатофакторного експерименту, потім йдуть роботи з інтерполяції поліномами та сплайнами, апроксимації за методом найменших квадратів, рішення систем нелінійних рівнянь за методом Ньютона-Рафсона, застосування методу скінченних елементів та прикладів роботи в сучасних CAE-системах з розрахунками конструктивних елементів автомобільної техніки. Приклади виконання надаються, окрім іншого, у вільновживаних комп’ютерних системах.

Лабораторна робота №1

Тема: Алгоритми розрахунку багатопараметричної моделі. Багатофакторний експеримент

1.1. Загальні положення

У прикладних задачах часто виникає необхідність визначення параметрів певної системи, що передбачає створення її формальної математичної моделі. Для побудови такої моделі використовуються експериментальні дані, які є результатом серії вимірювань, проведених відповідно до визначеного плану. У найпростішому випадку цей план містить опис умов, за яких здійснювались вимірювання, та значення вхідних параметрів (або *факторів*).

У загальному випадку реакція (відгук) системи описується функцією, яка залежить від *n* змінних

$$f = f(x_1, x_2, ..., x_n). \tag{1.1}$$

Математичну модель системи отримують шляхом наближення (апроксимації — див. лаб. роб. №2, 3) реальної функції певною обраною функцією, наприклад, лінійною.

$$y = a_0 + a_1 x_1 + a_2 x_2 + ... + a_n x_n. \tag{1.2}$$

де $a_0, a_1, a_2, ..., a_n$ — параметри моделі, що шукаються.

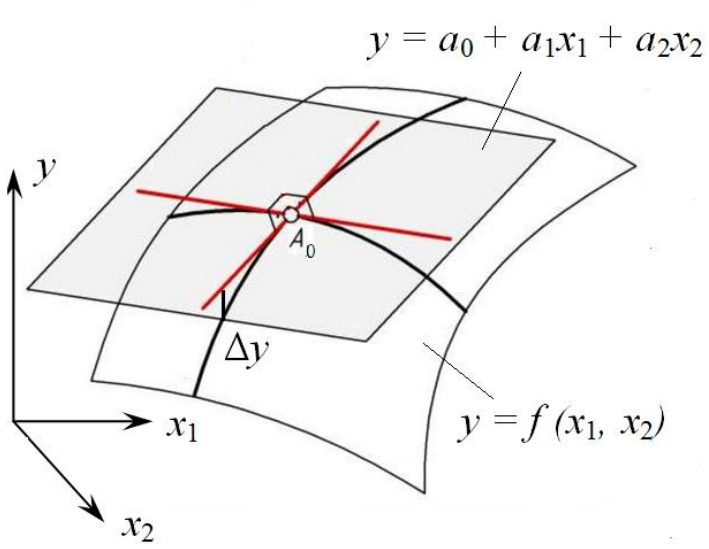


Рис. 1.1. Процес побудови лінійної моделі, яка наближено описує більш складну функціональну залежність

На рис. 1.1 схематично зображено процес побудови лінійної моделі, яка наближено описує більш складну функціональну залежність. Хоча сама функція є нелінійною, у безпосередній близькості до точки A_0 її можна замінити дотичною площиною. У межах цієї області похибка моделі не перевищує Δy.

Якщо відомі коефіцієнти цієї моделі, можна з певною точністю прогнозувати значення функції, а отже і поведінку системи поблизу точки A_0. Визначення цих коефіцієнтів і є основною метою експериментального дослідження.

Припустимо, що вхідні параметри технологічного процесу мають такі значення: $x_{1C} = 55$, $x_{2C} = 30$.

Візьмемо верхні та нижні значення обох факторів так, щоб вони розташовувалися симетрично щодо поточного значення, наприклад

$$x_{1B} = 60, x_{2B} = 35,$$
$$x_{1H} = 50, x_{2H} = 25.$$

Сформуємо таблицю, в якій усі можливі комбінації значень обох факторів представлені повним переліком, та виконаємо вимірювання в кожній із цих точок (значення відгуку наведено умовно)

x_1	x_2	y
50	25	140
50	35	210
60	25	170
60	35	220

Вважаючи, що лінійна модель процесу має вигляд

$$y = a_0 + a_1 x_1 + a_2 x_2 , \tag{1.3}$$

на основі заданих експериментальних даних можна побудувати систему з чотирьох рівнянь. Нижче наведено цю систему, а також її компактне подання у вигляді матриці. Такий тип матриці прийнято називати *матрицею експерименту*.

$$\begin{cases} a_0 + 50a_1 + 25a_2 = 140 \\ a_0 + 50a_1 + 35a_2 = 210 \\ a_0 + 60a_1 + 25a_2 = 170 \\ a_0 + 60a_1 + 35a_2 = 220 \end{cases}; \left(\begin{array}{cccc} x_0 & x_1 & x_2 & y \\ \hline 1 & 50 & 25 & 140 \\ 1 & 50 & 35 & 210 \\ 1 & 60 & 25 & 170 \\ 1 & 60 & 35 & 220 \end{array}\right).$$

У матриці експерименту другий і третій стовпці відображають значення факторів, четвертий — зафіксовані результати (відгук) системи, а перший стовпець заповнений одиницями, що відповідають вільному члену

моделі, тобто коефіцієнтам при a_0. Вважатимемо цей стовпець деяким віртуальним фактором x_0, який завжди набуває поодиноких значень.

Для спрощення розв'язання системи рівнянь здійснимо нормування факторів, тобто зробимо таку заміну змінних, щоб максимальному значенню фактора відповідало нормоване значення +1, мінімальному — значення -1, а середньому — 0. Таке нормування фактора вочевидь можна виконати за формулою:

$$\tilde{x}_i = \frac{2(x_i - x_{iC})}{x_{iB} - x_{iH}} = \frac{2x_i - x_{iB} - x_{iH}}{x_{iB} - x_{iH}}. \tag{1.4}$$

Перехід від нормованих до ненормованих факторів здійснюється тоді зворотним перетворенням

$$x_i = \tilde{x}_i \frac{x_{iB} - x_{iH}}{2} + x_{iC} = \tilde{x}_i \frac{x_{iB} - x_{iH}}{2} + \frac{x_{iB} + x_{iH}}{2}. \tag{1.5}$$

Будемо вважати, що рівняння моделі *формінваріантно* до вказаних перетворень, тобто відносно нормованих факторів модель буде мати аналогічний вигляд з точністю до позначень:

$$y = \tilde{a}_0 + \tilde{a}_1 \tilde{x}_1 + \tilde{a}_2 \tilde{x}_2 \,. \tag{1.6}$$

Тоді, щоб знайти параметри моделі для ненормованих координат, підставимо вирази для нормованих координат в рівняння моделі (1.6):

$$\begin{aligned}
&y = \tilde{a}_0 + \tilde{a}_1 \tilde{x}_1 + \tilde{a}_2 \tilde{x}_2 = \\
&= \tilde{a}_0 + \tilde{a}_1 \frac{2(x_1 - x_{1C})}{x_{1B} - x_{1H}} + \tilde{a}_2 \frac{2(x_2 - x_{2C})}{x_{2B} - x_{2H}} = \\
&= \tilde{a}_0 - \tilde{a}_1 \frac{2x_{1C}}{x_{1B} - x_{1H}} - \tilde{a}_2 \frac{2x_{2C}}{x_{2B} - x_{2H}} + \frac{2\tilde{a}_1}{x_{1B} - x_{1H}} x_1 + \frac{2\tilde{a}_2}{x_{2B} - x_{2H}} x_2 = \\
&= \tilde{a}_0 - \tilde{a}_1 \frac{x_{1B} + x_{1H}}{x_{1B} - x_{1H}} - \tilde{a}_2 \frac{x_{2B} + x_{2H}}{x_{2B} - x_{2H}} + \frac{2\tilde{a}_1}{x_{1B} - x_{1H}} x_1 + \frac{2\tilde{a}_2}{x_{2B} - x_{2H}} x_2.
\end{aligned}$$

Порівнюючи останній вираз з виразом для лінійної моделі в ненормованих координатах (1.3) отримаємо вирази для параметрів моделі:

$$\begin{aligned}
a_0 &= \tilde{a}_0 - \tilde{a}_1 \frac{x_{1B} + x_{1H}}{x_{1B} - x_{1H}} - \tilde{a}_2 \frac{x_{2B} + x_{2H}}{x_{2B} - x_{2H}}; \\
a_1 &= \frac{2\tilde{a}_1}{x_{1B} - x_{1H}}; \\
a_2 &= \frac{2\tilde{a}_2}{x_{2B} - x_{2H}};
\end{aligned} \tag{1.7}$$

З урахуванням нормування факторів система рівнянь та матриця експерименту набудуть наступного вигляду:

$$\begin{array}{l}\tilde{a}_0 - \tilde{a}_1 - \tilde{a}_2 = 140\\ \tilde{a}_0 - \tilde{a}_1 + \tilde{a}_2 = 210\\ \tilde{a}_0 + \tilde{a}_1 - \tilde{a}_2 = 170\\ \tilde{a}_0 + \tilde{a}_1 + \tilde{a}_2 = 220\end{array}; \left(\begin{array}{cccc} \tilde{x}_0 & \tilde{x}_1 & \tilde{x}_2 & y \\ \hline +1 & -1 & -1 & 140 \\ +1 & -1 & +1 & 210 \\ +1 & +1 & -1 & 170 \\ +1 & +1 & +1 & 220 \end{array}\right).$$

Така система є перевизначеною — число рівнянь більше кількості невідомих. Тому знайдемо деякі осереднені значення коефіцієнтів моделі наступним чином. Оскільки сума членів у другому та третьому стовпці матриці дорівнює нулю, вільний член моделі можна знайти, склавши всі чотири рівняння:

$$4\,\tilde{a}_0 = 140 + 210 + 170 + 220 = 740;$$
$$\tilde{a}_0 = 185.$$

Для визначення будь-якого іншого коефіцієнта моделі необхідно змінити знаки в системі рівнянь так, щоб у відповідному стовпці всі значення стали рівними одиниці. Після цього слід підсумувати всі чотири рівняння.

$$4\,\tilde{a}_1 = -140 - 210 + 170 + 220 = 40;$$
$$\tilde{a}_1 = 10.$$
$$4\,\tilde{a}_2 = -140 + 210 - 170 + 220 = 120;$$
$$\tilde{a}_2 = 30.$$

Отже, у близькому до точки (55, 30) діапазоні значень лінійна модель процесу має такий вигляд:

$$y = 185 + 10\tilde{x}_1 + 30\tilde{x}_2.$$

У загальному випадку рішення системи виглядатиме як

$$\tilde{a}_k = \frac{1}{2^n}\sum_{i=1}^{2^n} y_i \tilde{x}_{ki}$$

Зробимо зворотній перехід за формулами (1.7), і тоді отримаємо

$$a_0 = 185 - 10 \cdot \frac{60+50}{60-50} - 30 \cdot \frac{35+25}{35-25} = -105;$$
$$a_1 = \frac{2 \cdot 10}{60-50} = 2;$$
$$a_2 = \frac{2 \cdot 30}{35-25} = 6.$$

Остаточно маємо модель у природних координатах:

$$y = -105 + 2x_1 + 6x_2.$$

1.2. Повний факторний експеримент

У загальному вигляді матриця повного факторного експерименту з *n* факторами має вигляд

$$\left(\begin{array}{cccc|c} \tilde{x}_0 & \tilde{x}_1 & \dots & \tilde{x}_n & y \\ \hline +1 & -1 & \dots & -1 & y_1 \\ +1 & -1 & \dots & +1 & y_2 \\ \dots & \dots & \dots & \dots & \dots \\ +1 & +1 & \dots & +1 & y_{2^n} \end{array}\right). \tag{1.8}$$

Коефіцієнти лінійної моделі в нормованих координатах обчислюються за формулами:

$$\tilde{a}_k = \frac{1}{2^n} \sum_{i=1}^{2^n} y_i \tilde{x}_{ki} \qquad (k = 0...n); \tag{1.9}$$

Коефіцієнти лінійної моделі у природних (ненормованих) координатах обчислюються як:

$$a_0 = \tilde{a}_0 - \sum_{i=1}^{n} \tilde{a}_i \frac{x_{iB} + x_{iH}}{x_{iB} - x_{iH}};$$
$$a_k = \frac{2\tilde{a}_k}{x_{kB} - x_{kH}} \qquad (k = 1...n). \tag{1.10}$$

Перетворення природних факторів на нормовані і назад відбувається за формулами

$$\tilde{x}_i = \frac{2x_i - x_{iB} - x_{iH}}{x_{iB} - x_{iH}};$$
$$x_i = \tilde{x}_i \frac{x_{iB} - x_{iH}}{2} + \frac{x_{iB} + x_{iH}}{2}. \tag{1.11}$$

ЗАВДАННЯ: провести засобами доступних систем комп’ютерної математики

(СКМ) розрахунок повного двофакторного експерименту. Вхідні дані задаються індивідуально.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Скидання всіх змінних і визначень */
restart;

/* Підключення необхідних бібліотек */
load(combinat);
load(linalg);

/* Визначення кількості змінних */
n: 2;

/* Набір значень для змінних */
E: [1, -1];

/* Генерація декартового добутку */
T: cartprod([[1], E^n]);
i: 0;

/* Цикл для створення всіх можливих комбінацій */
while not(T.finished) do (
    T.nextvalue(),
    i: i + 1,
    M[i]: T.value()
);

/* Перетворення результату в список */
Ml: convert(M, list);

/* Визначення значень змінних */
y: array(1..length(Ml));
y[1]: 0.71;
y[2]: 0.9;
y[3]: 1.36;
y[4]: 7.28;

/* Визначення межі змінних */
Xmax: [4, 11];
Xmin: [0, 5];

/* Формування виразу і його перетворення в список */
A: sort(convert(sort(expand(product((x[ic]+1), ic=1..n)), [x[ic]$ic=1..n]),
list), length);

/* Обчислення значень функцій для кожної комбінації змінних */
for i: 1 thru length(Ml) do (
    f[i]: subst((x[ic]=Ml[i][ic+1])$ic=1..n, A)
);

/* Формування матриці функцій */
F: array(1..length(Ml), 1..length(Ml));
for i: 1 thru length(Ml) do (
    for j: 1 thru length(Ml) do (
        F[i, j]: f[i][j]
    )
);
```

```
/* Виведення матриці функцій та значень */
print(F);
print(y);

/* Перетворення векторів у формат Maxima */
Y: convert(y, vector);
Fm: convert(F, matrix);

/* Обчислення результату */
result: evalm(evalf(multiply(Y, Fm) / length(Ml)));
B: convert(result, vector);

/* Перетворення виразу векторів */
a: convert(A, vector);
C: multiply(B, A);

/* Нормалізація виразу */
c: expand(subst((x[ic]=2*(X[ic]-Xmin[ic])/(Xmax[ic]-Xmin[ic])-1)$ic=1..n,
C));
```

Система **Maple 10**:

```
restart:
with(combinat, cartprod):
with(linalg):
n:=2:

E:={1,-1}:
T:=cartprod([[1],E$n]):i:=0:
while not T[finished] do T[nextvalue]():i:=1+i: M[i]:=%%: od:
Ml:=convert(M,list);
y:=array(1..nops(Ml)):
y[1]:=0.71;
y[2]:=0.9;
y[3]:=1.36;
y[4]:=7.28;
Xmax[1]:=4:Xmin[1]:=0:
Xmax[2]:=11:Xmin[2]:=5:

A:=sort(convert(sort(expand(product((x[ic]+1),ic=1..n)),[x[ic]$ic=1..n]),list
),length);
for i to nops(Ml) do
f[i]:=subs((x[ic]=Ml[i][ic+1])$ic=1..n,A) od:
F:=array(1..nops(Ml),1..nops(Ml)):
for i to nops(Ml) do
for j to nops(Ml) do
F[i,j]:=f[i][j] od od:
print(F);
print(y);
Y:=convert(y,Vector):
Fm:=convert(F,matrix):
evalm(evalf(multiply(Y,Fm)/nops(Ml)));
B:=convert(%,vector);
a:=convert(A,Vector);
C:=multiply(B,A);
c:=expand(subs((x[ic]=2*(X[ic]-Xmin[ic])/(Xmax[ic]-Xmin[ic])-1)$ic=1..n,C));
```

$$y_1 := 0.71$$

$$y_2 := 0.9$$

$$y_3 := 1.36$$

$$y_4 := 7.28$$

$$A := [1, x_1, x_2, x_1 x_2]$$

$$\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

[0.71, 0.9, 1.36, 7.28]

[2.562500000, 1.757500000, 1.527500000, 1.432500000]

$$B := [2.562500000, 1.757500000, 1.527500000, 1.432500000]$$

$$a := \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$$

$$C := 2.562500000 + 1.757500000\, x_1 + 1.527500000\, x_2 + 1.432500000\, x_1 x_2$$

$$c := 0.551666667 - 1.031250000\, X_1 + 0.0316666667\, X_2 + 0.2387500000\, X_1 X_2$$

Контрольні питання.

1. Що таке багатофакторний експеримент і які його основні цілі в інженерному моделюванні?
2. Що розуміють під терміном «багатопараметрична модель» у контексті технічних систем?
3. Які етапи включає побудова багатофакторної моделі для технічного об'єкта?
4. Яким чином формується матриця планування експерименту?
5. Як і для чого проводиться нормування факторів?

Лабораторна робота №2

Тема: Інтерполяція поліномами Лагранжа, Ньютона та Ерміта. Інтерполяція сплайнами

2.1. Загальні положення

Як правило, процес моделювання в CAD-системах включає такі основні етапи:

- спочатку визначають координати обмеженої кількості точок, що називаються опорними (вони можуть бути отримані розрахунковим шляхом або в результаті експерименту);
- через ці опорні точки будуються гладкі криві або поверхні;
- отриману на певному етапі ітерацій модель (аналітичне представлення) поступово вдосконалюють і коригують до досягнення бажаної форми кривої чи поверхні.

Отже, основне завдання полягає в точному математичному описі геометричної форми об’єкта на основі координат точок, що розташовані на його поверхні (рис. 2.1).

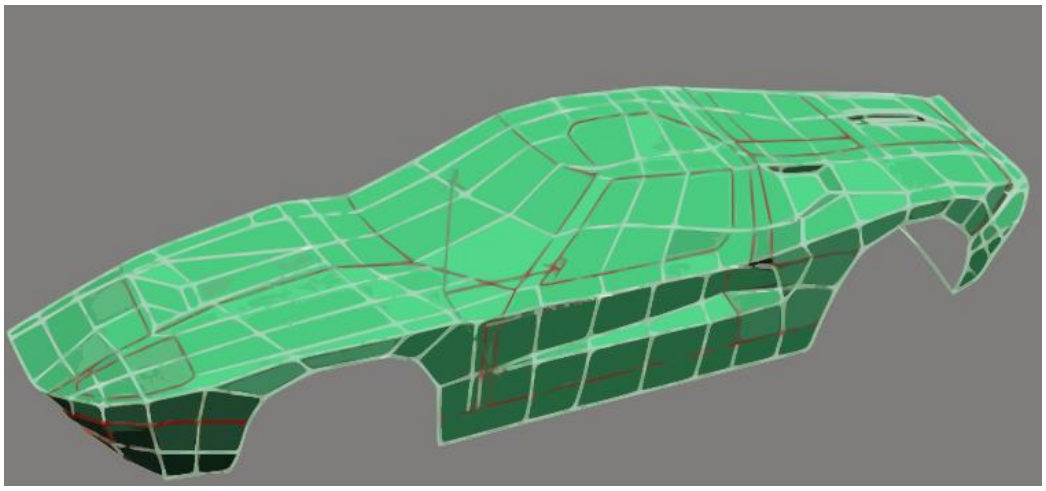


Рис. 2.1. Приклад інтерполяції поверхнями кузову автомобіля

У практиці моделювання кривих і поверхонь високого порядку широко застосовують параметричне представлення цих геометричних об’єктів. Відмову від явного та неявного способів завдання кусочно-гладких кривих пояснюють такими причинами:

- при явному представленні неможливо однозначно описати замкнені криві, оскільки їх потрібно розбивати на окремі сегменти і описувати кожен з них окремо; крім того, отримані описи залежать від орієнтації в просторі і не є інваріантними до поворотів, що ускладнює обчислення після трансформацій;

- при неявному представленні часто виникає проблема неоднозначності розв'язку рівнянь, також існують криві, які не мають точного аналітичного виразу або задаються системою рівнянь; це ускладнює об'єднання сегментів таких кривих.

При моделюванні кривих і поверхонь зазвичай постають такі задачі:

- *апроксимація* — заміна складних функцій, що описують криву або поверхню, простішими, які дають приблизне представлення без значної втрати точності; при цьому крива може не проходити через задані точки, але задовольняти певним умовам щодо них;

- *інтерполяція* — пошук гладких кривих або поверхонь, які проходять через задану множину точок (див. рис. 2.1);

- *згладжування* — забезпечення того, щоб шукані крива або поверхня описувалися функцією, яка відповідає заданим вимогам, наприклад, необхідному ступеню диференційовності.

Дуже часто задачу, яка відповідає задачі згладжування називають також задачею апроксимації (рис. 2.2).

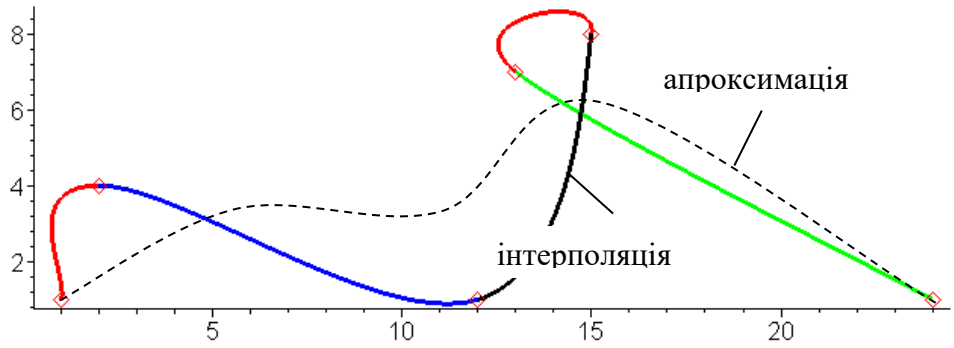


Рис. 2.2. Приклад інтерполяції й апроксимації по заданих точках

Сформульовані задачі відносяться до групи *чисельних методів*. В цій роботі будемо розглядати задачі інтерполяції.

2.2. Поліноміальні інтерполяції

Інтерполяція поліномами: нехай заданий набір точок на площині:

$$\mathbf{p} = \{[x_0,y_0], [x_1,y_1], \ldots, [x_n,y_n]\}.$$

У загальному вигляді поліном представляється як:

$$y(x) = a_0 + a_1x + a_2x^2 + \ldots + a_n{}^n\, x^n.$$

Задача зводиться до визначення $a_0, a_1, a_2, \ldots a_n$.

Невідомі коефіцієнти знаходять з формул:

$$\begin{cases} y(x_0) = y_0, \\ y(x_1) = y_1, \\ \dots \\ y(x_n) = y_n. \end{cases}$$

Взагалі, робота з поліномами обмежена невеликою кількістю точок, у зв'язку з тим, що зі збільшенням кількості точок збільшується порядок полінома, що приводить до некоректних результатів — так званих *осциляцій*.

2.3. Базисні форми

Представлення елемента деякого лінійного простору у вигляді лінійної комбінації базисних елементів (суми базисних елементів із числовими коефіцієнтами), наприклад

$$a = \sum_{i=1}^{n} \alpha_i a_i$$

називається розкладанням цього елемента *по базису*. В базисній формі можна представити також інтерполяційні поліноми, тоді в якості базисних елементів будуть використовувані базисні функції, що залежать від деякого параметру або від незалежної координати. Розглянемо базисну форму на прикладі поліномів Лагранжу.

2.4. Інтерполяція поліномами Лагранжа

Інтерполяція *поліномами Лагранжа* має вигляд (базисна форма):

$$y = \sum_{i=0}^{n} y_i \varphi_i(x),$$

де базисні функції будуть мати вигляд:

$$\varphi_i(x) = \prod_{j=0}^{n} \frac{x - x_j}{x_i - x_j}, i \neq j,$$

або

$$y = \sum_{i=0}^{n} y_i \prod_{j=0}^{n} \frac{x - x_j}{x_i - x_j}, i \neq j,$$

де n — порядок поліному.

Для першого порядку одержуємо лінійну інтерполяцію:

$$y = y_1 \frac{x - x_1}{x_0 - x_1} + y_0 \frac{x - x_0}{x_1 - x_0}.$$

Для полінома другого порядку:

$$y = y_0 \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2} + y_1 \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2}, n = 2.$$

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики візуалізацію інтерполяції Лагранжа за довільним масивом точок. Вхідні дані задаються індивідуально.

ПРИКЛАД ВИКОНАННЯ.

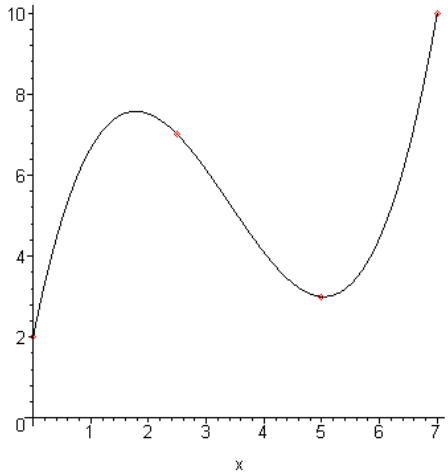
Система **Maxima**:

```
/* Очистка змінних */
restart;

/* Завдання точок */
P: [[0, 2], [2.5, 7], [5, 3], [7, 10]];

/* Визначення поліному */
n: length(P);  /* количество точек */
Y: sum(a[i] * x^(i - 1), i, 1, n);

/* Формування рівнянь */
Ur: makelist(subs(x = P[i][1], Y) = P[i][2], i, 1, n);

/* Рішення системи рівнянь */
Res: solve(Ur, a);

/* Побудова графіку */
A: plot2d([sum(a[i] * x^(i - 1), i, 1, n)], [x, 0, P[n][1]], [color, black]);
B: plot2d([discrete(P, style = point)]);

/* Відображення графіків */
display(A, B);
```

Система **Maple 10**:

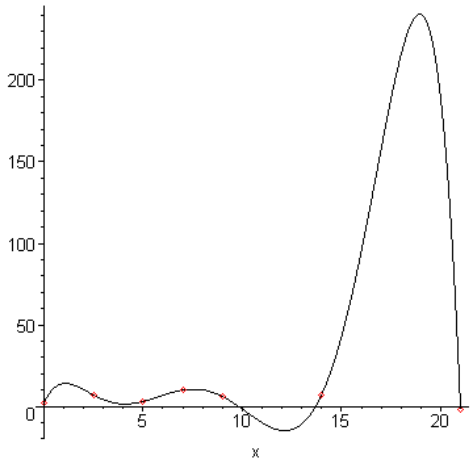
```
restart;
P:=[0, 2], [2.5, 7], [5, 3], [7, 10];
Y:=sum(a[ic]*x^(ic-1), ic=1..nops([P]));
for i to nops([P]) do
 Ur[i]:=subs( x=P[i][1], Y)  =P[i][2];
od;
Res:=solve( {Ur[ic]$ic=1..nops([P])}, [a[ic]$ic=1..nops([P])] );
```

```
A:=plot( [sum(a[ic]*x^(ic-1), ic=1..nops([P])), 0], x=0..P[-1][1],
color=black):
B:=plot([P], style=point):
plots[display]([A,B]);
```



Приклад осциляцій:

```
restart;
P:=[0, 2], [2.5, 7], [5, 3], [7, 10], [9, 6], [14, 7.3], [21, -2];
Y:=sum(a[ic]*x^(ic-1), ic=1..nops([P]));
for i to nops([P]) do
 Ur[i]:=subs( x=P[i][1], Y)  =P[i][2];
od;
Res:=solve( {Ur[ic]$ic=1..nops([P])}, [a[ic]$ic=1..nops([P])] );
A:=plot( [sum(a[ic]*x^(ic-1), ic=1..nops([P])), 0], x=0..P[-1][1],
color=black):
B:=plot([P], style=point):
plots[display]([A,B]);
```



2.5. Інтерполяція поліномами Ньютона

Інтерполяцію *поліномами Ньютона* можна так само представити у базисному вигляді:

$$\sum_{j=0}^{n} c_j N_j(x),$$

де $N_j(x)$ — базисні функції.

Базисні функції представляються у вигляді:

$$N_i(x) = \prod_{j=0}^{n}(x - x_j), N_0(x) = 1.$$

Коефіцієнти розкладення мають вигляд:

$$c_j = f_i - \sum_{j=0}^{n} c_j N_j(x),$$
$$c_0 = f_0.$$

Для двох точок коефіцієнти перепишуться у вигляді:

$$N_1 = t - t_0,$$
$$c_1 = f_1 - \frac{c_0}{t - t_0},$$
$$N_2 = (t - t_0)(t - t_1).$$

Для більшої кількості точок поліноми Ньютона поводяться так само, як поліноми Лагранжа.

2.6. Інтерполяція поліномами Ерміта

У деяких вузлах інтерполяції необхідно контролювати не тільки значення функції, але й кути її нахилу. У цьому випадку інтерполяція здійснюються *поліномами Ерміту*.

Для двохвузлової інтерполяції поліном Ерміта прийме вигляд:

$$y(x) = \sum_{i=0}^{n} z_i \varphi_i(x),$$

де

$$\varphi_0(x) = (2x + 1)(1 - x)^2,$$
$$\varphi_1(x) = (3 - 2x)x^2,$$
$$\varphi_2(x) = x(1 - x)^2,$$
$$\varphi_3(x) = (x - 1)x^2.$$

Перші два вирази контролюють положення вузлів, другі — кут нахилу у вузлах.

ЗАВДАННЯ: провести засобами доступних систем комп’ютерної математики візуалізацію інтерполяції поліномами Ньютона та Ерміта за довільним

масивом точок.

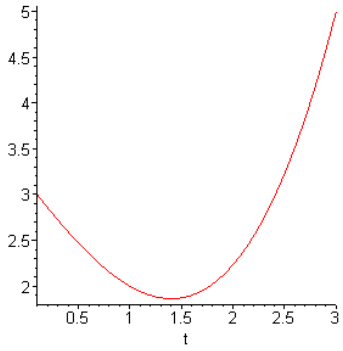
ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Очищення змінних */
restart;

/* Введення точок */
F: [[0.1, 3], [1, 2], [3, 5]];
m: length(F);

/* Ініціалізація N и c */
N: makelist(0, i, 0, m-1);
c: makelist(0, i, 0, m-1);

/* Завдання N[0] */
N[0]:= 1;

/* Заповнення N и c */
for i:1 thru m-1 do (
    N[i]:= product(t - F[j][1], j, 0, i-1),
    c[i]:= (F[i+1][2] - sum(c[j] * subst(t = F[i+1][1], N[j]), j, 0, i-1)) /
subst(t = F[i+1][1], N[i])
);

/* Ініціалізація c[0] */
c[0]:= F[1][2];

/* Завдання поліному */
polynomial: sum(c[i] * N[i], i, 0, m-1);

/* Відображення графіку */
plot2d(polynomial, [t, F[1][1], F[m][1]], [color, black], [xlabel, "t"],
[ylabel, "f(t)"], [title, "Piecewise Polynomial Interpolation"]);
```

Система **Maple 10**:

```
restart:
F:=[[0.1,3], [1, 2], [3,5]];
m:=3;
N[0]:=1;
for i to m do
 N[i]:=product(t-t[j], j=0..i-1);
 c[i]:=(f[i] - sum(c[j]*subs(t[j]=t[i],N[j]), j=0..i-1)/N[i])
od;
c[0]:=F[1][2];
for i to m-1 do
 N[i]:=subs((t[ic-1]=F[ic][1])$ic=1..m, product(t-t[j], j=0..i-1));
 c[i]:=(F[i+1][2] - sum(c[j]*subs(t=F[i+1][1], N[j]), j=0..i-
1))/subs(t=F[i+1][1], N[i])
od;
sum(c[ic]*N[ic],ic=0..m-1);
plot(sum(c[ic]*N[ic],ic=0..m-1), t=F[1][1]..F[-1][1]);
```

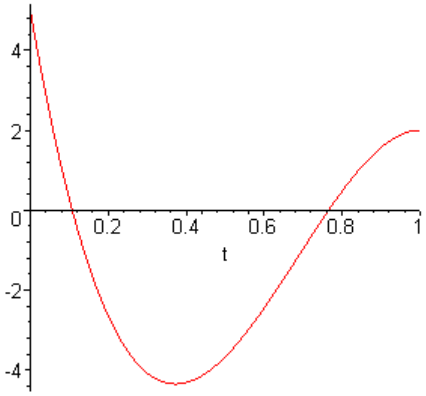


Система **Maxima**:

```
/* Визначення функцій phi */
phi[0] := (2*t + 1)*(1 - t)^2;
phi[1] := (3 - 2*t)*t^2;
phi[2] := t*(1 - t)^2;
phi[3] := (t - 1)*t^2;

/* Підставлення t = 0 и t = 1 в phi */
subs(t = 0, [phi[0], phi[1], phi[2], phi[3]]);
subs(t = 1, [phi[0], phi[1], phi[2], phi[3]]);

/* Визначення похідних phi по t та підставлення t = 0 и t = 1 */
subs(t = 0, diff([phi[0], phi[1], phi[2], phi[3]], t));
subs(t = 1, diff([phi[0], phi[1], phi[2], phi[3]], t));

/* Визначення функції v(t) */
v(t) := sum(z[i] * phi[i], i, 0, 3);

/* Підставлення значень для z та побудова функції V */
z_values: [5, 2, evalf(tan(%pi * 91 / 180)), tan(0)];
V(t) := subst(z[i] = z_values[i], v(t));

/* Побудова графіку функції V(t) */
plot2d(V(t), [t, 0, 1], [xlabel, "t"], [ylabel, "V(t)"], [title, "Plot of
V(t)"]);
```

Система **Maple 10**:

```
phi[0]:=(2*t+1)*(1-t)^2;
phi[1]:=(3-2*t)*t^2;
phi[2]:=t*(1-t)^2;
phi[3]:=(t-1)*t^2;
subs(t=0, [phi[ic]$ic=0..3]);
subs(t=1, [phi[ic]$ic=0..3]);
subs(t=0, diff([phi[ic]$ic=0..3],t));
subs(t=1, diff([phi[ic]$ic=0..3],t));
v(t):=sum(z[ic]*phi[ic], ic=0..3);
V:=subs(z[0]=5,z[1]=2,z[2]=evalf(tan(Pi*91/180)),z[3]=tan(0), v(t));
plot(V, t=0..1);
```



2.7. Лінійні та кубічні сплайни. Вимоги до кривих, що моделюються

Задачі апроксимації й інтерполяції можуть бути вирішені з використанням різних обчислювальних методів. Для запобігання осциляцій, що були наявні в попередніх інтерполяційних задачах, проводиться апроксимація та інтерполяція функціями, область визначення яких розбита на куски, при цьому на кожному з кусків функція є деяким поліномом, що дає набори відрізків, дуг окружностей, парабол й кривих більш високих порядків.

Усі ці методи використовують однаковий підхід:

- розбивка просторової або плоскої кривої на окремі сегменти (елементарні куски);
- опис цих сегментів поліномами;
- завдання умови з'єднання сегментів у єдині криві, які називають *сплайновими кривими*.

Розглянемо інтерполяцію найпростішим типом сплайнів — лінійними сплайнами.

2.8. Лінійні сплайни

Лінійний сплайн — це сплайн, складений з поліномів першого ступеня, тобто з відрізків прямих ліній. Точність інтерполяції лінійними сплайнами невисока, також слід зазначити, що вони не забезпечують безперервності навіть перших похідних. Однак у деяких випадках кусочно-лінійна апроксимація функції може виявитися переважніше, чим апроксимація більш високого порядку. Наприклад, лінійний сплайн зберігає монотонність переданого в нього набору точок.

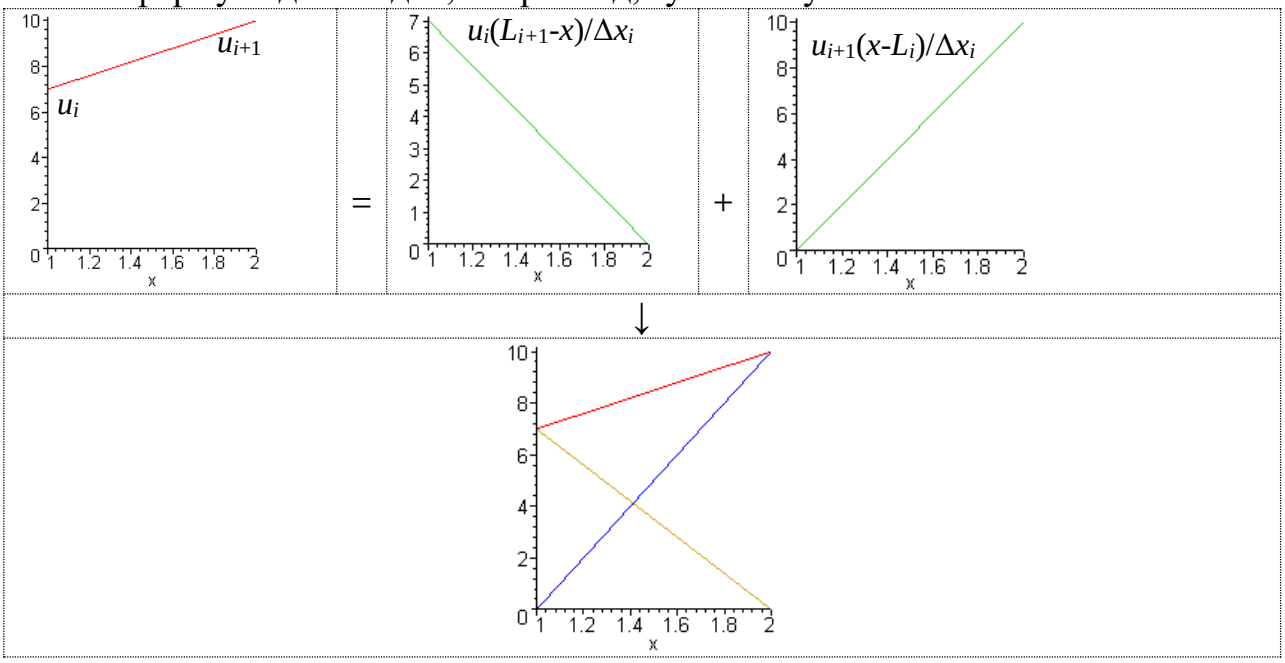
2.9. Базисна форма лінійних сплайнів

В задачах інтерполяції даних сплайнами, область визначення задачі розбита на n ділянок, на яких задаються т.з. *базисні функції* (див. 2.3). Але ж перед тим, як їх безпосередньо розглянути, згадаємо принципи лінійної інтерполяції даних. Нехай на відрізку Δx_i від L_i до L_{i+1} деяка функція приймає

значення u_i та u_{i+1} відповідно. Тоді лінійна інтерполяція на такій ділянці (пряма лінія $ax+b$, з граничними умовами $u(L_i) = u_i$, $u(L_{i+1}) = u_{i+1}$) може бути переписана інакше як

$$u_i(x)=u_i(L_{i+1} - x)/\Delta x_i + u_{i+1}(x - L_i)/\Delta x_i,$$

— в такому вигляді це класична формула лінійного балансу між точками, така ж формула дійсна для, наприклад, густини суміші газів:

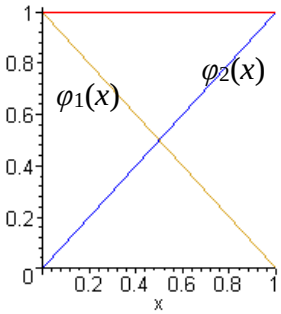


Функції

$$\varphi_{1i}(x) = (L_{i+1} - x)/\Delta x_i,$$
$$\varphi_{2i}(x) = (x - L_i)/\Delta x_i,$$

власне й мають назву *базисних функцій*. Вони змінюються від 0 до 1, та, наприклад, для L_i=0 та L_{i+1}=1 вони будуть мати особливо простий вигляд:

$$\varphi_1(x) = 1 - x;$$
$$\varphi_2(x) = x:$$

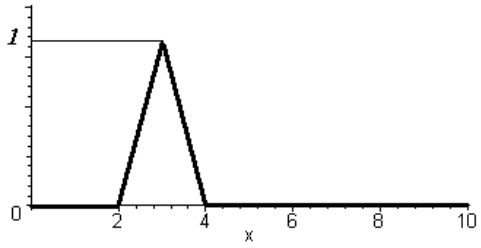


Більш того, ділянка [0,1] взаємно-однозначно відображується на ділянку $[L_i, L_{i+1}]$, але ж першою набагато легше працювати, тому звичайно працюють саме з ділянками від 0 до 1 з послідуючим відображенням. Ми

надалі будемо викладати в прямому вигляді, «як є», для більшої наочності. Таким чином загалом маємо:

$$\mathrm{u}_i(x) = u_i\varphi_{1i}(x) + u_{i+1}\varphi_{2i}(x),$$
$$\mathrm{u}(x) = \sum \mathrm{u}_i\omega_i$$

Функції ω_i є, вочевидь, *ваговими*, але ж їх крім того також називають, як вже оговорювалось, *базисними*, або *функціями форми*. Дійсно, важливо те, що ми перейшли до форми, яка покладена в основу базисного представлення, але ж з особливими функціями, які мають для нашого випадку лінійної інтерполяції характерний трикутний вигляд:



та, крім того, як ми записали — є адитивними (підсумовуються), і *лінійно незалежними*.

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики візуалізацію інтерполяції лінійними сплайнами за довільним масивом точок.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

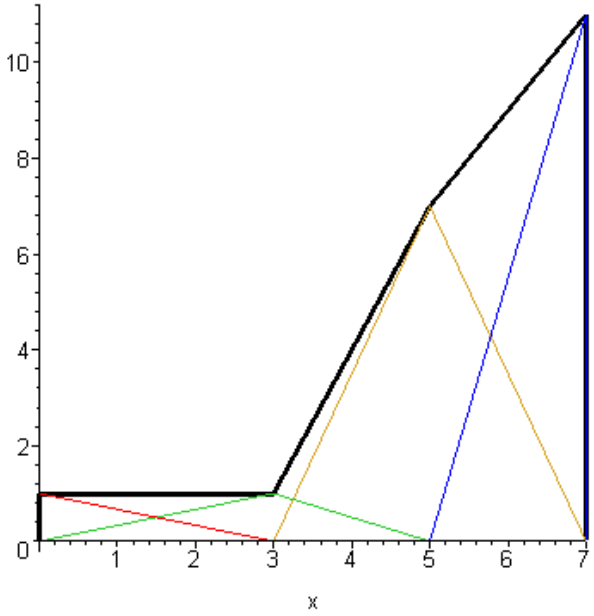
```
/* Очистка всіх змінних */
restart;

/* Визначення точок */
Points: [[0, 1], [3, 1], [5, 7], [7, 11]];
N: length(Points);
Leng: Points[N][1];

/* Ініціалізація масивів dx, L та функцій u */
dx: makelist(0, i, 1, N);
L: makelist(0, i, 1, N + 1);
u: makelist(0, i, 1, N);

/* Визначення функцій u[i] */
for i:1 thru N do (
    if i = 1 then (
        dx[i]:= Points[i + 1][1] - Points[i][1],
        L[i]:= Points[i][1],
        L[i + 1]:= Points[i + 1][1],
        u[i](x):= piecewise([x > L[i] and x < L[i + 1], (1 / dx[i]) * (-x +
L[i + 1])], [x > L[i + 1], 0])
    ) elseif i = N then (
```

```
        dx[i]:= Points[i][1] - Points[i - 1][1],
        L[i - 1]:= Points[i - 1][1],
        L[i]:= Points[i][1],
        u[i](x):= piecewise([x < L[i - 1], 0], [x > L[i - 1] and x < L[i], (1
/ dx[i]) * (x - L[i - 1])])
    ) else (
        dx[i]:= Points[i][1] - Points[i - 1][1],
        dx[i + 1]:= Points[i + 1][1] - Points[i][1],
        L[i - 1]:= Points[i - 1][1],
        L[i]:= Points[i][1],
        L[i + 1]:= Points[i + 1][1],
        u[i](x):= piecewise([x < L[i - 1], 0],
                            [x > L[i - 1] and x < L[i], (1 / dx[i]) * (x -
L[i - 1])],
                            [x > L[i] and x < L[i + 1], (1 / dx[i + 1]) * (-x
+ L[i + 1])],
                            [x > L[i + 1], 0])
    )
);

/* Визначення сплайну */
Spline: sum(Points[i][2] * u[i](x), i, 1, N);

/* Побудова графіків */
P1: plot2d([discrete([Points[i][2] * u[i](x)], i, 1, N)], [x, 0, Leng],
[style, point]);
P2: plot2d(Spline, [x, 0, Leng], [color, black], [thickness, 3]);

/* Відображення графіків */
display(P1, P2);
```

Система **Maple 10**:

```
restart;
Points:=[[0,1],[3,1],[5,7], [7,11]];
N:=nops(Points);
Leng:=Points[-1][1];

for i from 1 to N do
 if i=1 then
  dx[i]:=Points[i+1][1]-Points[i][1];
  L[i]:=Points[i][1]:
  L[i+1]:=Points[i+1][1]:
  u[i](x):=piecewise(x>L[i] and x<L[i+1], (1/dx[i])*(-x+L[i+1]), x>L[i+1],
0):
 elif i=N then
  dx[i]:=Points[i][1]-Points[i-1][1];
  L[i-1]:=Points[i-1][1]:
  L[i]:=Points[i][1]:
  u[i](x):=piecewise(x<L[i-1], 0, x>L[i-1] and x<L[i], (1/dx[i])*(x-L[i-1])):
 else
  dx[i]:=Points[i][1]-Points[i-1][1];
  dx[i+1]:=Points[i+1][1]-Points[i][1]:
  L[i-1]:=Points[i-1][1]:
  L[i]:=Points[i][1]:
  L[i+1]:=Points[i+1][1]:
  u[i](x):=piecewise(x<L[i-1], 0, x>L[i-1] and x<L[i], (1/dx[i])*(x-L[i-1]),
                              x>L[i] and x<L[i+1], (1/dx[i+1])*(-x+L[i+1]),
x>L[i+1], 0):
 fi:
od:
Spline:=sum(Points[ic][2]*u[ic](x), ic=1..N);
```

```
P1:=plot([Points[icc][2]*u[icc](x)$icc=1..4], x=0..Leng):
P2:=plot(Spline, x=0..Leng,color=black, thickness=3):
plots[display]([P1,P2]);
```



2.10. Інтерполяція кубічними сплайнами

Як вже зазначалося, *сплайн* (від англ. spline — гнучка рейка) — це функція, графік якої утворений із кількох поліноміальних відрізків. Ці відрізки з'єднані таким чином, що похідні отриманої функції до порядку, який на одиницю менший за степінь полінома, є неперервними на всьому проміжку, що розглядається.

Щоб краще зрозуміти, як сплайн-функції пов’язані з креслярськими роботами, розглянемо приклад з гнучкою сталевою лінійкою (рис. 2.3). Якщо таку лінійку встановити на ребро, зафіксувати один її кінець у певній точці, а інші частини пропустити через опори, розташовані на площині в заданих координатах, то отримана форма лінійки буде відповідати формі сплайн-кривої.

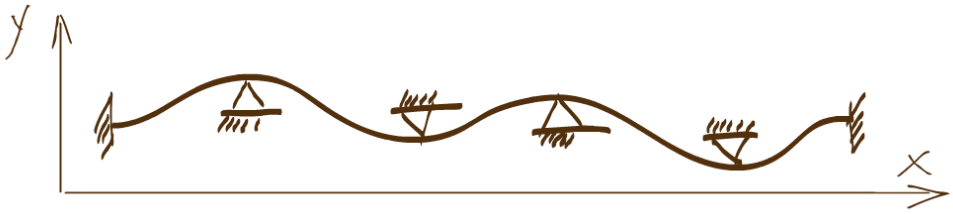


Рис. 2.3. Механічне представлення сплайна

2.11. Особливості застосування методів моделювання сплайнів

До сплайнів як до засобу апроксимації та інтерполяції функцій висувається низка вимог, що забезпечують їхню точність, гладкість та ефективність у чисельному моделюванні. Основні з них такі:

- безперервність та гладкість — сплайн має бути безперервним не лише сам по собі, але й разом зі своїми похідними певного порядку (залежно від задачі). Зазвичай вимагається, щоб були безперервними перша та друга похідні (наприклад, для кубічних сплайнів);
- точне проходження через вузлові точки (інтерполяція). У класичній задачі інтерполяції сплайн повинен точно проходити через задані точки, тобто задовольняти умову $S(x_i) = f(x_i)$, де x_i — вузли;
- локальність впливу. Зміна одного вузла або умов на відрізку має впливати лише на локальну частину сплайна, не змінюючи всю криву. Це важливо для стабільності обчислень і зручності у редагуванні;
- мінімізація викривлення (вигину). Бажано, щоб побудований сплайн мав мінімальну кривизну або інший оптимум, наприклад, мінімум другої похідної по всій довжині. Це забезпечує плавність кривої;
- чисельна стабільність. Методи побудови сплайнів повинні бути стійкими до похибок обчислень, особливо при великій кількості вузлів або складній геометрії;
- обмеження на ступінь поліномів. Щоб уникнути осциляцій (*ефекту Рунге*), застосовують низькостепеневі поліноми (найчастіше другого чи третього порядку), з яких і складається сплайн на кожному відрізку;
- гнучкість у заданнях граничних умов. Сплайн повинен легко підлаштовуватись під додаткові умови — наприклад, відомі значення похідних на кінцях.

Ці вимоги забезпечують точне, плавне й контрольоване представлення складних функцій або форм, що особливо важливо в інженерному моделюванні, графіці та CAD/CAE — системах.

2.12. Параметричні кубічні сплайни (загальні відомості)

У разі подання фізичного сплайна як тонкої пружної балки, його форма буде змінюватися під дією зовнішніх сил. При цьому геометрія кривої залежить від розподілу навантаження та граничних умов (див. рис. 2.3). Вигин балки y описується рівнянням Ейлера-Бернуллі (2.1), яке пов’язує момент в перерізі балки $M(x)$ з її радіусом кривизни уздовж довжини балки

$$\frac{1}{\rho(x)} = \frac{M(x)}{EI}, \tag{2.1}$$

де E — модуль Юнга, що залежить від властивостей матеріалу балки; I — момент інерції, обумовлений формою кривій; $\rho(x)$ — радіус кривизни.

Величина в лівій частині рівняння (2.1) (обернено пропорційна $\rho(x)$) називається кривизною. Для малих y $(y' << 1)$ кривизна буде наближено дорівнювати другій похідній вигину:

$$\frac{1}{\rho(x)} = \frac{y''}{(1+y'^2)^{3/2}} \approx y''.$$

де штрих позначає похідну по x — відстані уздовж балки.

Моменти розподіляються лінійно (плече пропорційно *x*), тобто $M(x) = Ax + B$, і, підставляючи цю величину в (2.1), з врахуванням вищесказаного одержуємо:

$$y'' = \frac{Ax + B}{EI}.$$

Інтегрування дає:

$$y = A_1x^3 + B_1x^2 + C_1x + D_1. \qquad (2.2)$$

Отже, геометрія сплайна в постановці задачі у відповідності з рис. 2.3. визначається кубічним поліномом (2.2).

Кубічний сплайн забезпечує неперервність похідних до другого порядку включно в точках з'єднання. Сплайни, складені з поліномів невисокого ступеня, особливо зручні для інтерполяції, оскільки вони не потребують значних обчислювальних ресурсів і не спричиняють числових нестабільностей, характерних для поліномів високого порядку.

Далі розглядається набір кубічних сегментів, кожен з яких проходить через пару заданих точок. Такі сплайни можуть мати точки перегину й зміну кривини в тривимірному просторі.

2.13. Кубічний сегмент сплайну

Карще за все записувати формулу для сегменту сплайна в параметричному вигляді. Окрім того, у звязку з тим, що ми шукаємо запис сегменту в тривимірному просторі, будемо використовувати векторне представлення:

$$\mathbf{P}(t) = \sum_{i=1}^{4} \mathbf{b}_i t^{i-1}, t_1 \le t \le t_2, \qquad (2.3)$$

де t_1 й t_2 — значення параметрів на границях сегмента; $\mathbf{P}(t) = [x(t) \quad y(t) \quad z(t)]$ — це векторна функція, де три складові — декартові координати вектора, $\mathbf{b}_i$ — векторні коефіцієнти.

Покомпонентний запис буде мати вигляд

$$x(t) = \sum_{i=1}^{4} b_{xi} t^{i-1}, t_1 \le t \le t_2;$$

$$y(t) = \sum_{i=1}^{4} b_{y_i} t^{i-1}, t_1 \le t \le t_2;$$

$$z(t) = \sum_{i=1}^{4} b_{zi} t^{i-1}, t_1 \le t \le t_2.$$

Для знаходження векторних коефіцієнтів $\mathbf{b}_i$ треба використати чотири граничні умови для сегмента. Розкриємо у (2.3) суму:

$$\mathbf{P}(t) = \mathbf{b}_1 + \mathbf{b}_2 t + \mathbf{b}_3 t^2 + \mathbf{b}_4 t^3, t_1 \le t \le t_2. \tag{2.4}$$

Позначемо як $\mathbf{P}_1$ і $\mathbf{P}_2$ вектори, що відповідають кінцям сегмента, тоді $\mathbf{P}_1'$ і $\mathbf{P}_2'$ — похідні по t (які є дотичними векторами в кінцях сегмента (рис. 2.4)).

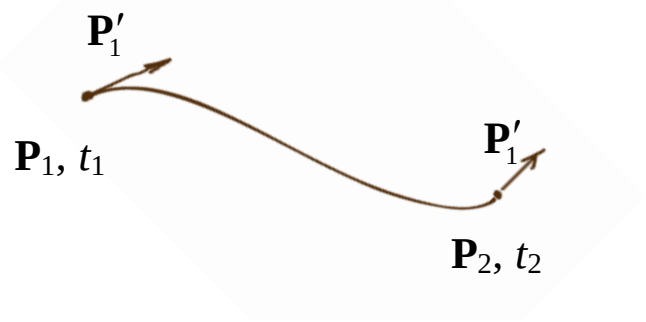


Рис. 2.4. Сегмент сплайну

Знайдемо похідні по параметру:

$$\mathbf{P}'(t) = \sum_{i=1}^{4} \mathbf{b}_i (i-1) t^{i-2}, t_1 \le t \le t_2$$

і, таким чином, отримаємо поліном другого ступеня

$$\mathbf{P}'(t) = \mathbf{b}_2 + 2\mathbf{b}_3 t + 3\mathbf{b}_4 t^2, t_1 \le t \le t_2 \tag{2.5}$$

Приймемо, що на початку сегменту $t_1 = 0$. Тоді маємо

$$\begin{aligned} \mathbf{P}(0) &= \mathbf{P}_1, \\ \mathbf{P}(t_2) &= \mathbf{P}_2, \\ \mathbf{P}'(0) &= \mathbf{P}_1', \\ \mathbf{P}'(t_2) &= \mathbf{P}_2'. \end{aligned} \tag{2.6}$$

Звідси знаходемо $\mathbf{b}_i$, що дає

$$\mathbf{P}(t) = \mathbf{P}_1 + \mathbf{P}_1' t + \left[\frac{3(\mathbf{P}_2 - \mathbf{P}_1)}{t_2^2} - \frac{2\mathbf{P}_1'}{t_2} - \frac{\mathbf{P}_2'}{t_2} \right] t^2 + \left[\frac{2(\mathbf{P}_1 - \mathbf{P}_2)}{t_2^3} + \frac{\mathbf{P}_1'}{t_2^2} + \frac{\mathbf{P}_2'}{t_2^2} \right] t^3. \quad (2.7)$$

Рівняння (2.7) можна далі використовувати для будь-якої пари суміжних кубічних сегментів. Можна також застосовувати параметричні представлення вищого порядку, проте, як вже зазначалось, це часто призводить до появи небажаних осциляцій.

Основні варіанти завдання параметричних кубічних кривих:

- завдання кривими Ерміта;
- параболічна інтерполяція (метод Оверхаузера);
- завдання кривими Без'є (визначається за допомогою скінченної кількості проміжних контрольних точок);
- метод B-сплайнів (кінцеві точки можуть не належати самій кривій, чим забезпечується неперервність як першої, так і другої похідних на межах сегментів).

Зазвичай інтерполяційні задачі розв'язуються першими двома методами, тоді як останні підходи частіше застосовують у випадках апроксимації.

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики візуалізацію інтерполяції кубічними сплайнами за довільним масивом точок.

ПРИКЛАД ВИКОНАННЯ.

*Наведено розрахунок кубічного сплайну зі зв'язкою сегментів в якості прикладу вигляду $\mathbf{P'}_{\text{зліва}} = \mathbf{P'}_{\text{справа}}$ = середнє зліва та справа ($\Delta\mathbf{P}/\Delta$t).

Система **Maxima**:

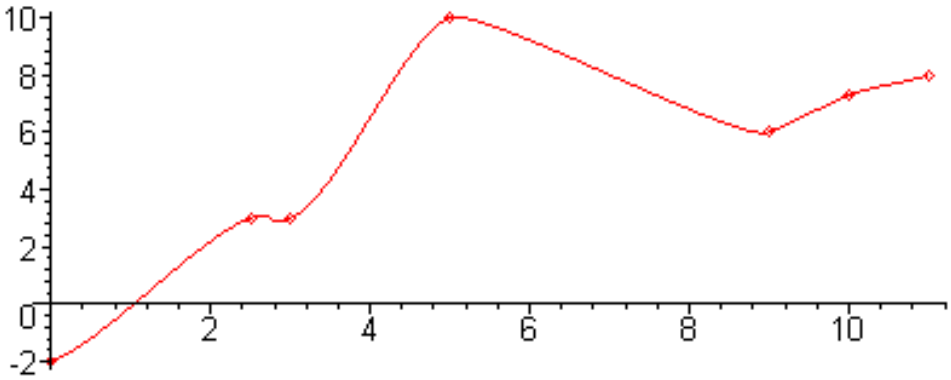
```
/* Очищення всіх змінних та налаштувань */
restart;

/* Визначення функції f та її похідної */
f: B1 + B2*t + B3*t^2 + B4*t^3;
df: diff(f, t);

/* Визначення системи рівнянь для розв'язання */
eq1: subs(t=t1, f) = P1;
eq2: subs(t=t2, f) = P2;
eq3: subs(t=t1, df) = dP1;
eq4: subs(t=t2, df) = dP2;

/* Розв'язання системи рівнянь для B1, B2, B3, B4 */
ResB: solve({eq1, eq2, eq3, eq4}, {B1, B2, B3, B4});

/* Витягування значень B1, B2, B3, B4 */
for i:1 thru 4 do B[i]:= rhs(ResB[i]) end;
```

```
/* Визначення функції F за знайденими коефіцієнтами B */
F: sum(B[ic]*t^(ic-1), ic, 1, 4);

/* Визначення точок */
Points: [[0, -2], [2.5, 3], [3, 3], [5, 10], [9, 6], [10, 7.3], [11, 8]];

/* Ініціалізація змінних для кусочних функцій */
forma1: NULL;
forma2: NULL;

/* Визначення кусочних функцій */
for i:1 thru nops(Points) - 1 do (
    t1: Points[i][1],
    t2: Points[i + 1][1],
    if i = 1 then (
        dP:  (Points[i  +  1][2]  -  Points[i][2])  /  (Points[i  +  1][1]  -
Points[i][1]),
        Ff[i][1]:=subs(P1=Points[i][1], P2=Points[i + 1][1], dP1=dP, dP2=dP,
F),
        Ff[i][2]:=subs(P1=Points[i][2], P2=Points[i + 1][2], dP1=dP, dP2=dP,
F)
    ) else (
        dP:  0.5  *  (Points[i][2]  -  Points[i  -  1][2])  /  (Points[i][1]  -
Points[i - 1][1]) +
            0.5 * (Points[i + 1][2] - Points[i][2]) / (Points[i + 1][1] -
Points[i][1]),
        Ff[i][1]:=subs(P1=Points[i][1], P2=Points[i + 1][1], dP1=dP, dP2=dP,
F),
        Ff[i][2]:=subs(P1=Points[i][2], P2=Points[i + 1][2], dP1=dP, dP2=dP,
F)
    ),
    forma1: forma1, t1 <= t and t <= t2, Ff[i][1],
    forma2: forma2, t1 <= t and t <= t2, Ff[i][2]
);

/* Створення кусочних функцій для графіку */
Forma1: piecewise(forma1);
Forma2: piecewise(forma2);

/* Побудова графіків */
PL1: plot2d([Forma1, Forma2], [t, 0, 6]),
PL2: plot2d([Points], [style, point]);

/* Відображення графіків */
plots[display]([PL1, PL2]);
```

Система **Maple 10**:

```
f:=B1+B2*t+B3*t^2+B4*t^3;
df:=diff(f,t);
eq1:=subs( t=t1, f)=P1;
eq2:=subs( t=t2, f)=P2;
eq3:=subs( t=t1, df)=dP1;
eq4:=subs( t=t2, df)=dP2;
ResB:=solve({eq1,eq2,eq3,eq4},  {B1,B2,B3,B4});
for i to 4 do B[i]:=rhs(ResB[i]) od;
F:=sum(B[ic]*t^(ic-1), ic=1..4);
Points:=[0, -2], [2.5, 3], [3, 3], [5, 10], [9, 6], [10, 7.3], [11, 8];
forma1:=NULL:
forma2:=NULL;
for i to nops([Points])-1 do
```

```
 t1:=i-1:
 t2:=i;
 if i=1 then
  dP := (Points[i+1][2]-Points[i][2])/(Points[i+1][1]-Points[i][1]) :
  Ff[i][1]:=subs(P1=Points[i][1], P2=Points[i+1][1], dP1= dP, dP2= dP , F):
  Ff[i][2]:=subs(P1=Points[i][2], P2=Points[i+1][2], dP1= dP, dP2= dP , F)
 else
   dP:=0.5* (Points[i][2]-Points[i-1][2])/(Points[i][1]-Points[i-1][1])+
       0.5* (Points[i+1][2]-Points[i][2])/(Points[i+1][1]-Points[i][1]) :
   Ff[i][1]:=subs(P1=Points[i][1], P2=Points[i+1][1], dP1= dP, dP2= dP, F );
   Ff[i][2]:=subs(P1=Points[i][2], P2=Points[i+1][2], dP1= dP, dP2= dP, F )
  fi;
  forma1:=forma1,t1<=t and t<=t2,Ff[i][1];
  forma2:=forma2,t1<=t and t<=t2,Ff[i][2];
od:
Forma1:=piecewise(forma1);
Forma2:=piecewise(forma2);
PL1:=plot([Forma1,Forma2, t=0..6]):
  PL2:=plot([Points], style=point):
plots[display]([PL1,PL2]);
```



2.13. Інтерполяція ермітовими сплайнами. Сегмент ермітової кривої

Розглянемо сегмент, що задається кривою Ерміта. Перепозначимо коефіцієнти та перепишемо рівняння (2.4) як:

$$\mathbf{P}(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3, \quad (0 \le u \le 1) \qquad (2.8)$$

або

$$\mathbf{P}(u) = \sum_{i=0}^{3} \mathbf{a}_i u^i ,$$

де $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ — векторні коефіцієнти.

Приймемо, що параметр змінюється в діапазоні від 0 до 1 — тому, на відміну від (2.4) позначаємо цей параметр як u. З врахуванням граничних умов ($u = 0$ для $\mathbf{P}_0$ й $\mathbf{P}_0'$; $u = 1$ для $\mathbf{P}_1$ й $\mathbf{P}_1'$) з формули (2.8) одержимо:

$$\begin{cases} \mathbf{P}_0 = \mathbf{P}(0) = \mathbf{a}_0; \\ \mathbf{P}_1 = \mathbf{P}(1) = \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3; \\ \mathbf{P}_0' = \mathbf{P}'(0) = \mathbf{a}_1; \\ \mathbf{P}_1' = \mathbf{P}'(1) = \mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3. \end{cases} \tag{2.9}$$

Вирішуючі (2.9) відносно $\mathbf{a}_0, \ldots, \mathbf{a}_3$:

$$\begin{cases} \mathbf{a}_0 = \mathbf{P}_0; \\ \mathbf{a}_1 = \mathbf{P}_0'; \\ \mathbf{a}_2 = -3\mathbf{P}_0 + 3\mathbf{P}_1 - 2\mathbf{P}_0' - \mathbf{P}_1'; \\ \mathbf{a}_3 = 2\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_0' + \mathbf{P}_1'. \end{cases} \tag{2.10}$$

Тоді, з врахуванням (2.8), одержимо:

$$\mathbf{P}(u) = \mathbf{P_0} + \mathbf{P_0'}u + (-3\mathbf{P_0} + 3\mathbf{P_1} - 2\mathbf{P_0'} - \mathbf{P_1'})u^2 + (2\mathbf{P_0} - 2\mathbf{P_1} + \mathbf{P_0'} + \mathbf{P_1'})u^3, \tag{2.11}$$

Це, по суті, перетворене рівняння (2.7) з врахуванням діапазону параметра.

Після перетворень (2.11) прийме вигляд

$$\mathbf{P}(u) = (1 - 3u^2 + 2u^3)\mathbf{P}_0 + (3u^2 - 2u^3)\mathbf{P}_1 + (u - 2u^2 + u^3)\mathbf{P}_0' + (-u^2 + u^3)\mathbf{P}_1'$$

або

$$\mathbf{P}(u) = \begin{bmatrix} 1 - 3u^2 + 2u^3 & 3u^2 - 2u^3 & u - 2u^2 + u^3 & -u^2 + u^3 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_0'} \\ \mathbf{P_1'} \end{bmatrix}. \tag{2.12}$$

Відповідні вектори $\mathbf{P}_0, \mathbf{P_1}, \mathbf{P}_0', \mathbf{P}_1'$ мають назву *геометричних*, а рівняння (2.12) — *рівнянням ермітової кривої*.

Ми можемо сформувати матрицю геометричних коефіцієнтів:

$$\mathbf{B} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_0' \\ \mathbf{P}_1' \end{bmatrix}. \tag{2.13}$$

Функції параметра *u*, що утворюються в (2.12) називаються *функціями сполучення*:

$$\begin{cases} f_1(u) = 1 - 3u^2 + 2u^3; \\ f_2(u) = 3u^2 - 2u^3; \\ f_3(u) = u - 2u^2 + u^3; \\ f_4(u) = -u^2 + u^3. \end{cases} \tag{2.14}$$

Приклад кривої Ерміта наведено на рис. 2.5.

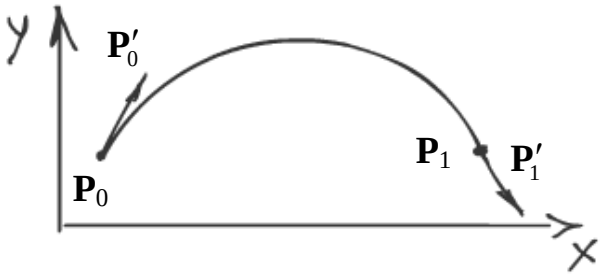


Рис. 2.5. Ермітова крива

Сімейство ермітових кривих, у яких напрямки векторів у заданих точках збігаються, показано на рис. 2.6. Водночас довжина цих векторів істотно впливає на форму кривої: збільшення довжини векторів призводить до підняття кривої вгору, при цьому найвища крива в цьому сімействі утворює петлю.

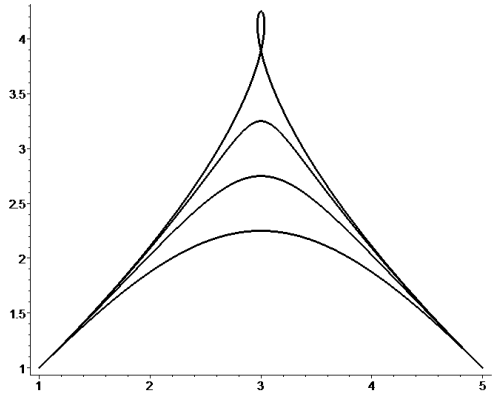


Рис. 2.6. Сімейство кривих Ерміта

ЗАВДАННЯ: провести засобами доступних систем комп’ютерної математики візуалізацію сімейства ермітових кривих, що з’єднують довільні точки з різними дотичними. Вхідні дані задаються індивідуально.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Імплементація бібліотеки лінійної алгебри */
load(linalg);

/* Оголошення символічних змінних */
u: 'u';
a1: 'a1'; b1: 'b1'; a2: 'a2'; b2: 'b2';
c1: 'c1'; d1: 'd1'; c2: 'c2'; d2: 'd2';

/* Визначення кольорів для графіків */
g[1]: color = red;
g[2]: color = blue;
g[3]: color = black;
g[4]: color = cyan;

/* Оголошення матриці U */
U: matrix([[1 - 3*u^2 + 2*u^3, 3*u^2 - 2*u^3, u - 2*u^2 + u^3, -u^2 + u^3]]);

/* Ініціалізація значень для a1, b1, a2, b2, c1, d1, c2, d2 */
a1[1]:=1;  b1[1]:=1;  a2[1]:=5;  b2[1]:=1;  c1[1]:=5;  d1[1]:=5;  c2[1]:=5;
d2[1]:=-5;
a1[2]:=1;  b1[2]:=1;  a2[2]:=5;  b2[2]:=1;  c1[2]:=7;  d1[2]:=7;  c2[2]:=7;
d2[2]:=-7;
a1[3]:=1;  b1[3]:=1;  a2[3]:=5;  b2[3]:=1;  c1[3]:=9;  d1[3]:=9;  c2[3]:=9;
d2[3]:=-9;
a1[4]:=1;  b1[4]:=1;  a2[4]:=5;  b2[4]:=1;  c1[4]:=13;  d1[4]:=13;  c2[4]:=13;
d2[4]:=-13;

/* Генерація графіків для кожного набору параметрів */
for i:1 thru 4 do (
    /* Оголошення матриці B */
    B:  matrix([a1[i],  b1[i],  0],  [a2[i],  b2[i],  0],  [c1[i],  d1[i],  0],
[c2[i], d2[i], 0]]);

    /* Обчислення S як добуток U на B */
    S: U . B;

    /* Визначення x та y */
    x: S[1,1];
    y: S[1,2];

    /* Побудова графіку */
    C[i]: plot2d([x, y, u = 0 .. 1], [color, g[i]], [scaling, constrained],
[thickness, 3]);

    /* Відображення графіку */
    display(C[i]);

    /* Скидання символічної змінної u */
    u: 'u';
);

/* Відображення всіх графіків разом */
display(C[1], C[2], C[3], C[4]);
```

Система **Maple 10**:

```
with(linalg):u:='u';

a1:='a1';b1:='b1';a2:='a2';b2:='b2';c1:='c1';d1:='d1';c2:='c2';d2:='d2';
g[1]:=color=red;
g[2]:=color=blue;
g[3]:=color=black;
g[4]:=color=cyan;
U:=Matrix([[1-3*u2+2*u3, 3*u2-2*u3, u-2*u2+u3, -u2+u3]]);
a1[1]:=1;b1[1]:=1;a2[1]:=5;b2[1]:=1;c1[1]:=5;d1[1]:=5;c2[1]:=5;d2[1]:=-5;
a1[2]:=1;b1[2]:=1;a2[2]:=5;b2[2]:=1;c1[2]:=7;d1[2]:=7;c2[2]:=7;d2[2]:=-7;
a1[3]:=1;b1[3]:=1;a2[3]:=5;b2[3]:=1;c1[3]:=9;d1[3]:=9;c2[3]:=9;d2[3]:=-9;
a1[4]:=1;b1[4]:=1;a2[4]:=5;b2[4]:=1;c1[4]:=13;d1[4]:=13;c2[4]:=13;d2[4]:=-13;
for i from 1 to 4 do L[i]:

B:=Matrix([[a1[i],b1[i],0],[a2[i],b2[i],0],[c1[i],d1[i],0],[c2[i],d2[i],0]]):
   S:=U.B; x:=S[1,1]: y:=S[1,2];
   C[i]:=plot([x,y,    u=0..1],g[i],    scaling=CONSTRAINED,    thickness=3):
display(C[i]); u:='u':
od:
display(C[1],C[2],C[3],C[4]);
```

2.14. Інтерполяція ермітовою кривою

Перейдемо безпосередньо к моделюванню сплайна. Інтерпольовані точки позначемо $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_i, \ldots, \mathbf{P}_{n-1}, \mathbf{P}_n$. Так як кількість точок дорівнює $n+1$, то число сегментів Ермітових кривих буде рівно n. Вказані криві позначено $\mathbf{P}_1(u), \mathbf{P}_2(u), \ldots, \mathbf{P}_i(u), \ldots, \mathbf{P}_n(u)$ (рис. 2.7).

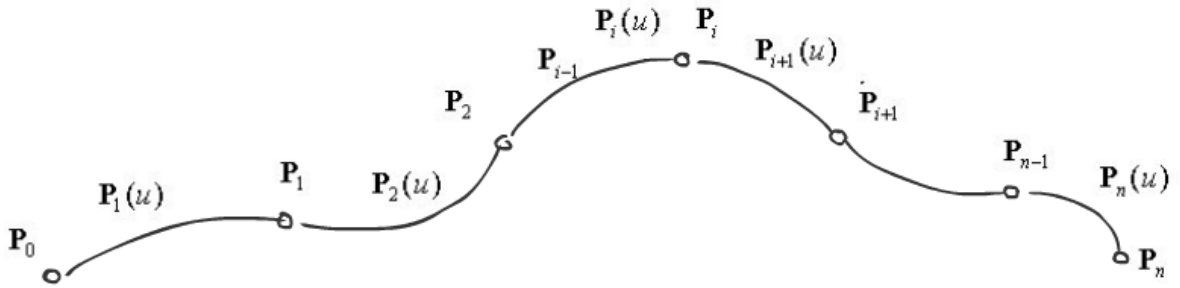


Рис. 2.7. Точки, що інтерполюються та ермітові криві

З врахуванням (2.8), будь-який сегмент ермітової кривої можна подати у такому вигляді:

$$\mathbf{P}_i(u) = \mathbf{P}_{i-1} + \mathbf{P}'_{i-1}u + [3(\mathbf{P}_i - \mathbf{P}_{i-1}) - 2\mathbf{P}'_{i-1} - \mathbf{P}'_i]u^2 + [2(\mathbf{P}_{i-1} - \mathbf{P}_i) + \mathbf{P}'_{i-1} + \mathbf{P}'_i]u^3, \quad (2.15)$$

де $\mathbf{P}'_{i-1}$ й $\mathbf{P}'_i$ — вектори дотичних у точках P_{i-1} і P_i відповідно.

З метою забезпечення безперервності другої похідної в точках стику сегментів накладаємо відповідну граничну умову:

$$\left.\frac{d^2\mathbf{P}_i(u)}{du^2}\right|_{u=1} = \left.\frac{d^2\mathbf{P}_{i+1}(u)}{du^2}\right|_{u=0}, \qquad i=1, 2, ..., n-1. \tag{2.16}$$

Підставивши (2.15) у (2.16), отримуємо рівняння (2.19), яке виводиться таким чином. Обчислюємо першу та другу похідні (2.15):

$$\begin{aligned}\mathbf{P}_i'(u) &= \mathbf{P}_{i-1}' + 2\left[3\left(\mathbf{P}_i - \mathbf{P}_{i-1}\right) - 2\mathbf{P}_{i-1}' - \mathbf{P}_i'\right]u + 3\left[2\left(\mathbf{P}_{i-1} - \mathbf{P}_i\right) + \mathbf{P}_{i-1}' + \mathbf{P}_i'\right]u^2 = \\ &= \mathbf{P}_{i-1}' + \left[6\mathbf{P}_i - 6\mathbf{P}_{i-1} - 4\mathbf{P}_{i-1}' - 2\mathbf{P}_i'\right]u + \left[6\mathbf{P}_{i-1} - 6\mathbf{P}_i + 3\mathbf{P}_{i-1}' + 3\mathbf{P}_i'\right]u^2; \\ \mathbf{P}_i''(u) &= 6\mathbf{P}_i - 6\mathbf{P}_{i-1} - 4\mathbf{P}_{i-1}' - 2\mathbf{P}_i' + 2\left[6\mathbf{P}_{i-1} - 6\mathbf{P}_i + 3\mathbf{P}_{i-1}' + 3\mathbf{P}_i'\right]u. \end{aligned} \tag{2.17}$$

Далі вираз (2.15) запишемо для $i+1$:

$$\mathbf{P}_{i+1}(u) = \mathbf{P}_i + \mathbf{P}_i'u + \left[3\left(\mathbf{P}_{i+1} - \mathbf{P}_i\right) - 2\mathbf{P}_i' - \mathbf{P}_{i+1}'\right]u^2 + \left[2\left(\mathbf{P}_i - \mathbf{P}_{i+1}\right) + \mathbf{P}_i' + P_{i+1}'\right]u^3.$$

Знаходимо першу й другу похідні для $i+1$:

$$\begin{aligned}\mathbf{P}_{i+1}'(u) &= \mathbf{P}_i' + \left[6\mathbf{P}_{i+1} - \mathbf{P}_i - 4\mathbf{P}_i' - 2\mathbf{P}_i' + 1\right]u + \left[6\mathbf{P}_i - 6\mathbf{P}_{i+1} + 3\mathbf{P}_i' + 3\mathbf{P}_{i+1}'\right]u^2; \\ \mathbf{P}_{i+1}''(u) &= 6\mathbf{P}_{i+1} - 6\mathbf{P}_i - 4\mathbf{P}_i' - 2\mathbf{P}_{i+1}' + 6\left[2\left(\mathbf{P}_i - \mathbf{P}_{i+1}\right) + \mathbf{P}_i' + \mathbf{P}_{i+1}'\right]u. \end{aligned} \tag{2.18}$$

Використовуючи граничні умови (2.16) знаходимо другу похідну для значення $u=1$, а з виразу (2.17) для значення $u=0$:

$$\begin{aligned}\mathbf{P}_i''(u=1) &= 6\mathbf{P}_i - 6\mathbf{P}_{i-1} - 4\mathbf{P}_{i-1}' - 2\mathbf{P}_i' + 12\mathbf{P}_{i-1} - 12\mathbf{P}_i + 6\mathbf{P}_{i-1}' + 6\mathbf{P}_i'; \\ \mathbf{P}_{i+1}''(u=0) &= 6\mathbf{P}_{i+1} - 6\mathbf{P}_i - 4\mathbf{P}_i' - 2\mathbf{P}_{i+1}'. \end{aligned}$$

Враховуючи умову стикування сегментів $\mathbf{P}_i''(1) = \mathbf{P}_{i+1}''(0)$, одержуємо потрібне рівняння:

$$\mathbf{P}_{i-1}' + 4\mathbf{P}_i' + \mathbf{P}_{i+1}' = 3\mathbf{P}_{i+1} - 3\mathbf{P}_{i-1}. \tag{2.19}$$

Якщо провести підстановку в (2.19) усіх значень i від 1 до $n-1$

$$\begin{aligned} i=1:&\ \ \mathbf{P}_0' + 4\mathbf{P}_1' + \mathbf{P}_2' = \mathbf{3}P_2 - 3\mathbf{P}_0, \\ i=2:&\ \ \mathbf{P}_1' + 4\mathbf{P}_2' + \mathbf{P}_3' = 3\mathbf{P}_3 - 3\mathbf{P}_1, \\ i=3:&\ \ \mathbf{P}_2' + 4\mathbf{P}_3' + \mathbf{P}_4' = 3\mathbf{P}_4 - 3\mathbf{P}_2, \\ i=4:&\ \ \mathbf{P}_3' + 4\mathbf{P}_4' + \mathbf{P}_5' = 3\mathbf{P}_5 - 3\mathbf{P}_3, \end{aligned}$$

$$\dots$$

$$i = n-2:\ \mathbf{P}'_{n-3} + 4\mathbf{P}'_{n-2} + \mathbf{P}'_{n-1} = 3\mathbf{P}_{n-1} - 3\mathbf{P}_{n-3},$$

$$i = n-1:\ \mathbf{P}'_{n-2} + 4\mathbf{P}'_{n-1} + \mathbf{P}'_{n} = 3\mathbf{P}_{n} - 3\mathbf{P}_{n-2},$$

то отримаємо матричне рівняння:

$$\begin{bmatrix} 4 & 1 & 0 & . & . & . & . & . & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & . & . & . & . & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & . & . & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & . & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & . & . & . & . & 0 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}'_1 \\ \mathbf{P}'_2 \\ \mathbf{P}'_3 \\ . \\ . \\ . \\ \mathbf{P}'_{n-2} \\ \mathbf{P}'_{n-1} \end{bmatrix} = \begin{bmatrix} 3\mathbf{P}_2 - 3\mathbf{P}_0 - \mathbf{P}'_0 \\ 3\mathbf{P}_3 - 3\mathbf{P}_1 \\ 3\mathbf{P}_4 - 3\mathbf{P}_2 \\ . \\ . \\ . \\ 3\mathbf{P}_{n-1} - 3\mathbf{P}_{n-3} \\ 3\mathbf{P}_n - 3\mathbf{P}_{n-2} - \mathbf{P}'_n \end{bmatrix}. \qquad (2.20)$$

Для відомих значень $\mathbf{P}_0'$ й $\mathbf{P}_n'$ із правої частини рівняння (2.20) можна знайти значення $n-1$ невідомих змінних $\mathbf{P}_1'$, $\mathbf{P}_2'$, ..., $\mathbf{P}_{n-1}'$. Після знаходження значень усіх похідних їх можна підставити у рівняння (2.15), що дозволить отримати повністю визначену ермітову криву.

Вектори дотичних на кінцях кривої $\mathbf{P}_0'$ й $\mathbf{P}_n'$ можемо знайти наступним чином. Будемо виходити з припущення, що на кінцях відсутнє кручення, що еквівалентно присвоєнню значень $\mathbf{P}_0'' = 0$ і $\mathbf{P}_n'' = 0$, оскільки друга похідна пропорційна крученню:

$$\left.\frac{d^2\mathbf{P}_1(u)}{du^2}\right|_{u=0} = -3\mathbf{P}_0 + 3\mathbf{P}_1 - 2\mathbf{P}_0' - \mathbf{P}_1' = 0; \qquad (2.21)$$

$$\left.\frac{d^2\mathbf{P}_n(u)}{du^2}\right|_{u=1} = 2\left[3\left(\mathbf{P}_n - \mathbf{P}_{n-1}\right) - 2\mathbf{P}'_{n-1} - \mathbf{P}_n'\right] +$$

$$+6\left[2\left(\mathbf{P}_{n-1} - \mathbf{P}_n\right) + \mathbf{P}'_{n-1} + \mathbf{P}'_n\right] = 0. \qquad (2.22)$$

Після перетворень (2.21) і (2.22) отримаємо:

$$2\mathbf{P}_0' + \mathbf{P}_1' = 3\mathbf{P}_1 - 3\mathbf{P}_0, \qquad (2.23)$$

$$2\mathbf{P}_n' + \mathbf{P}'_{n-1} = 3\mathbf{P}_n - 3\mathbf{P}_{n-1}. \qquad (2.24)$$

Тепер додамо рівняння (2.23) і (2.24) до системи (2.20), у результаті отримаємо нове матричне рівняння:

$$\mathbf{K} \cdot \mathbf{P}' = \mathbf{M}, \tag{2.25}$$

де

$$\mathbf{K} = \begin{bmatrix} 2 & 1 & 0 & . & . & . & . & . & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & . & . & . & . & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & . & . & . & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 0 & 0 & . & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & . & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & . & . & . & . & 0 & 1 & 2 \end{bmatrix}; \quad \mathbf{P}' = \begin{bmatrix} \mathbf{P}'_0 \\ \mathbf{P}'_1 \\ \mathbf{P}'_2 \\ \mathbf{P}'_3 \\ . \\ . \\ \mathbf{P}'_{n-1} \\ \mathbf{P}'_n \end{bmatrix}; \quad \mathbf{M} = \begin{bmatrix} 3\mathbf{P}_1 - 3\mathbf{P}_0 \\ 3\mathbf{P}_2 - 3\mathbf{P}_0 \\ 3\mathbf{P}_3 - 3\mathbf{P}_1 \\ 3\mathbf{P}_4 - 3\mathbf{P}_2 \\ . \\ . \\ 3\mathbf{P}_n - 3\mathbf{P}_{n-2} \\ 3\mathbf{P}_n - 3\mathbf{P}_{n-1} \end{bmatrix}.$$

Звідси можемо визначити всі внутрішні дотичні як

$$\mathbf{P}' = \mathbf{K}^{-1} \cdot \mathbf{M}. \tag{2.26}$$

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики візуалізацію ермітового сплайну для довільного набору точок та дотичних. Вхідні дані задаються індивідуально.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Завантаження бібліотек для лінійної алгебри та графіків */
load(linalg);
load(plots);

/* Оголошення символічної змінної u */
u: 'u';

/* Оголошення кольорів для графіків */
g[1]: color = green;
g[2]: color = blue;
g[3]: color = black;
g[4]: color = green;
g[5]: color = black;

/* Оголошення матриці K */
K: matrix([[2.0, 1.0, 0, 0, 0, 0],
           [1.0, 4.0, 1.0, 0, 0, 0],
           [0, 1.0, 4.0, 1.0, 0, 0],
           [0, 0, 1.0, 4.0, 1.0, 0],
           [0, 0, 0, 1.0, 4.0, 1.0],
           [0, 0, 0, 0, 1.0, 2.0]]);
```

```
/* Обчислення оберненої матриці K */
Ko: inverse(K);

/* Оголошення значень для a та b */
a[0]:=4.0; b[0]:=3.0;
a[1]:=5.0; b[1]:=4.0;
a[2]:=11.0; b[2]:=1.0;
a[3]:=17.0; b[3]:=9.0;
a[4]:=12.0; b[4]:=7.0;
a[5]:=25.0; b[5]:=2.0;

/* Оголошення матриць P */
P[0]:=matrix([a[0], b[0]]);
P[1]:=matrix([a[1], b[1]]);
P[2]:=matrix([a[2], b[2]]);
P[3]:=matrix([a[3], b[3]]);
P[4]:=matrix([a[4], b[4]]);
P[5]:=matrix([a[5], b[5]]);

/* Оголошення матриці M для обчислення похідних */
M: matrix([[3*P[1] - 3*P[0]],
           [3*P[2] - 3*P[0]],
           [3*P[3] - 3*P[1]],
           [3*P[4] - 3*P[2]],
           [3*P[5] - 3*P[3]],
           [3*P[5] - 3*P[4]]]);

/* Оголошення матриці U для обчислення функції */
U: matrix([[1 - 3*u^2 + 2*u^3,
            3*u^2 - 2*u^3,
            u - 2*u^2 + u^3,
            -u^2 + u^3]]);

/* Обчислення матриці Pr як добуток оберненої матриці K та M */
Pr: Ko . M;

/* Побудова графіків для кожного сегмента */
for i:1 thru 5 do (
    /* Оголошення змінних для коефіцієнтів c та d */
    L[i]: NULL;
    c[i-1]:=Pr[i,1];
    d[i-1]:=Pr[i,2];
    c[i]:=Pr[i+1,1];
    d[i]:=Pr[i+1,2];

    /* Оголошення матриці B для поточного сегмента */
    B: matrix([[a[i-1], b[i-1], 0],
               [a[i], b[i], 0],
               [c[i-1], d[i-1], 0],
               [c[i], d[i], 0]]);

    /* Обчислення функції S як добуток U на B */
```

```
        S: U . B;
        x: S[1,1];
        y: S[1,2];

        /* Побудова графіку для поточного сегмента */
        C[i]: plot2d([x, y, u = 0 .. 1], [color, g[i]], [scaling,
constrained], [thickness, 3]);

        /* Скидання символічної змінної u */
        u: 'u';
    );

    /* Відображення всіх графіків разом */
    display(C[1], C[2], C[3], C[4], C[5]);
```

Система **Maple 10**:

```
with(linalg): with(plots): u:='u':
g[1]:=color=green:
g[2]:=color=blue:
g[3]:=color=black:
g[4]:=color=green:
g[5]:=color=black:
K:=Matrix([[2.0,1.0,0,0,0,0],
[1.0,4.0,1.0,0,0,0], [0,1.0,4.0,1.0,0,0],[0,0,1.0,4.0,1.0,0],
   [0,0,0,1.0,4.0,1.0], [0,0,0,0,1.0,2.0]]):
Ko:=0.3:
a[0]:=4.0: b[0]:=3.0:
a[1]:=5.0: b[1]:=4.0:
a[2]:=11.0: b[2]:=1.0:
a[3]:=17.0: b[3]:=9.0:
a[4]:=12.0: b[4]:=7.0:
a[5]:=25.0:b[5]:=2.0:
P[0]:=Matrix([a[0],b[0]]); P[1]:=Matrix([a[1],b[1]]);
P[2]:=Matrix([a[2],b[2]]);
P[3]:=Matrix([a[3],b[3]]); P[4]:=Matrix([a[4], b[4]]);
P[5]:=Matrix([a[5],b[5]]);

M:=Matrix([[3*P[1]-3*P[0]], [3*P[2]-3*P[0]], [3*P[3]-3*P[1]],
   [3*P[4]-3*P[2]], [3*P[5]-3*P[3]], [3*P[5]-3*P[4]]]);
U:=Matrix([[1-3*u^2+2*u^3,3*u^2-2*u^3,u-2*u^2+u^3,-u^2+u^3]]);


Pr:=Ko.M:
for i from 1 to 5 do L[i]: c[i-1]:=Pr[i,1]; d[i-1]:=Pr[i,2]:
   c[i]:=Pr[i+1,1]; d[i]:=Pr[i+1,2]:
   B:= Matrix([[a[i-1],b[i-1],0],[a[i],b[i],0],[c[i-1],d[i-
1],0],[c[i],d[i],0]]):
   S:=U.B; x:=S[1,1]: y:=S[1,2];
   C[i]:=plot([x,y, u=0..1], g[i], scaling=CONSTRAINED,
   thickness=3): u:='u':
od:
display(C[1], C[2], C[3], C[4], C[5]);
```

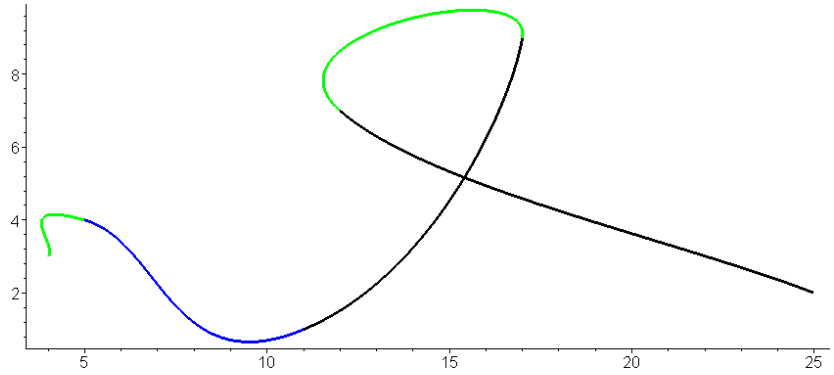


Рис. 2.7. Сплайн ермітової кривої

Контрольні питання.

1. У чому полягає суть інтерполяції та яка її основна мета?
2. Чим відрізняється форма полінома Ньютона від полінома Лагранжа?
3. Які переваги має поліном Ерміта у порівнянні з іншими інтерполяційними методами?
4. У чому полягає основна ідея інтерполяції сплайнами?
5. У яких випадках доцільніше використовувати сплайнову інтерполяцію замість поліноміальної?

Лабораторна робота №3

Тема: Апроксимація та згладжування

Як вже визначалось в попередній роботі, *апроксимація* — наближене вираження одних математичних об'єктів іншими, близькими за значенням, але простішими, наприклад, кривих ліній — ламаними, неперервних функцій — многочленами. Можна знайти більш детальне визначення таких задач, а саме, задача знаходження функції, що проходить близько від великого числа заданих точок є задачею *згладжування*, задача ж знаходження функції, близької до деякої іншої в сенсі деякої вибраної норми є задачею *апроксимації*. Але найчастіше сьогодні задачу згладжування також відносять до методів апроксимації.

3.1. Метод найменших квадратів

Метод найменших квадратів (МНК) є одним із найпоширеніших підходів до згладжування даних (у випадку апроксимації застосовується його *інтегральна форма*). Цей метод широко використовується для обробки експериментальних даних, а також у побудові та дослідженні моделей у техніці, фізиці, біології, економіці та багатьох інших науках.

Суть методу полягає у визначенні параметрів функції заданої форми, яка найкращим чином описує залежність змінної y від змінної x.

Припустимо, що необхідно встановити функціональний зв'язок між двома емпіричними величинами x та y, значення яких наведені у наступній таблиці:

x	x_1	x_2	…	x_i	…	x_n
y	y_1	y_2	…	y_i	…	y_n

Точки $(x_i; y_i)$ координатної площини прийнято називати *експериментальними*.

Оцінюючи розміщення експериментальних точок на координатній площині, можна сформулювати припущення щодо характеру залежності між досліджуваними величинами та визначити відповідну форму функції $y = f(x)$.

В найпростішому випадку, що зображений на рис. 3.1, ми спробуємо апроксимувати дані за допомогою лінійної залежності між x і y, яку можна описати формулою:

$$y = kx + b. \qquad (3.1)$$

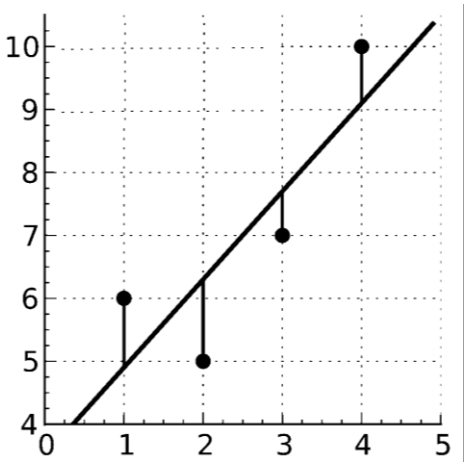


Рис.3.1. Апроксимація лінійною залежністю.

Рівняння прямої представимо у вигляді

$$y-(kx+b)=0\,.$$

Підставляючи замість x и y значення координат цих точок $(x_1;y_1),(x_2;y_2),\ \ldots,\ (x_n;y_n)$ у вираз $y-(kx+b)$, одержуємо:

$$y_1-(kx_1+b)=\delta_1,\ y_2-(kx_2+b)=\delta_2,\ \ldots,\ y_n-(kx_n+b)=\delta_n,$$

де $\delta_1,\delta_2,\ \ldots,\ \delta_n$ — називають *похибками* (*відхиленнями, нев'язками*).

Очевидно, що чим менші ці похибки за абсолютним значенням, тим точніше пряма, задана рівнянням $y=kx+b$, відображає залежність між змінними.

Суть методу найменших квадратів полягає у підборі таких коефіцієнтів k і b апроксимуючої функції, які мінімізують суму квадратів відхилень (похибок):

$$S=\delta_1^2+\delta_2^2+\ldots+\delta_n^2=\sum_{i=1}^{n}\delta_i^2=\sum_{i=1}^{n}(y_i-(kx_i+b))^2\to\min \qquad (3.2)$$

Зазначимо, що необхідність підсумовувати саме квадрати похибок виникає остільки, оскільки при додаванні самих похибок додатні й від'ємні значення можуть взаємно компенсуватись, що спотворює реальну оцінку точності апроксимації δ_i.

У зв'язку з тим, що у вказаній рівності x_i і y_i — задані числа, а k і b — невідомі, то суму S можна розглянути як функцію двох змінних k і b: $S=S(k,b)$. Поставлена задача буде вирішена, якщо ми знайдемо екстремум цієї функції, і цей екстремум буде мінімумом.

Запишемо необхідну умову існування екстремума функції двох змінних:

$$\begin{cases} \dfrac{\partial S}{\partial k} = 0, \\ \dfrac{\partial S}{\partial b} = 0; \end{cases}$$

$$\frac{\partial S}{\partial k} = 2\sum_{i=1}^{n}(y_i - (kx_i + b))(-x_i) = -2\sum_{i=1}^{n}(y_i - (kx_i + b))\,x_i,$$

$$\frac{\partial S}{\partial b} = 2\sum_{i=1}^{n}(y_i - (kx_i + b))(-1) = -2\sum_{i=1}^{n}(y_i - (kx_i + b)).$$

Прирівнюючи часткові похідні до нуля, одержуємо лінійну систему двох рівнянь із двома невідомими k і b:

$$\begin{cases} -2\sum_{i=1}^{n}(y_i - (kx_i + b))\,x_i = 0, \\ -2\sum_{i=1}^{n}(y_i - (kx_i + b)) = 0. \end{cases} \tag{3.3}$$

У результаті перетворення першого рівняння системи отримаємо:

$$-\sum_{i=1}^{n} y_i x_i + k\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = 0\,.$$

Перетворюючи друге рівняння системи, отримаємо

$$-\sum_{i=1}^{n} y_i + k\sum_{i=1}^{n} x_i + bn = 0\,.$$

Збираючи результати, отримуємо систему:

$$\begin{cases} k\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i x_i, \\ k\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i. \end{cases} \tag{3.4}$$

Ця система називається *нормальною системою*.

Із (3.4) визначаємо значення k і b, та підставляємо їх у рівняння прямої. Той факт, що функція $S = S(k,b)$ в знайденій точці (k,b) є точкою мінімуму, визначається за допомогою часткових похідних другого порядку, зокрема — аналізом знаку визначника матриці Гессе.

$$\frac{\partial^2 S}{\partial k^2} = -2\sum_{i=1}^{n}(-x_i)\,x_i = 2\sum_{i=1}^{n}(x_i)^2,$$

$$\frac{\partial^2 S}{\partial b^2} = -2\cdot\sum_{i=1}^{n}(-1) = 2n,$$

$$\frac{\partial^2 S}{\partial k\,\partial b} = -2\sum_{i=1}^{n}(-x_i) = 2\sum_{i=1}^{n}x_i\,.$$

Обчислимо $\Delta = \dfrac{\partial^2 S}{\partial k^2}\cdot\dfrac{\partial^2 S}{\partial b^2} - \left(\dfrac{\partial^2 S}{\partial k\,\partial b}\right)^2$.

$$\Delta = 4n\sum_{i=1}^{n}(x_i)^2 - \left(2\sum_{i=1}^{n}x_i\right)^2 = 2\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2.$$

Очевидно, $\Delta > 0$, і, таким чином, у знайденій точці (k,b) функція $S = S(k,b)$ має екстремум; а так як $\partial^2 S/\partial k^2 > 0$, то, згідно з достатньою умовою екстремума функції двох змінних, у точці (k,b) функція має мінімум.

Отримана функція $y = kx + b$ називається *лінійною регресією*, а коефіцієнти *k* і *b* — *коефіцієнтами регресії*.

Зв'язок між експериментальними даними може наближено мати квадратичний характер (як показано на рис. 3.2). У такому разі завдання зводиться до визначення коефіцієнтів a_2, a_1, a_0 для складання рівняння виду $y = a_2x^2 + a_1x + a_0$.

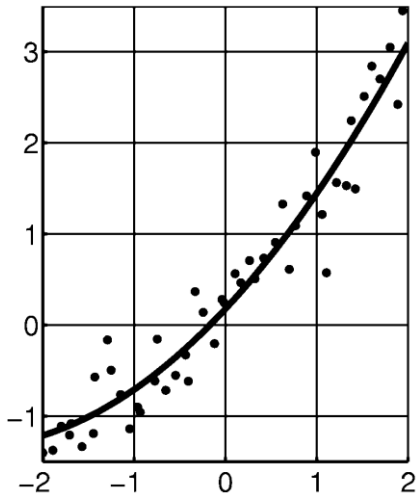


Рис. 3.2. Апроксимація квадратичною залежністю.

Проводячи аналогічні розрахунки, легко показати, що для визначення коефіцієнтів a_2, a_1, a_0 слід розв'язати систему рівнянь:

$$\begin{cases} n a_0 + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i, \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i, \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i. \end{cases} \quad (3.5)$$

На практиці в якості наближуючи функцій, крім $y = kx + b$ й $y = a_2 x^2 + a_1 x + a_0$, залежно від характеру точкового графіка часто використовуються такі функції:

$$y = ax^m,\ y = ae^{mx},\ y = \frac{1}{ax+b},\ y = \frac{a}{x} + b,\ y = \frac{x}{ax+b},\ y = a \ln x + b.$$

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики розрахунок та візуалізацію апроксимації методом найменших квадратів системи точок, що в логарифмічній системі описується лінійним графіком. Вхідні дані задаються індивідуально.

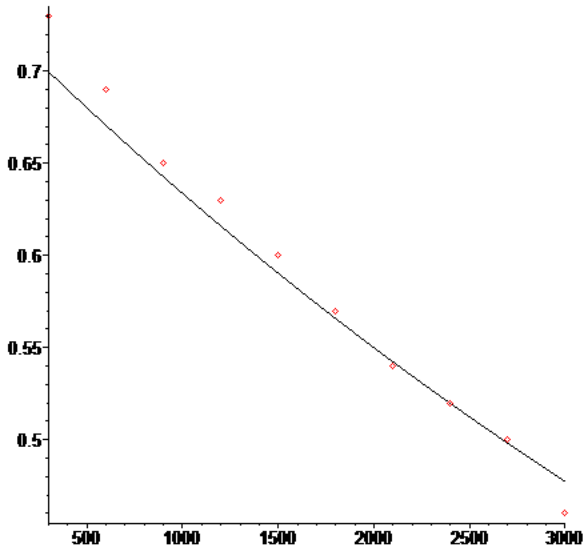
ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Оновлення середовища Maxima */
restart;

/* Завантаження бібліотек для підгонки кривих та графіків */
load(CurveFitting);
load(plots);

/* Визначення векторів з даними */
Tau: [300, 600, 900, 1200, 1500, 1800, 2100, 2400, 2700, 3000];
dcr: [0.73, 0.69, 0.65, 0.63, 0.6, 0.57, 0.54, 0.52, 0.5, 0.46];

/* Обчислення натуральних логарифмів значень dcr */
lndcr: [log(dcr[i])$i:1..length(Tau)];

/* Підгонка кривої методом найменших квадратів для логарифмів dcr */
LeastSquares(Tau, lndcr, tau, curve = log(dcr[1]) - b*tau);

/* Обчислення функції Y на основі підгонки */
Y: dcr[1] * exp(diff(% , tau) * tau);

/* Побудова графіку точок даних */
A: plot2d([[Tau[i], dcr[i]]$i:1..length(Tau)], [style, points]);

/* Побудова графіку підігнаної кривої */
B: plot2d(Y, [tau, min(Tau)..max(Tau)], [color, black]);

/* Відображення обох графіків разом */
display(A, B);
```

Система **Maple 10**:

```
restart:
with(CurveFitting):
with(plots):
Tau:=[300,600,900,1200,1500,1800,2100,2400,2700,3000];
dcr:=[0.73,0.69,0.65,0.63,0.6,0.57,0.54,0.52,0.5,0.46];

lndcr:=[ln([0.73,0.69,0.65,0.63,0.6,0.57,0.54,0.52,0.5,0.46][ic])$ic=1..nops(
Tau)]:
LeastSquares(Tau, lndcr, tau, curve=ln(dcr[1])-b*tau):
Y:=dcr[1]*exp(diff(%,tau)*tau);
A:=plot([[Tau[ic],dcr[ic]]$ic=1..nops(Tau)],style=point):
B:=plot(Y,tau=Tau[1]..Tau[nops(Tau)],color=black):

display(A,B);
```



Контрольні питання.

1. У чому полягає різниця між інтерполяцією та апроксимацією?
2. Сформулюйте основну ідею методу найменших квадратів.
3. Яка функція називається апроксимуючою в методі найменших квадратів?
4. Яке рівняння потрібно мінімізувати при знаходженні апроксимуючої функції методом найменших квадратів?
5. Який вигляд має апроксимуючий поліном другого ступеня, побудований за методом найменших квадратів?

Лабораторна робота №4

Тема: Розв’язок систем нелінійних рівнянь методом Ньютона-Рафсона

Метод Ньютона-Рафсона — це чисельний алгоритм для знаходження коренів (нулів) нелінійного рівняння вигляду $f(x) = 0$. Метод базується на ітераційному наближенні. Починаючи з деякого початкового наближення x_n, нове значення x_{n+1} обчислюється за формулою:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{4.1}$$

Таким чином кожна наступна точка визначається як точка перетину дотичної до графіка функції в поточній точці з віссю x. Функція $f(x)$ має бути диференційовною в околиці кореня. Початкове наближення має бути досить близьким до істинного розв’язку, інакше метод може не збігатися. Метод має низку переваг, серед яких основними є висока швидкість збіжності (один з найшвидших методів) при хорошому початковому наближенні та простота реалізації.

Розв'язок систем нелінійних рівнянь у принципі не відрізняється від розв'язку одного рівняння.

Нехай задана система рівнянь у вигляді

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0; \\ f_1(x_1, x_2, ..., x_n) = 0; \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0. \end{cases}$$

Побудуємо три матриці: перша — це стовпчиковий вектор змінних, друга — стовпчиковий вектор значень функцій, а третя — *матриця Якобі*, що містить часткові похідні функцій за відповідними змінними:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ .... \\ x_n \end{bmatrix}; \quad F = \begin{bmatrix} f_1(x_1, x_2, ..., x_n) \\ f_2(x_1, x_2, ..., x_n) \\ ..... \\ f_n(x_1, x_2, ..., x_n) \end{bmatrix}; \quad F' = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ ..... \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

Тоді ітераційна формула буде мати вигляд

$$X_{n+1} = X_n - (F')^{-1} F. \qquad (4.1)$$

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики розрахунок системи алгебраїчних рівнянь методом Ньютона-Рафсона. Виконати у вигляді процедури. Вхідні дані задаються індивідуально.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Визначення функції NLSNd */
NLSNd(equations_set, num_iterations, initial_approximation) :=
block(
    [num_vars, i, j, indets, var_set, matrix, vector, X, min_x, min_y, max_x,
max_y,
    index, var_matrix, mm1_matrix, var_approx, mm_matrix, sit, k, mBm, mMm,
cm, MMM,
    LIST_X, XV, Xstring, Nbracket, NameX, Nindex, vector, rhs_string,
lhs_string],

    /* Ініціалізація змінних */
    num_vars: length(equations_set),
    X: map(lambda([index], equations_set[index]), makelist(i, i, 1,
num_vars)),
    if initial_approximation = [] then (
        LIST_X: [],
        XV: makelist(0, i, 1, num_vars)
    ) else (
        LIST_X: initial_approximation[1],
        XV: initial_approximation[2]
    ),
    /* Друк початкових значень */
    print(map(lambda([index], XV[index]), makelist(index, index, 1,
num_vars))),
    /* Обробка строки індексів */
    Xstring: string(equations_set[1]),
    Nbracket: position("[", Xstring),
    NameX: parse(substring(Xstring, 1, Nbracket - 1)),
    Nindex: length(split(",", Xstring)) + 1,
    /* Ітерації */
    for sit:1 thru num_iterations do (
        MMM: matrix(num_vars, num_vars),
        for j:1 thru num_vars do (
            indets: indets(equations_set[j], indexed),
            for i:1 thru length(indets) do (
                if not(member(indets[i], LIST_X, 'k')) and sit = 1 then (
                    LIST_X: append(LIST_X, [indets[i]]),
                    MMM[j, length(LIST_X)]:=
evalf(substitute(LIST_X[i]:=XV[i],
                    diff(equations_set[j], indets[i])))
                ) else (
                    MMM[j, k]:= evalf(substitute(LIST_X[i]:=XV[i],
diff(equations_set[j],
                                indets[i])))
                )
            )
        ),
        print(MMM),
        /* Рішення системи та оновлення значень */
```

```
        vector: substitute(LIST_X[i]:=XV[i], X),
        mm1_matrix: inverse(MMM),
        var_approx: mm1_matrix . vector,

        for i:1 thru num_vars do (
            XV[i]:= XV[i] - var_approx[i]
        ),
        print(map(lambda([index], XV[index]), makelist(index, index, 1,
num_vars)))
    ),
    /* Формування та вивід строки результату */
    lhs_string: string(LIST_X),
    rhs_string: string(XV),
    lhs_string: delete(rhs_string, length(lhs_string)),
    rhs_string: delete(rhs_string, length(rhs_string)),
    parse(string(join([lhs_string, ":=", rhs_string], " ")))
);
/* Приклад виклику функції */
NLSNd([eq1, eq2, eq3], 10, [[x1, x2], [0, 1]])
```

Система **Maple 10**:

```
NLSNd := proc (equations_set, num_iterations, initial_approximation)
  local num_vars, i, j, Indets, var_set, matrix, vector, X, min_x, min_y,
max_x, max_y, ind, var_matrix,
  mm1_matrix, var_approx, mm_matrix, sit, k, mBm, mMm, cm, MMM;
  num_vars := nops(equations_set); print(num_vars);
  X := [`$`(equations_set[ind], ind = 1 .. num_vars)];
  if `or`(initial_approximation = 0, initial_approximation = NULL) then
   LIST_X := [];
   XV := array(sparse, 1 .. num_vars)
  else
   LIST_X := initial_approximation[1];
   XV := convert(initial_approximation[2], array)
  end if;
   Xstring := convert(indets(equations_set, indexed), string);
   Xstring1 := convert(indets(equations_set[1], indexed)[1], string);
   print(Xstring);
   Nbracket := StringTools[Search]("[", Xstring1);
   NameX := parse(StringTools[SubString](Xstring1, 1 .. `+`(Nbracket, `-
`(1))));
   print(NameX);
   Nindex := `+`(nops({StringTools[SearchAll](",", Xstring1)}), 1);
   for sit to num_iterations do
    MMM := array(sparse, 1 .. num_vars, 1 .. num_vars);
    for j to num_vars do Indets := indets(equations_set[j], indexed);
     for i to nops(Indets) do
      if `and`(`not`(member(Indets[i], LIST_X, 'k')), sit = 1) then
       LIST_X := [op(LIST_X), Indets[i]]; print(LIST_X, jj);
       MMM[j, nops(LIST_X)] := evalf(subs(`$`(LIST_X[ic] = XV[ic], ic = 1 ..
nops(LIST_X)), diff(equations_set[j], Indets[i])))
      else
       MMM[j, k] := evalf(subs(`$`(LIST_X[ic] = XV[ic], ic = 1 ..
nops(LIST_X)), diff(equations_set[j], Indets[i])))
      end if
     end do
    end do;
    print(MMM);
    subs(`$`(LIST_X[ind] = XV[ind], ind = 1 .. num_vars), X);
    vector := convert(%, array);
    mm1_matrix := Matlab[inv](MMM);
    Matlab[setvar]("mMm1vsl", mm1_matrix);
```

```
    Matlab[setvar]("mBmvsl", vector);
    Matlab[evalM]("cm = (mMm1vsl) * mBmvsl");
    var_approx := Matlab[getvar]("cm");
    for i to num_vars do
     XV[i] := `+`(XV[i], `-`(var_approx[i]))
    end do;
    print(seq(XV[ind], ind = 1 .. num_vars))
   end do;
   LHS := convert(LIST_X, string);
   LLHS := StringTools[Length](LHS);
   LHS := StringTools[Delete](LHS, LLHS .. LLHS);
   LHS := StringTools[Delete](LHS, 1 .. 1);
   RHS := convert(convert(XV, list), string);
   LRHS := StringTools[Length](RHS);
   RHS := StringTools[Delete](RHS, LRHS .. LRHS);
   RHS := StringTools[Delete](RHS, 1 .. 1);
   parse(StringTools[Join]([LHS, ":=", RHS], " "));
   %
end proc:
```

Завдання рівнянь та запуск процедури:

```
Ur:={x[1]^2-5*x[2]-3,4*x[1]+2*x[2]-1};
NLSNd(Ur,3,0);
```

0, 0
0.5500000000, -0.6000000000
0.5227477478, -0.5454954955
0.5226805090, -0.5453610179

Контрольні питання.

1. У чому полягає суть методу Ньютона-Рафсона для одного нелінійного рівняння?
2. Як записується метод Ньютона-Рафсона для системи нелінійних рівнянь?
3. Які переваги та недоліки має метод Ньютона-Рафсона при розв’язуванні систем?
4. Як визначити, що ітераційний процес досягнув необхідної точності?
5. Що таке якобіан і яку роль він відіграє в методі Ньютона для систем?

Лабораторна робота №5

Тема: Метод скінченних елементів

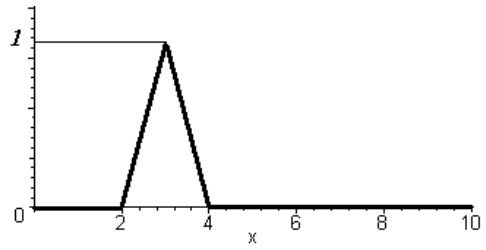
5.1. Основи метода

В лабораторній роботі № 2 розглядалась інтерполяція лінійними сплайнами. На основі вказаної інтерполяції вже можна побудувати модель теорії скінченно-елементного аналізу.

Нагадаємо, що лінійний сплайн можна представити як комбінацію

$$\omega(x) = \sum \alpha_i \varphi_i(x).$$

Функції φ_i є *ваговими*, їх крім того також називають *базисними*, або *функціями форми*, бо вони тотожні таким функціям у методі зважених неув'язок. Як вже зазначалося, важливо те, що ми перейшли до форми, яка покладена в основу метода зважених нев'язок, але ж з особливими функціями, які мають для випадку *лінійної інтерполяції* характерний трикутний вигляд:



та, крім того, як ми записали — є адитивними (підсумовуються), та *лінійно незалежними*.

Таким чином, можна намагатися застосувати їх для отримання наближеного рішення *методом Галеркіна*. Цей метод в одній з постановок, що знайшла своє поширення серед інженерних застосувань, має наступний сенс. Нехай ми шукаємо наближене рішення деякого неоднорідного рівняння $L\omega=f$ з диференціальним оператором L у тому самому вигляді: $\omega(x) = \sum \alpha_i \varphi_i$, тобто

$$L\omega(x) = L\sum \alpha_i \varphi_i(x) \approx f\,. \tag{5.1}$$

Таким чином величина $L\sum \alpha_i \varphi_i(x) - f$ ненульова, позначимо її як

$$L\sum \alpha_i \varphi_i(x) - f = \delta(\omega_1, \omega_2, ..., x) \tag{5.2}$$

— *нев'язка* наближеного рішення. Основна задача — мінімізувати цю нев'язку.

Мінімізувати нев'язку можна тільки за рахунок правильного підбору $\alpha_1, \alpha_2, ..., \alpha_n$. Саме рівняння $L\sum\alpha_i\varphi_i(x) - f = \delta(\alpha_1, \alpha_2, ..., \alpha_n, x)$ має купу невідомих α_i — наближених значень невідомої функції у вузлах сітки.

Уявимо функцію $v_i(x)$ вектором в деякому просторі (кожне значення $v_i(x)$ в деякій точці *x* — компонента цього вектора в такому просторі, а самий *x* — «номер» вісі деякої координатної системи в такому просторі). Хоча такий простір може здатися нескінченновимірним, але ж він має границі, що співпадають з областю визначення функції, в даному випадку від 0 до *L*. Усі функції $v_1(x), v_2(x), ..., v_n(x)$ задані звичайно в одному й тому ж просторі (а саме, на тієї ж множині *x* дійсних чисел — тобто множина, на якій задана функція, уявляється як простір, в якому вона є вектором, і порівнювати можна тільки ті функції, що задані в одному просторі — як ми звичайно порівнюємо двовимірні фігури з двовимірними, тривимірні з тривимірними). З такого представлення випливає, що для вектору $\omega(x)$ запис $\omega(x) = \sum v_i\varphi_i$ є розкладанням його по базису $v_1(x), v_2(x), ..., v_n(x)$ у просторі $\{x: x \in [0, L]\}$, — таким самим чином в тривимірному просторі можна розкласти вектор по, наприклад, двох заданих наперед інших векторах (він буде знаходитися в одній площині з цими двома). Саме від цього пішла назва функцій — базисні.

Введемо ще деяку іншу функцію $\psi(x)$. Її теж будемо вважати вектором, що заданий в тому ж просторі. Проєкція одного вектора на іншій буде пропорційна, як відомо з векторного аналізу, скалярному їх добутку, тобто сумі добутків відповідних компонент (наприклад, $\mathbf{AB} = \sum A_i B_i$ для класичного векторного аналізу). Якщо подовжити таке визначення для наших функцій-векторів, то *скалярним добутком однієї функції на іншу* буде теж сума, але сума нескінченної кількості компонент, — тобто інтеграл по цьому простору, а саме: $\int_0^L v_i(x)\psi(x)dx$. Різниця між інтегралом від цих функцій та безпосередньо функціями така ж, як між скалярним добутком та векторами. Довжина вектора $v_i(x)$ може тепер бути знайдена як $\sqrt{\int_0^L v_i^2(x)dx}$. Уявимо, що всі $v_1, v_2, ..., v_n$ вже задані, тобто нев'язка $\delta(x)$ є вектор в тому ж просторі, що і кожна $v_i(x)$ та $\psi(x)$. Якщо скалярний добуток нульовий, то це означає, що вектори *ортогональні*. Якщо б проєкції $\delta(x)$ на деякі *n* функцій $\psi_i(x)$ стали всі нульовими, то це означало б, що вектор у просторі, що розглядається, став нульовим. У методі Гальоркіна в якості такого простору вибираємо все ті ж самі базисні функції φ_i:

$$\int_\Omega \delta(\alpha_1, \alpha_2, ..., \alpha_n, x)\varphi_i dx = 0, i = 1..N \quad . \qquad (5.3)$$

Це дає систему з N рівняннями та N невідомими, що може бути вирішена також чисельними методами (наприклад, методом Ньютона-Рафсона). Також для зведення до системи алгебраїчних рівнянь використовують методи *колокацій* (вибирається простір з дельта-функцій), та *інтегральний метод найменших квадратів*.

Розглянемо рішення на прикладі рівняння прогину однопрогонної балки (рис. 5.1):

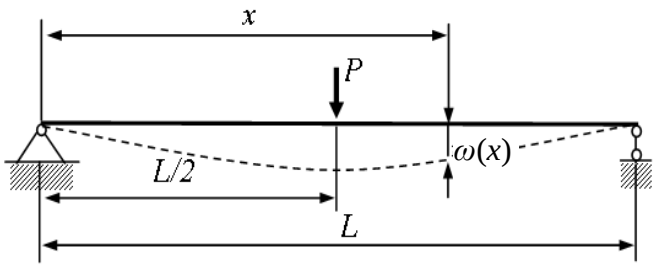


Рис. 5.1. Розрахункова задача для ілюстрації метода скінченних елементів

Диференціальне рівняння деформації балки (ми з ним стикалися в роботі №2) має вигляд

$$\frac{d^2\omega}{dx^2} = \frac{M(x)}{EJ}. \tag{5.4}$$

Зазвичай, для того, щоб адекватно врахувати граничні умови, та для позбавлення від можливих дельтаподібних функцій, нам треба провести інтегрування по частинам рівняння (5.4) в проєкції на вагові функції (у відповідності з методом Гальоркіна)

$$\int_0^L \omega''\varphi_i dx = \omega'\varphi_i\Big|_0^L - \int_0^L \omega'\varphi_i' dx = \frac{1}{EJ}\int_0^L M\varphi_i dx.$$

З врахуванням граничних умов, перша складова середньої частини обертається в нуль, тому маємо

$$-\int_0^L \omega'\varphi_i' dx = \frac{1}{EJ}\int_0^L M\varphi_i dx. \tag{5.5}$$

Такий запис вихідного рівняння в методі скінченних елементів має назву *варіаційної* (*слабкої*) *форми*. Далі необхідно замість реального прогину $\omega(x)$ підставити відповідну інтерполяцію лінійним сплайном $\omega(x) = \sum \alpha_i \varphi_i(x)$.

Таким чином, у зв'язку з лінійністю диференціального рівняння маємо лінійну систему рівнянь, що в матричному вигляді для, наприклад, 9-ти розбиттів балки та використання на кожній з ділянок базисних функцій лінійного сплайна має вигляд:

$$\begin{bmatrix} 2. & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -1. & 2. & -1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 2. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & -1. & 2. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & 2. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & -1. & 2. & -1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & -1. & 2. & -1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & -1. & 2. & -1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -1. & 2. \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} -0.02 \\ -0.04 \\ -0.06 \\ -0.08 \\ -0.09 \\ -0.08 \\ -0.06 \\ -0.04 \\ -0.02 \end{bmatrix}$$

Відповідна матриця системи має назву *матриці жорсткості*. В результаті маємо наступний вигляд деформованої балки:

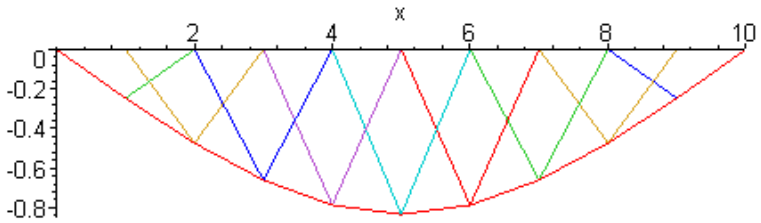


Рис. 5.2. Форма прогину балки за МСЕ з використанням лінійних сплайнів (лінійні скінченні елементи)

Якщо в якості основного сплайна взяти кубічний сплайн, то також отримуємо систему лінійних рівнянь, але ж більшого порядку (рис. 5.3).

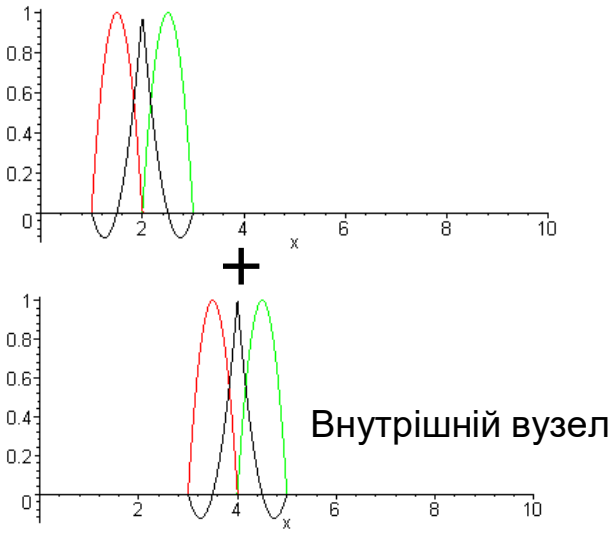


Рис. 5.3. Ілюстрація кубічних скінченних елементів

$$
\begin{bmatrix}
4.666666667 & 0.3333333333 & 0 & 0 & 0 & 0 & 0 & 0 & -2.666666667 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.3333333333 & 4.666666667 & 0.3333333333 & 0 & 0 & 0 & 0 & 0 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.3333333333 & 4.666666667 & 0.3333333333 & 0 & 0 & 0 & 0 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.3333333333 & 4.666666667 & 0.3333333333 & 0 & 0 & 0 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3333333333 & 4.666666667 & 0.3333333333 & 0 & 0 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.3333333333 & 4.666666667 & 0.3333333333 & 0 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.3333333333 & 4.666666667 & 0.3333333333 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.3333333333 & 4.666666667 & -5.333333333 & -5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3333333333 & 4.666666667 & -5.333333333 & -2.666666667 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2.666666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2.666666667 & -2.666666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -2.666666667 & -2.666666667 & 0 & 0 & 0 & 0 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2.666666667 & -2.666666667 & 0 & 0 & 0 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2.666666667 & -2.666666667 & 0 & 0 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2.666666667 & -2.666666667 & 0 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2.666666667 & -2.666666667 & 0 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2.666666667 & -2.666666667 & 0 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.666666667 & -2.666666667 & 10.66666667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.666666667 & 5.333333333 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ U_8 \\ U_9 \\ U_{1/2} \\ U_{3/2} \\ U_{5/2} \\ U_{7/2} \\ U_{9/2} \\ U_{11/2} \\ U_{13/2} \\ U_{15/2} \\ U_{17/2} \\ U_{19/2}
\end{bmatrix}
=
\begin{bmatrix}
-0.006666666667 \\ -0.01333333333 \\ -0.02000000000 \\ -0.02666666667 \\ -0.03333333333 \\ -0.02666666667 \\ -0.02000000000 \\ -0.01333333333 \\ -0.006666666667 \\ -0.006666666667 \\ -0.02000000000 \\ -0.03333333333 \\ -0.04666666667 \\ -0.06000000000 \\ -0.06000000000 \\ -0.04666666667 \\ -0.03333333333 \\ -0.02000000000 \\ -0.006666666667
\end{bmatrix}
$$

Вирішуючи вказану систему отримаємо деформовану форму з більшою гладкістю (рис. 5.4).

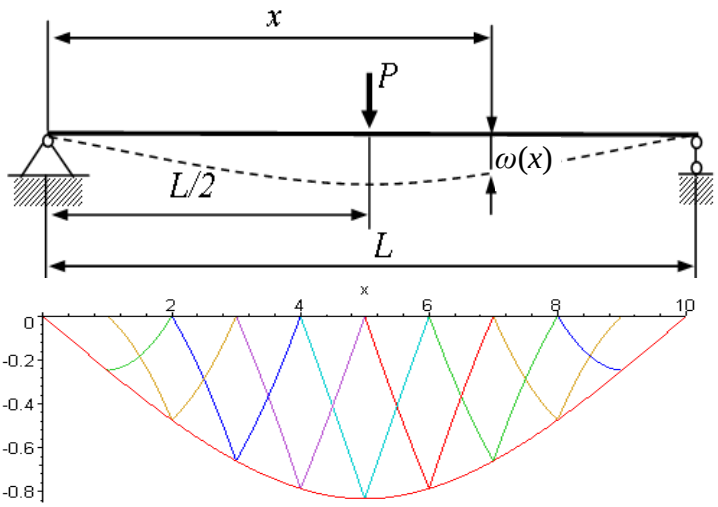


Рис. 5.4. Форма прогину балки за МСЕ з використанням кубічних сплайнів (кубічні скінченні елементи)

В якості інтерполяційної функції можна використовувати також сплайни іншого типу. Для демонстрації використання сплайнів з *поліномами Ерміту* (див. лаб. роботу №2), розглянемо приклад деформації балки з врахуванням роподіленного на неї навантаження. В цьому випадку плоского згину балки з постійними жорсткісними характеристиками, рівняння рівноваги записується у вигляді:

$$EI\frac{d^4\omega}{dx^4} = q(x), x \in [0, L],$$

де $q(x)$ — відоме розподілене навантаження на одиницю довжини (Н/м).

Для повної постановки задачі необхідно задати граничні умови. Залежно від типу закріплення кінців балки можливі різні варіанти. Розглянемо випадок балки, жорстко закріпленої на обох кінцях (консольно затиснута балка):

$$w(0) = 0, \quad w'(0) = 0, \quad w(L) = 0, \quad w'(L) = 0.$$

Тут $\omega(0) = \omega\,(L) = 0$ — переміщення кінців балки дорівнює нулю; $\omega'(x)$ — кут повороту поперечного перерізу, також дорівнює нулю на кінцях.

Оскільки рівняння є диференціальним рівнянням четвертого порядку, для застосування МСЕ необхідно знизити порядок похідних. Це також досягається через варіаційну (слабку) постановку задачі. Для цього множимо рівняння на тестову функцію $v(x) \in V$, де V — відповідний функціональний простір, та інтегруємо по області:

$$\int_0^L EI \frac{d^4\omega}{dx^4} v(x)dx = \int_0^L q(x)v(x)dx.$$

Застосовуючи інтегрування частинами двічі, і враховуючи, що $v(0)=v'(0)=v(L)=v'(L)=0$, отримаємо слабку форму задачі:

$$\int_0^L EI \frac{d^4\omega}{dx^4} v dx = \left[\frac{d^3\omega}{dx^3} v \right]_0^L - \int_0^L \frac{d^3\omega}{dx^3} \frac{dv}{dx} dx.$$

Таким чином

$$EI\left(\left[\frac{d^3\omega}{dx^3} v \right]_0^L - \int_0^L \frac{d^3\omega}{dx^3} \frac{dv}{dx} dx \right) = \int_0^L q(x)v(x)dx.$$

Друга складова в дужках розкривається як

$$\int_0^L \frac{d^3\omega}{dx^3} \frac{dv}{dx} dx = \left[\frac{d^2\omega}{dx^2} \frac{dv}{dx} \right]_0^L - \int_0^L \frac{d^2\omega}{dx^2} \frac{d^2v}{dx^2} dx.$$

Тоді

$$EI\left(\left[\frac{d^3\omega}{dx^3} v \right]_0^L - \left[\frac{d^2\omega}{dx^2} \frac{dv}{dx} \right]_0^L + \int_0^L \frac{d^2\omega}{dx^2} \frac{d^2v}{dx^2} dx \right) = \int_0^L q(x)v(x)dx.$$

Таким чином отримаємо

$$\int_0^L EI \frac{d^2\omega}{dx^2}\frac{d^2v}{dx^2}dx = \int_0^L q(x)v(x)dx.$$

Знову ділимо відрізок $[0, L]$ на N однакових елементів довжини h, тобто:

$$h = \frac{L}{N}.$$

Зазначимо координати вузлів:

$$x_0 = 0 < x_1 < \cdots < x_N = L,$$

де $x_i = ih$, $i = 0,1,\ldots,N$.

Кожен інтервал $[x_i, x_{i+1}]$ утворює один скінченний елемент. Оскільки слабка форма містить другі похідні від функції $\omega(x)$, для побудови апроксимації необхідно, щоб функції належали простору C^1 — були неперервними разом з похідною.

Як вже зазначалось, будемо далі використовувати кубічні поліноміальні елементи Ерміта (*Hermite beam elements*), які забезпечують неперервність як функції, так і її першої похідної. На кожному елементі $e=[x_i, x_{i+1}]$ шукану функцію $\omega(x)$ апроксимують як:

$$\omega_h(x) = \sum_{j=1}^{4} a_j^{(e)} \phi_j^{(e)}(x),$$

де $a_j^{(e)}$ — *локальні степені свободи* на елементі (значення прогину та кутів повороту в кінцевих точках), $\phi_j^{(e)}(x)$ — кубічні поліноміальні базисні функції Ерміта.

Для кожного вузла (вузлової точки) у балці маємо дві степені свободи:

1. ω_i — прогин у точці x_i,
2. $\theta_i = \omega_i'$ — кут повороту перерізу балки у точці x_i.

Таким чином, загальна кількість степенів свободи в задачі:

$$n = 2(N + 1),$$

але з урахуванням граничних умов $\omega_0 = \theta_0 = \omega_N = \theta_N = 0$ кількість невідомих зменшується.

Глобальна апроксимація прогину виглядає як лінійна комбінація глобальних базисних функцій $\phi_i(x)$:

$$\omega_h(x) = \sum_{j=1}^{4} W_i \phi_i(x).$$

На кожному елементі використовуються чотири базисні функції ϕ_1, ϕ_2, ϕ_3, ϕ_4, що відповідають:

1. ϕ_1: одиничний прогин у лівому вузлі, нуль в усіх інших ступенях свободи,
2. ϕ_2: одиничний кут повороту в лівому вузлі,
3. ϕ_3: одиничний прогин у правому вузлі,
4. ϕ_4: одиничний кут повороту в правому вузлі.

Їхня аналітична форма у локальних координатах $\xi \in [0, h]$ має вигляд (у безрозмірному вигляді, див. також (2.14)):

$$\phi_1(\xi) = 1 - 3\left(\frac{\xi}{h}\right)^2 + 2\left(\frac{\xi}{h}\right)^3,$$
$$\phi_2(\xi) = \xi\left(1 - \frac{\xi}{h}\right)^2,$$
$$\phi_3(\xi) = 3\left(\frac{\xi}{h}\right)^2 - 2\left(\frac{\xi}{h}\right)^3,$$
$$\phi_4(\xi) = \xi\left(\frac{\xi}{h} - 1\right)^2.$$

Або аналогічно в інтервалі $\xi \in [-1, 1]$ при зміні змінної.

Нехай елемент розташований на відрізку $x \in [0, h]$, де h — довжина елемента. На цьому елементі прогин $\omega(x)$ апроксимується кубічним поліномом:

$$\omega_h(x) = \sum_{j=1}^{4} a_i \phi_i(x),$$

де:

- $a_1 = \omega(0)$, прогин на лівому кінці,
- $a_2 = \omega'(0)$, кут повороту на лівому кінці,
- $a_3 = \omega(h)$, прогин на правому кінці,
- $a_4 = \omega'(h)$, кут повороту на правому кінці,
- $\phi_i(x)$, $i=1,\ldots,4$ — базисні функції Ерміта.

У векторному вигляді:

$$\omega_h(x) = \boldsymbol{\varphi}^T(x)\mathbf{a},$$

де

$$\boldsymbol{\varphi}(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_3(x) \\ \phi_4(x) \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} \omega(x) \\ \omega'(x) \\ \omega(h) \\ \omega'(h) \end{bmatrix}.$$

На одному елементі слабка форма записується як:

$$\int_0^h EI\omega_h'' v_h'' dx = \int_0^h q(x)v(x)dx.$$

Оскільки $v_h(x)=\boldsymbol{\phi}^\top(\mathbf{x})\mathbf{b}$, де $\mathbf{b}$ — довільний вектор степенів свободи, то підставляємо апроксимацію:

$$\int_0^h EI\left(\boldsymbol{\varphi}''^T(x)\mathbf{a}\right)\left(\boldsymbol{\varphi}''^T(x)\mathbf{b}\right)dx.$$

Або, згрупувавши:

$$\mathbf{b}^T\left(\int_0^h EI\boldsymbol{\varphi}''(x)\boldsymbol{\varphi}''^T(x)\mathbf{a}dx\right).$$

Отже, *локальна матриця жорсткості* елемента визначається як:

$$\mathbf{K}^{(e)} = \int_0^h EI\boldsymbol{\varphi}''(x)\boldsymbol{\varphi}''^T(x)dx.$$

Для стандартного елемента довжини h, при сталій жорсткісті, значення інтегралів можна обчислити аналітично. Відомо, що аналітично обчислена локальна матриця жорсткості має вигляд:

$$\mathbf{K}^{(e)} = \frac{EI}{h^3}\begin{bmatrix} 12 & 6h & -12 & 6h \\ 6h & 4h^2 & -6h & 2h^2 \\ -12 & -6h & 12 & -6h \\ 6h & 2h^2 & -6h & 4h^2 \end{bmatrix}$$

Це симетрична матриця розміром 4×4, де:

- рядки та стовпці відповідають локальним степеням свободи: ω_1, θ_1, ω_2, θ_2,
- коефіцієнти враховують як переміщення, так і кути повороту на обох кінцях елемента.

Якщо розподілене навантаження $q(x)$ є сталою функцією q_0, тоді *вектор правої частини* має вигляд:

$$\mathbf{F}^{(e)} = \frac{q_0 h}{12} \begin{bmatrix} 6 \\ h \\ 6 \\ -h \end{bmatrix}$$

Цей вираз отримується шляхом інтегрування базисних функцій Ерміта, помножених на сталу силу q_0.

Після побудови всіх локальних матриць жорсткості та векторів навантаження виконується *глобальна збірка*:

- для кожного елемента його 4×4 матриця вставляється у відповідні позиції *глобальної матриці* **K**,
- враховується, які глобальні номери вузлів відповідають локальним степеням свободи,
- аналогічно формується *глобальний вектор навантаження* **F**.

Після чого отримуємо систему рівнянь:

$$\mathbf{KW} = \mathbf{F},$$

де **W** — глобальний вектор переміщень і кутів повороту у всіх вузлах.
Кожен вузол у задачі згинання балки має дві степені свободи:

- ω_i — прогин у вузлі i,
- θ_i — кут повороту в тому ж вузлі.

Отже, при кількості вузлів $N + 1$ (тобто N елементів), маємо:

$$n_{\text{DOF}} = 2(N + 1)$$

всього глобальних степенів свободи.

Для кожного елемента $e = 1,\ldots,N$, розташованого між вузлами i та $i + 1$, маємо локальну матрицю жорсткості $\mathbf{K}^{(e)}$ розміром 4×4. Цю матрицю потрібно вставити у глобальну матрицю **K** розміром $2(N + 1)\times2(N + 1)$ у позиції, що відповідає локальним степеням свободи:

$$\text{Global indexes} = [2i,\ 2i + 1,\ 2i + 2,\ 2i + 3].$$

Тобто:

- $2i$ — це ω_i,
- $2i + 1$ — це θ_i,
- $2i + 2$ — це ω_{i+1},
- $2i + 3$ — це θ_{i+1}.

Збірка полягає у додаванні відповідних елементів до глобальної матриці:

$$\mathbf{K}_{\text{global}}[\mathrm{I}, \mathrm{J}] += \mathbf{K}^{(e)}[i, j],$$

де **I**, **J** ∈ {2*i*, 2*i* + 1, 2*i* + 2, 2*i* + 3}, а *i*, *j*=0,…,3.

Аналогічно формується *глобальний вектор навантаження* **F** з локальних векторів:

$$\mathbf{F}_{\text{global}}[\mathrm{I}] += \mathbf{F}^{(e)}[i].$$

Для балки з жорстко закріпленими кінцями (обидва кінці нерухомі) граничні умови будуть:

$$w(0) = 0, \quad \theta(0) = 0, \quad w(L) = 0, \quad \theta(L) = 0.$$

Це означає, що степені свободи з індексами:

$$0\ (w_0), \quad 1\ (\theta_0), \quad 2N\ (w_N), \quad 2N + 1\ (\theta_N)$$

є заданими (нульовими), тобто не входять до системи рівнянь як змінні.

Існують два способи врахування граничних умов.

1. *Метод викреслювання*:

– Видаляємо відповідні рядки і стовпці з глобальної матриці **K**,
– Видаляємо відповідні компоненти з вектора **F**,
– Розв’язуємо систему з редукованою кількістю невідомих.

Цей підхід найчастіше застосовується в реалізаціях чисельно.

2. *Метод модифікації системи*.

Задаємо в рядку та стовпці, які відповідають відомим степеням свободи:

- 1 на діагоналі,
- 0 в інших елементах,
- значення вектора **F** дорівнює значенню граничної умови (в нашому випадку — 0).

Після врахування граничних умов маємо редуковану СЛАР:

$$\tilde{\mathbf{K}}\tilde{\mathbf{W}} = \tilde{\mathbf{F}}$$

яку можна розв’язати чисельно, наприклад, *методом Гаусса* або будь-яким прямим/ітераційним методом.

Після обчислення $\tilde{\mathbf{W}}$ — значень прогинів і кутів повороту у вузлах — можемо відновити повний розв’язок по всій області, зокрема побудувати:

- прогин $\omega(x)$,
- кут $\theta(x) = \omega'(\mathrm{x})$,
- момент $M(x) = -EI\,\omega''(x)$,
- силу $Q(x) = -EI\,\omega'''(x)$.

Для двовимірного та тривимірного випадків приведення до слабкого формулювання не становить особливих складностей. Узагальнення інтегрування по частинах дає

$$\int_{\Omega} \frac{\partial^2 \omega}{\partial x^2} v dV = \int_{\partial\Omega} v \frac{\partial \omega}{\partial x} n_x dS - \int_{\Omega} \frac{\partial \omega}{\partial x} \frac{\partial v}{\partial x} dV ,$$

так що з урахуванням граничних умов маємо

$$\int_{\Omega} \frac{\partial^2 \omega}{\partial x^2} v dV = -\int_{\Omega} \frac{\partial \omega}{\partial x} \frac{\partial v}{\partial x} dV .$$

Складаючи відповідні вираження для інших осей, одержимо *тотожність Гріна* для оператора Лапласа

$$\int_{\Omega} v \Delta \omega dV = \int_{\partial\Omega} v \nabla \omega \mathbf{n} dS - \int_{\Omega} \nabla \omega \nabla v dV ,$$

або, з урахуванням граничних умов

$$\int_{\Omega} v \Delta \omega dV = -\int_{\Omega} \nabla \omega \nabla v dV .$$

Розрахункову область необхідно далі апроксимувати сіткою скінченних елементів. Для двовимірних задач сітка найчастіше представляється трикутниками (*триангуляція*) або ж чотирьохкутниками (рис. 5.5).

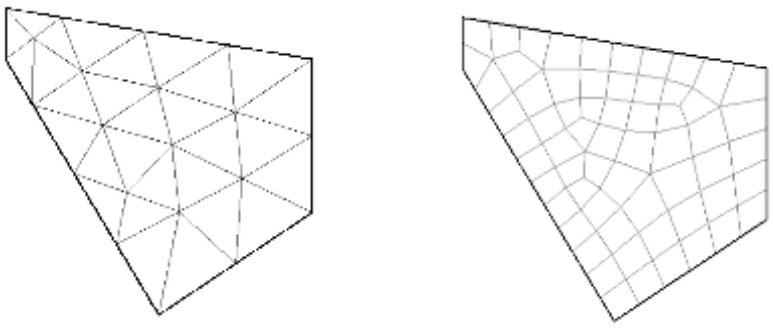


Рис. 5.5. Варіанти розбиття області в методі скінченних елементів

Приклади базисних функцій для двовимірного випадку показані на рисунках нижче.

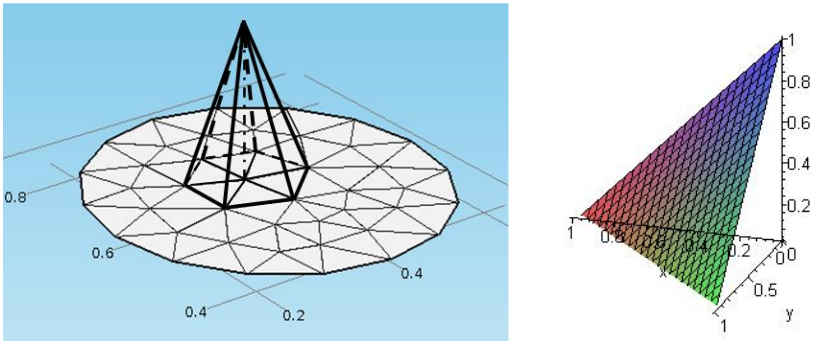


Рис. 5.6. Лінійні трикутні скінченні елементи (базисні функції)

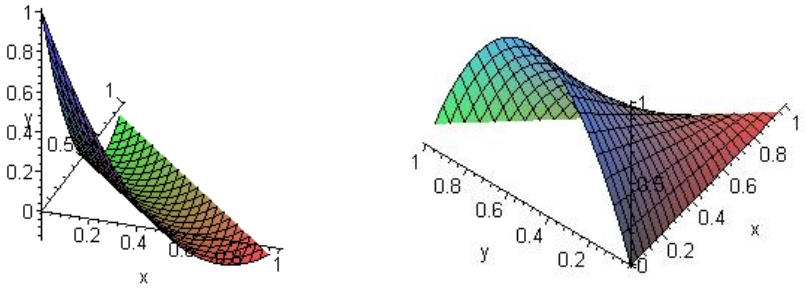


Рис. 5.7. Квадратичні трикутні скінченні елементи (базисні функції),

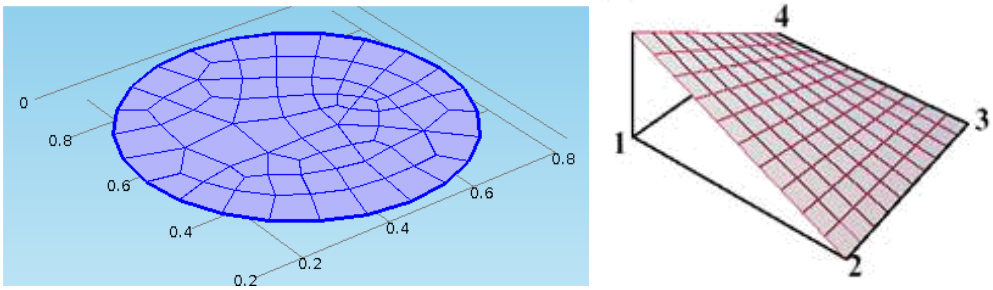


Рис. 5.8. Білінійні чотирикутні скінченні елементи (базисні функції).

ЗАВДАННЯ: провести засобами доступних систем комп'ютерної математики розрахунок деформації балки за методом скінченних елементів. Сила P = 1 та направлена вверх.

ПРИКЛАД ВИКОНАННЯ.

Система **Maxima**:

```
/* Оновлення середовища Maxima */
restart;

/* Визначення кількості елементів та довжини */
N: 10;
Leng: 10.0;
dx: Leng / N;

/* Ініціалізація та визначення функцій для базисних функцій */
for i from 1 to N do (
```

```
    L[i-2]:= dx * (i-2),
    L[i-1]:= dx * (i-1),
    L[i]:= dx * i,
    L[i+1]:= dx * (i+1),
    L[i+2]:= dx * (i+2),
    u[i](x):= piecewise([x < L[i-1], 0,
                         x > L[i-1] and x < L[i], (1/dx) * (x - L[i-1]),
                         x > L[i] and x < L[i+1], (1/dx) * (-x + L[i+1]),
                         x > L[i+1], 0]),
    du[i](x):= diff(u[i](x), x),
    ddu[i](x):= diff(u[i](x), x, 2)
);

/* Визначення моментної функції */
Mom: piecewise([x <= Leng / 2, 2 * x / Leng,
                x > Leng / 2, 2 * (Leng - x) / Leng]);

/* Ініціалізація матриць для обчислень */
M: array(1..N-1, 1..N-1);
B: array(1..N-1);

/* Обчислення матриці M і вектора B */
for i from 1 to N-1 do (
    for j from 1 to N-1 do (
        M[i, j]:= integrate(du[i](x) * du[j](x), x, 0, Leng, numeric)
    ),
    B[i]:= integrate(Mom * u[i](x), x, 0, Leng, numeric),
    print(B[i])
);

/* Виведення матриці M */
print(M);

/* Перетворення вектора B в матрицю і обчислення результату */
Bm: convert(B, matrix);
U: convert(evalm(inverse(M) . B), list);

/* Формування та побудова графіку */
Us: sum(U[ic] * u[ic](x), ic:1..N-1);
plot2d([Us, (U[ic] * u[ic](x))$ic:1..N-1], [x, 0..Leng]);
```

Система **Maple 10**:

```
restart;
N:=10;
Leng:=10.;
dx:=Leng/N;
for i from 1 to N do
L[i-2]:=dx*(i-2):
L[i-1]:=dx*(i-1):
L[i]:=dx*i:
L[i+1]:=dx*(i+1):
L[i+2]:=dx*(i+2):
u[i](x):=piecewise(x<L[i-1], 0, x>L[i-1] and x<L[i], (1/dx)*(x-L[i-1]),
x>L[i] and x<L[i+1], (1/dx)*(-x+L[i+1]), x>L[i+1], 0):
du[i](x):=diff(u[i](x),x):
ddu[i](x):=diff(u[i](x),x$2):
od:
Mom:=piecewise( x<=Leng/2, 2*x/Leng, x>Leng/2, 2*(Leng-x)/Leng);
M:=array(1..N-1, 1..N-1);
B:=array(1..N-1);
for i from 1 to N-1 do
```

```
for j from 1 to N-1 do
M[i,j]:=int(du[i](x)*du[j](x), x=0..Leng, numeric);
od:
B[i]:=int(Mom*u[i](x),x=0..Leng, numeric):
print(B[i]);
od;
print(M);
```

$$\begin{bmatrix} 2. & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -1. & 2. & -1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 2. & -1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & -1. & 2. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & -1. & 2. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & -1. & 2. & -1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & -1. & 2. & -1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & -1. & 2. & -1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -1. & 2. \end{bmatrix}$$

```
Bm:=convert(B, matrix);
```

$$Bm := \begin{bmatrix} 0.20000000 \\ 0.40000000 \\ 0.60000000 \\ 0.80000000 \\ 0.93333333 \\ 0.80000000 \\ 0.60000000 \\ 0.40000000 \\ 0.20000000 \end{bmatrix}$$

```
U:=convert(evalm(Minv&*B), list);
Us:=sum(U[ic]*u[ic](x), ic=1..N-1):
plot([Us, (U[ic]*u[ic](x))$ic=1..N-1],x=0..Leng);
```

Контрольні питання.

1. У чому полягає основна ідея методу скінчених елементів?
2. Яке диференціальне рівняння описує вигин балки?
3. Що таке матриця жорсткості у методі скінчених елементів?
4. Які переваги має МСЕ в порівнянні з аналітичними методами для задач механіки?
5. Як впливає кількість елементів (розбиття сітки) на точність отриманого розв’язку?

Лабораторна робота №6

Тема: Метод скінченних елементів в сучасних CAE-системах для задач в автомобільній галузі

6.1. Задачі лінійної пружності на прикладі розрахунку рами вантажного автомобіля

Розрахунки за допомогою методу скінченних елементів (МСЕ) дозволяють моделювати різні фізичні явища, такі як розподіл напруг, температури, потоки рідин та електромагнітні поля. Проте, найбільше застосування цей метод знайшов у розрахунках *напружено-деформованого стану* (НДС) конструкцій.

Сучасні CAE-системи або надають розширені інструменти для моделювання, або функціонують у тісному зв'язку з CAD-середовищами (наприклад, з AutoCAD), а часто — поєднують обидві можливості. Все більшої популярності набувають інтерфейси типу «direct on CAD», які працюють безпосередньо з геометричними даними, уникаючи необхідності у проміжному перетворенні. Завдяки такому підходу зменшується ризик втрати інформації та скорочується час виконання циклу «проєктування – аналіз – коригування». Також CAD-системи відіграють важливу роль у спрощенні побудови моделей і забезпечують можливість роботи з геометрично складними об'єктами. Сьогодні активно застосовуються гібридні системи моделювання, які поєднують об'ємне, поверхневе та каркасне моделювання з параметричними й об'єктно-орієнтованими методами. Це дозволяє генерувати практично будь-яку геометрію, необхідну для подальшого інженерного аналізу. У більшості систем МСЕ реалізовано функцію імпорту геометрії як через універсальні формати на зразок IGES, так і шляхом прямого підключення до CAD-програм.

Під час етапу *препроцесінгу* для аналізу напружено-деформованого стану (НДС) побудова сітки та розташування вузлів, як правило, виконується безпосередньо в середовищі самої CAE-системи. Водночас деякі платформи передбачають інтеграцію з зовнішніми генераторами сіток і дозволяють імпортувати вже сформовану сіткову структуру. У сучасних системах зміни в геометричній моделі часто автоматично призводять до оновлення сітки, що значно спрощує робочий процес та зменшує ймовірність помилок. Важливо, щоб тип сіткових елементів відповідав просторовій розмірності задачі: для одномірного аналізу — лінійні елементи, для двовимірного — плоскі, а для об'ємного — просторові. Також елементи, що використовуються, повинні бути підтримані обраною системою МСЕ, аби забезпечити коректне формування матриці жорсткості та інших характеристик моделі. Чим більше типів елементів міститься в бібліотеці програми, тим ширший спектр задач вона здатна вирішувати — від простих до складних інженерних сценаріїв.

Тип скінченних елементів обирається залежно від характеру аналізу — зокрема, чи є задача статичною або динамічною, лінійною або нелінійною, а

також від того, які саме характеристики аналізуються: напруження, деформації, температурні поля тощо. Кожному вузлу моделі призначаються ступені свободи — невідомі величини, які можуть включати переміщення, повороти, температуру, теплові потоки або інші параметри залежно від задачі. Наступним кроком є визначення граничних умов. У випадках, коли границі моделі мають неперервну форму, ці умови можуть задавати фіксацію зсувів, прикладення сил або завдання температур. Граничні умови задаються у вузлах, розташованих на границі геометричного об'єкта, і подаються у вигляді конкретних значень відповідних змінних. Щоб врахувати вплив навантаження в конкретних точках, вузли слід розміщувати саме в цих точках, що дозволяє точніше відтворити локальні ефекти. Багато сучасних систем, інтегрованих із CAD, підтримують можливість завдання граничних умов прямо на геометричному представленні, після чого система автоматично трансформує ці умови у вузловий формат. Гнучкість у виборі методів прикладення навантажень і інших граничних умов робить можливим точне моделювання широкого спектра фізичних сценаріїв, що виникають у реальних технічних об'єктах.

На етапі препроцесінгу обов'язковим є призначення матеріальних характеристик для кожного скінченного елемента. Найчастіше до основних параметрів належать модуль пружності (Юнга) та коефіцієнт Пуассона, які визначають механічну (реологічну) поведінку матеріалу. Для таких об'єктів, як оболонки та пластини, товщина зазвичай задається як додаткова матеріальна характеристика, а не геометричний параметр — це дозволяє уникнути необхідності тривимірного аналізу. У задачах з іншою фізичною природою можуть також враховуватись теплоємність**,** теплопровідність, в'язкість або інші параметри. Можливість задавати різні характеристики для різних елементів відкриває шлях до моделювання складених конструкцій або композиційних матеріалів. Однією з найбільших складностей при цьому є точне описання контактів і границь між різнорідними матеріалами, де можуть виникати локальні ефекти чи особливі граничні умови.

Після завершення побудови моделі всі введені параметри передаються в модуль чисельного аналізу, який виконує розрахунки. Отримані результати готуються до візуалізації й аналізу в *постпроцесорі* — програмному засобі для інтерпретації числових даних. Сучасні CAE-системи дозволяють виводити обчислені результати у вигляді таблиць, графіків, а також кольорових карт для візуалізації розподілу деформацій, напружень та переміщень.

Найпоширенішим способом візуального подання результатів є контурні графіки (трасування), де різні кольори позначають значення фізичних величин на поверхні чи в об'ємі тіла. Крім цього, програми підтримують побудову ізоповерхонь — поверхонь з однаковими значеннями заданого параметра, а також поперечних перерізів моделі. Для задач динаміки та часово-залежного аналізу доступні анімаційні інструменти, які дозволяють спостерігати зміну стану системи в часі, зокрема для нелінійних процесів.

Також більшість сучасних програмних продуктів передбачає експорт графіків, зображень та відео у поширені формати, зручні для подальшого використання в звітах, презентаціях або на вебресурсах.

ЗАВДАННЯ: провести засобами доступних скінченноелементних CAE систем розрахунок деформації рами вантажного автомобіля за методом скінченних елементів. Використовувати плоский оболонковий скінченний елемент типу shell, товщиною 5 мм. Вхідні дані задаються індивідуально. Для виконання роботи можна рекомендувати вільновживаний (обмеженння за кількістю скінченних елементів) програмний продукт ANSYS Student (https://www.ansys.com/academic/students).

ПРИКЛАД ВИКОНАННЯ.

CAD-модель рами вантажного автомобіля має вигляд, що казаний на рис. 6.1.

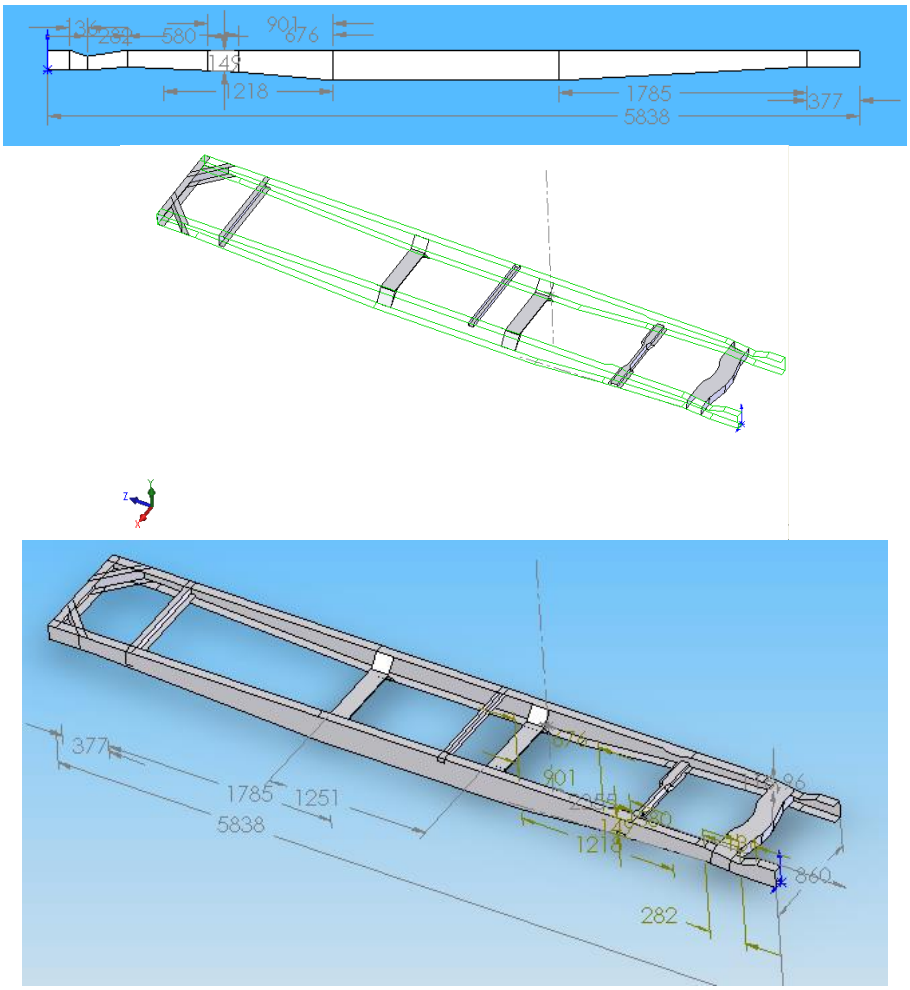
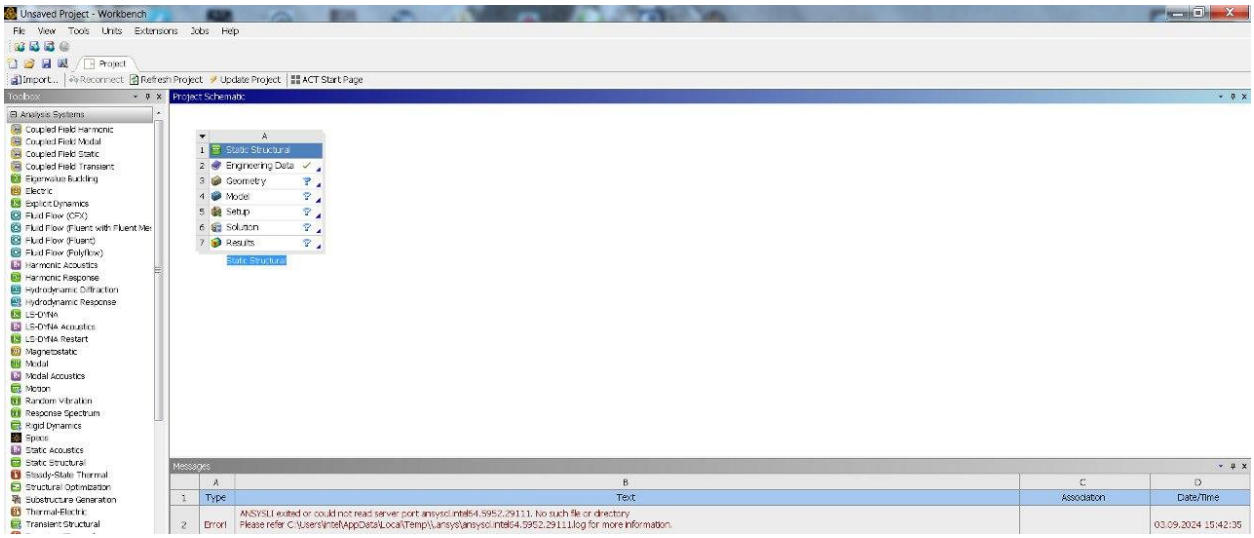
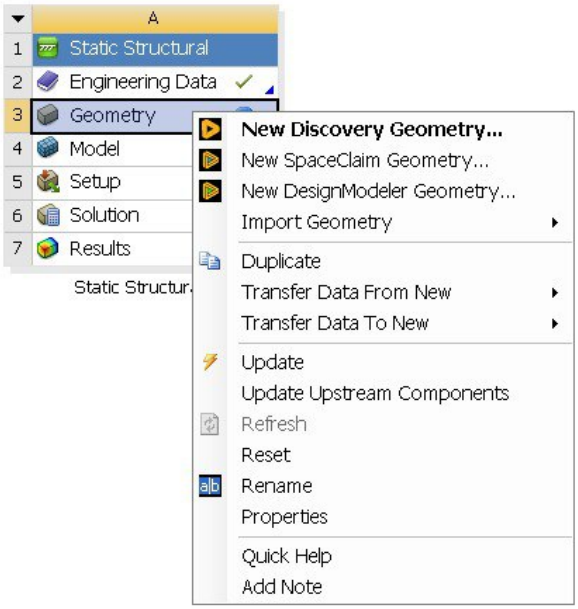


Рис. 6.1. Поперечини рами що з’єднані з лонжеронами

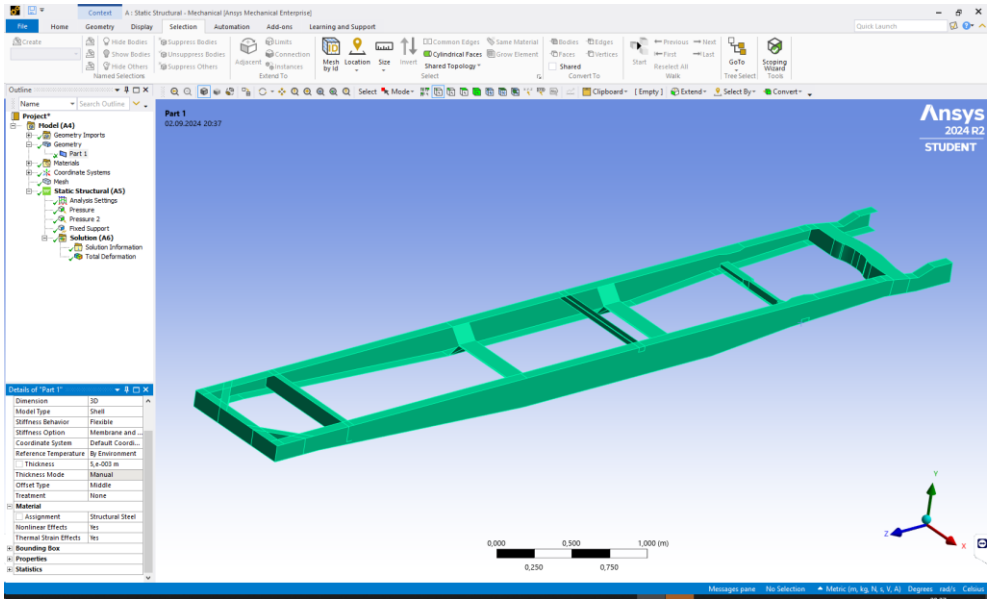
Розрахунок проводиться засобами системи ANSYS Workbanch. В системі ANSYS Workbanch в полі проєкту активується послідовність аналізу напружено-деформованого стану простим перетягуванням з Toolbox:



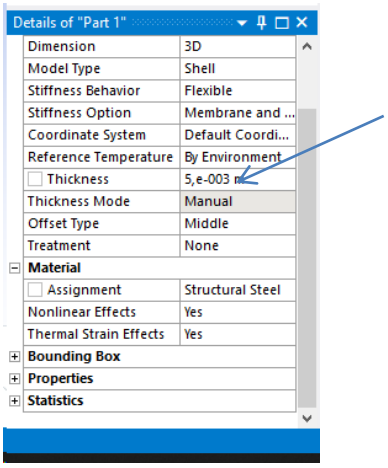
Імпортуємо в модуль геометрії CAD-модель в нейтральному форматі (правий клік на відповідному ярлику):



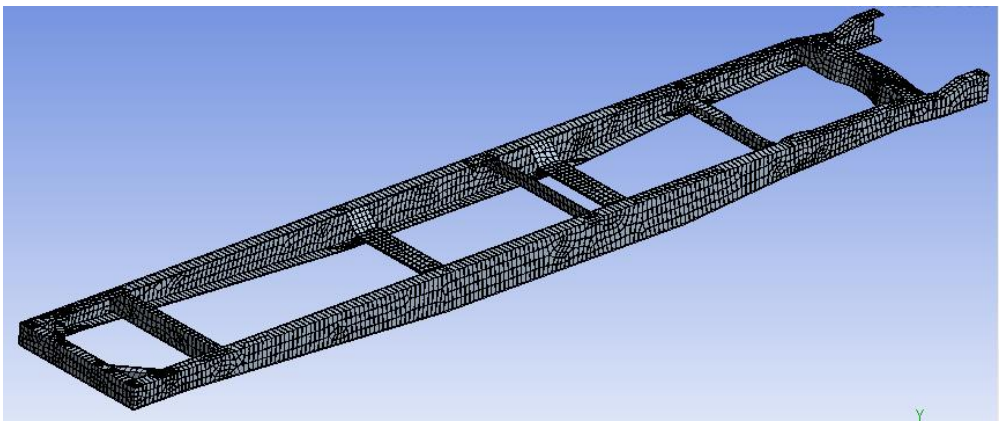
Після цього запускається подвійним кліком налаштування моделі Model:



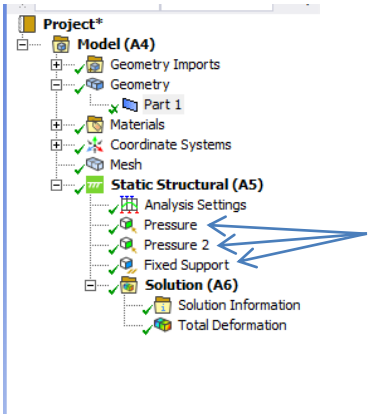
В позиції Geometry вказується товщина скінченного елементу Shell:



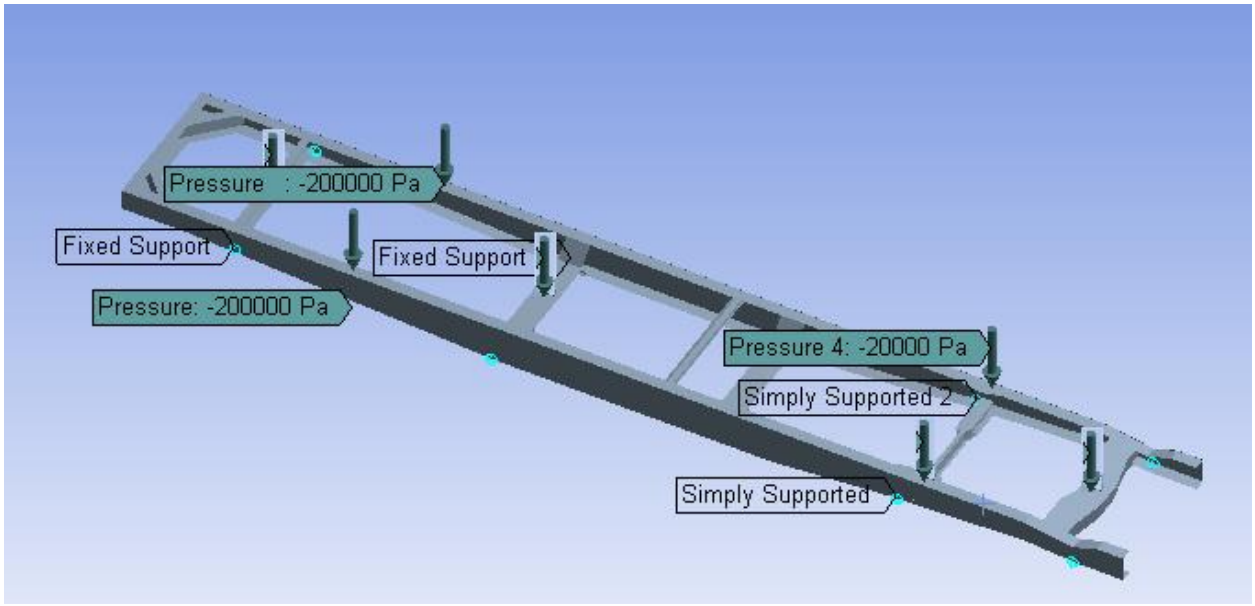
Проводиться розбиття на скінченні елементи в позиції дерева проєкту Mesh – Generate:



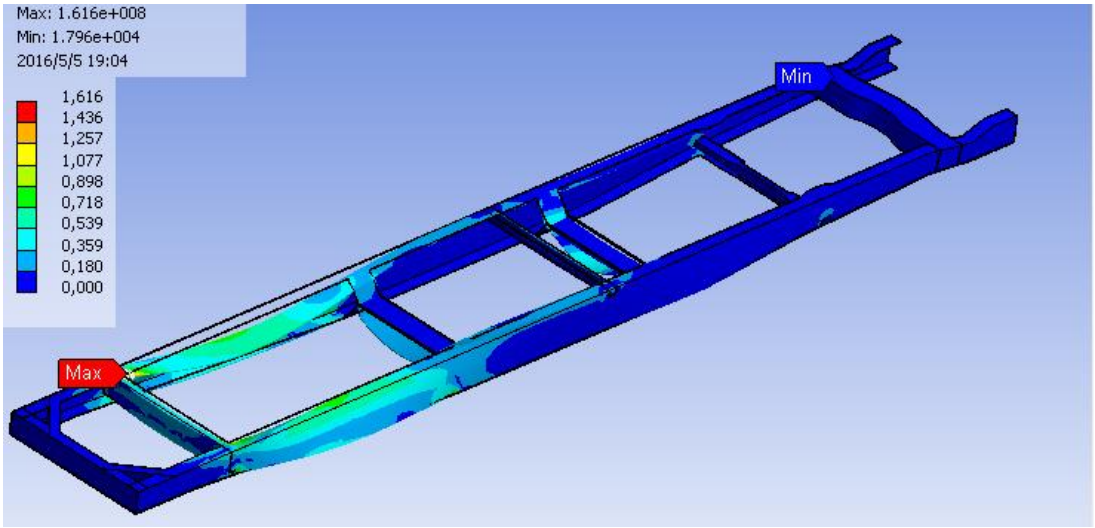
В дереві проєкту в позиції Static Structural задаються відповідні граничні умови



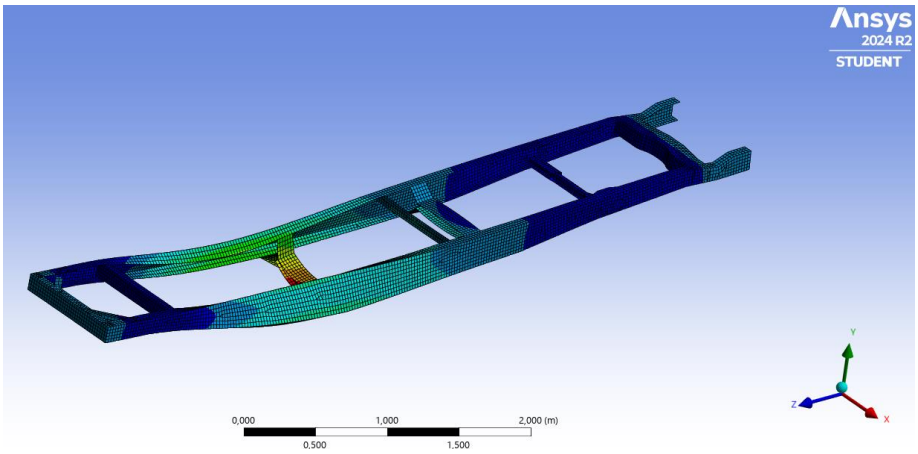
як вказано на рисунку:



Проводиться генерування розрахунку в позиції Solution – Generate. Виводяться (Solution – Insert) результати:



– напруги за Мізесом,



та загальні деформації.

Максимальні внутрішні напруги дорівнюють σ_{max}=162 МПа. Відповідні реакції опор становлять:

$$R_п = 2100 \text{ Н}, R_з = 26000 \text{ Н},$$

що відповідає сумарному навантаженню

$$\Sigma = 2 R_п + 2 R_з = 56200 \text{ Н} = 5620 \text{ кГ}.$$

Дане навантаження відрізняється від заданого на

$$1 – 620/(4000+1700) = 1{,}4\%,$$

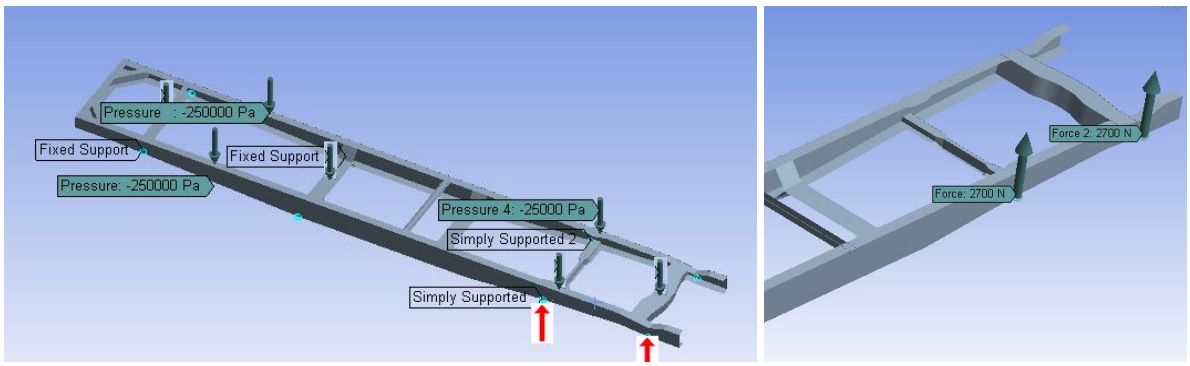
що говорить про досить хороше сіткове наближення та розрахунок тисків.

Максимальні прогини становлять f_{max}=3,9 мм, що відповідає рекомендаціям статичного навантаження.

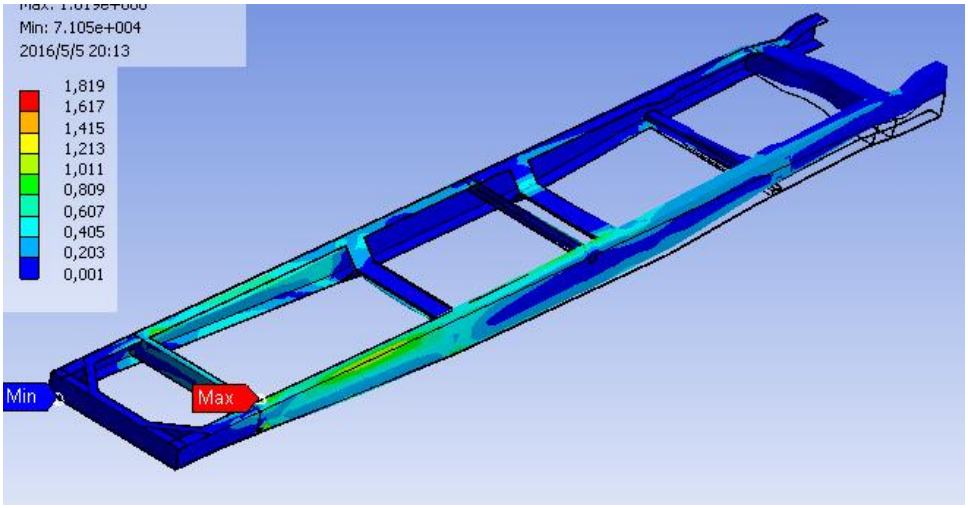
Динамічні навантаження при наїзді на перешкоду враховується коефіцієнтом динамічності $k_д$, який дорівнює $k_д$ = 2,5. Зробимо розрахунок наїзду правим переднім колесом на перешкоду. Динамічна навантаження при цьому випадку перевершить статичну реакцію в $k_д$ раз, і, відповідно вийде:

$$R_{пу} = k_д R_п = 2{,}5 \cdot 2100 \approx 5200 \text{ Н},$$

Така схема навантаження дасть можливість також розрахувати міцність рами при крутильному несиметричному навантаженні:

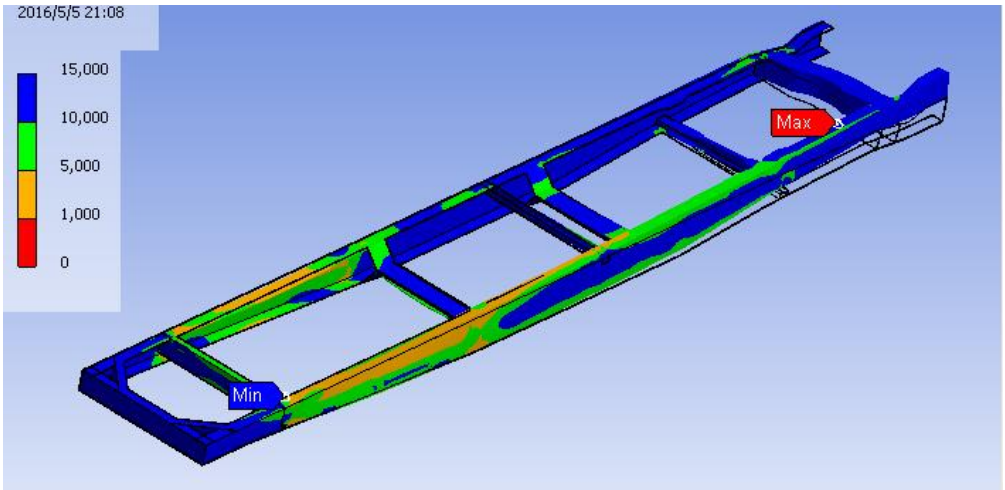


Отримані напруги та деформована форма рами мають вигляд:



Максимальні внутрішні напруги дорівнюють σ_{max} = 182 МПа. Максимальні прогини складають f_{max} = 19,6 мм

У всіх випадках міцність рами виконується, що можна проаналізувати за коефіцієнтом запасу міцності — Safety Factor:



Мінімальний коефіцієнт запасу міцності складає 1,374.

6.2. Контактні задачі на прикладі розрахунку пружної муфти

Контактні задачі є важливою складовою чисельного моделювання в системах автоматизованого інженерного проєктування (CAE). Вони стосуються ситуацій, коли різні частини або компоненти конструкції контактують або взаємодіють один з одним. В таких задачах необхідно точно врахувати, як сили передаються між частинами конструкції і як вони впливають на загальну поведінку системи. Контактні задачі є критично важливими для аналізу, проєктування і оптимізації структурних систем, таких як автомобільні компоненти, будівельні конструкції, машинобудівні елементи та багато іншого.

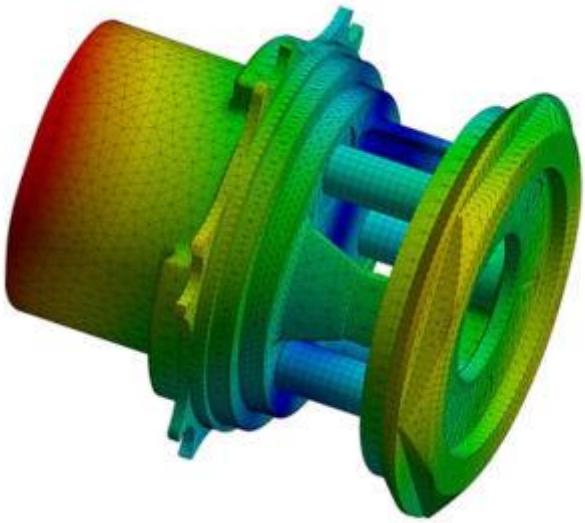


Рис. 6.2. Скінченно-елементна модель контактної задачі

Контактні задачі в CAE системах зазвичай включають кілька основних аспектів, що перераховані нижче.

1. Типи контактів:
    - лінійний контакт — два елементи торкаються вздовж лінії;
    - поверхневий контакт — контакт між двома поверхнями;
    - точка контакту — контакт у конкретній точці.
2. Моделі контакту:
    - тверді контакти — моделюють ситуації, коли поверхні не проникають одна в одну і взаємодіють без пружних або пластичних деформацій;
    - теплові контакти — враховують теплообмін між контактуючими поверхнями;
    - механічні контакти — моделюють механічні взаємодії, включаючи тертя, зсув та ін.
3. Методи вирішення:
    - метод згладжування — застосовується для уникнення великих деформацій і спрощення розрахунків;

- метод контактних поверхонь — моделює контакт як зв’язок між визначеними поверхнями;
- метод тертя — включає моделювання тертя і можливого припливу матеріалу у зонах контакту.

Процес моделювання контактних задач в CAE системах зазвичай включає кілька основних етапів:

1. створення геометрії: формування детальної геометричної моделі конструкції, що дозволяє чітко визначити області контакту;
2. визначення контактних пар: ідентифікація пар поверхонь або частин, які вступають у контакт. Це включає налаштування типів контактів і визначення граничних умов;
3. завдання властивостей контактів: введення характеристик контактних взаємодій, таких як коефіцієнти тертя, модулі пружності, температура і т.д.;
4. створення сітки: розбиття геометрії на скінченні елементи, з урахуванням особливостей контактних зон;
5. налаштування розрахунків: вибір відповідного чисельного методу і налаштування параметрів розрахунку для моделювання контактних взаємодій;
6. аналіз результатів: інтерпретація отриманих даних, таких як розподіл напружень, деформацій і тертя в контактних зонах.

Моделювання контактних задач може бути досить складним і вимагати уважності та досвіду від проєктувальника у зв’язку з наступними можливими проблемами:

- контактні задачі часто супроводжуються чисельними нестабільностями, такими як перепади напружень і деформацій. Це може бути вирішено шляхом вдосконалення сітки або правильного налаштування чисельних методів;
- проблеми з точністю — висока точність моделювання контактів може бути досягнута за рахунок детального розбиття сітки в областях контакту та використання більш складних моделей тертя;
- моделювання контактних задач може вимагати значних обчислювальних ресурсів, особливо для великих і складних моделей. Це може бути вирішено шляхом оптимізації моделі або використанням потужних обчислювальних систем.

ЗАВДАННЯ: провести засобами доступних скінченноелементних CAE систем розрахунок напружено-деформованого стану муфти типу «Renk». Вхідні дані задаються індивідуально. Для виконання роботи можна рекомендувати вільновживаний (обмеженння за кількістю скінченних елементів) програмний продукт ANSYS Student (https://www.ansys.com/academic/students). Пружний елемент муфти є пакетом вставлених одна в одну n гільзових пружин із витком прямокутного

поперечного перерізу. Розрахунки провести для контактної задачі. Розрахункова модель — складання сегмента, що є 1/12 частиною муфти.

ПРИКЛАД ВИКОНАННЯ.

Схема муфти муфти з пакетами гнутих гільзових пружин (виконання для сполучення маховика дизеля з колінчастим валом) вказана на рис. 6.3.

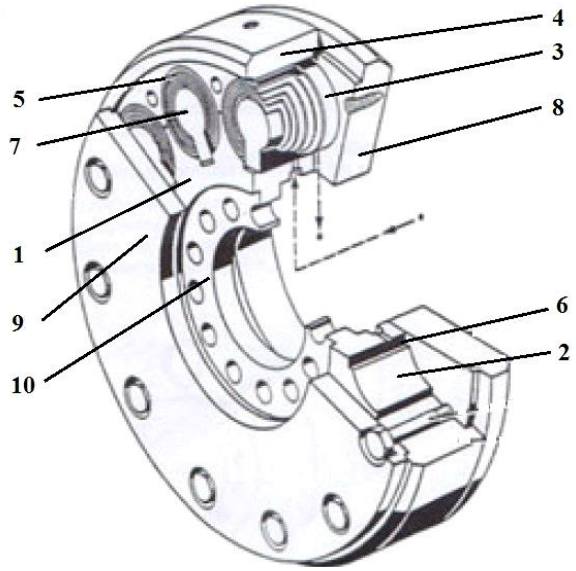
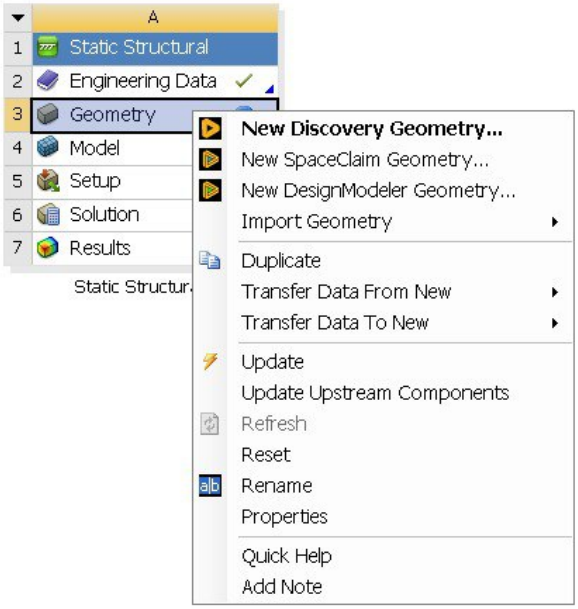


Рис. 6.3. Муфта типу «Renk»

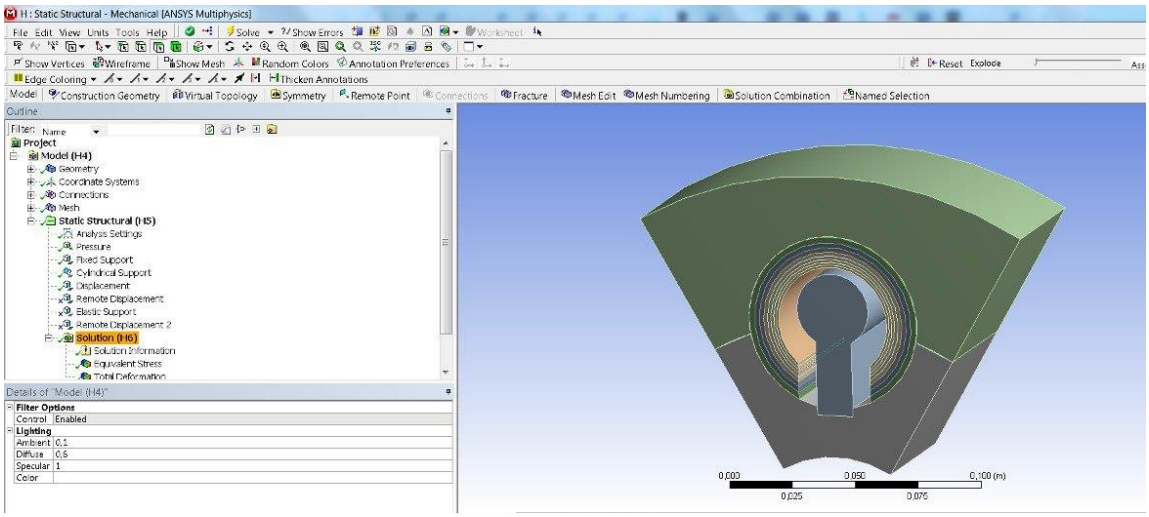
Муфта типу «Renk» містить внутрішню півмуфту (зірку) 1 з відфрезерованими на зовнішній поверхні западинами 2 півколового профілю (їх кількість *z* відповідіє кількості пружних елементів 3 у вигляді пакетів гільзових пружин), встановлену співвісно зірці 1 зовнішню півмуфту 4 (обойму), де півколові западини 5 виконані на внутрішній поверхні з тим же кутовим кроком. У складеній муфті западини зірки та обойми утворюють отвори в які встановлено пакети пружин 3. У западинах 2 внутрішньої півмуфти 1 виконані також пази 6 прямокутного профілю, в яких закріплено напрямні 7, що мають голівки круглого перерізу для установки на них пружних елементів та хвостовики для закріплення в пазах 6. У дизелях обойма 4 муфти кріпиться до маховика 8, і закривається з іншого боку кришкою 9, а фланець 10 зірки 1 закріплюється на колінчастому валу дизеля.

Розрахунок проводиться засобами системи ANSYS Workbanch. Як і в попередній роботі в системі ANSYS Workbanch в полі проєкту активується послідовність статичного аналізу напружено-деформованого стану Static Structural простим перетягуванням з Toolbox.

Імпортуємо в модуль геометрії CAD-модель в нейтральному форматі (правий клік на відповідному ярлику):

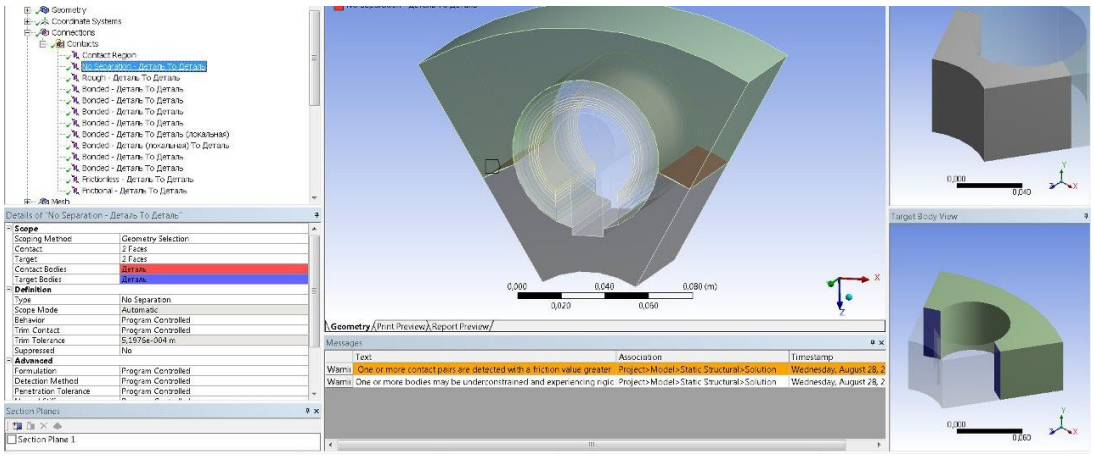


Після цього запускається подвійним кліком налаштування моделі Model:

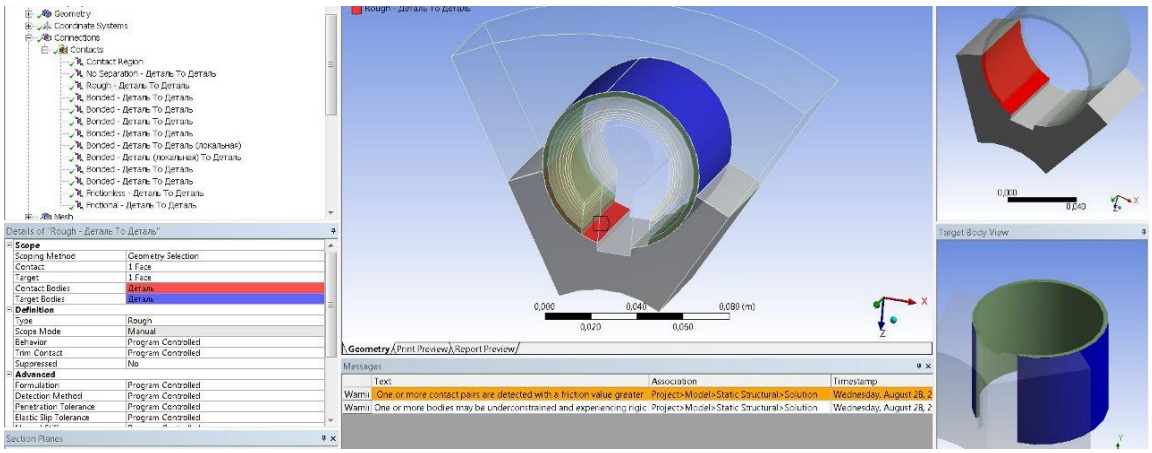


ANSYS автоматично знаходить вірогідні контактні пари заносячи їх в дерево проєкту в позиції Connections, але нам необхідно провести їх переналаштування, та розбити декілька пар на додаткові. Підсвічуючи кожну контактну пару, ми можемо бачити, які поверхні до неї входять.

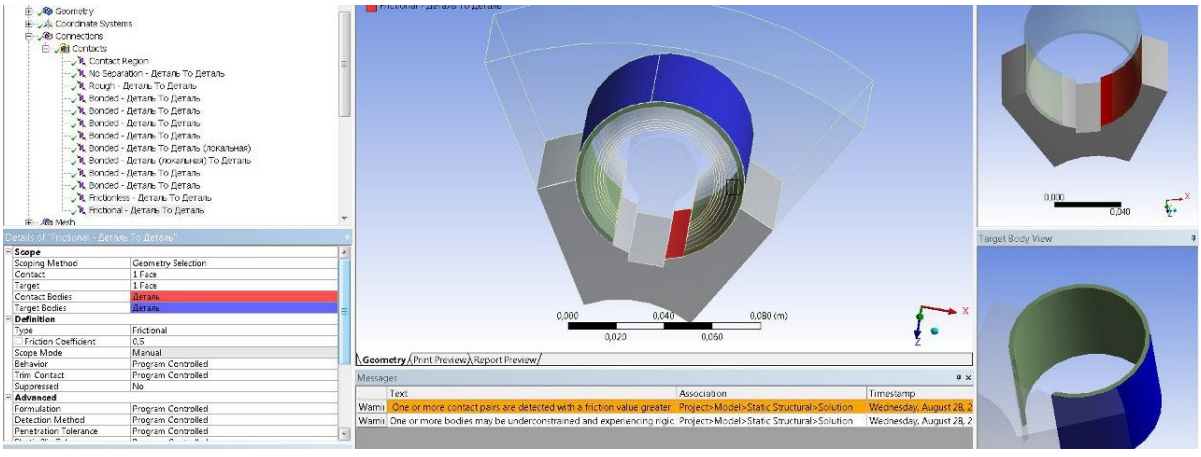
По поверхням контакту між полу муфтами задаємо умову No Separation (проходить проковзування без розділення поверхонь):



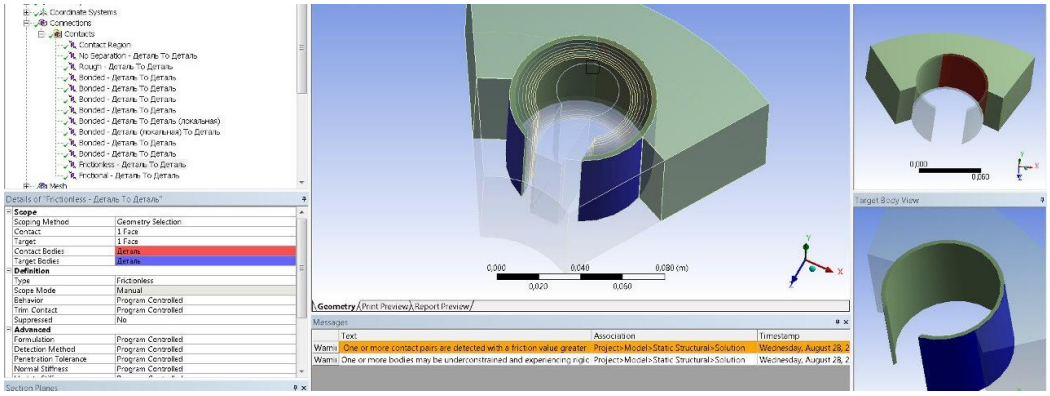
На поверхнях контакту лівої частини внутрішньої полумуфти з зовнішньою поверхнею пружин задається умова Rough (тобто вважаємо наближено, що поверхні можуть відриватися, але не проковзувати), при цьому параметр Interface treatment ставимо як Adjust to Touch (автоматичне прибирання зазорів в початковій геометрії):



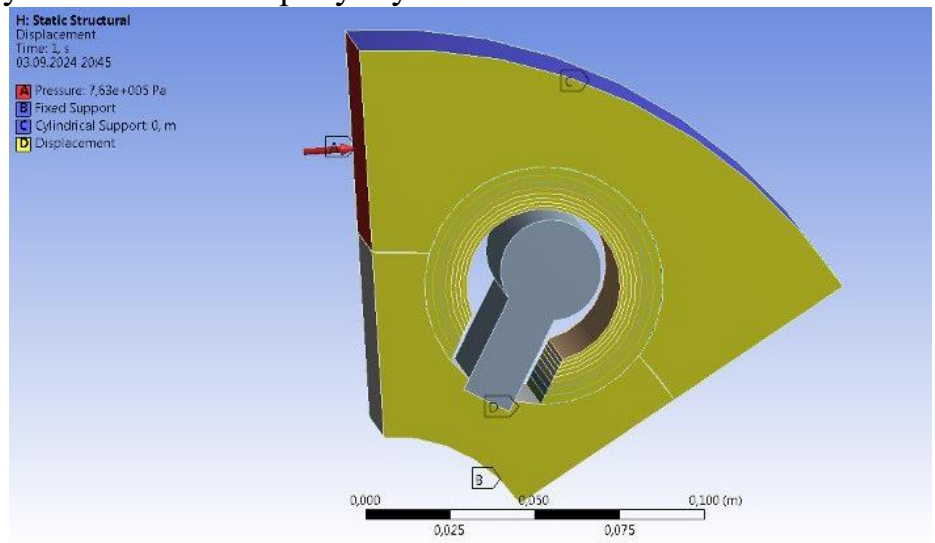
На поверхнях контакту правої частини внутрішньої полумуфти з зовнішньою поверхнею пружин задається умова Frictional (тобто вважаємо, що поверхні можуть як відриватися, так і проковзувати з коефіцієнтом тертя 0,5). Аналогічно параметр Interface treatment ставимо як Adjust to Touch:



На поверхнях контакту правої частини зовнішньої полумуфти з зовнішньою поверхнею пружин задається умова Frictionless (тобто вважаємо, що поверхні можуть як відриватися, так і вільно проковзувати). Аналогічно параметр Interface treatment ставимо як Adjust to Touch:

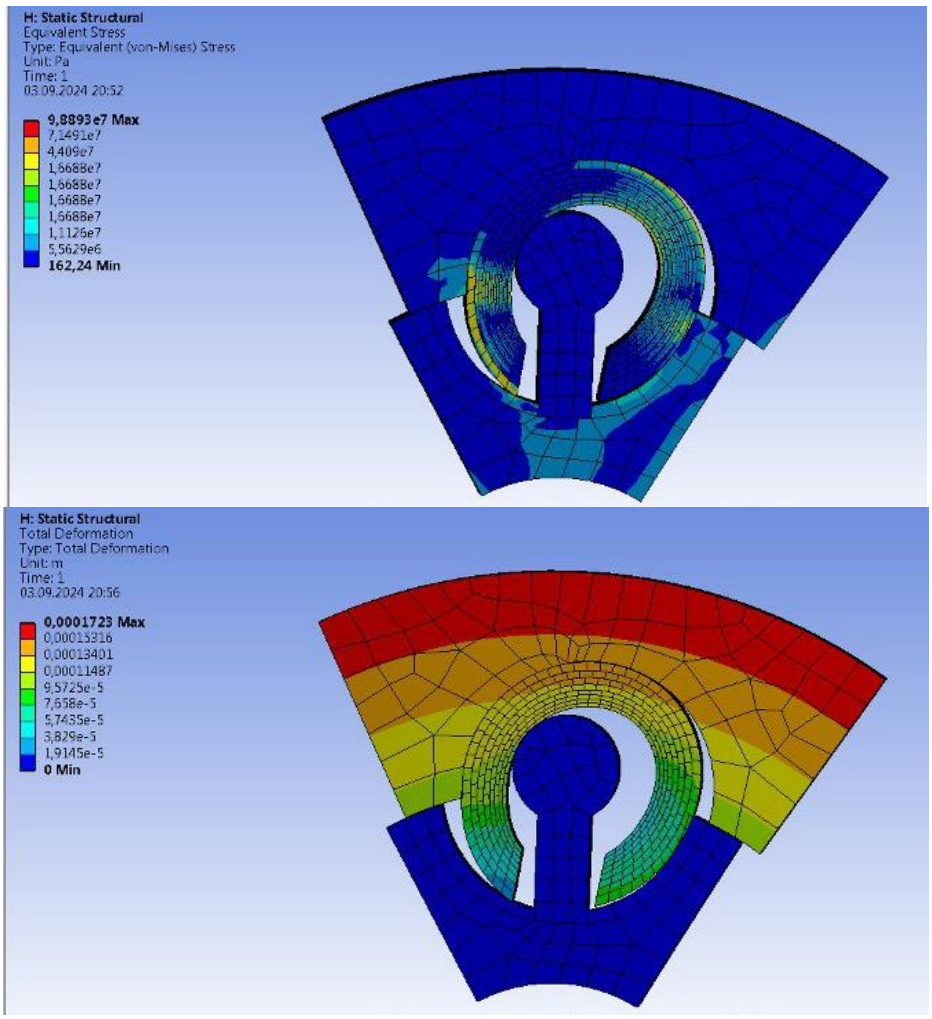


Граничні умови вказані на рисунку нижче:



A) Тиск $7{,}63 \cdot 10^5$;
B) Fixed Support;
C) Cylindrical Support — занулення радіальних переміщень;
D) Displacement — занулення поперечних переміщень.

Після розбиття на скінчені елементи проводимо рішення з візуалізацією напруг за Мізесом та значень деформації:



Контрольні питання.

1. Які етапи включає типове моделювання конструкцій в CAE-системі?
2. Які типи скінченних елементів найчастіше використовуються для моделювання рами (балкові, оболонкові, об’ємні)?
3. Які фізичні властивості матеріалу необхідно задати при моделюванні задачі пружності?
4. Чим відрізняється лінійний та нелінійний аналіз в задачах на міцність?
5. Наведіть приклади вузлів автомобіля, де потрібно вирішувати контактну задачу.

Література

1. Бугаєць Н.О. Математика з програмою Maxima: навчальний посібник. — Ніжин: НДУ ім. М. Гоголя, 2023. — 163 с.
2. Бугаєць Н.О. Засоби програми Maxima для створення графічних зображень та математичних досліджень // Науковий часопис НПУ ім. М.П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання. К.: НПУ ім. М.П. Драгоманова. 2015. №15(22). С. 105 – 114.
3. Жалдак М.І. Математика з комп'ютером / Жалдак М.І., Горошко Ю.В., Вінниченко Є.Ф. К.: Видавництво НПУ імені М.П.Драгоманова, 2015. – 315 с.
4. Kunwoo Lee. Principles of CAD/CAM/CAE. — Pearson; 1st edition (January 20, 1999). — 560 p.
5. ДСТУ 2226-93. Автоматизовані системи. Терміни та визначення. – [Чинний від 1994-07-01]. Вид. офіц. Київ: Держстандарт України, 1994. — 93 с.
6. Pierre Germain-Lacour. Mathematiques et CAO. — Universite de Metz: Hermes Publishing, 1986-1987. — 264 p.
7. Баганов Є.О. Методи розрахунків на ЕОМ. Навчальний посібник. — Херсон: Олді-плюс, 2009. — 288 с.
8. Рівняння математичної фізики. Узагальнені розв'язки крайових задач: Навч. посіб. для студ. техн. спец. вищ. закл. освіти / Ю. К. Рудавський, П. П. Костробій, М. А. Сухорольський, І. М. Зашкільняк, В. М. Колісник; Нац. ун-т «Львів. політехніка». — Л., 2002. — 236 с.
9. Волков В. П., Вільський Г. Б. Теорія руху автомобіля: підручник. Суми : Університетська книга, 2010. 320 с.
10. Луняка К. В., Русанов С. А., Клюєв О. І., Клюєва О. О. Використання методу факторного експерименту при дослідженні умов роботи теплоакумулятора в системі передпускової підготовки автомобільного двигуна. // Стан, досягнення і перспективи холодильної техніки і технології : зб. тез доп. Всеукр. наук.-техн. онлайн-конф. молодих вчених та здобувачів вищої освіти, Одеса, 19-20 квіт. 2022 р. / Одес. нац. технол. ун-т, Навч.-наук. ін-т холоду, кріотехнологій та екоенергетики ім. В. С. Мартиновського ; під ред. М. Г. Хмельнюка. — Одеса, 2022. — С. 39–41.
11. Pedro X. La Hera. Active suspension control for a heavy-duty vehicle with hydraulic cylinders at the wheels // Swedish cluster of forest technology / Swedish University of Agricultural Sciences. Technical Report. 2012. – 17 p.
12. Protsenko, V., Rusanov, S., Kliuev, O., Voitovych, O., Protasov, R. Improvement of the internal combustion engine sleeve spring packets equipped coupling calculation model // Strojnicky Casopis, 2024, 74(3), p. 89–96.
13. Edgar J. A Century of Car Aerodynamics: the Science and Art of Cars and Airflow. Amazon Digital Services LLC, 2021. 218 p.
14. Piechna J., Kurec K., Broniszewski J., Remer M., Piechna A., Kamieniecki K., Bibik P. Influence of the car movable aerodynamic elements on fast road car cornering. Energies. 2022. Vol. 15, No. 3. P. 689.

ДОДАТОК 1

Короткий огляд систем Maxima, Maple та ANSYS

В циклі лабораторних робіт наводяться приклади їх виконання у системах Maxima, Maple (лабораторні роботи №1-5) та ANSYS (лабораторна робота №6). Інтеграція таких різноманітних програмних інструментів дозволяє значно підвищити ефективність розробки, оптимізації та верифікації технічних рішень. Одним із ключових напрямів такої інтеграції є поєднання систем комп'ютерної математики (СКМ), інша назва – системи комп'ютерної алгебри (СКА) та систем автоматизованого проєктування (САПР) типу CAD/CAE/CAM.

СКМ, такі як *Matlab, Maple, Mathematica, Mathcad, Maxima* та інші, забезпечують широкі можливості для аналітичних і чисельних розрахунків, моделювання складних процесів, обробки експериментальних даних. У той час САПР-системи, в першу чергу CAD, CAE, CAM — системи, призначені для створення креслень, тривимірного моделювання об'єктів, моделювання поведінки об'єктів та програмування технологічних процесів виготовлення деталей і підготовки проєктної документації (PDM). Інтеграція цих двох класів програмного забезпечення дозволяє не лише автоматизувати окремі етапи проєктування, а й побудувати єдиний безперервний цикл «розрахунок – моделювання – аналіз – корекція». Синтаксис та принципи роботи незначно відрізняються в системах Maxima, Maple та Mathematica, тому перехід за необхідністю одна до одної не буде значною проблемою.

Maxima — це одна з найвідоміших систем комп'ютерної математики, історія розробки якої почалася у Масачусетському технологічному інституті (MTI) в кінці 60-х років минулого століття у вигляді проєкту MACSYMA (Проєкт Mac's SYmbolic MAnipulation System), що виконувався групою Mathlab у лабораторії комп'ютерних наук MTI (спочатку відомої як Project MAC). Ліцензування програмних кодів Macsyma призвело до створення інших прикладних математичних пакетів — Maple фірми Waterloo Maple Inc. та Mathematica фірми Wolfram Research та ін. Спільність цих програмних продуктів виражається як у схожому синтаксисі, так і в спільних алгоритмах.

Новий етап у розвитку Maxima настав у 1999 році, коли минув термін дії патенту, і права на Maxima повернулись до одного з її авторів — Вільяма Шелтера, який виконав повну переробку системи та залучив до її відкритої розробки провідних фахівців. З листопада 2001 року проєкт Maxima підтримується роботою команди на чолі з Джеймсом Амундсоном.

На даний час Maxima – це вільно поширювана відкрита система, що

розповсюджується за ліцензією GPL.

Maple — це одна з найпотужніших сучасних систем комп'ютерної алгебри, яка розроблена у 1980-х роках у Університеті Ватерлоо (Канада) групою дослідників під керівництвом Гастона Гонне. У подальшому розвитком та комерціалізацією системи зайнялася компанія Waterloo Maple Inc. (згодом перейменована на Maplesoft). Метою створення Maple було забезпечення ефективного інструменту для виконання символічних і чисельних обчислень, який би поєднував математичну потужність з високою швидкістю виконання.

Як вже зазначалося, система Maple успадкувала багато ідей від свого попередника — проєкту MACSYMA, і є близькою до інших систем комп'ютерної математики як за синтаксисом, так і за реалізованими алгоритмами.

Maple забезпечує широкий спектр функцій: від елементарних обчислень до розв'язування складних задач математичного аналізу, алгебри, диференціальних рівнянь, теорії графів, статистики, оптимізації тощо. Завдяки зручному графічному інтерфейсу, підтримці інтерактивних документів і можливості вбудовування в інші програмні продукти, Maple знайшла широке застосування в наукових дослідженнях, інженерії та освіті.

На сьогодні Maple є комерційним продуктом, що активно розвивається і підтримується компанією Maplesoft, та використовується у провідних університетах та наукових установах по всьому світу.

ANSYS — це одна з провідних у світі САПР, що призначена для інженерного моделювання та чисельного аналізу, вирішення складних задач механіки суцільних середовищ, теплопередачі, динаміки, електромагнетизму та інших фізичних процесів методом скінченних елементів (МСЕ). Розробка ANSYS розпочалася на початку 1970-х років американським інженером Джоном Свенсоном, який заснував компанію Swanson Analysis Systems у Піттсбурзі (США). Пізніше компанія була перейменована на ANSYS Inc., яка і досі є основним розробником та постачальником цієї системи.

ANSYS став відповіддю на потребу інженерної спільноти у потужному, гнучкому та універсальному інструменті для комп'ютерного моделювання фізичних процесів, які важко або неможливо дослідити експериментально. Основною ідеєю при створенні ANSYS було створення модульної системи, яка дозволяє користувачеві моделювати складні інженерні системи шляхом поєднання різних фізичних полів у єдиній обчислювальній моделі.

З часом ANSYS еволюціонував у комплексну мультифізичну платформу, що поєднує численні модулі для аналізу міцності конструкцій, гідро- та аеродинаміки (CFD), термічного аналізу, електромагнетизму, оптимізації. Значну частину функціональності ANSYS було реалізовано завдяки купівлі провідних спеціалізованих компаній, таких як Fluent Inc. (CFD), CFX, Ansoft (електромагнітні задачі) та інших.

ANSYS є комерційним продуктом, ліцензованим для широкого кола користувачів — від університетів до великих інженерних корпорацій.

Програмне забезпечення активно використовується у машинобудуванні, енергетиці, авіації, автомобілебудуванні, електроніці та інших галузях. Постійна розробка нових модулів і вдосконалення чисельних методів дозволяє ANSYS залишатися однією з найавторитетніших та найпотужніших систем інженерного аналізу у світі. Програмний продукт ANSYS Student є вільновживаним для студентської спільноти, однак має обмеженння за кількістю скінченних елементів.

ДОДАТОК 2

Основи роботи в системі Maxima

Д1. Базові операції у системі Maxima

Система комп'ютерної алгебри Maxima підтримує виконання базових арифметичних операцій над числовими значеннями та змінними. Оператори додавання, віднімання, множення та ділення використовуються у традиційній математичній нотації:

```
2 + 3;
5 - 7;
4 * 6;
8 / 2;
```

Результати обчислень мають вигляд:

$$2 + 3 = 5,\ 5 - 7 = -2,\ 4 \times 6 = 24,\ 8 / 2 = 4$$

Присвоювання значень змінним здійснюється за допомогою оператора ':'. Наприклад:

```
x : 5;
y : 7;
x + y;
```

$$5$$
$$7$$
$$12$$

Д2. Перетворення та спрощення виразів

У Maxima передбачені команди для виконання алгебраїчних перетворень, таких як розкриття дужок, розкладання многочленів на множники та спрощення дробів.

```
expand((x + 2)^3);
```

$$x^3+6\,x^2+12\,x+8$$

Операція розкладання многочлена на множники виконується командою 'factor':

```
factor(x^3 + 6*x^2 + 12*x + 8);
```

$$(x+2)^3$$

Для скорочення дробів та приведення їх до раціонально спрощеного вигляду використовується команда 'ratsimp':

```
ratsimp((x^2 - 4)/(x - 2));
```

$$x+2$$

Для спрощення тригонометричних виразів використовується команда 'trigsimp':

```
trigsimp(sin(x)^2+cos(x)^2);
```

$$1$$

Д3. Розв’язування рівнянь

Одним із ключових завдань комп’ютерної алгебри є розв’язання рівнянь різного типу. У Maxima для цієї мети передбачені функції 'solve' та 'linsolve', які дозволяють знаходити корені алгебраїчних рівнянь, а також розв’язувати системи лінійних рівнянь.

```
solve(x^2 - 5*x + 6 = 0, x);
```

x = 2, x = 3

Для розв’язання системи лінійних рівнянь застосовується наступний синтаксис:

```
solve([x + y = 5, x - y = 1], [x, y]);
```

x = 3, y = 2

У випадку великих систем лінійних рівнянь можна використовувати функцію 'linsolve', яка приймає матрицю коефіцієнтів і вектор вільних членів.

Д4. Диференціювання та інтегрування

Maxima надає можливості для символьного диференціювання функцій за допомогою команди 'diff':

```
diff(x^3 + 2*x, x);
```

$$3x^2+2$$

Інтегрування виконується командою 'integrate':

```
integrate(sin(x), x);
```

$$-\cos(x)$$

Д5. Робота з матрицями

Maxima підтримує створення та опрацювання матриць. Матриця створюється командою 'matrix':

```
A : matrix([1, 2], [3, 4]);
```

Двовимірна матриця A має вигляд:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Операція множення матриць здійснюється за допомогою оператора '.':

```
B : matrix([5, 6], [7, 8]);
C : A . B;
```

Результат:

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Обчислення визначника матриці виконується командою 'determinant':

```
determinant(A);
```

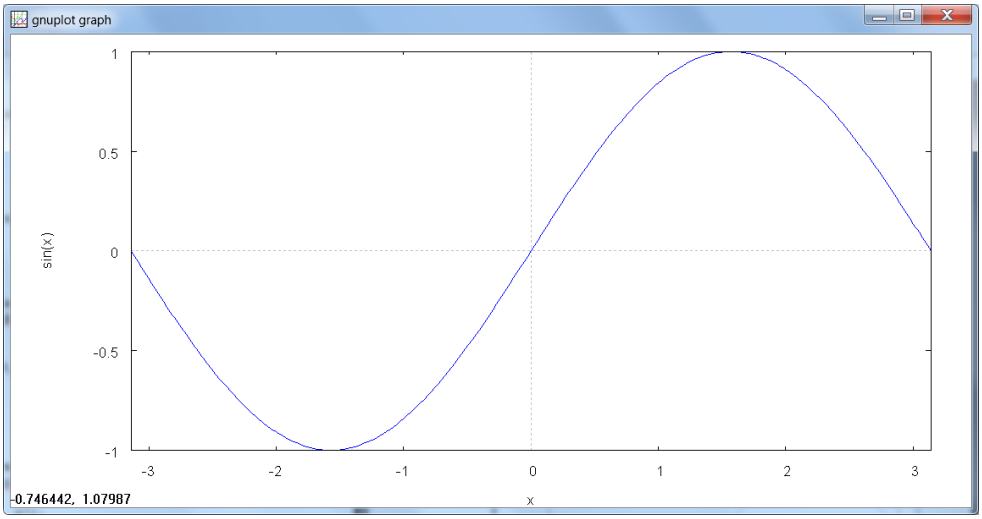
$$-2$$

Д6. Побудова графіків

Maxima забезпечує можливість побудови двовимірних та тривимірних графіків функцій. Для побудови 2D графіка застосовується команда 'plot2d'.

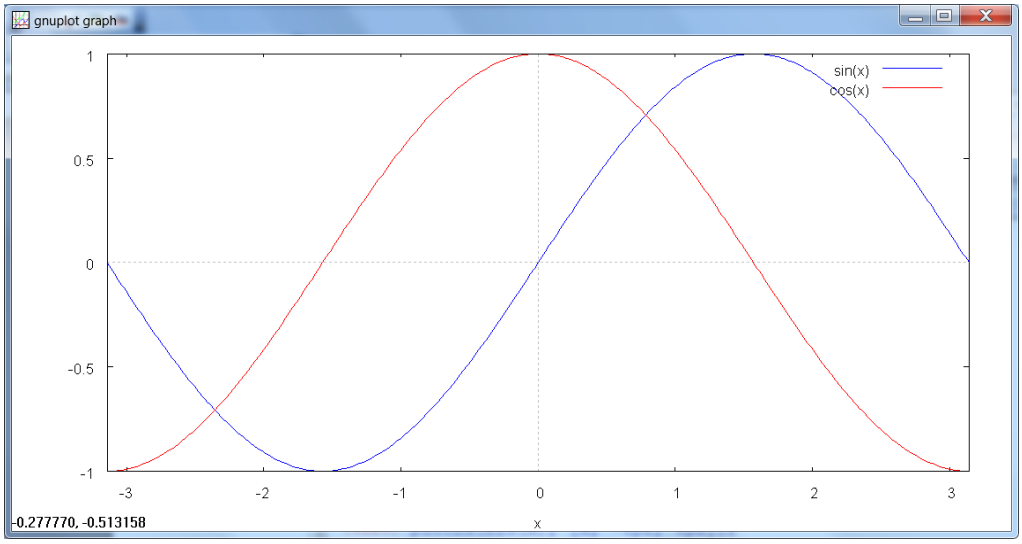
```
plot2d(sin(x), [x, -%pi, %pi]);
```

Ця команда побудує графік функції sin(x) на проміжку від -π до π:



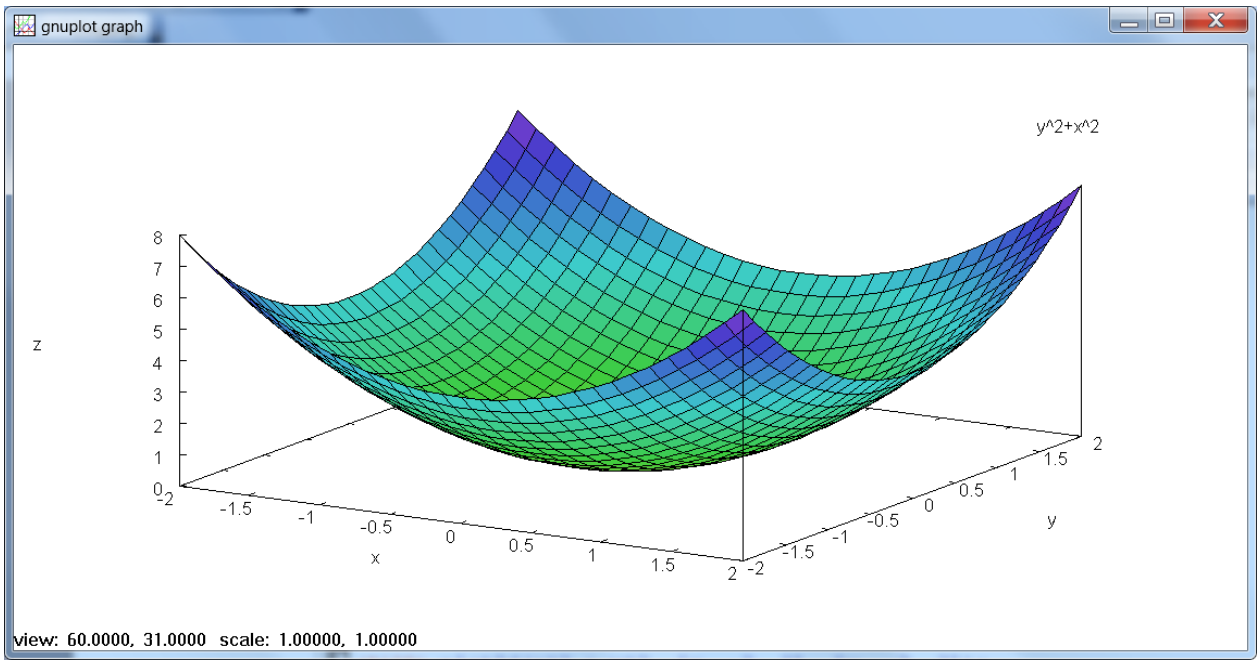
Для побудови графіків декількох функцій використовується список функцій:

```
plot2d([sin(x), cos(x)], [x, -%pi, %pi]);
```



Для побудови тривимірних графіків застосовується команда 'plot3d':

```
plot3d(x^2 + y^2, [x, -2, 2], [y, -2, 2]);
```



Д7. Чисельні методи

У Maxima передбачено функції для наближеного розв’язування рівнянь та чисельного інтегрування.

Для чисельного розв’язання нелінійного рівняння використовується команда 'find_root':

```
find_root(sin(x) = 0, x, 3, 4);
```

Ця команда знаходить корінь функції sin(x) на проміжку [3, 4].

```
3.14159
```

Для чисельного інтегрування використовують функцію 'quad_qags' з пакету 'quadpack':

```
load(quadpack);
quad_qags(sin(x), x, 0, %pi);
```

```
2.0
```

Д8. Елементи програмування в системі Maxima

Maxima підтримує прості конструкції програмування, що дозволяють автоматизувати обчислення, створювати функції та виконувати ітеративні обчислення.

Д9.1 Оголошення функцій

Для визначення функції в Maxima використовується синтаксис:

```
f(x) := x^2 + 3*x + 1;
```

Ця функція *f* приймає аргумент x і повертає значення x^2+3x+1. Виклик функції здійснюється звично:

```
f(2);
```

Результат:

```
11
```

Д9.2 Умовні оператори

Умовні конструкції реалізуються за допомогою if-then-else:

```
if x > 0 then print("+") else print("-");
```

Для складніших перевірок можна вкладати умови:

```
if x > 0 then print("+")
elseif x = 0 then print("0")
else print("-");
```

Д9.3 Цикли

Maxima підтримує цикли for і while.
Приклад циклу for:

```
for i:1 thru 5 do print(i);
```

```
1
2
3
4
5
```

Цикл while виконує блок, доки умова істинна:

```
i : 1;
```

```
while i <= 5 do (
    print(i),
    i : i + 1
);
```

```
1
2
3
4
5
```

Д10. Список основних команд

Нижче наведено перелік основних команд Maxima, які часто використовуються у процесі розв’язання задач.

- Арифметичні операції: `+`, `-`, `*`, `/`, `^`
- Присвоєння змінних: `:`
- Спрощення виразів: `expand()`, `factor()`, `ratsimp()`
- Розв’язання рівнянь: `solve()`, `linsolve()`
- Диференціювання: `diff()`
- Інтегрування: `integrate()`
- Побудова графіків: `plot2d()`, `plot3d()`
- Чисельне інтегрування: `quad_qags()`
- Чисельне розв’язання рівнянь: `find_root()`

ДОДАТОК 3

Основи роботи в системі Maple

Д1. Базові операції у системі Maple

Синтаксис арифметичних операцій в системі комп'ютерної алгебри Maple не відрізняється від такого в системі Maxima.

Присвоювання значень змінним здійснюється за допомогою оператора ':='. Наприклад:

```
x:= 5;
y:= 7;
x+y;
```

$$x := 5$$

$$y := 7$$

$$12$$

Д2. Перетворення та спрощення виразів

У Maple як і в Maxima передбачені команди для виконання алгебраїчних перетворень. Синтаксис команд expand та factor повністю ідентичний до системи Maxima

```
expand((x + 2)^3);
```

$$x^3 + 6\,x^2 + 12\,x + 8$$

```
factor(x^3 + 6*x^2 + 12*x + 8);
```

$$(x + 2)^3$$

Окремо треба виділити потужну команду спрощення виразів 'simplify':

```
simplify(sin(x)^2+cos(x)^2);
```

$$1$$

Д3. Розв'язування рівнянь

В сисемі Maple для розв'язання рівнянь, як і в системі Maxima, передбачені функція 'solve':

```
solve(x^2 - 5*x + 6 = 0, x);
```

$$3, 2$$

Для розв'язання системи лінійних рівнянь застосовується синтаксис також еквівалентний такому в системі Maxima:

```
solve([x + y = 5, x - y = 1], [x, y]);
```

$$[[x = 3, y = 2]]$$

Д4. Диференціювання та інтегрування

Система Maple має аналогічну команду для аналітичного диференціювання команди 'diff':

```
diff(x^3 + 2*x, x);
```

$$3\,x^2 + 2$$

Інтегрування на відміну від Maxima виконується командою 'int':

```
integrate(sin(x), x);
```

$$-\cos(x)$$

Д5. Робота з матрицями

Maple як і Maxima підтримує створення та роботу з матрицями. Матриця створюється командою 'matrix' пакету linalg:

```
with(linalg):

A:=Matrix([[1,2], [3,4]]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Операція множення матриць також здійснюється за допомогою оператора '.':

```
B:= Matrix([[5, 6], [7, 8]]);
C:= A . B;
```

$$B := \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$C := \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Обчислення визначника матриці виконується командою 'det:

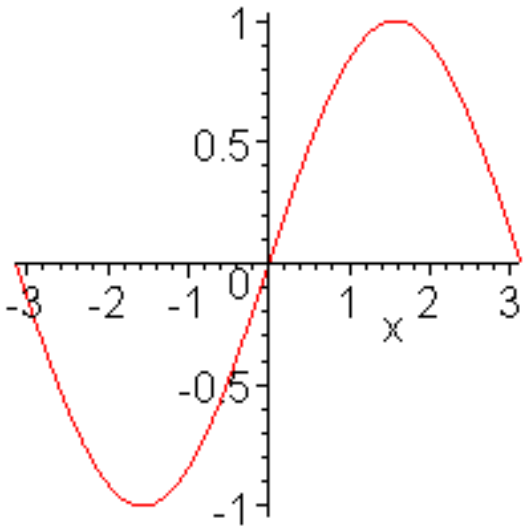
```
det(A);
```

-2

Д6. Побудова графіків

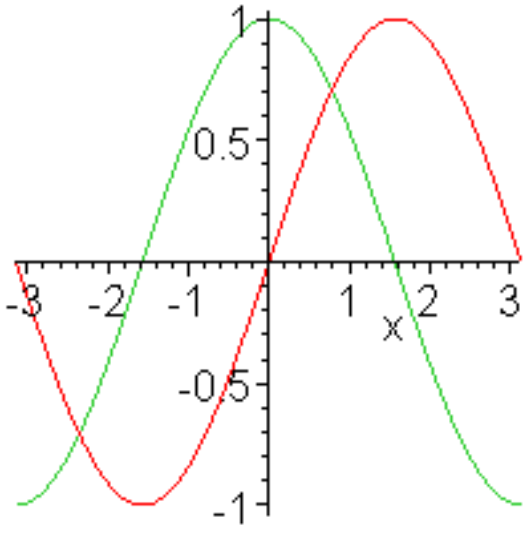
Як і Maxima, Maple забезпечує можливість побудови двовимірних та тривимірних графіків функцій. Для побудови 2D графіка функції sin(x) на проміжку від -π до π застосовується команда 'plot' з наступним синтаксисом:

```
plot(sin(x), x=-Pi..Pi);
```



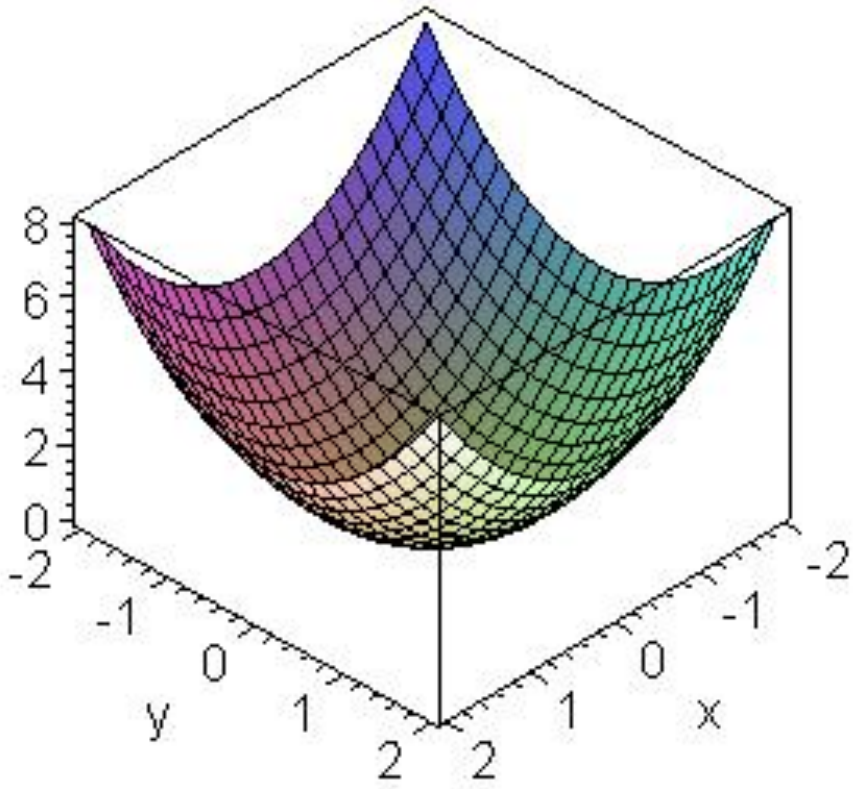
Для побудови графіків декількох функцій також, як і в системі Maxima, використовується список функцій:

```
plot([sin(x), cos(x)], x=-Pi..Pi);
```



Для побудови тривимірних графіків застосовується команда 'plot3d' бібліотеки plots:

```
with(plots):

plot3d(x^2 + y^2, x=-2..2, y=-2..2);
```



Д7. Чисельні методи

Для чисельного розв’язання нелінійного рівняння використовується команда fsolve. Знайдемо корінь функції sin(x) на проміжку [3, 4].

```
fsolve(sin(x) = 0, x=3..4);
```

3.141592654

Для чисельного інтегрування використовують параметр 'numeric':

```
int(sin(x), x=0..Pi, numeric);
```

2.

Д8. Елементи програмування в системі Maple

Д9.1 Оголошення функцій

Для визначення функції в Maple використовується синтаксис:

```
f:=x->x^2+3*x+1;
```

Виклик функції здійснюється аналогічно:

```
f(2);
```

Результат:

11

Д9.2 Умовні оператори

Умовні конструкції в Maple також реалізуються за допомогою if-then-else:

```
if x > 0 then print("+") else print("-");
```

```
if x > 0 then print("+")
   elif x = 0 then print("0")
else print("-");
```

Д9.3 Цикли

Приклад циклу for в системі Maple:

```
for i from 1 to 5 do print(i) od;
```

1

2

3

4

5

Цикл while:

```
i:=1;
while i <= 5 do
    print(i);
    i:=i + 1
od;
```

$$i := 1$$

$$1$$

$$i := 2$$

$$2$$

$$i := 3$$

$$3$$

$$i := 4$$

$$4$$

$$i := 5$$

$$5$$

Д10. Список основних команд

Нижче наведено перелік основних команд Maple, які часто використовуються у процесі розв’язання задач.

- Арифметичні операції: `+`, `-`, `*`, `/`, `^`
- Присвоєння змінних: `:=`
- Спрощення виразів: `expand()`, `factor()`, `simplify()`
- Розв’язання рівнянь: `solve()`
- Диференціювання: `diff()`
- Інтегрування: `int ()`
- Побудова графіків: `plot()`, `plot3d()`
- Чисельне інтегрування: параметр`numeric`
- Чисельне розв’язання рівнянь: `fsolve()`

ЗМІСТ

Навчальне електронне видання

Сергій Русанов
Олег Клюєв
Владислав Проценко

МАТЕМАТИЧНІ ТА КОМП'ЮТЕРНІ ЗАСОБИ РОЗРАХУНКІВ.

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Навчально-методичний посібник

для студентів технічних закладів вищої освіти

ISBN 978-617-8187-56-9 (електронне видання)

Підписано до видання 15.09.2025 р. Формат 60×84/8.
Гарнітура Times.
Ум. друк. арк. 12,52. Обл.-вид. арк. 13,46.
Замовлення №3240.

ISBN 978-617-8187-56-9
9 786178 187569

Книжкове видавництво ФОП Вишемирський В.С.
Свідоцтво про внесення до державного реєстру суб'єктів видавничої справи:
Серія ХС №48 від 14.04.2005 р.
Видано Управлінням у справах преси та інформації
73000, Україна, м. Херсон, вул. Соборна, 2.
Тел. +38(050)133-10-13
e-mail: printvvs@gmail.com