ЛЕКЦІЯ 1
МОДЕЛЮВАННЯ ФІЗИЧНИХ ПРОЦЕСІВ, ОПИСУВАНИХ ДИФЕРЕНЦІАЛЬНИМИ РІВНЯННЯМИ

Вибір з великої кількості спеціалізованих пакетів для математичних обчислень як базового програмного засобу саме системи MatLab обумовлений, зокрема, тим, що:

- в пакет інтегрований потужний математичний апарат, що дозволяє розв'язувати складні задачі та допомагає знаходити розв'язки:
  - лінійних та нелінійних алгебраїчних рівнянь і систем;
  - задачі Коші та крайової задачі;
  - ДР в частинних похідних;
  - задач статистичної оброки даних (обчислення статистичних параметрів, інтерполяція, апроксимація, згладжування тощо);
  - задач лінійної алгебри (операції з матрицями й векторами);
  - задач пошуку екстремумів функціональних залежностей;
- пакет має потужні засоби графічного подання інформації (графіки функцій, поверхонь, карти ліній рівня, векторні поля тощо);
- пакет забезпечений засобами анімації, що дозволяє розглядати тимчасову еволюцію математичних моделей в динаміці тощо;
- в пакет інтегрований математичний апарат, що реалізує символьні обчислення.

1.1 Розв'язання систем звичайних диференціальних рівнянь

Для розв'язання систем звичайних ДР (ЗДР) в системі MatLab існує кілька вбудованих процедур-розв'язувачів (солверів).

Приклад 1.1. Розв'яжемо рівняння $\frac{dy}{dx} = -2xy$ з початковою умовою $y(0) = 1$

.

Розглянемо застосування солверу ode45. Як один з можливих форматів виклику можна запропонувати такий:

[T,y]=ode45(@DiffEquatFunc, [Tstart, Tfinal], StartVector).

Функція, що описує праву частину рівняння, – файл func1.m – має наступний вигляд:

```
function dd = func1(x,y)
% Функція, що описує ДР
dd=-2*x*y;
end
```

Викликатися така функція може зі скрипта (як в нашому випадку) (див. рис. 1.1), функції або з командного вікна: [T,y]=ode45(@func1, [0, 2], 1).

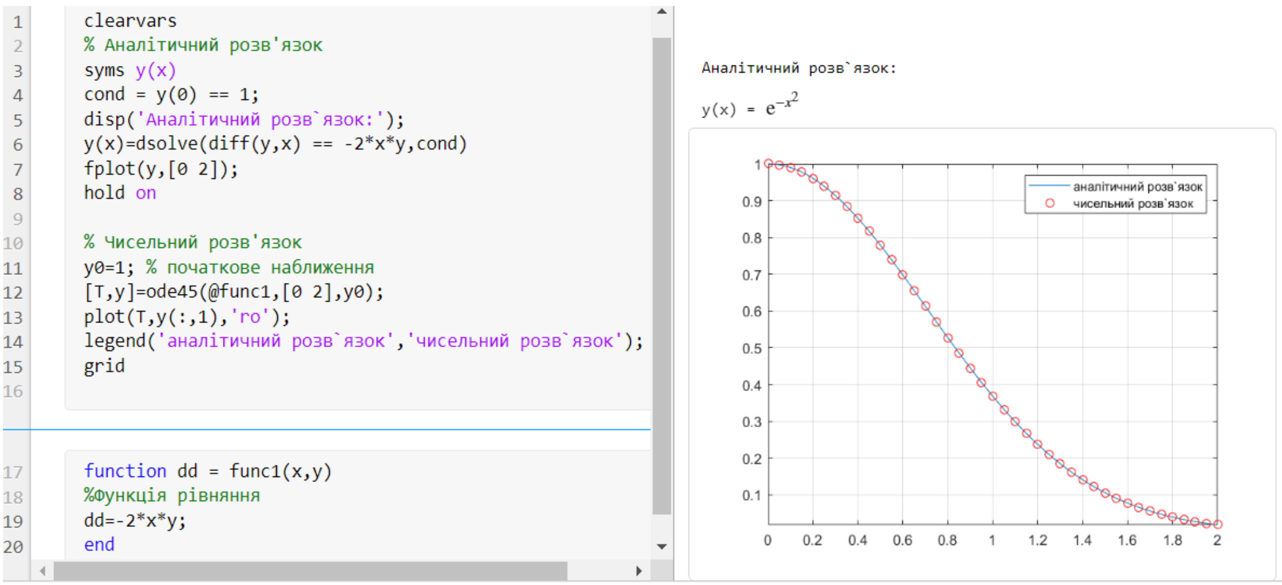


Рисунок 1.1

Тут заданий часовий інтервал від Tstart=0 до Tfinal=2 та початкове значення функції StartVector=1. Графік отриманої таким чином функції y(T) виводиться за допомогою функції plot. Для даного прикладу точності 10^{-3}, закладеної за замовчуванням у процедурі ode45, достатньо.

У загальному випадку солвер ode45 може розв'язувати систему рівнянь такого вигляду: $\frac{d}{dt}X(t)=F(t,x_1,x_2,\ldots,x_n)$, де $X(t)$ – вектор-стовпець $\begin{pmatrix} x_1(t) \\ x_2(t) \\ \ldots \\ x_n(t) \end{pmatrix}$, $F(t,x_1,x_2,\ldots,x_n)$ – функція-стовпець, що залежить від часу та компонент вектору *X*.

Наше ДР можна розв'язати в MatLab ще й аналітично за допомогою таких дій (див. рис.1.1):

```
syms y(x)
cond = y(0) == 1;
y(x)=dsolve(diff(y,x) == -2*x*y,cond)
```

Наявність аналітичного розв'язку дозволяє перевірити результати чисельних розрахунків. У випадку відсутності точного аналітичного розв'язку необхідно проаналізувати отриманий чисельний результат на його так звану «фізичність», виконання відомих граничних випадків, а також прослідкувати, як змінюється результат залежно від зміни точності.

На рис.1.1 наведений скрін екрану, що відповідає аналітичному та чисельному розв'язанню прикладу 1.1 в системі MatLab версії R2020b, де func1 – функція рівняння. Результати аналітичного та чисельного розв'язків виведені у графічне вікно (Figure 1) за допомогою команд fplot та plot, відповідно.

Розглянемо два відомі приклади з механіки.

Приклад 1.2. Рух зарядженої частинки. Закон Кулона.

Даний приклад ілюструє створення функції DiffEquatFunc для виклику її процедурою ode45. Нехай деяка точка маси *m* із зарядом *q* рухається в електричному полі двох нерухомих зарядів Q_1 та Q_2 (рис.1.2).

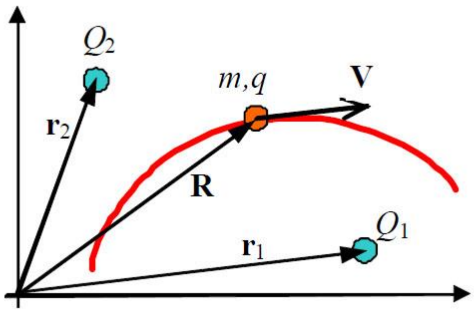


Рисунок 1.2 – Рух зарядженої частинки в полі двох нерухомих зарядів

На заряджену частинку з боку зарядів Q_1 та Q_2 діятимуть кулонівські сили, і її рух описуватиметься таким рівнянням (для простоти припустимо, що рух відбувається у вакуумі):

$$m\ddot{\vec{R}} = \frac{qQ_1}{\left|\vec{R}-\vec{r}_1\right|^3}\left(\vec{R}-\vec{r}_1\right)+\frac{qQ_2}{\left|\vec{R}-\vec{r}_2\right|^3}\left(\vec{R}-\vec{r}_2\right). \tag{1.1}$$

Як бачимо, дане ДР має другий порядок. Але його можна звести до системи ДР першого порядку:

$$\begin{cases} \dot{\vec{R}} = \vec{V} \\ \dot{\vec{V}} = \dfrac{Q_1}{\left|\vec{R}-\vec{r}_1\right|^3}\left(\vec{R}-\vec{r}_1\right)+\dfrac{Q_2}{\left|\vec{R}-\vec{r}_2\right|^3}\left(\vec{R}-\vec{r}_2\right) \end{cases}. \tag{1.2}$$

Нехай маса частинки $m=1$, $q=1$. Крім того, для простоти одразу перейдемо до безрозмірних одиниць і вважатимемо, що дана задача є «плоскою». Введемо такі позначення: $\vec{R}=\left(x_1,x_2\right)$, $\vec{r}_1=\left(C_{1x},C_{1y}\right)$, $\vec{r}_2=\left(C_{2x},C_{2y}\right)$, $\vec{V}=\left(x_3,x_4\right)$. Тоді систему ДР, що описують рух частинки в полі двох нерухомих зарядів, можна подати таким чином:

$$\begin{cases} \dot{x}_1 = x_3, \\ \dot{x}_2 = x_4, \\ \dot{x}_3 = \dfrac{Q_1\left(x_1-C_{1x}\right)}{\left(\left(x_1-C_{1x}\right)^2+\left(x_2-C_{1y}\right)\right)^{\frac{3}{2}}}+\dfrac{Q_2\left(x_2-C_{2x}\right)}{\left(\left(x_1-C_{2x}\right)^2+\left(x_2-C_{2y}\right)\right)^{\frac{3}{2}}}, \\ \dot{x}_4 = \dfrac{Q_1\left(x_1-C_{1y}\right)}{\left(\left(x_1-C_{1x}\right)^2+\left(x_2-C_{1y}\right)\right)^{\frac{3}{2}}}+\dfrac{Q_2\left(x_2-C_{2y}\right)}{\left(\left(x_1-C_{2x}\right)^2+\left(x_2-C_{2y}\right)\right)^{\frac{3}{2}}}. \end{cases} \tag{1.3}$$

Розглянемо найпростіший випадок фінітного руху з Q_1=-50, Q_2=0, C_1=5.0, C_2=10. За таких початкових параметрів (Q_2=0 та Q_1<0) наша точка рухається у притягуючому полі лише першого заряду й, як ми пам'ятаємо з класичної механіки, повинна описувати навколо нього еліпс. Запишемо праву частину системи рівнянь як таку функцію:

```
function f=pointq12(t,x)
global Q1 Q2 C1x C1y C2x C2y
f=[x(3); x(4); Q1*(x(1)-C1x)/(sqrt((x(1)-C1x)^2+(x(2)-C1y)^2))^3+...
  Q2*(x(1)-C2x)/(sqrt((x(1)-C2x)^2+(x(2)-C2y)^2))^3;...
  Q1*(x(2)-C1y)/(sqrt((x(1)-C1x)^2+(x(2)-C1y)^2))^3+...
  Q2*(x(2)-C2y)/(sqrt((x(1)-C2x)^2+(x(2)-C2y)^2))^3];
```

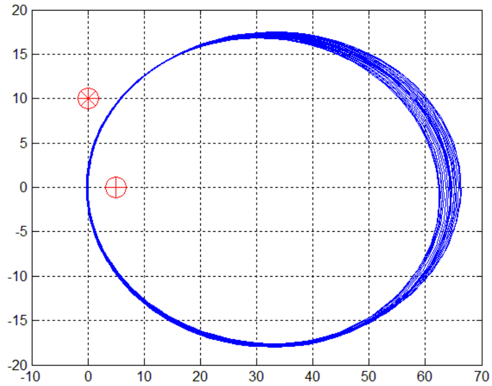
Розв'яжемо систему ДР, викликавши солвер ode45 з функції pointDyn.m та взявши v_{x0}=0; v_{y0}=4.3; T_1=4000:

```
function pointDyn()
clear all
global Q1 Q2 C1x C1y C2x C2y
Q1=-50; Q2=-0.;
C1x=5; C1y=0; C2x=0; C2y=10;
x0=0; y0=0;
vx0=0; vy0=4.3;
T1=4000;
[t,h]=ode45(@pointq12,[0,T1],[x0,y0,vx0,vy0]);
x=h(:,1); y=h(:,2);
x1=C1x; y1=C1y;
x2=C2x; y2=C2y;
plot(x,y,'b-'); % виведення траєкторії
hold on
% виведення положення нерухомих зарядів
plot(x1,y1,'r+',x2,y2,'r*','MarkerSize',15);    plot(x1,y1,'ro',x2,y2,'ro','MarkerSize',15);
comet(x,y); % рух
```

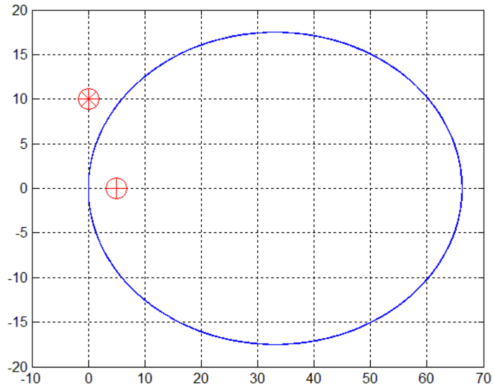
Передача додаткових параметрів у функцію відбувається через опис констант за допомогою global.

Ми отримали фінітний рух, але отримана траєкторія лише віддалено нагадує еліпс (рис.1.3,а) (хрестиком та зірочкою на графіку позначені положення зарядів

Q_1 та Q_2, відповідно). Особливо добре це видно, якщо прослідкувати за траєкторією за допомогою процедури comet(x,y).



а)



б)

Рисунок 1.3 – Траєкторія зарядженої частинки в полі нерухомого заряду: а – з точністю за замовчуванням, б – з підвищеною точністю

Для даного прикладу точності 10^{-3}, закладеної за замовчуванням у процедурі ode45, недостатньо. Тому змінимо точність розв'язку на три порядки – 10^{-6}, це робиться за допомогою визначення нової точності:

```
tol=1e-6;
[t,h]=ode45(@pointq12,[0,T1],[x0,y0,vx0,vy0],odeset('RelTol',tol));
```

Час розрахунку збільшився, але тепер ми отримали досить прийнятний результат (рис.1.3,б).

Зазначимо, що ми використали модель точкових зарядів, тобто знехтували можливістю «потрапляння» зарядів один в одного.

<u>Приклад 1.3. Рух під дією сил тяжіння й тертя.</u>

Розглянемо траєкторію руху кулі під дією сили тяжіння (рис.1.4).

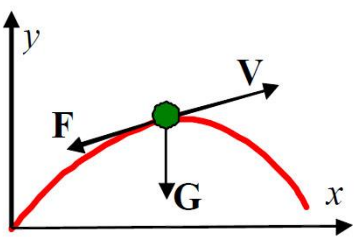


Рисунок 1.4 – Рух кулі під дією сили тяжіння G та опору повітря F

За відсутності опору повітря це буде парабола. При швидкості кулі більше швидкості звуку сила опору повітря пропорційна квадрату швидкості й протилежна напряму руху. Рівняння руху кулі масою m виглядатиме таким чином: $m\vec{w} = m\ddot{\vec{r}} = \vec{G} + \vec{F} = m\vec{g} - kV\vec{V}$. Припустимо для простоти, що коефіцієнт пропорційності k у силі тертя залежить від густини повітря ρ, яка, у загальному випадку, може змінюватися с висотою y, площини поперечного перерізу кулі S та деякого постійного безрозмірного параметру b порядку одиниці, який враховує форму кулі. З міркувань розмірності $k=b\rho S$. Враховуючи, що прискорення – похідна від швидкості за часом, запишемо це рівняння у векторному вигляді:

$$m\dot{\vec{V}} = m\vec{g} - k\left|\vec{V}\right|^2 \cdot \frac{\vec{V}}{\left|\vec{V}\right|}, \quad k = b\rho(y)S. \tag{1.4}$$

Розпишемо це рівняння по координатах:

$$\begin{cases} \dot{V}_x = -\dfrac{k}{m} V_x \sqrt{V_x^2 + V_y^2} \\ \dot{V}_y = -g - \dfrac{k}{m} V_y \sqrt{V_x^2 + V_y^2} \end{cases}. \tag{1.5}$$

У такому вигляді система ДР готова для розв'язання її за допомогою процедури ode45. З такими даними складемо функцію bullet.m:

```
function u=bullet(t,v)
global g ro s m b
k=b*ro*s/m;
u=[-k*sqrt(v(1)^2+v(2)^2)*v(1);   -g-k*sqrt(v(1)^2+v(2)^2)*v(2)];
end
```

Таким чином, ми знайдемо швидкість кулі залежно від часу. Отримані масиви точок V_{xi}, V_{yi}, t_i можна у подальшому обробити, провести інтерполяцію, апроксимацію та т.ін. Знайдемо координати точок простим інтегруванням

$$\begin{cases} x(t) = x_0 + \int\limits_{t_0}^{t} V_x(\tau) d\tau \\ y(t) = y_0 + \int\limits_{t_0}^{t} V_y(\tau) d\tau \end{cases}. \tag{1.6}$$

Для простоти інтегрування замінимо підсумовуванням:

$$\begin{cases} x_i = x_0 + \sum\limits_{k=2}^{i} V_{xk}(t_k - t_{k-1}) \\ y_i = y_0 + \sum\limits_{k=2}^{i} V_{yk}(t_k - t_{k-1}) \end{cases}. \tag{1.7}$$

У цьому випадку початковий момент часу дорівнює 0, куля знаходиться у точці (x_0, y_0). Створимо функцію bulletDyn, з якої викличемо процедуру ode45. Нехай маса кулі m=9 г, поперечний переріз S=0.5 см^2 (приблизно відповідає калібру 7.62 мм). Нехай густина повітря не залежить від висоти й дорівнює $\rho=\rho(0)$=1.22 кг/м^3, прискорення вільного падіння g=9.8 м/с^2, коефіцієнт b=0.5. Переведемо все до системи СІ та знерозміримо. У початковий момент часу куля перебувала у початку координат, а швидкість кулі по горизонталі складала 800 м/с, по вертикалі – 100 м/с.

```
function bulletDyn()
clear all
global g ro s m b
g=9.8;  ro=1.22;  s=0.00005;  m=0.009;  b=0.5;
V0x=800; V0y=100;
x0=0; y0=0;
T=6.5;  % підібраний час розрахунків
[t,h]=ode45(@bullet,[0 T],[V0x V0y]);
vx=h(:,1); vy=h(:,2);
ll=length(t);
x(1)=x0; y(1)=y0;
for i=2:ll
   x(i)=x(i-1)+vx(i-1)*(t(i)-t(i-1));
    y(i)=y(i-1)+vy(i-1)*(t(i)-t(i-1));
end
plot(x,y,'b-','LineWidth',2);
hold on
grid on
```

Результати наведені на рис.1.5. Часовий діапазон [0, 6.5] був підібраний емпірично таким чином, щоб траєкторія руху була показана від моменту вильоту до падіння. Рівень «землі» відповідає значенню ординати 0.

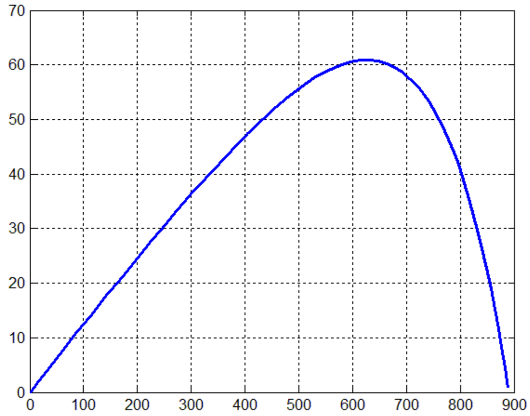


Рисунок 1.5 – Траєкторія кулі під дією сили тяжіння й опору повітря

Нагадаємо, що ми розв'язували систему ДР (1.5) відносно компонент швидкості руху кулі. При розв'язанні системи (1.5) відносно координат вона набуде вигляду:

$$\begin{cases} \dot{x} = V_x \\ \dot{y} = V_y \\ \dot{V}_x = -\dfrac{k}{m} V_x \sqrt{V_x^2 + V_y^2} \\ \dot{V}_y = -g - \dfrac{k}{m} V_y \sqrt{V_x^2 + V_y^2} \end{cases} . \qquad (1.8)$$

1.2 Осцилятор Ван дер Поля. Жорсткість системи звичайних диференціальних рівнянь

Вище ми застосовували однокроковий метод Рунге-Кутта високого порядку ode45. Це класичний, досить швидкий метод, рекомендований для початкової проби розв'язку. В багатьох випадках він дає хороші результати. Розглянемо тепер відомий приклад осцилятора Ван дер Поля й побачимо, що застосування ode45 або сильно збільшує час розв'язку, або взагалі не може призвести до розв'язку. Отже, ДР, що описує осцилятор Ван дер Поля, виглядає таким чином:

$$\frac{d^2x}{dt^2} - \mu\left(1 - x^2\right)\frac{dx}{dt} + x = 0, \tag{1.9}$$

де μ – параметр. Перепишемо це рівняння другого порядку у вигляді системи ЗДР першого порядку. Вважатимемо $x_1 = x$, $x_2 = \dot{x}_1 = \dot{x}$, тоді

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx2}{dt} = \mu\left(1 - x_1^2\right)x_2 - x_1 \end{cases}. \tag{1.10}$$

За функцію, що описує систему ДР (1.10), візьмемо стандартну функцію MatLab vanderpoldemo:

```
function dydt = vanderpoldemo(t,y,Mu)
%VANDERPOLDEMO Defines the van der Pol equation for ODEDEMO.
% Copyright 1984-2002 The MathWorks, Inc.
% $Revision: 1.2 $  $Date: 2002/06/17 13:20:38 $
dydt = [y(2); Mu*(1-y(1)^2)*y(2)-y(1)];
```

Для малих μ порядку одиниці будь-який солвер MatLab зможе ефективно розв'язати рівняння Ван дер Поля.

Для великих значень $\mu>0$ система ЗДР стає жорсткою. Причина в тому, що при збільшенні параметра μ починають сильно розрізнятися порядки коефіцієнтів при різних доданках. Саме степінь цієї відмінності найчастіше й визначає жорсткість системи ЗДР $\frac{d}{d}X(t) = F(t,X)$. Як відповідну характеристику обирають матрицю Якобі (якобіан) векторної функції $F(t,X)$, що визначає праву частину системи ЗДР. Чим сильніше вироджена матриця Якобі, тобто функціональна матриця, складена з похідних $F(t,X)$, тим жорсткішою є система рівнянь. Для швидкого й ефективного інтегрування таких систем потрібно використовувати спеціальні методи, реалізовані в солверах ode15s, ode23s, ode23t, ode23tb тощо.

Контрольні питання

1. За допомогою яких процедур розв'язуються ЗДР у MatLab та Octave? Наведіть алгоритм розв'язання.

2. Які солвери використовуються для розв'язання жорстких систем ДР?

3. Які особливості програмної реалізації наведених вище моделей у MatLab?