фМІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

# МОДЕЛЮВАННЯ СИСТЕМ

Методичні вказівки
до виконання лабораторних робіт
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 (122) «Комп’ютерні науки»,
F6 (126) «Інформаційні системи і технології»,
A5 (015.39) «Професійна освіта (Цифрові технології)»

Київ 2025

УДК 004.94;519.876
М74

Укладач О. В. Горда, канд. техн. наук, доцент

Рецензент О. А. Поплавський, д-р техн. наук, доцент

Відповідальна за випуск Т. А. Гончаренко, д-р техн. наук, доцент

*Затверджено на засіданні кафедри інформаційних технологій, протокол № 11 від 17 березня 2025 року.*

В авторській редакції.

**Моделювання** систем : методичні вказівки до виконання
М74 лабораторних робіт / уклад. О.В. Горда. – Київ : КНУБА, 2025. – 104 с.

Містять зміст, порядок оформлення і поради до виконання окремих лабораторних робіт.

Призначено для здобувачів першого (бакалаврського) рівня вищої освіти спеціальностей F3 (122) «Комп’ютерні науки», F6 (126) «Інформаційні системи і технології», A5 (015.39) «Професійна освіта (Цифрові технології)».

**ЗМІСТ**

## ЗАГАЛЬНІ ПОЛОЖЕННЯ

Дисципліна «Моделювання систем» входить до навчальних планів усіх комп'ютерних, а також ряду природничо-наукових та спрямованих на бізнес спеціальностей широкого профілю. Це викликано вимогами науково-технічного прогресу, в процесі якого людство йде шляхом використання все більш складних технічних та інформаційних засобів, а також використання складних природних об'єктів або систем.

Методичні вказівки є доповненням до конспекту лекцій «Моделювання систем» і охоплюють основні типи моделей.

**Метою** методичних вказівок до виконання лабораторних завдань є формування знань та закріплення практичних навичок у студентів загальних принципів постановки та розв'язання завдань, пов'язаних з моделюванням систем різного типу та призначення, а також використання спеціалізованих програмних додатків до задач математичного та імітаційного моделювання.

Методичні вказівки містять рекомендації для вирішення задач, які безпосередньо, в практичному зрізі відображають якість засвоєння здобувачами вивчення дисципліни, а саме:

- причинно-наслідковий аналіз проблеми;
- оцінювати результати натурного та імітаційного експерименту;
- побудова моделі на основі експериментальних даних;
- побудова та дослідження моделі неперервних та дискретних детермінованих систем;
- побудова та дослідження моделі стохастичних систем;
- аналіз та побудова моделі складних систем.

Для кращого розуміння та полегшення засвоєння матеріалу в методичних вказівках наведені типові приклади вирішення завдань.

Завершивши вивчення дисципліни, здобувач повинен вміти здійснювати постановку проблеми; ідентифікувати об'єкт моделювання та правильно визначати математичний апарат для побудови моделі; виконувати реалізацію та дослідження моделей різного типу із застосуванням програмного інструментарію.

В якості інструментального засобу для вирішення задач моделювання пропонується середовище Matlab, яке сьогодні для студентів та викладачів є у вільному online доступі https://matlab.mathworks.com/.

## Лабораторна робота №1
## Тема: Аналіз системи

***Мета:*** отримання практичних навичок з аналізу системи – виділення системи та визначення зовнішнього середовища та за необхідності надсистеми, визначення типу системи; побудова моделі «чорна скринька». Проведення структурного та функціонального аналізу системи та схематичне представлення результатів. Визначення критеріїв ефективності оцінки системи та їх типи.

***Завдання для самостійного виконання***

Для вибору теми для самостійного виконання бажано обирати тему за напрямом дипломної або наукової роботи студента. Також тему можна обрати із списку, наведеному нижче:

1. Технічна система: механізм, машина, прилад.
2. Транспортна система: потік транспорту, транспортна мережа або її елемент: перехрестя, пішохідний перехід, розв’язка.
3. Громадський заклад: освітній заклад, бібліотека.
4. Система управління: освітнім закладом, освітнім процесом, технологічним або виробничим процесом, торговельним процесом.
5. Система оцінювання або прийняття рішень.
6. Інформаційна система: система ідентифікації та аутентифікації, система обліку, система моніторингу і т.д.

***Теоретичні відомості***

Необхідний теоретичний матеріал викладено у дисципліні «Системний аналіз» та в лекціях 1, 2 [1].

***Порядок виконання***

Розглянемо заклад вищої освіти як єдину систему з позиції надання освітніх послуг.

*Система*: КНУБА –університет.

*Головна мета системи*: накопичення та передача знань для підготовки кваліфікованих кадрів відповідно до необхідних компетенцій, які передбачені стандартами відповідних спеціальностей як інструментарію для вирішення нагальних задач та проблем, що висуваються потребами часу.

*Головна мета дослідження*: організація учбового процесу.

*Зовнішнє середовище*: країна – замовник освітніх послуг. Регламентує попит та вимоги до випускників необхідних спеціальностей. Регламентує правову та економічну діяльність закладу на рівні держави.

*Надсистема*: МОН – головний орган у системі центральних органів виконавчої влади, що забезпечує формування та реалізує державну політику у сферах освіти і науки, наукової, науково-технічної та інноваційної діяльності, а також забезпечує формування та реалізацію державної політики у сфері здійснення державного нагляду (контролю) за діяльністю закладів освіти, підприємств, установ та організацій, які надають послуги у сфері освіти.

Залежно від мети дослідження, МОН може виступати як зовнішнє середовище.

Під час розгляду навчального підрозділу, наприклад, факультету, надсистемою буде ректорат (рис. 1.1), а МОН – зовнішнім середовищем.

Рис. 1.1. Модель «чорна скринька» ЗВО

*Тип системи*:

– за походженням – штучна;

– за об’єктивністю існування – реальна;

– за характером зв’язків – відкрита (обмінюється ресурсами та інформацією із зовнішнім середовищем);

– за структурою – складна, гетерогенна (характеризуються великим числом елементів, внутрішніх зв’язків, їх неоднорідністю, структурною різноманітністю, виконує ряд функцій). Можна віднести до третього типу складності – важко формалізується;

– за характером розвитку – та, що розвивається;

– за поведінкою та управлінням – самоорганізуюча, частково централізована.

В процесі функціонування ЗВО можна виділити основні види діяльності (рис. 1.2 – 1.4):

– управлінська;

– навчальна;

– наукова;

– забезпечуюча;

– партнерська.

Рис. 1.2. Організаційна структура ЗВО

Рис. 1.3. Структура організації процесу навчання у ЗВО

Рис. 1.4. Маршрут підготовки студента

*Критерії оцінки ефективності*. Оцінки ефективності навчальної діяльності ЗВО, як правило, визначаються МОН України, оцінювання виконується за такими критеріями:

- міжнародна діяльність;
- якість фінансового забезпечення освітнього процесу;
- якість формування студентського контингенту;
- якість реалізації змісту підготовки фахівців;
- якість наукової та науково-дослідної роботи під час підготовки наукових кадрів;
- якість організаційного, науково-методичного та інформаційного забезпечення освітнього процесу;
- якість навчальних і творчих досягнень студентів і випускників, результати працевлаштування;
- якість науково-педагогічного складу.

В якості оцінок беруться відношення фактичних значень до відповідних показників держзамовлення, де в якості фактичних значень виступає частка від загальної кількості, що дозволяє отримати нормовані, однорідні оцінки за кожним критерієм. Тоді остаточну оцінку можна вивести у вигляді адитивної згортки частинних критеріїв: $O = \sum_i g_1 o_i$, де $o_i$ – оцінка за частинним критерієм, $g_i$ – відповідний ваговий коефіцієнт частинного критерію.

Приклади технічної та інформаційної системи наведені у методичних вказівках до курсової роботи [3].

## Лабораторна робота №2

## Тема: Основи роботи в середовищі Matlab

***Мета:*** ознайомлення студентів з програмним середовищем Matlab, його структурою та можливостями. Способи вводу та початкового аналізу вхідних даних. Арифметичні операції та виконання обчислень. Робота у командному вікні.

***Завдання для самостійного виконання***

В результаті дослідження деякої характеристики системи при сталих значеннях вхідних даних $x$ виконано п’ять серій вимірювань вихідної величини $y$. Результати вимірювань занесені у таблицю. Необхідно:

- оцінити придатність результатів вимірювань для аналізу системи та подальшого використання у задачах побудови на їх основі математичної моделі;
- оцінити точність, з якою можна на основі заданих вимірювань побудувати математичну модель.

***Варіанти завдань***

**№ 1** $\left(f(x)=2x\cdot\sin(2x)\right)$

| $x$ | Прогони | | | | |
|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 0 | 0 | 0.0006 | 0.119 | 0.107 | 0.601 |
| 1 | 2.728 | 2.825 | 2.737 | 3.303 | 2.761 |
| 2 | -4.541 | -4.248 | -4.009 | -4.407 | -3.854 |
| 3 | -2.515 | -2.340 | -1.913 | -1.943 | -1.877 |
| 4 | 11.872 | 12.284 | 12.039 | 11.981 | 12.884 |
| 5 | -8.160 | -8.073 | -7.710 | -7.648 | -7.371 |
| 6 | -9.658 | -9.303 | -9.601 | -9.463 | -8.648 |
| 7 | 20.803 | 20.955 | 21.586 | 21.280 | 20.935 |
| 8 | -6.910 | -6.864 | -6.390 | -6.404 | -6.533 |
| 9 | -20.277 | -20.203 | -19.401 | -20.191 | -19.933 |
| 10 | 27.388 | 27.883 | 28.344 | 27.973 | 27.557 |

**№ 2** $\left(f(x)=10\dfrac{\sin(2x+3)}{x}\right)$

| $x$ | Прогони | | | | |
|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 2 | 3.285 | 3.702 | 3.819 | 3.468 | 3.512 |
| 3 | 1.374 | 1.674 | 1.626 | 2.259 | 2.294 |
| 4 | -2.500 | -2.374 | -2.255 | -1.508 | -1.837 |
| 5 | 0.840 | 0.841 | 0.927 | 1.888 | 1.333 |
| 6 | 1.084 | 1.487 | 1.765 | 1.444 | 1.581 |
| 7 | -1.373 | -1.268 | -1.250 | -1.221 | -0.864 |
| 8 | 0.187 | 0.464 | 0. 251 | 1.129 | 0.876 |
| 9 | 0.930 | 0.987 | 1. 442 | 1.661 | 1.536 |

| | | | | | |
|---|---|---|---|---|---|
| 10 | -0.846 | -0.470 | -0.410 | -0.759 | -0.840 |
| 11 | -0.120 | 0.151 | 0.207 | 0.664 | -0.020 |
| 12 | 0.797 | 1.015 | 1.169 | 0.923 | 1.660 |

**№ 3** $\left(f(x) = -5x + \sin(2x)\right)$

| $x$ | Прогони | | | | |
|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 0 | 0 | 0.174 | 0.319 | 0.722 | 0.322 |
| 1 | -4.091 | -3.815 | -3.901 | -4.076 | -3.981 |
| 2 | -10.757 | -10.106 | -10.480 | -10.049 | -10.704 |
| 3 | -15.279 | -15.215 | -14.802 | -15.062 | -14.821 |
| 4 | -19.011 | -18.381 | -18.922 | -18.842 | -18.572 |
| 5 | -25.544 | -25.430 | -25.478 | -25.203 | -25.295 |
| 6 | -30.537 | -29.467 | -30. 061 | -30.169 | -29.890 |
| 7 | -34.009 | -33.843 | -33. 995 | -33.207 | -33.782 |
| 8 | -40.288 | -39.470 | -40.260 | -39.761 | -39.572 |
| 9 | -45.751 | -45.694 | -45.685 | -44.953 | -45.156 |
| 10 | -49.087 | -48.683 | -48.655 | -48.942 | -48.798 |

**№ 4** $\left(f(x) = 3e^{-0,5x} + 2\right)$

| $x$ | Прогони | | | | |
|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 0 | 5 | 5.199 | 5.602 | 5.022 | 6.042 |
| 1 | 3.820 | 4.293 | 4.396 | 4.335 | 3.916 |
| 2 | 3.104 | 3.371 | 3.326 | 3.172 | 3.310 |
| 3 | 2.669 | 3.016 | 2.674 | 3.428 | 3.600 |
| 4 | 2.406 | 2.536 | 2.488 | 2.626 | 2.659 |
| 5 | 2.246 | 2.509 | 3.011 | 3.032 | 2.251 |
| 6 | 2.149 | 2.349 | 2. 96 | 2.383 | 2.860 |
| 7 | 2.091 | 2.383 | 2. 682 | 2.385 | 2.464 |
| 8 | 2.397 | 2.470 | 2.582 | 2.230 | 2.101 |
| 9 | 2.251 | 2.694 | 2.176 | 2.226 | 2.260 |
| 10 | 2.020 | 2.024 | 2.548 | 2.742 | 2.204 |

**№ 5** $\left(f(x) = 2\ln(3x + 1)\right)$

| $x$ | Прогони | | | | |
|---|---|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ |
| 0 | 0.149 | 0.010 | 0.443 | 0.954 | 0.185 |
| 1 | 2.920 | 2.773 | 2.870 | 2.857 | 3.357 |
| 2 | 3.932 | 3.892 | 4.215 | 4.601 | 4.519 |
| 3 | 4.741 | 4.605 | 5.060 | 4.842 | 4.655 |
| 4 | 5.203 | 5.136 | 5.591 | 5.903 | 5.390 |
| 5 | 5.567 | 5.545 | 5.621 | 6.270 | 6.440 |
| 6 | 5.929 | 5.889 | 6.176 | 6.418 | 6.607 |
| 7 | 6.313 | 6.182 | 6.654 | 6.658 | 6.576 |
| 8 | 6.575 | 6.438 | 6.727 | 7.148 | 7.342 |
| 9 | 6.680 | 6.640 | 6.792 | 7.804 | 7.053 |
| 10 | 6.874 | 8.891 | 7.048 | 7.338 | 7.222 |

***Теоретичні відомості***

*Основні положення роботи в середовищі Matlab.*

В середовищі Matlab можна створювати математичні та імітаційні моделі різними способами:

– в режимі інтерпретатора командному вікні (Command Window);

– в режимі компілятора (створення підпрограм функцій та скриптів);

– вбудованою графічною мовою Simulink.

Мова Matlab є регістрозалежною: велика та маленька літера – різні символи.

У командному вікні символ «>>» – запрошення до вводу.

Для блокування виводу результату виконання оператора в кінці ставиться «;». Якщо в одному рядку записується декілька операторів, то їх можна розділяти або «,» або «;». Якщо оператор не вміщується в одному рядку, то символ «…» в кінці рядка означає, що він продовжується на наступний рядок.

Символ «%» на початку рядка позначає коментар.

*Типи даних.*

Matlab не вимагає опису типу чи вимірювання змінних (табл. 2.1). Кожен раз, коли Matlab зустрічає нове ім’я змінної, він створює змінну і виділяє відповідно простір пам’яті (за замовчуванням – масив дійсних чисел подвійної точності). Проста змінна визначається як матриця розмірності 1x1.

Всі значення змінних зберігаються у загальнодоступному просторі Workspace. Вміст робочого простору можна зберегти у файлі за допомогою команди головного меню Save Workspace, а потім завантажити командою головного меню Open і вибрати необхідний файл або за допомогою команди:

load ім’я_файлу.

*Таблиця 2.1*

**Типи змінних**

| ***Тип*** | ***Опис*** |
|---|---|
| char | символ або масив символів (рядок). Значення береться в апострофи: x = ‘a”; y = ‘hello world’. Також описати символьні змінні можна ключовим словом: syms a b c d |
| uint8[16, 32] | ціле без знаку (різної розрядності) |
| int8[16,32] | ціле із знаком (різної розрядності) |
| single | дійсне звичайної точності (4 байти) |
| double | дійсне подвійної точності (8 байтів) |
| complex | комплексне число $z = a + b * i$. За замовчуванням $i$ або $j$ – |

| | |
|---|---|
| | системні константи для позначення уявної одиниці $\sqrt{-1}$ |
| struct | структура (опис та доступ подібний до мови С) |
| cell | комірка, дозволяє об'єднати пов'язані дані, можливо різних розмірів та типів, в єдину структуру:<br>myCell = {1, 2, 3; 'text', rand(5,10,2), {11; 22; 33}} |
| table | таблиці. Можна створити на основі даних Workspace:<br>T = table(Gender,Smoker,Height,Weight); T(1:5,:) – таблиця з чотирьох стовпчиків та п'яти рядків;<br>або завантажити з файлу: T2 = readtable('patients.dat');<br>T2(1:5,:) |
| handle_fun | вказівник на функцію створюється за допомогою операції адресації @. Функція може бути описана безпосередньо при створенні вказівника: MySqr=@(x)[x;x.^2]; або можна створити вказівник на підпрограму-функцію:<br>function f = sqr(x) – опис функції<br>f=x.^2;<br>end<br>f=@sqr – створення вказівника |
| UserObject | тип, що визначається користувачем: inline( ) |

В Matlab передбачені системні змінні, на відміну від констант їх значення можна перевизначати:

- $i$ або $j$ – системні константи для позначення уявної одиниці $\sqrt{-1}$;
- pi – число $\pi = 3{,}1415926...$;
- eps – похибка операцій над дійсними числами $eps = 2^{52}$;
- realmin, realmax – найменше $\left(2^{1022}\right)$ та найбільше $\left(2^{1023}\right)$ дійсне число;
- inf – позначення машинної нескінченності;
- NaN – вказує на нечисловий характер даних (Not-a-Number) або на невизначеність нуль-ділити на-нуль;
- ans – змінна для збереження результату останньої операції.

*Введення та виведення значень змінних.*

Для введення значень змінних існує декілька способів:

- за допомогою оператора присвоєння « = », значення задаються у [ ]:
  - x = 2; – $x$ визначається як матриця дійсних чисел розмірності 1×1;
  - x= [1 2 3] – задання значень вектора-рядка з трьох значень;
  - x= [1;2;3] – задання значень вектора-стовпчика з трьох значень;
  - x= [1 2 3;4 5 6] – задання значень матриці розмірності 2x3;
- для задання значень змінної на деякому проміжку можна використовувати операцію діапазону « : »: x = 0:0.5:4 (x = початкове

значення : крок : кінцеве значення) – *x* визначається як вектор рядок, що приймає значення від 1 до 4 з кроком 0,5, запис x = 0:4 означає, що замовчуванням крок приймається рівним одиниці;

– зчитати з текстового файлу або таблиці (наприклад, Excel) за допомогою команди головного меню Import Data (рис. 2.1). Після вибору необхідного файлу відкривається вікно налаштування та попереднього перегляду даних (рис. 2.1) де визначається роздільник між даними, діапазон таблиці, який буде зчитуватись, тип. Після натискання на команду Import Selection дані записуються у робочий простір з іменем VarName1, яке можна змінити.

– задати безпосередньо у робочому просторі. У випадаючому меню налаштування, яке з'являється під час натискання правої кнопки «миші» на робочому просторі, вибирається пункт New. У Workspace з'явиться змінна unnamed, її можна перейменувати і задати необхідний ідентифікатор. Далі необхідно двічі натиснути на квадратик, що стоїть ліворуч ідентифікатора, відкриється таблиця, в яку можна вносити значення змінної (рис. 2.2).

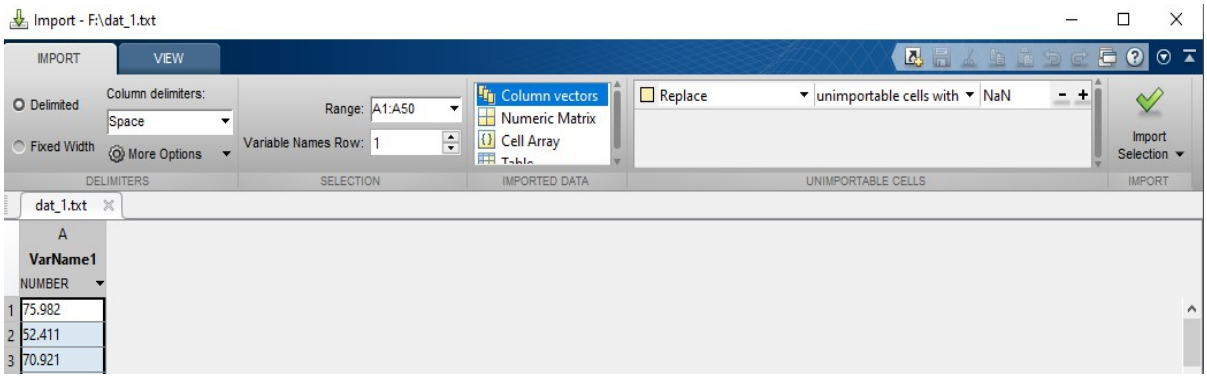



Рис. 2.1. Вікно налаштування Import Data

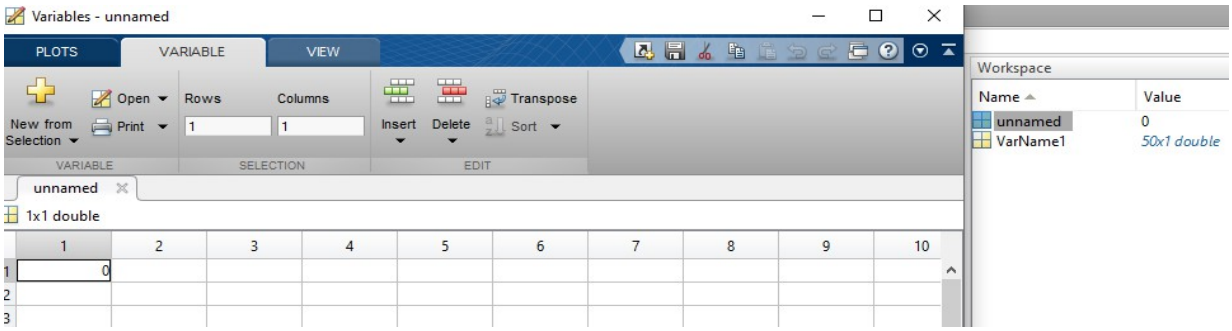



Рис. 2.2. Вікно значень змінної

***Зауваження***. Нумерація масивів за замовчуванням починається з 1. Щоб звернутись до елементу масиву використовують ( ):

– name(n) – виділяє *n*-й елемент вектора;

– name(n,m) – виділяє елемент матриці з *n*-го рядка *m*-го стовпчика;
– для виділення частини масиву зручно використовувати операцію діапазону:
– name(n,:) – виділяє всі стовпчики *n*-го рядка;
– name(n:m) – виділяє елементи вектора, починаючи з *n*-го до *m*-го.

Для об'єднання масивів існує спеціальна команда:

x=cat(dim,A1,A2,…),

де x – масив, утворений шляхом об'єднання масивів A1,A2,…, dim – параметр, який визначає спосіб об'єднання: 1 – вертикально; 2 – горизонтально; 3 – шляхом накладання (використовується під час обробки колірних зображень).

Для обробки масивів в різних мовах програмування застосовують цикл for, який в середовищі Matlab має наступний вигляд:

for x=x_start:x_step:x_end
тіло циклу
end

Змінна x може бути будь-якого простого типу (цілого, дійсного, символьного), наприклад, якщо описати змінну a11 як символьну (sum a11), то результатом виконання дії a11='a':'d' буде вектор рядок abcd.

***Зауваження.*** Для виконання операцій з масивами можна не користуватись циклом, відповідна операція буде виконана для всіх елементів масиву, але при цьому необхідно, щоб виконувались наступні умови:

– масиви мали правильні розмірності;
– в Matlab розрізняються поелементні операції та матричні. Для того, щоб операція виконувалась поелементно, то перед нею ставлять крапку, наприклад, якщо $a$ і $b$ матриці однакової розмірності, то в результаті обчислення $a*b$ отримаємо матрицю як добуток за матричними правилами, якщо запишемо $a.*b$, то кожний елемент матриці $a$ буде помножений на відповідний елемент матриці $b$. Зрозуміло, що для операцій додавання та віднімання крапку можна не ставити.

*Арифметичні операції.*

Арифметичні операції мови Matlab наведені у табл. 2.2.

*Таблиця 2.2*

**Арифметичні операції**

| Позначення | Операція | Позначення | Операція |
|---|---|---|---|
| + | додавання | - | віднімання |
| * | добуток | ^ | степінь |
| / | ділення | \ | ліве ділення |
| ‘ | комплексно спряжене транспонування | | |

***Приклад.*** Нехай задано дві матриці розмірності 2 х 2:

```
>> a=[1 2; 3 4];
>> b=[2 1;4 3];
a =   1   2                    b =…..2…..1
      3   4                    …….4……3
```

Знайдемо добуток матриць та добуток відповідних елементів:

```
      a*b                          a.*b
ans =  10   7              ans =   2   2
       22  15                     12...12
```

*Деякі статистичні оцінки.*

В середовищах Matlab та Mathcad вбудована велика кількість спеціальних функцій різного типу та призначення, включаючи статистичні оцінки (табл. 2.3).

*Таблиця 2.3*

**Основні стандартні функції статистичних оцінок**

| Числова характеристика | Функція Mathcad | Функція Matlab |
|---|---|---|
| Математичне сподівання | ***mean*(*x*)** | ***mean*(*x*)** |
| Дисперсія | ***var*(*x*)** | ***var*(*x*)** |
| Виправлене середньоквадратичне відхилення | ***stdev*(*x*)** | ***std*(*x*)** |
| Медіана | ***median*(*x*)** | ***median*(*x*)** |

***Порядок виконання***

Для виконання обчислень застосуємо спеціалізовані програмні середовища Mathcad або Matlab.

1. Введення початкових даних. Вхідні дані задамо як два масиви: вектор $xata$, матриця $ydata$. Оскільки значення вектора $x$ задані з постійним кроком, то в середовищі Matlab для їх задання доцільно скористатись операцією діапазон:

>> xdata=0:10;  % – вектор-рядок із значеннями від 0 до 10 з кроком 1.

Значення матриці *ydata* задамо безпосередньо у Workspace (рис. 2.3).

Збережемо наші дані на диску, для цього на панелі головного меню передбачена кнопка Save Workspace.

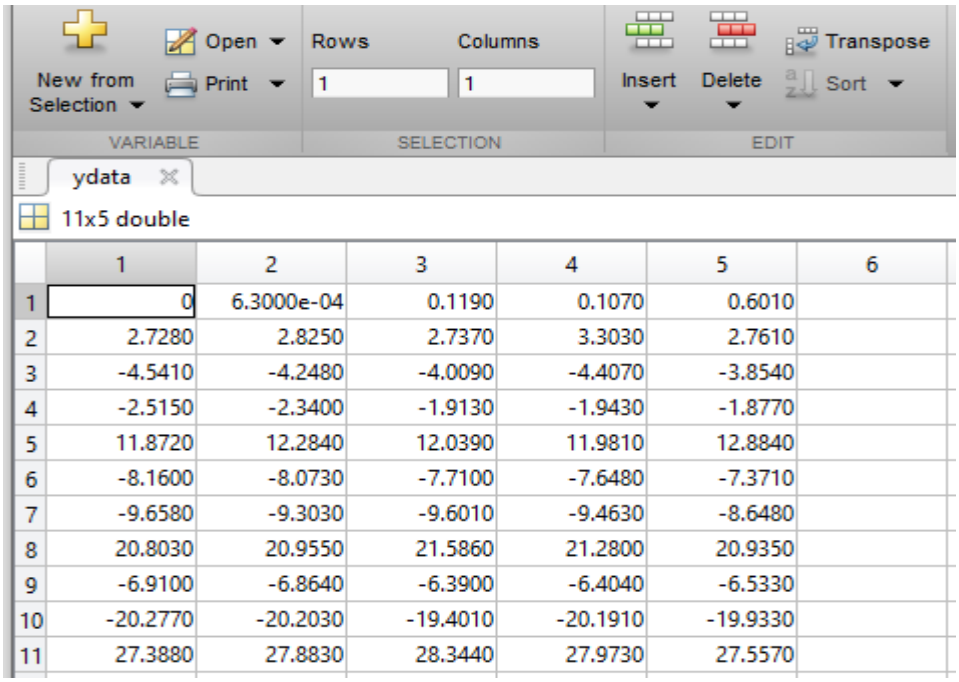
В середовищі Mathcad, щоб не було конфлікту типу даних, краще задати наступним чином:

– *xata* – вектор стовпчик;

– *ydata* – матриця розмірності 11 x 5.

2. Перевірка придатності даних вимірювань полягає у перевірці статистичних гіпотез:

– для кожного фіксованого значення $x$ значення $y$ повинні мати нормальний закон розподілу (існує значення, до якого всі інші прямують і воно може бути взяте за істине значення величини $y$).




| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 6.3000e-04 | 0.1190 | 0.1070 | 0.6010 | |
| 2 | 2.7280 | 2.8250 | 2.7370 | 3.3030 | 2.7610 | |
| 3 | -4.5410 | -4.2480 | -4.0090 | -4.4070 | -3.8540 | |
| 4 | -2.5150 | -2.3400 | -1.9130 | -1.9430 | -1.8770 | |
| 5 | 11.8720 | 12.2840 | 12.0390 | 11.9810 | 12.8840 | |
| 6 | -8.1600 | -8.0730 | -7.7100 | -7.6480 | -7.3710 | |
| 7 | -9.6580 | -9.3030 | -9.6010 | -9.4630 | -8.6480 | |
| 8 | 20.8030 | 20.9550 | 21.5860 | 21.2800 | 20.9350 | |
| 9 | -6.9100 | -6.8640 | -6.3900 | -6.4040 | -6.5330 | |
| 10 | -20.2770 | -20.2030 | -19.4010 | -20.1910 | -19.9330 | |
| 11 | 27.3880 | 27.8830 | 28.3440 | 27.9730 | 27.5570 | |

Рис. 2.3. Введення значень матриці

Для перевірки нормального закону можна застосовувати критерій Пірсона;

– при малих вибірках критерій Пірсона може дати дуже наближений результат, тому достатньо знайти статистичні оцінки середнього вибіркового $\bar{y}$, медіану $me_y$, моду $mo_y$ та

середньоквадратичне відхилення $\sigma_y$. Оскільки значення з'являлись один раз, то моди немає.

Для обчислення статистичних оцінок скористаємось стандартними функціями (табл. 2.3). Обчислимо, також, коефіцієнт варіації *cv*. Щоб спростити обчислення, скористаємось циклом for:

```
>> for i=1:11
yvs(i)=mean(ydata(i,:));
me(i)=median(ydata(i,:));
sigma=std(ydata(i,:));
cv=sigma./abs(yvs).*100;
end
```

Для зручності перегляду та порівняння отриманих оцінок запишемо їх у матрицю *Y* і виведемо на екран (рис. 2.4). Видно, що, крім першого рядка, середнє вибіркове та медіана майже співпадають, середньоквадратичне відхилення не є значним. Це дає підставу вважати закон розподілу значень нормальним і у якості істинних вихідних значень взяти значення *yvs*.

```
>> Y(:,1)=yvs; Y(:,2)=me; Y(:,3)=sigma; Y(:,4)=cv

Y =

    0.1655    0.1070    0.3733  225.5094
    2.8708    2.7610    0.3733   13.0025
   -4.2118   -4.2480    0.3733    8.8626
   -2.1176   -1.9430    0.3733   17.6273
   12.2120   12.0390    0.3733    3.0566
   -7.7924   -7.7100    0.3733    4.7903
   -9.3346   -9.4630    0.3733    3.9989
   21.1118   20.9550    0.3733    1.7681
   -6.6202   -6.5330    0.3733    5.6385
  -20.0010  -20.1910    0.3733    1.8663
   27.8290   27.8830    0.3733    1.3413
```

Рис. 2.4. Статистичні оцінки (Matlab)

В середовищі Mathcad завдання виконується аналогічно. Для обчислення статистичних оцінок для кожного рядка можна скористатись операцією транспонування та виділення стовпчика матриці (рис. 2.5).

$m_i := \text{mean}\left[\left(y^T\right)^{\langle i \rangle}\right]$ $me_i := \text{median}\left[\left(y^T\right)^{\langle i \rangle}\right]$ $\sigma_i := \text{stdev}\left[\left(y^T\right)^{\langle i \rangle}\right]$ $kv_i := \left|\frac{\sigma_i}{m_i}\right| \cdot 100$

Рис. 2.5. Статистичні оцінки (Mathcad)

***Висновок.*** Дані, отримані в результаті експериментальних вимірювань, можна вважати придатними для побудови математичної моделі, так як, крім першого рядка, середні значення і медіана практично співпадають, середньоквадратичне відхилення в межах допустимого значення. В якості абсолютної похибки моделі можна застосувати першу норму і взяти максимальне значення виправленого середньоквадратичного відхилення, а в якості відносної – коефіцієнт варіації. Для зменшення похибки точку $x = 0$ з розгляду можна виключити або провести повторне вимірювання.

## Лабораторна робота №3

## Тема: Побудова математичної моделі за принципом «чорної скриньки» в середовищі Matlab

***Мета:*** ознайомити студентів та надати практичних навичок з програмним середовищем Matlab і можливостями роботи у режимі компілятора та побудови математичної моделі за принципом «чорної скриньки» (на основі експериментальних даних).

***Завдання для самостійного виконання***

На основі даних, наведених у роботі №2

- визначити тип функції $y = f(x)$, який лежить в основі математичної моделі;
- методом найменших квадратів побудувати математичну модель та оцінити її точність.

***Теоретичні відомості***

Поширеним методом побудови моделі за експериментальними даними є метод найменших квадратів (МНК). Його перевагою є те, що математичну модель можна представляти не тільки у вигляді полінома, а і за допомогою елементарних функцій та їх комбінацією, що дуже важливо, коли процес, що моделюється, є періодичним, або монотонним і модель необхідно отримати на великому проміжку зміни аргументу.

Основні функції наближення експериментальних даних функції однієї змінної для середовищ Mathcad та Matlab наведені відповідно у табл. 3.1, 3.2.

*Таблиця 3.1*

**Основні функції апроксимації та згладжування в Mathcad**

| Тип згладжування | Функція | Додаткові параметри |
|---|---|---|
| Лінійне $y = ax + b$ | ***line (x,y)*** | |
| Поліноміальне $y = P_n(x)$ | ***regress (x, y, n)*** | n – степінь полінома |
| Експоненціальне $y = ae^{bx} + c$ | ***expfit (x,y,g)*** | g – вектор початкових наближень параметрів $a, b, c$ |
| Логістична функція $y = a/(1 + be^{cx})$ | ***igsfit(x,y,g)*** | g – вектор початкових наближень параметрів $a, b, c$ |
| Степенева функція $y = ax^b + c$ | ***pwfit(x,y,g)*** | g – вектор початкових наближень параметрів $a, b, c$ |

*Закінчення табл. 3.1*

| Тип згладжування | Функція | Додаткові параметри |
|---|---|---|
| Логарифмічна функція $y = a\ln(x + b) + c$ | ***logfit (x,y, g)*** | g – вектор початкових наближень параметрів $a, b, c$ |
| 2-во параметрична лог. функція $y = a\ln(x) + b$ | ***infit(x,y)*** | |
| Синусоїда $y = a\sin(x + b) + c$ | ***sinfit (x,y,g)*** | g – вектор початкових наближень параметрів $a, b, c$ |
| комбінацією основних функцій | ***genfit (x,y,g,F)*** | g – вектор початкових наближень параметрів, F – вектор функцій користувача |

*Таблиця 3.2*

**Функції МНК та побудови графіка в Matlab**

| Функція | Параметри | Призначення |
|---|---|---|
| ***p=polyfit(x,y,n)*** | ***x,y*** – задані точки функції<br>***n*** – степінь полінома | Повертає коефіцієнти полінома |
| ***y= polyval(p,x)*** | ***p*** – коефіцієнти полінома<br>***x*** – точки, у яких необхідно знайти значення полінома | Повертає значення полінома, заданого своїми коефіцієнтами |
| x= | починаючи з точки x0, | Повертає вектор |

| | | |
|---|---|---|
| lsqnonlin(fun,x0,[lb,ub]) | знаходить мінімум суми квадратів функцій, описаних у fun. Функція fun має повертати вектор значень, а не суму квадратів значень,<br>lb, ub визначає нижні і верхні межі змінних x, тому рішення завжди знаходиться в діапазоні lb ≤ x ≤ ub. Можна зафіксувати компонент рішення x(i), вказавши lb(i) = ub(i) | коефіцієнтів заданої функції |

***Графічне представлення даних***

В середовищі Matlab передбачена велика бібліотека різноманітних функцій для побудови 2-D та 3-D графіків різного типу. Графіки виводяться в окремому вікні figer, яке є незалежним додатком, у якому передбачені широкі можливості роботи з графіками: збереження у різних форматах, налаштування системи координат, налаштування та редагування графіків та багато інших можливостей.

В одному графічному вікні можна розмістити декілька графіків:

– в одній координатній системі для утримання / відключення графічного вікна застосовується команда hold on / off;

– за допомогою команди subplot(i, j, n) графічне вікно можна розбити на декілька, де і та j – кількість вікон по вертикалі та горизонталі, а n – номер вікна (відраховується від лівого верхнього кута за рядками).

Графіки можна побудувати декількома способами:

– безпосередньо з робочого вікна Workspace. Для цього необхідно вибрати дані, які будуть виводитись на графік, натиснувши правою кнопкою миші, відкрити випадаюче меню, з якого вибрати необхідний тип графіка, з’явиться вікно figure з відповідним графіком. Необхідно брати до уваги, що в цьому випадку вважається, що вибрані значення функції розташовані рівномірно;

– за допомогою спеціальної функції. Для побудови поточкового 2-D графіка застосовується функція:

plot(X1,Y1,[LineSpec1],...,[Xn,Yn,LineSpecn]),

де $X_i$, $Y_i$ – відповідно значення аргумента і функції, LineSpec – параметри типу рядок для налаштування типу лінії: стиль та колір лінії, тип маркерів.

У вікні figure передбачена можливість наближення таблично заданої функції поліномами різного порядку та сплайн-апроксимації. Для цього в головному меню вікна figure необхідно вибрати пункт Tools – Basic Fitting, відкриється вікно наближення, в якому задаються дані та спосіб наближення. Якщо позначити пункт Show equations, то відобразиться рівняння відповідної функції (рис. 3.1).

Якщо натиснути праву нижню жирну стрілочку, то відобразяться коефіцієнти полінома та похибка наближення.

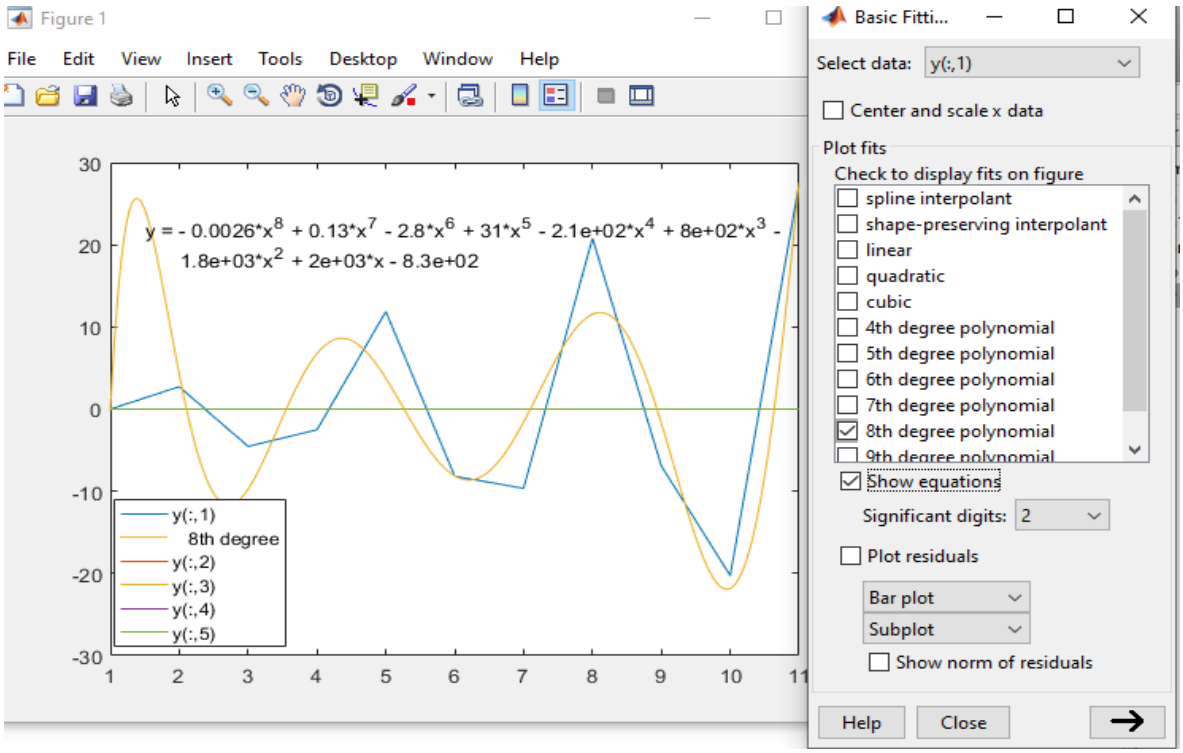



Рис. 3.1. Наближення таблично заданої функції поліномом

***Порядок виконання***

Використовуючи результати, отримані в попередній роботі, сформуємо таблицю залежності між вхідними та вихідними даними *xdata* та *yvs* та нанесемо їх на графік для того, щоб за розташуванням точок *yvs* визначити характер функції, яка лежить в основі нашої

таблично заданої функції. Оскільки значення *xdata* задано з постійним кроком, рівним 1, то графік можна побудувати безпосередньо з вікна ***Workspace***. Для цього натиснемо правою кнопкою миші на змінну $yvs$, у меню, що відкриється, оберемо plot(yvs). На екрані з'явиться вікно ***figure,*** у якому буде відображений графік розташувань точок $yvs$, з'єднаних ламаною лінією. Використовуючи інструменти пункту меню ***Edit***, нанесемо сітку та позначимо точки. З графіка видно, що ми маємо справу з періодичною функцією, причому при $xdata = 0$, $yvs = 0$, отже, в основу покладений синусоїдальний закон, при цьому амплітуда зростає пропорційно зростанню значень *xdata*, зсув за фазою не спостерігається. З цього можна зробити припущення, що загальний вигляд функції такий: $y(x) = a \cdot x \cdot \sin(b \cdot x)$ (рис. 3.2).

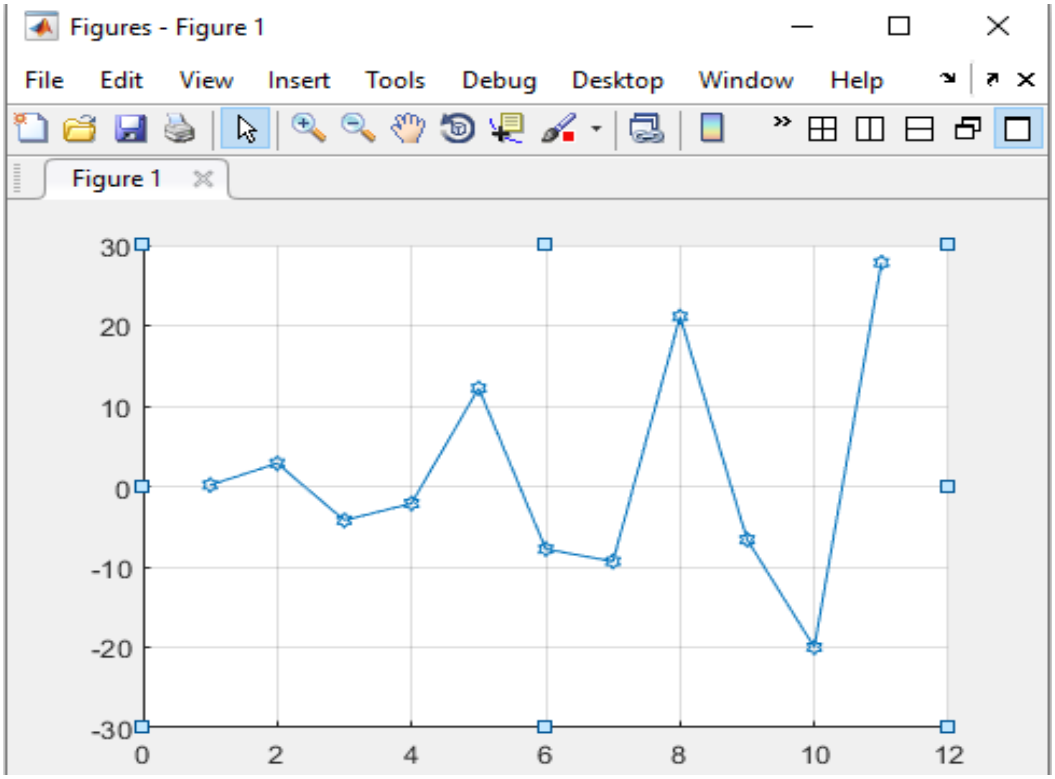



Рис. 3.2. Графік таблично заданої функції

***Зауваження.*** Для визначення характеру закону, який лежить в основі залежності між вхідними та вихідними даними, їх наносять на графік і на підставі елементарних функцій та їх композицій висувають гіпотезу про вигляд закону. В якості додаткової перевірки можна скористатись середньоарифметичними та середньогеометричними значеннями крайніх точок таблиці (таблиця наведена у методичних

вказівках (частина 1) до практичних занять з дисципліни «Чисельні методи» (метод найменших квадратів)).

Щоб наблизити таку періодичну функцію поліномом, необхідно взяти поліном високої степені, що може дати велику похибку за рахунок обчислень та поліноміального «розгойдування». Для апроксимації краще за основу взяти визначену базову функцію і, в цьому випадку, для реалізації методу найменших квадратів застосувати стандартну функцію *lsqnonlin*(). Для цього опишемо функцію, що представляє різницю значень між теоретичними значеннями, які визначаються нашою базовою функцією, та емпіричними значеннями $yvs$ і створимо на неї вказівник. Шукані коефіцієнти $a$ та $b$ опишемо відповідно як $x(1)$, $x(2)$. Задамо їх нижню та верхню границі, звернемось до функції *lsqnonlin*().

```
>> xdata=0:10;
>> y=@(x)x(1).*xdata.*sin(x(2)*xdata)-yvs';
>> x=lsqnonlin(y,[2,3],[2,1.5],[3.5,2.2])
```

Отримаємо значення $x(1)=3.2$, $x(2)=2.004$.

Для знаходження коефіцієнтів функції методом найменших квадратів в середовищі Mathcad скористаємось стандартною функцією з бібліотеки «Апроксимація та згладжування» для апроксимації комбінацією елементарних функцій ***genfit (x,y,g,F)***,. Сформуємо вектор *F*, задамо початкове наближення шуканих параметрів (рис. 3.3).

$$F(a,b,X):=\begin{pmatrix} a\cdot X\cdot\sin(b\cdot X) \\ X\cdot\sin(b\cdot X) \\ a\cdot X^2\cdot\cos(b\cdot X)\end{pmatrix} \quad g:=\begin{pmatrix}3.2\\2\end{pmatrix} \quad c:=\text{genfit}(x,y,g,F)=\begin{pmatrix}3.216\\2.001\end{pmatrix}$$

Рис. 3.3. Наближення таблично заданої функції в середовищі Mathcad

Наступним кроком є нанесення результатів експерименту на графік у вигляді точок і побудова графіка функції, що отримана в результаті моделювання. Для кращого представлення отриманої математичної моделі обчислимо її значення з кроком 0,1:

```
xd=0:0.1:10;
y=x(1).*xd.*sin(x(2)*xd);
```

Для побудови графіка застосуємо функцію plot(), нанесемо сітку;

```
plot(xdata,ydata,xd,y)
grid
```

Засобами графічного вікна (пункти головного меню Edit та Insert) позначимо точки маркерами і вставимо легенду (рис. 3.4.).

З графіків видна різниця між експериментальними та теоретичними значеннями. Для оцінки точності знаходимо середньоквадратичне відхилення між експериментальними значеннями та значеннями, які отримані з моделі як середня відстань між експериментальними значеннями функції і значеннями, отриманими з моделі:

$$\varepsilon = \frac{\sqrt{\sum_{i=1}^{n}\left(y_i - F(x_i)\right)^2}}{n}$$

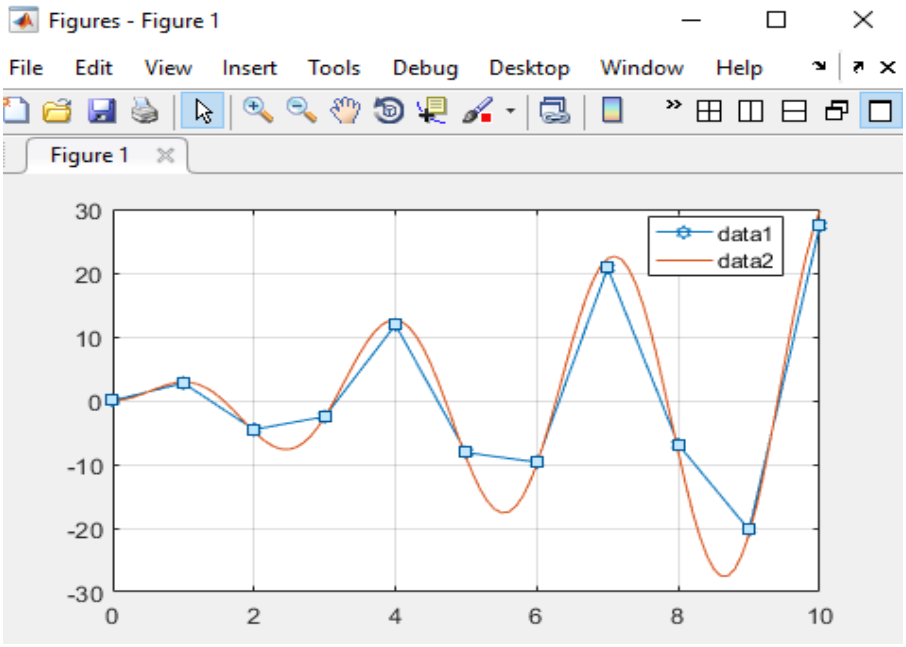



Рис. 3.4. Графічне представлення експериментальних та модельних даних

```
xdata=0:10;
y=x(1).*xdata.*sin(x(2)*xdata);
eps=sqrt(sum(ydata-y').^2)/11
eps= 0.1251
```

Операція транспонування застосована для приведення обох векторів до одного типу.

Якщо отримані результати не задовольняють вимогам до задачі, то у вигляд модель вносяться зміни і будується нова модель.

## Лабораторна робота №4

## Тема: Моделювання зовнішніх впливів

***Мета:*** ознайомити студентів та надати практичних навичок створення власних функцій і скриптів та моделювання зовнішніх впливів та їх врахування в математичній та імітаційній моделі.

***Завдання для самостійного виконання***

На основі даних, наведених у роботі №2, та математичної моделі, отриманої в роботі №3, дослідити вплив зовнішніх факторів. Засобами Matlab необхідно:

- генерувати випадковий шум з математичним сподіванням $M = 0$ та дослідити його вплив на корисний сигнал, що визначається моделлю за різних значень амплітуди (нерівність математичного сподівання нулю породжує систематичну похибку);
- оцінити стійкість та область адекватності моделі відносно впливу випадкового шуму.

***Теоретичні відомості***

*Моделювання випадкового сигналу*

В середовище Matlab вбудовані стандартні функції для генерації випадкових чисел за заданим законом розподілу. Розглянемо основні.

1. Моделювання випадкового сигналу з рівномірним законом розподілу:

– X = rand – повертає одне рівномірно розподілене випадкове число на інтервалі (0,1);

– X = rand(n) – повертає матрицю розмірності $n \times n$ рівномірно розподілених випадкових чисел на інтервалі (0,1).

Моделювання $n$ рівномірно розподілених випадкових чисел на інтервалі $(a,b)$:

```
n=100;
a=2;  b=8;
r = a + (b-
a).*rand(n,1);
```

Моделювання $n$ рівномірно розподілених випадкових чисел із заданим математичним сподіванням $mu$ та дисперсією $D$:

```
r = mu + sqrt(D).*rand(n,1);
```

2. Моделювання випадкового сигналу з нормальним законом розподілу

– X = normrnd(mu,sigma) – генерує нормальні випадкові числа із середнім *mu* та стандартним відхиленням *sigma*. *mu* та *sigma* можуть бути векторами, матрицями або багатовимірними масивами, які мають однаковий розмір, який також є розміром X.

У пакеті MatLab є ще одна корисна функція, яка дозволяє додавати до сигналу білий шум із заданим рівнем:

X = awgn(x, snr, sigpower, state, ‘powertype’), де ***x*** – вектор відліку сигналу, *snr* – скаляр, що задає співвідношення сигнал/шум в одиницях, що визначаються параметром *powertype* (за замовчуванням – в децибелах), *sigpower* – потужність сигналу, *state* – примусове встановлення генератора випадкових чисел (останні три параметри є обов’язковими).

*Робота в режимі компілятора*

Для створення режиму компілятора власних підпрограм функцій та скриптів необхідно відкрити пункт головного меню New і обрати необхідну опцію.

Для створення функцій відкривається вікно (рис. 4.1.), де треба задати необхідні параметри і написати текст функції. Функція зберігається на диску у файлі з іменем, яке відповідає імені функції. Як і в інших мовах програмування, в тілі функції імені функції повинно присвоюватись значення.

*Створення script-файлів та власних функцій*

В середовищі Matlab передбачена можливість створювати додатки в режимі компілятора і зберігати їх на диску у М-файлах. Для великих завдань, а також для завдань, які потрібно запускати на виконання кілька разів, зручніше зберегти послідовність дій у так званих скриптах (script) та підпрограмах-функціях. Скрипти є звичайними текстовими файлами, які можна створювати та редагувати в будь-якому текстовому редакторі або безпосередньо в середовищі MATLAB. Запустити *script-файл* можна у командному вікні за його іменем. З файлами функціями можна працювати так, як і в інших мовах програмування.

Для створення script-файлів та функцій у головному меню необхідно вибрати пункт New, у випадаючому вікні вибрати необхідний тип додатка. Відкриється відповідне вікно, в якому записується програмний код. Script може представляти будь-яку частину програми, функції ж мають свої правила запису і структуру (рис. 4.1).

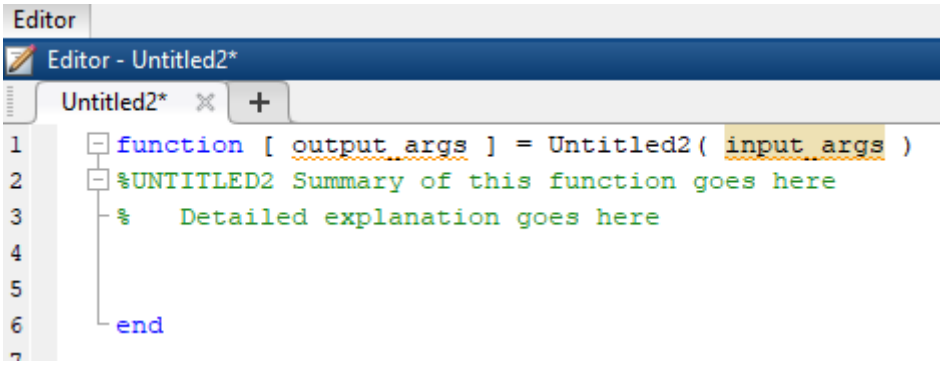



Рис. 4.1. Вікно для створення функції

Функція починається із заголовку:

function [ output_args ] = Untitled2( input_args ),

– [ output_args ] – ім'я вихідного параметру (якщо вихідних параметрів декілька і вони однорідні, то їх можна об'єднувати у масив);

– Untitled – ім'я функції (ім'я файлу на диску, де зберігається функція, повинно точно співпадати з ім'ям функції;

– input_args – список вхідних параметрів.

Після заголовку функції можна вставляти коментарі, але це не обов'язково. Якщо між заголовком і коментарем немає пустого рядка, то цей коментар можна вивести у командному вікні за допомогою команди help, що дає змогу, наприклад, швидко ознайомитись з призначенням функції. Якщо коментар відокремлений від заголовку пустим рядком, то він є внутрішнім.

Між заголовком і командою end записується тіло функції. Необхідно пам'ятати, що вихідним параметрам обов'язково повинно присвоюватись значення.

В Matlab передбачена можливість побудови графіка за описом функції:

fplot(h_fun, [xinterval,] [Linespace]),

де h_fun – вказівник на функцію, xinterval = [xmin xmax] – відрізок, на якому будується функція (замовчуванням [-5 5]), Linespace – символьний парамеметр налаштування вигляду графіка.

Наприклад:

fp = fplot(@(x) sin(x)) – побудова графіка функції синус.

Якщо функція зберігається в М-файлі, то необхідно створити на неї вказівник:

h_myfun = @my_fun;

fplot(h_myfun)

*Оператор умови та створення випадаючого меню*

Для створення логічних виразів використовують операції відношення та логічні операції (табл. 4.1).

В Matlab існує команда для створення випадаючого меню:

choice = menu(‘заголовок’,’пункт_1’,’пункт_2’,...,’пункт_n’),

яка повертає значення choice – кількість пунктів меню.

***Порядок виконання***

Створимо підпрограми функції для опису отриманої математичної моделі (рис. 4.2, 4.3) та моделі, зашумленої випадковим сигналом.

*Таблиця 4.1*

| ***Операції відношення*** | | | |
|---|---|---|---|
| < | менше | > | більше |
| <= | менше або дорівнює | >= | більше або дорівнює |
| == | дорівнює | ~= | не дорівнює |
| ***Логічні операції*** | | | |
| &, and | і | | |
| \|, or | або | | |
| xor | виключаючи або | | |
| ~, not | заперечення | | |
| any() | істино, якщо всі елементи вектора дорівнюють нулю | | |
| all() | істино, якщо всі елементи вектора не дорівнюють нулю | | |

Для реалізації розгалуженого процесу застосовують оператори if або switch:

| Оператор if: | Оператор switch: |
|---|---|
| if умова<br>дії<br>[else / elseif<br>дії]<br>….<br>end | switch вираз<br>case значення_1<br>оператори_1<br>case значення_2<br>оператори_2<br>...<br>otherwise<br>оператори_N<br>end |

Editor - D:\Моделирование\практика\my_fun3.m

my_fun3.m

```
function y = my_fun3( x )
% опис математичної моделі y=2.5.*X.*sin(2*X)

%   лабораторна робота №3
y=2.5.*x.*sin(2*x);
end
```

Рис. 4.2. Опис математичної моделі

Editor - D:\Моделирование\практика\YRnoise.m

Lab_3.m | my_fun3.m | YRnoise.m

```
function Yl = YRnoise( x,a )
% зашумлення випадковим шумом

%  лаб №3
% a - амплітуда шуму
 n=length(x); % визначення довжини вектора x
r=randn(n); R=r(1,:);% генерування вектора випадкових чисел
        Yl=my_fun3(x)+a.*R;

end
```

Рис. 4.3. Зашумлення випадковим сигналом

Вплив випадкового шуму будемо досліджувати за різних значень амплітуди, наприклад, коли $a = 1, 2, 3.$ Знайдемо відхилення та середнє відхилення між значеннями, отриманими з моделі та зашумленого сигналу. Для цього створимо script-файл (рис. 4.4).

Editor - D:\Моделирование\практика\YA_noise.m

YR_graph.m | YA_noise.m | YA_noise.asv

```
x=0:10;
y=my_fun3(x);    % значення функції
yl=YRnoise(x,1);% зашумлене значення a=1
y2=YRnoise(x,2);% зашумлене значення a=2
y3=YRnoise(x,3);% зашумлене значення a=3
dl=abs(y-yl);    % відхилення
d2=abs(y-y2);
d3=abs(y-y3);
disp([y' yl' y2' y3' dl' d2' d3'])
% середнє відхилення
ds1=sum(dl)/11; ds2=sum(d2)/11; ds3=sum(d3)/11;
disp('Середні відхилення = '), disp([ds1 ds2 ds3])
```

Рис. 4.4. Script-файл

Команда disp( ) виводить результати на екран, але треба брати до уваги, що вивести можна значення лише однієї змінної. Якщо необхідно вивести декілька значень і вони одного типу, то їх можна об’єднати у вектор.

За отриманими зашумленими значеннями знайдемо коефіцієнти для нашої моделі (рис. 4.5).

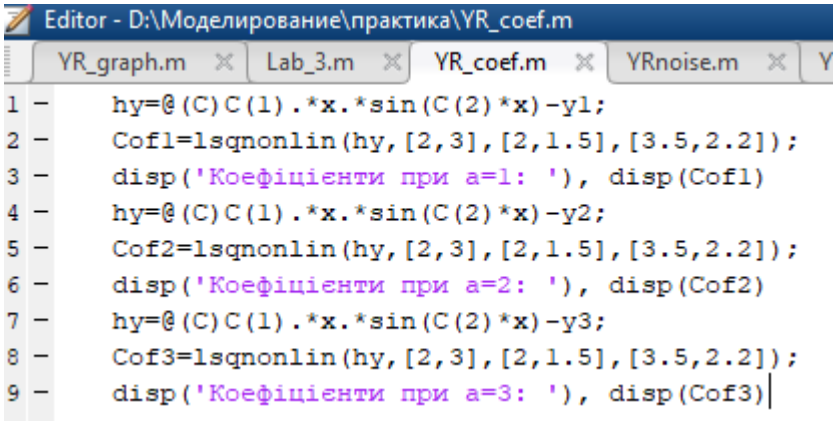



Рис. 4.5. Знаходження параметрів моделі при зашумленому сигналі

Отримаємо наступні значення:

– коефіцієнти при a=1: 2.4190  2.0043;

– коефіцієнти при a=2: 2.6246  2.0020;

– коефіцієнти при a=3: 2.4347  1.9989.

Порівняємо їх з отриманими з результатів вимірювань, щоб визначити наскільки вплинув заданий випадковий шум на систему. Видно, що вплив випадкового шуму суттєво не впливає на сигнал системи.

Для побудови гладкого графіка зменшимо крок параметру x=0:0.1:10, обчислимо значення моделі та зашумленого сигналу (рис.4.6).

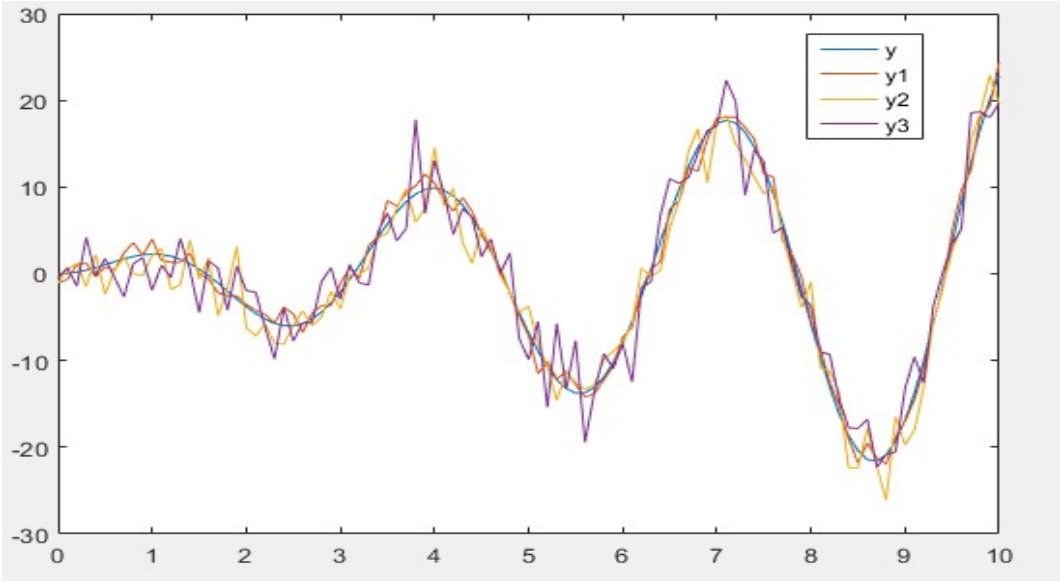



Рис. 4.6. Графіки сигналу та зашумленого сигналу випадковим шумом

З графіків видно, що випадковий шум має більший вплив в початковий період роботи системи (що визначає наявність перехідного процесу) і при збільшенні аргументу $x$ вплив зменшується.

Відповідний script-файл наведено на рис. 4.7.

Editor - D:\Моделирование\практика\YR_gra

YR_graph.m

```
1 -    x=0:0.1:10;
2 -    y=my_fun3(x);
3 -    y1=YRnoise(x,1);
4 -    y2=YRnoise(x,2);
5 -    y3=YRnoise(x,3);
6 -    plot(x,y,x,y1,x,y2,x,y3)
```

Рис. 4.7. Script-файл для побудови графіка

Стійкість та область адекватності отриманої моделі будемо досліджувати, імітуючи зовнішні впливи у вигляді адитивного шуму.

Для спрощення демонстрації роботи окремих кроків завдання їх можна об’єднати в один script-файл на основі випадаючого меню (рис. 4.8).

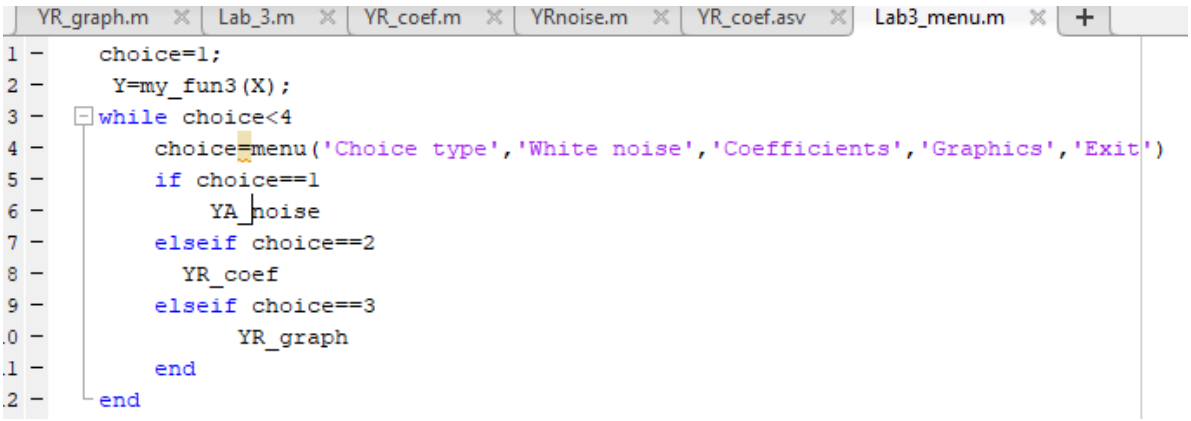



Рис. 4.8. Script-файл меню

## Лабораторна робота №5

## Тема: Створення графічного інтерфейсу імітаційної моделі

***Мета:*** ознайомити студентів та надати практичних навичок створення графічного інтерфейсу на прикладі моделювання зовнішніх впливів та їх врахування в математичній та імітаційній моделі.

***Завдання для самостійного виконання***

На основі даних, наведених у роботі №2, та математичної моделі, отриманої в роботі №3, дослідити вплив зовнішніх факторів. Засобами Matlab необхідно:

- генерувати періодичний шум та дослідити його вплив на корисний сигнал (варіант 1 – прямокутний, варіант 2 – трапецевидний, варіант 3 – Діріхле, варіант 4 – пилковидний, варіант 5 – трикутний). Для періодичного шуму провести дослідження за різних значень амплітуди, ширини сигналу та зсуву за фазою. Скласти план експерименту;
- оцінити стійкість та область адекватності моделі відносно впливу періодичного шуму.

***Теоретичні відомості***

***Моделювання періодичного сигналу***

Генерувати періодичні сигнали різної форми можна за допомогою вбудованих функцій середовища MatLab (бібліотека Signal Processing Toolbox; табл. 5.1).

*Таблиця 5.1*

**Функції моделювання періодичних сигналів**

| Функція | Призначення |
|---|---|
| Прямокутні імпульси<br>x = square(t,[ duty]) | Генерує прямокутну хвилю з періодом 2π для елементів вектора часу *t* з піками ±1. Параметр *duty* генерує прямокутну хвилю із заданим циклом заповнення, duty, яке є числом від 0 до 100. Цикл заповнення – це відсоток періоду, протягом якого сигнал позитивний |
| x = rectpuls(t,[w]) | Повертає неперервний аперіодичний прямокутний імпульс одиничної висоти в момент часу t, з центром в точці t = 0 і шириною, w = 1. Зауваження: інтервал ненульової амплітуди визначається як відкритий праворуч, тобто rectpuls(-0,5) = 1, а rectpuls(0,5) = 0 |

*Закінчення табл. 5.1*

| | |
|---|---|
| Трикутні імпульси<br>x=tripuls(T,[w,[s]]) | Повертає неперервний, аперіодичний, симетричний трикутний імпульс одиничної висоти в момент часу, зазначений у масиві *T*, з центром в точці *T*=0 і шириною 1. Параметр w визначає ширину імпульсу, а *s* $(-1 \le s \le 1)$ – перекос, при $s = 0$ генерується симетричний трикутний імпульс |
| Пилковидні імпульси<br>x=sawtooth(t,[width]) | Генерує пилкоподібну хвилю з періодом 2π для елементів вектору часу *t* з піками –1 і 1. Пилкоподібна хвиля визначається як –1 при кратності 2π і лінійно зростає з часом із нахилом 1/π в усі інші моменти часу. *Width* – ширина, скалярний параметр від 0 до 1, визначає точку між 0 і 2π, у якій виникає максимум. Функція зростає від –1 до 1 на інтервалі від 0 до 2π × *width*, потім лінійно спадає. Таким чином, параметр 0,5 визначає стандартну трикутну хвилю, симетричну відносно моменту часу π з амплітудою 1 |
| Функція Діріхле<br>X = diric(x,n) | Функція Діріхле може розглядатись як сигнал, n має бути додатним цілим числом |
| Гаусівські імпульси<br>yi = gauspuls(t,fc,bw) | Генерує синусоїдальні імпульси, модульовані за Гаусом з одиничною амплітудою в момент часу, зазначений у масиві t, із центральною частотою fc у герцах і частковою смугою пропускання bw > 0. Значення за замовчуванням для fc становить 1000 Гц, а для bw – 0,5 |

**Приклад 1**. Моделювання прямокутних сигналів.

А. Сформуємо прямокутний сигнал (рис. 5.1) з відношенням періоду до довжини імпульсу 50% на інтервалі часу від 0 до 20 с. з амплітудою 0,7:

```
t=0:0.1:20;
y=0.7.*square(t,50);
plot(t,y)
```

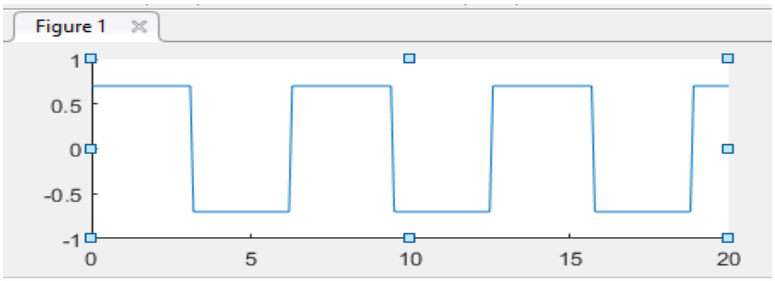



Рис. 5.1. Прямокутний сигнал

Б. Створимо прямокутну хвилю 30 Гц із частотою дискретизації 1 кГц протягом 70 мс., робочий цикл 37%. Додамо білий шум Гаусса з дисперсією 1/100 (рис. 5.2):

```
t = 0:1/1e3:0.07;
y = square(2*pi*30*t,37)+randn(size(t))/10;
plot(t,y)
```

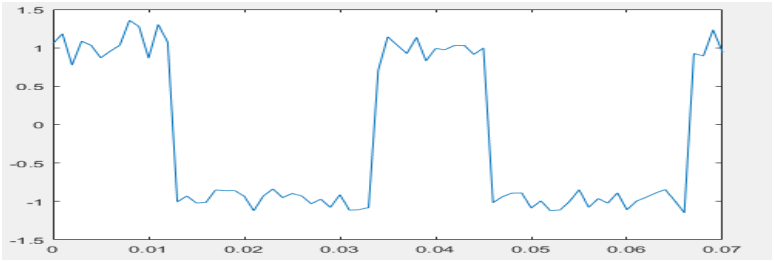



Рис. 5.2. Прямокутний сигнал з випадковим шумом

**Приклад 2.** Трикутні імпульси

А. Генерація 200 мс симетричного трикутного імпульсу з частотою дискретизації 10 кГц і шириною 40 мс (рис. 5.3):

```
fs = 10e3; w = 40e-3;
t = -0.1:1/fs:0.1;
x = tripuls(t,w);
plot(t,x)
```

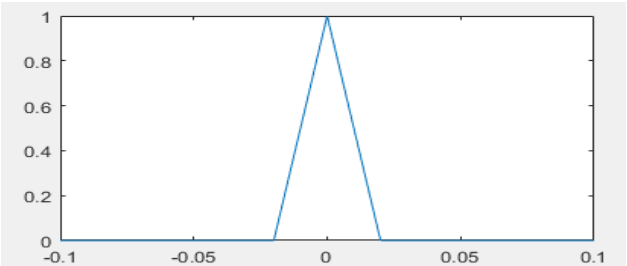


Рис. 5.3. Трикутний імпульс

Б. Сформуємо симетричний трапецієвидний імпульс з амплітудою 10 і розмірами верхньої і нижньої основи 20 і 60 мс відповідно. Частота дискретизації рівна 1 кГц (рис. 5.4):

```
fs = 10e3; A=10;
t = -50e-3:1/fs: 50e-3;
T1 = 20e-3; T2 = 60e-3;
x = A*(T2* tripuls(t,T2)- T1* tripuls(t,T1))/(T2-T1);
plot(t,x)
```

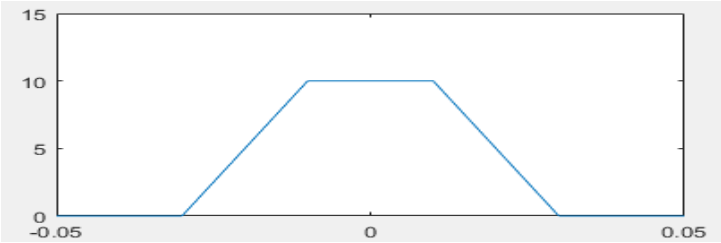



Рис. 5.4. Трапецієвидний сигнал

**Приклад 3.** Пилковидні імпульси.

Сформуємо трикутний сигнал на інтервалі часу від 0 до 20 с. Дослідимо вплив параметра *width* на форму сигналу (рис. 5.5):

```
t = 0:0.1:20; width = 0 [0.5] [1];
x = sawtooth(t, width);
plot(t,x)
```

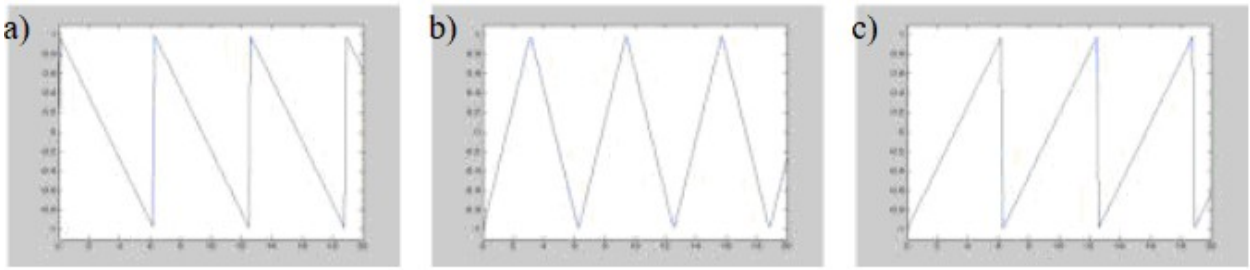



Рис. 5.5. Пилкоподібний сигнал: a) width = 0, b) width = 0.5; c) width = 1

**Приклад 4.** Моделювання сигналу на основі функції Діріхле для різних значень параметра *n* (рис. 5.6):

```
x = linspace(-2*pi,2*pi,301);
d7 = diric(x,7);   d8 = diric(x,8);
subplot(2,1,1)
plot(x/pi,d7), ylabel('N = 7'), title('Dirichlet Function')
subplot(2,1,2)
plot(x/pi,d8), ylabel('N = 8'), xlabel('x / \pi')
```

Для створення відліків імпульсних сигналів різної форми служить функція: x = pulstran(t,d,'func'[,p1,p2,...]). Форма задається параметром func, який може мати значення:

•gauspuls – синусоїда, модульована за законом Гауса ;

•rectpuls – прямокутний імпульс;

•tripuls – трикутний імпульс.

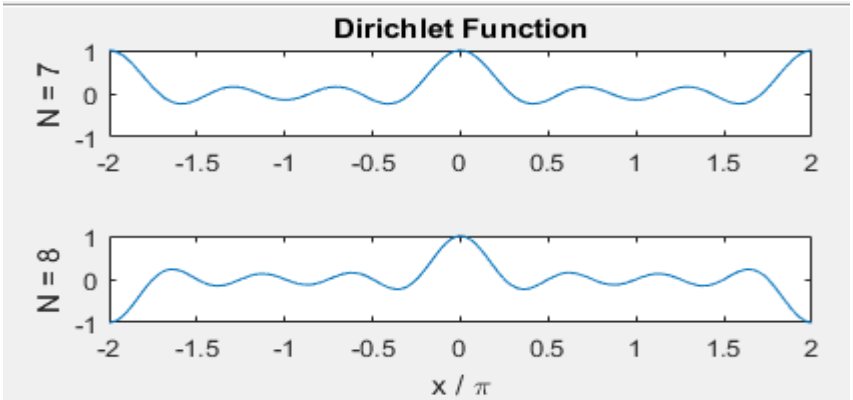



Рис. 5.6. Моделювання сигналу на основі функції Діріхле

Число імпульсів у заданому інтервалі часу задається довжиною вектора d. Необов'язкові параметри p1,p2,… дозволяють задати додаткові параметри звернення до 'func', наприклад, типу func(t-d(1), p1,p2,…). Під час запису функцій у вигляді y=pulstran(t,d,p,[fs]) можна задати частоту дискретизації fs (за замовчуванням 1 Гц).

**Приклад 5.** Побудова трикутного імпульсного сигналу за допомогою функції pulstran (рис. 5.7):

```
t = 0 : 1/1e3 : 1;        % частота 1 кГц протягом 1 с
d = 0 : 1/3 : 1;          % частота повторення 3 Гц
y = pulstran (t,d,'tripuls',0.1,-1);
plot(t,y)
xlabel 'Time (s)', ylabel Waveform
```

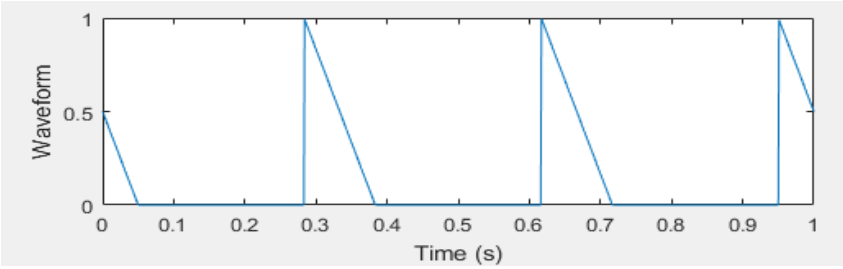



Рис. 5.7. Імпульсний трикутний сигнал

***Graphic User Interface (GUI)***

Повний перелік функцій GUI середовища Matlab можна вивчити за допомогою команди help uitools.

Для створення GUI зручно користуватися вбудованим редактором, який викликається з основного меню: New – App – GUIDE. Відкривається вікно, в якому необхідно вибрати необхідний тип інтерфейсу (рис. 5.8).

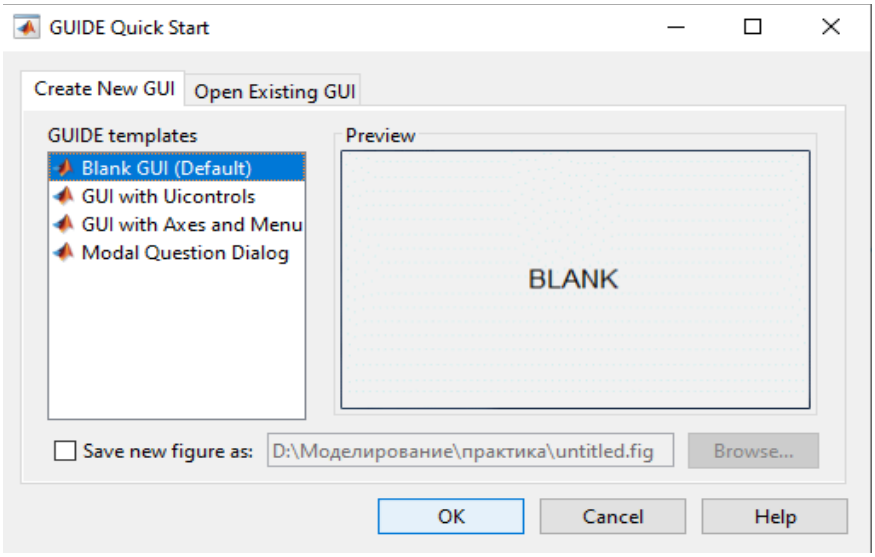



Рис. 5.8. Вікно вибору бланку для створення графічного інтерфейсу

Загальному конструктору графічного інтерфейсу відповідає Blank CUI (Default) (рис. 5.9).

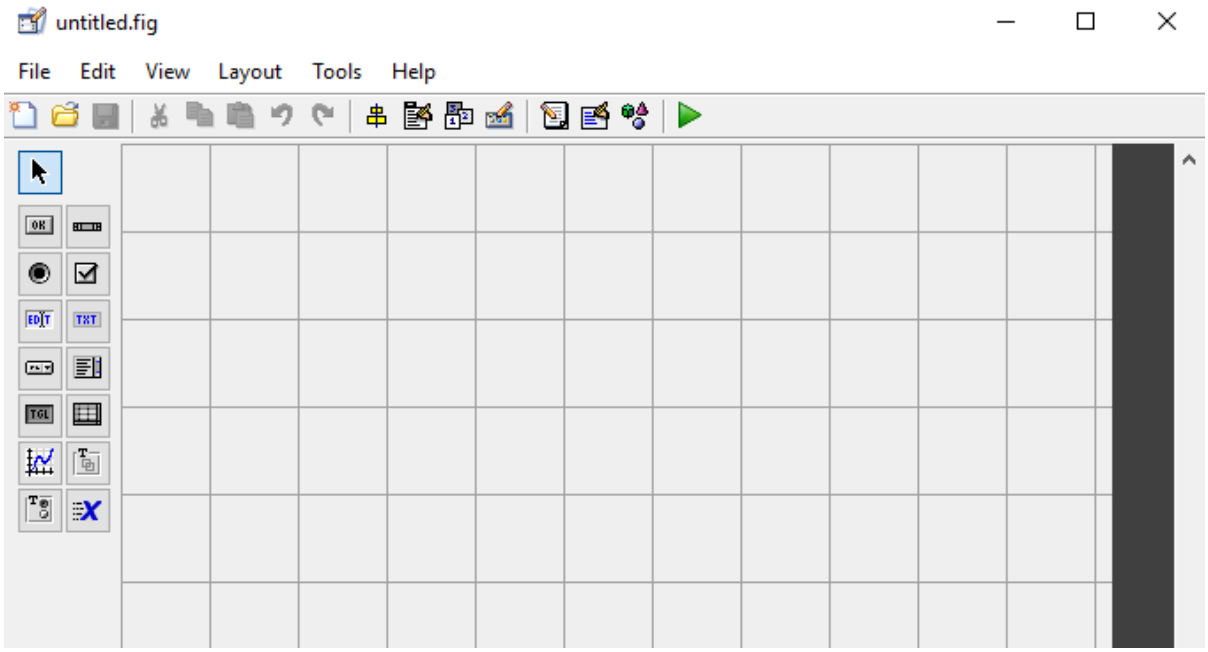



Рис. 5.9. Вікно конструктора графічного інтерфейсу

Конструктор інтерфейсу користувача містить:

• рядок меню;

• панель інструментів керування програмою;

• заготовку вікна програми з нанесеною сіткою;

• вертикальну та горизонтальну лінійки;

• панель інструментів для додавання елементів інтерфейсу до вікна програми.

Дизайн інтерфейсу компонується з елементів панелі інструментів шляхом їх перетягування у вікно заготовки програми. Вікно інспектора налаштування параметрів елементу відкривається подвійним кліком. Для налаштування методів простіше, клікнувши правою кнопкою миші на елемент, вибрати з випадаючого меню пункт View Callbacks, де знаходиться список методів.

В поточний момент програма знаходиться в режимі редагування. Будь-який об’єкт можна видалити з вікна за допомогою кнопки Delete, попередньо виділивши його. Запуск програми здійснюється за допомогою кнопки Run або вибором пункту меню Tools. Можлива поява діалогового вікна середовища із пропозицією зберегти проєкт. Натисніть Yes та збережіть у файлі з іменем, наприклад, mygui.

Графічний додаток в MATLAB зберігається у двох файлах з розширеннями .fig і .m, перший містить опис форми вікна, а другий – код програми. Додавання елемента інтерфейсу в редакторі призводить

до автоматичної генерації шаблону обробника події (наприклад, обробка кліка на кнопці) у файлі з кодом програми. У функцію активації шаблону function untitled_OpeningFcn після рядка guidata(hObject, handles) можна задати глобальні змінні та встановити параметри елементів за замовчуванням, якщо це не було зроблено під час їх налагодження (рис.5. 10).

```
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled (see VARARGIN)

% Choose default command line output for untitled
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% Опис даних та параметрів
  global x,y
```

Рис. 5.10. Функції активації шаблону

Якщо необхідно, щоб значення змінних були доступними у різних частинах програми, то їх необхідно описати як глобальні у функції активації та в кожній функції, де вони використовуються ключовим словом global.

Далі необхідно написати код для кожної функції методу. Для встановлення та зчитування значень застосовуються функції set( ) та get():

set(handles.тег, ‘Назва властивості’, ‘значення’)

get(handles.тег, ‘Назва властивості’).

Якщо ці функції задаються в середині функції, що стосується даного об’єкта, то вказівник handles.тег можна записувати спрощено hObject.

Функції перетворення типів мають наступний формат: тип2тип( ).

***Порядок виконання***

1. За допомогою GUIDE створити графічний шаблон інтерфейсу. Вигляд шаблону, що пропонується, наведений на (рис. 5.11). Передбачити можливість зміни параметрів шуму, наприклад, амплітуди та частоти. Для цього можна застосувати компонент Slider або компоненти Edit та Txt.

Також передбачити можливість вибирати різний тип шуму, наприклад, за допомогою RadioButton.

2. Описати математичну модель та задати шуми. У вікні axes1 вивести графіки безпосередньо моделі та зашумленої моделі.

3. У заданих точках параметра *x* оцінити та вивести значення середньоквадратичної похибки.

4. За заданими точками параметра *x* та отриманими значеннями *y* знайти методом найменших квадратів параметри моделі (див. практичне завдання 4) і вивести їх.

5. У вікні axes2 вивести графіки безпосередньо моделі та моделі з новими параметрами, отриманими за результатами зашумлення.

6. Зробити висновки.

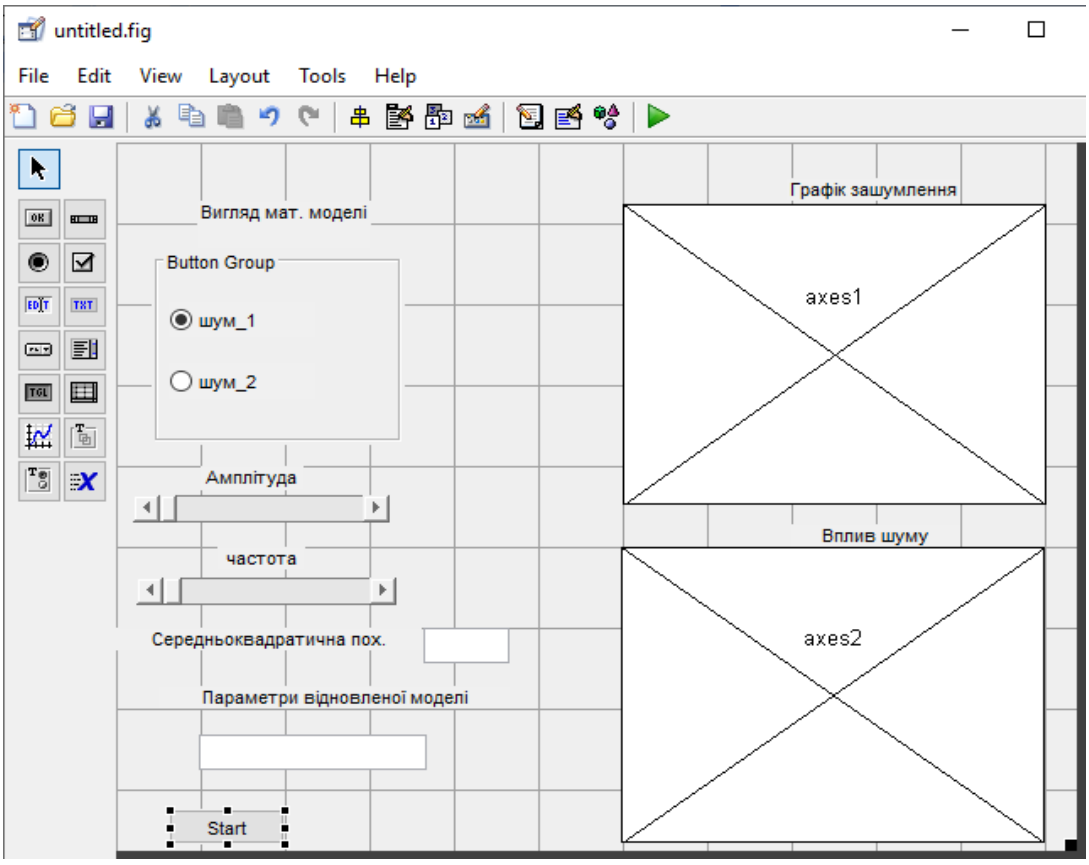



Рис. 5.11. Вигляд шаблону інтерфейсу

## Лабораторна робота № 6

## Тема: Побудови математичних моделей динамічних систем

***Мета:*** ознайомити студентів та надати практичних навичок на основі експериментальних даних за допомогою системи ідентифікації Ident середовища Matlab.

***Завдання для самостійного виконання***

Згенерувати набір вхідних та вихідних даних, що представляють результати вимірювань роботи деякої динамічної системи $y(t)=f(u(t))$. Вхідні дані $u(t)$ та вихідні дані $y(t)$ повинні бути представлені у вигляді векторів стовпчиків.

За допомогою інструментарію бібліотеки System Identification Toolbox побудувати математичні моделі процесу та вибрати модель кращого наближення.

***Теоретичні відомості***

Бібліотека System Identification Toolbox містить функції Matlab, блоки Simulink і додаток для побудови математичних моделей динамічних систем на основі виміряних даних. Це дозволяє створювати та використовувати моделі динамічних систем, які важко змоделювати на основі фізичних законів чи специфікацій. Побудувати моделі можна, застосовуючи команди бібліотеки або за допомогою графічного вікна System Identification. Найпростіший спосіб – за допомогою графічного вікна.

Для завантаження графічного вікна у командному рядку необхідно задати ключове слово ident або systemIdentification. На екрані з'явиться вікно системи ідентифікації (рис. 6.1).

Робота з додатком складається з декількох послідовних кроків:

– побудова та обробка даних (ліва сторона вікна);

– побудова та дослідження моделі (права сторона вікна).

Порядок ідентифікації моделі у графічному вікні позначений стрілочками.

***Введення та обробка даних***

Першим кроком роботи з додатком є введення даних Import data, причому вхідні та вихідні сигнали були представлені в часовій або частотній області. Дані можна вводити різними способами, найпростіший – завантажити з робочої області Workspace, для цього необхідно вибрати Import data – Time domain data. Відкриється діалогове вікно імпорту даних, в якому необхідно вказати вхідні та вихідні дані (рис. 6.2.).

– Input/Output – назва змінної або вираз MATLAB, який представляє вхідні/вихідні дані і визначається, як обчислюватися, як вектор-стовпець або матриця.

– Data name – назва набору даних, який з'явиться у вікні ідентифікації системи після завершення операції імпорту.

– Starting time – початкове значення часу для часових графіків.

– Sample time – фактичний час вибірки в експерименті.

Для задання властивостей даних в області Data Information необхідно натиснути More, щоб розгорнути діалогове вікно (рис. 6.2.), де вказуються параметри Input Properties – властивості введення:

– InterSample – параметр визначає поведінку вхідних сигналів між вибірками: zoh (утримання нульового порядку) – вхідний сигнал був кусково-постійним між вибірками під час збору даних; foh (утримання першого порядку) – вихід був кусково-лінійним під час збору даних; bl (поведінка з обмеженням пропускної здатності) – вхідний сигнал неперервного часу має нульову потужність вище частоти Найквіста (дорівнює оберненому часу вибірки).

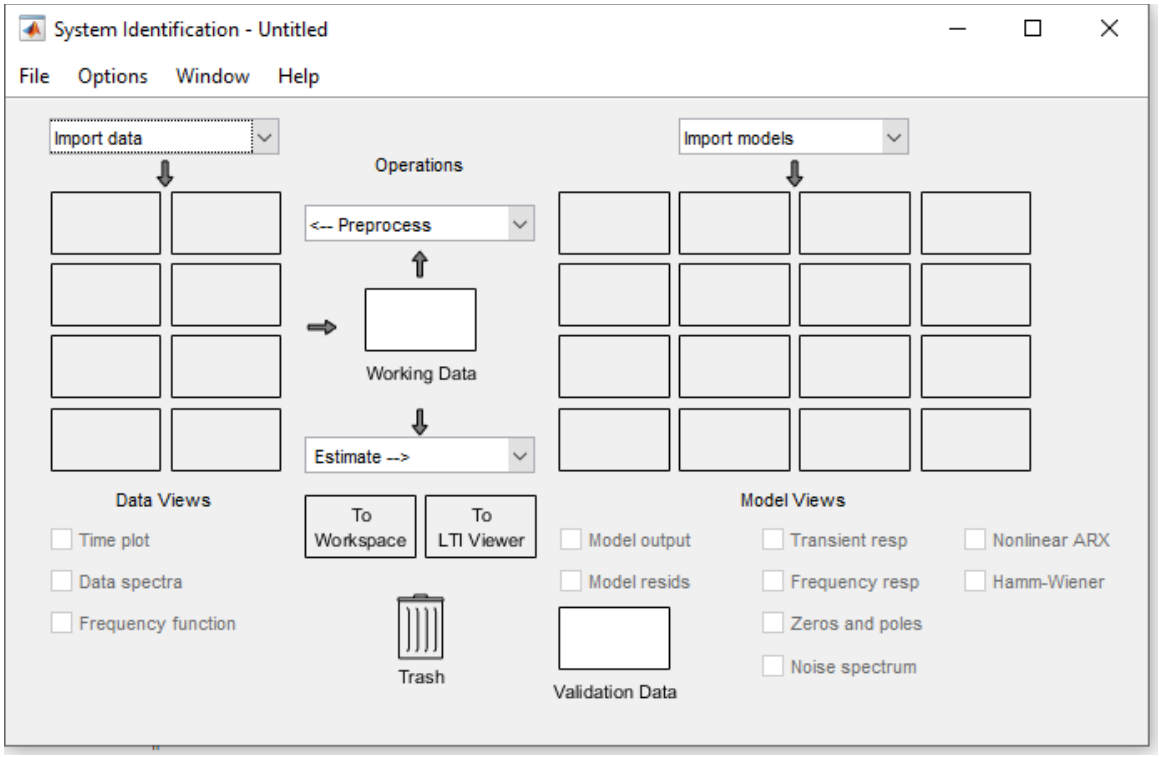



Рис. 6.1. Вікно системи ідентифікації та аналізу моделі

– Period – прийняти inf, щоб указати неперіодичне введення. Для періодичного введення введіть цілу кількість періодів вхідного сигналу в експерименті.

– Channel Names – назви каналів допомагають ідентифікувати дані на графіках.

– Physical Units of Variables – фізичні одиниці змінних.

– Notes – примітки для коментарів щодо експерименту або даних. Наприклад, можна ввести назву експерименту, дату та опис умов експерименту.

Щоб завантажити дані у вікно System Identification, необхідно натиснути Import і закрити вікно імпорту даних.

У прямокутнику даних з'являться дані. Їх можна представити графічно, вибравши пункти з меню Dada Views.

Оскільки експериментальні дані можуть містити похибки, їх можна коригувати за допомогою операцій пункту <--Preprocess, наприклад, відняти середні значення (Remove means) або розділити на частини (Select Range), що додасть новий набір даних.

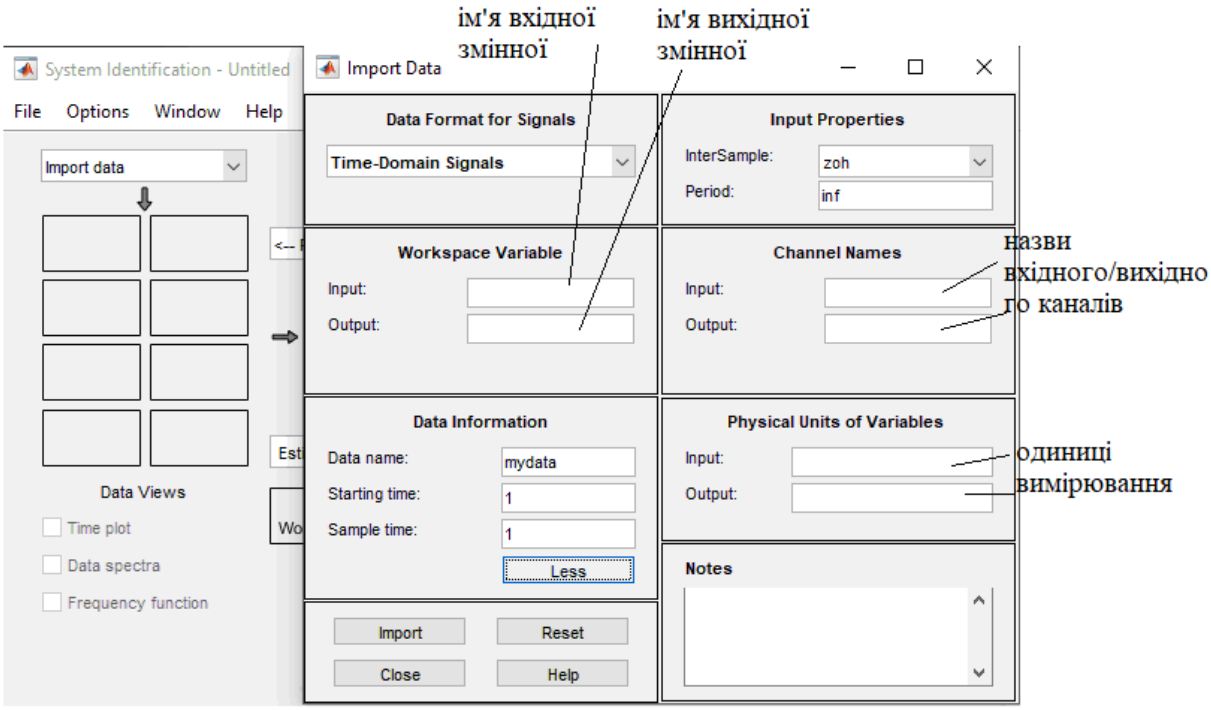



Рис. 6.2. Вікна налаштування вхідних/вихідних даних

Для ідентифікації та перевірки моделі необхідно мати два набори даних, один – для побудови моделі, а другий – для перевірки. Тут є різні варіанти:

– можна будувати модель на основі експериментальних даних, а для перевірки використати кориговані дані;

– можна будувати модель на основі коригованих даних, а для перевірки ввести новий тестовий набір даних;

– можна одну частину вхідних даних використовувати для побудови, а другу – для перевірки.

Дані, які будуть використовуватись для побудови моделі, необхідно перетягти у прямокутник Working Data, а дані для перевірки – у прямокутник Validation Data.

Щоб отримати інформацію про набір даних, клацніть його значок правою кнопкою миші. Наприклад, клацніть правою кнопкою миші data, щоб відкрити діалогове вікно Data/model Info, в якому можна переглянути дані та внести деякі зміни (рис. 6.3):

– назви набору даних у полі Data name;

– кольору значка даних у полі Color. Кольори визначаються як значення RGB (відносна кількість червоного, зеленого та синього). Кожне значення знаходиться в діапазоні від 0 до 1. Наприклад, [1,0,0] вказує, що присутній лише червоний, а зелений і синій не домішуються в загальний колір.

Область Diary and Notes – перегляд або редагування команд містить командний рядок, еквівалентний обробці, виконаній за допомогою програми ідентифікації системи. Наприклад, як показано у вікні Data/model Info: estimate», набір даних datad є результатом усунення тенденції середніх значень:

```
% Import   data
 datad = detrend(data,0)
```

***Ідентифікація моделі***

Перед побудовою моделі дані повинні бути вже оброблені. Процес ідентифікації системи складається з наступних кроків:

– вибір структури моделі;

– застосування методу оцінки для значення регульованих параметрів у структурі моделі-кандидата;

– оцінку передбачуваної моделі для визначення відповідності потребам.

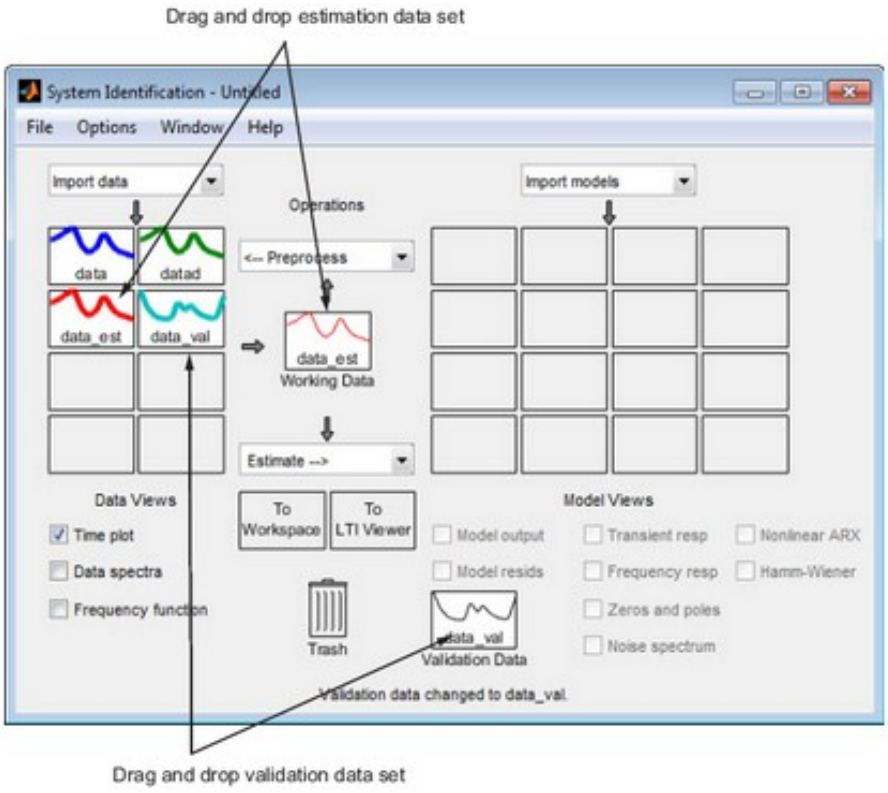



Рис. 6.3.Розділення даних

Спливаюче меню Estimate (Оцінка) в головному вікні містить методи оцінки моделей на основі набору робочих даних:

– Transfer Function Models – моделі передатних функцій, дозволяють оцінити функції передачі за допомогою даних у часовій або частотній області;

– Process Models – моделі процесів, дозволяють генерувати прості динамічні лінійні моделі неперервного часу, що характеризуються статичним посиленням, постійними часом і часовими затримками;

– State Space Models – просторові моделі стану, дозволяють оцінити моделі простору станів у неперервному та дискретному часі;

– Polynomial Models – поліноміальні моделі, дозволяють генерувати поліноміальні моделі даних введення-виведення та часових рядів. Підтримувані структури моделей включають AR (для часових рядів), ARX, ARMAX, Output Error (OE), Box-Jenkins (BJ). Додатково можна вказати поведінку інтеграції на шумі, що призводить до структур моделі типу «ARIMA» та «ARIMAX»;

– Nonlinear models – нелінійні моделі, дозволяють генерувати нелінійні динамічні моделі двох типів: нелінійні моделі ARX і моделі Хаммерштейна-Вінера;

– Refine Existing Models – удосконалення існуючих моделей, дозволяють запускати інтеграції і таким чином оновлювати параметри будь-якої лінійної моделі, яка існує на панелі моделей програми або в базовій робочій області;

– Spectral model – спектральна модель, оцінює частотну характеристику системи на основі даних за допомогою методів перетворення Фур'є або підходу Блекмана-Тьюкі;

– Correlation model – кореляційна модель, оцінює перехідну характеристику системи (імпульсну характеристику) шляхом кореляції відфільтрованих версій даних введення-виведення;

– Quick start – швидкий старт, виконує кореляційний і спектральний аналіз. Обчислює модель ARX за замовчуванням із затримкою, яка визначається евристично на основі оціненої імпульсної характеристики системи, а також модель простору станів порядку за замовчуванням.

Для вибору методу ідентифікації відкривається вікно налаштування, де представлений загальний вигляд моделі і необхідно задати для нього параметри. Основними параметрами є *na* – кількість полюсів, *nb*+1 – кількість нулів, *nk* – чиста затримка часу (мертвий час) у системі. Якщо модем представляється як відношення, то полюсами передаточної функції називають корені характеристичного полінома знаменника, нулі – корені характеристичного полінома чисельника. Для системи управління вибірковими даними зазвичай *nk* = 1, якщо немає мертвого часу.

Розглянемо ідентифікацію за допомогою Polynomial Models, де застосуємо модель ARX – це лінійне різницеве рівняння, яке пов'язує вхідні дані u(t) із вихідними даними y(t) наступним чином:

$$y(t) + a_1 \cdot y(t-1) + ... + a_{n \cdot a} y(t - na) = b_1 \cdot u(t - nk) + ... + b_{nb} \cdot u(t - nk - nb + 1).$$

Для систем із кількома входами *nb* і *nk* є векторами-рядками, де *i*-й елемент задає порядок/затримку, пов'язаний з *i*-м входом. У випадку з декількома виходами *na*, *nb* і *nk* – це матриці з такою кількістю рядків, скільки виходів.

Виберемо пункт Polynomial Models, відкриється вікно налаштування моделі, де необхідно в полі Strucnure вибрати метод (наприклад, ARX), в полі Orders задати параметри моделі $[na\ nb\ nk]$. Для автоматичного визначення параметрів на основі вхідних даних натисніть

Order Selection, відкриється вікно з кращими визначеними параметрами, під час натискання Insert параметри перенесуться у вікно налаштування моделі (рис. 6.4).

Для завершення процесу побудови моделі натисніть Estimate і закрийте вікно налаштування параметрів. Після виконання процесу Import Model модель з'явиться у вікні моделей. Подвійний клік миші на модель відкриває вікно перегляду моделі (рис. 6.5).

Для збереження моделі у робочому просторі у вікні перегляду моделі натисніть Export. Для графічного перегляду результатів ідентифікації порівняно з набором даних для перевірки натисніть на вікно моделі і у вікні System Identificatio виберіть пункти Model output або Model resids (нев'язка). Відкриються вікна з відповідним графічним представленням моделі, побудованої на основі робочого та перевірочного наборів даних, в правому вікні виводиться відсоток співпадання. Чим вище значення цього відсотка, тим краще наближення. Розриви на графіку нев'язки визначають несумісність моделей на робочому та перевірочному наборах даних.

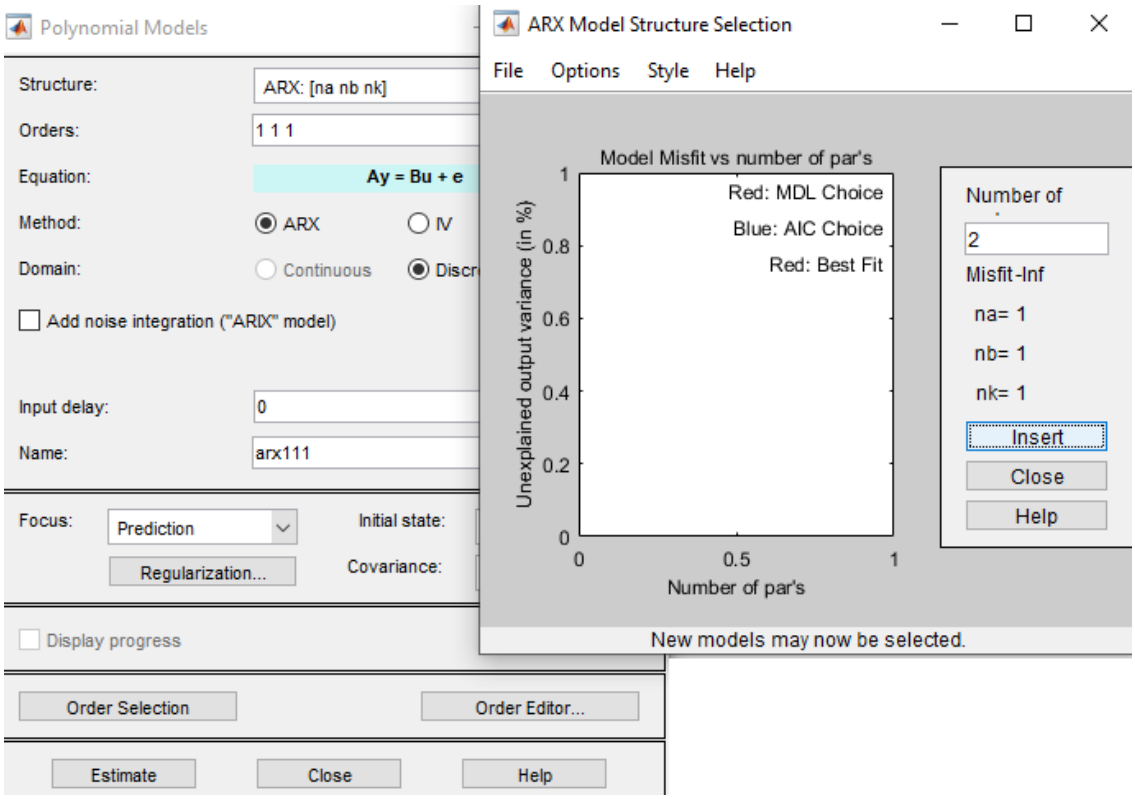



Рис. 6.4. Вікно налаштування параметрів моделі

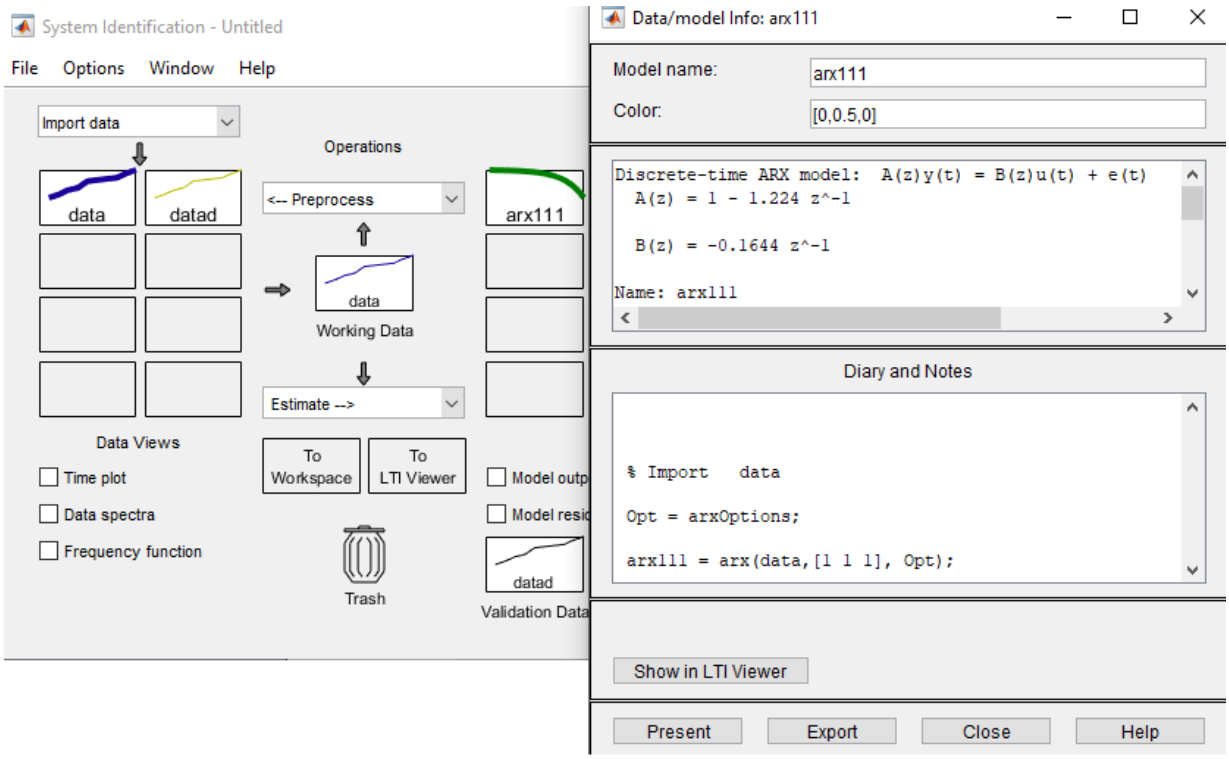



Рис. 6.5. Вікно перегляду моделі

Переглянути графічно характеристики моделі, можна вибравши відповідний пункт Model Views.

Щоб оцінити лінійні моделі, можна скористатися функцією швидкого запуску в програмі System Identification. Quick Start може створити остаточні лінійні моделі, які ви вирішите використовувати, або надати вам інформацію, необхідну для налаштування оцінки точних параметричних моделей, таких як постійні часу, вхідні затримки та резонансні частоти.

Ця дія генерує графіки ступінчастої характеристики, частотної характеристики та вихідних даних простору станів і поліноміальних моделей. Щоб отримати додаткові відомості про ці графіки, перегляньте перевірку моделей швидкого запуску. Швидкий старт оцінює наступні чотири типи моделей і додає наступне до програми ідентифікації системи з іменами за замовчуванням:

– imp – відповідь з кроком протягом певного періоду часу з використанням імпульсного алгоритму;

– spad – частотна характеристика в діапазоні частот із використанням алгоритму spa. АЧХ – це перетворення Фур'є імпульсної характеристики лінійної системи;

– arxqs – модель авторегресії четвертого порядку (ARX):

$y(t)+a_1y(t-1)+...+a_{na}y(t-na)=b_1u(t-nk)+...+b_{nb}u(t-nk-nb+1)+e(t)$,

де $y(t)$ представляє вихідний сигнал у момент часу t, $u(t)$ – вхідний сигнал у момент часу *t*, *na* – кількість полюсів, *nb* – кількість параметрів *b* (що дорівнює числу нулів плюс 1), *nk* – кількість вибірок перед тим, як вхідний сигнал вплине на вихідний сигнал системи (затримка), $e(t)$ – збурення білого шуму. Алгоритм оцінює параметри $a_1,...,a_n$ та $b_1,...,b_n$ використовуючи вхідні та вихідні дані з набору даних оцінки. В arxqs, *na*=*nb*=4, та *nk* оцінюється за моделлю ступінчастої реакції imp.

– n4s3 – модель простору станів, розрахована за допомогою n4sid. Алгоритм автоматично вибирає порядок моделі, яка є параметричною і має таку структуру:

$$\frac{dx}{dt}=Ax(t)+Bu(t)+Ke(t),\ y(t)=Cx(t)+Du(t)+e(t).$$

Алгоритм оцінює матриці простору станів A, B, C, D і K.

Перевірка моделей швидкого запуску Quick Start генерує такі графіки під час оцінювання моделі, щоб допомогти вам перевірити якість моделей:

– діаграма крок-відповідь;
– діаграма частотної характеристики;
– модельно-вихідний сюжет.

Після обробки даних можна видалити всі не потрібні набори даних, перемістивши їх в блок Trash і зберегти сеанс. Щоб зберегти модель у робочому просторі, її необхідно перетягти у прямокутник To Workspace, звідки модель можна завантажити у блок Simulink.

## Лабораторна робота №7

## Тема: Основи роботи з графічною мовою Simulink

***Мета:*** ознайомити студентів та надати практичних навичок роботи з графічною мовою Simulink. Ознайомлення з основними бібліотеками. Налаштування параметрів моделі. Задання математичних виразів та функцій. Простіші прийоми розв’язання лінійних і нелінійних систем та рівнянь.

***Завдання для самостійного виконання***

1. Розв’язати систему лінійних рівнянь.

2. Задати математичний вираз функції засобами графічної мови Simulink.

2.1. Обчислити значення виразу для заданого значення аргументу $x$ (пряма задача).

2.2. При заданому значенні функції $y$ знайти значення аргументу (розв'язати нелінійне рівняння, обернена задача).

2.3. Дослідити поведінку функції відносно параметра $a$ (графічно, параметрична задача).

3. Розв'язати систему нелінійних рівнянь методом простих інтеграцій.

***Варіант*** №1

1. Система лінійних рівнянь: $\begin{cases} 0.21x_1 - 0.18x_2 + 0.75x_3 = 0.11 \\ 0.13x_1 + 0.75x_2 - 0.11x_3 = 2.00 \\ 3.01x_1 - 0.33x_2 + 0.11x_3 = 0.13 \end{cases}$

2. Нелінійна функція: $y = a \cdot x + \lg(x-1) - 0.5$, $x = 2{,}5$, $y = 1$, $a = 0{,}5$.

3. Система нелінійних рівнянь: $\begin{cases} \sin(x+1) - y = 1.2 \\ 2x + \cos y = 2 \end{cases}$

***Варіант*** №2

1. Система лінійних рівнянь: $\begin{cases} 3.01x_1 - 0.14x_2 - 0.15x_3 = 1.00 \\ 1.11x_1 + 0.13x_2 - 0.75x_3 = 0.13 \\ 0.17x_1 - 2.11x_2 + 0.71x_3 = 0.17 \end{cases}$

2. Нелінійна функція: $y = \sqrt{x} - \cos(a \cdot x)$, $x = 2{,}5$, $y = 0{,}7$, $a = 0{,}387$

.

3. Система нелінійних рівнянь: $\begin{cases} \sin(x+1) - y = 1 \\ 2x + \cos y = 2 \end{cases}$

***Варіант*** №3

1. Система лінійних рівнянь: $\begin{cases} 0.92x_1 - 0.83x_2 + 0.62x_3 = 2.15 \\ 0.24x_1 - 0.54x_2 + 0.43x_3 = 0.62 \\ 0.73x_1 - 0.81x_2 - 0.67x_3 = 0.88 \end{cases}$

2. Нелінійна функція: $y = \lg(1 + a \cdot x) - 2 + x$, $x = 1{,}5$, $y = 0$, $a = 2$.

3. Система нелінійних рівнянь: $\begin{cases} \cos(x-1) + y = 0.5 \\ x - \cos y = 3 \end{cases}$

***Варіант*** №4

1. Система лінійних рівнянь: $\begin{cases} 0.34x_1 + 0.71x_2 + 0.63x_3 = 2.08 \\ 0.71x_1 - 0.65x_2 - 0.18x_3 = 0.17 \\ 1.17x_1 - 2.35x_2 + 0.75x_3 = 1.28 \end{cases}$

2. Нелінійна функція: $y = \sin(a \cdot x) + 1 - x^2$, $x = 2{,}5$, $y = 0$, $a = 0{,}5$.

3. Система нелінійних рівнянь: $\begin{cases} \sin y + 2x = 2 \\ \cos(x - 1) + y = 0.7 \end{cases}$

***Варіант*** №5

1. Система лінійних рівнянь: $\begin{cases} 3.75x_1 - 0.28x_2 + 0.17x_3 = 0.75 \\ 2.11x_1 - 0.11x_2 - 0.12x_3 = 1.11 \\ 0.22x_1 - 3.17x_2 + 1.81x_3 = 0.05 \end{cases}$

2. Нелінійна функція: $y = a \cdot e^{\sqrt{x}} + x - 2$, $x = 2{,}5$, $\mathcal{Y} =$, $a = 2$.

3. Система нелінійних рівнянь: $\begin{cases} \cos x + y = 1.2 \\ 2x - \sin(y - 0.5) = 2 \end{cases}$

***Теоретичні відомості***

Графічна мова Simulink є достатньо самостійним додатком до пакету MATLAB. Під час моделювання з використанням Simulink реалізується принцип візуального програмування, відповідно до якого користувач на екрані з бібліотеки стандартних блоків створює модель та здійснює розрахунки. Під час роботи з Simulink користувач має можливість модернізувати бібліотечні блоки, створювати власні, а також складати нові бібліотеки блоків.

Для запуску стартової сторінки Simulink необхідно натиснути на відповідну кнопку головного меню (рис. 7.1).

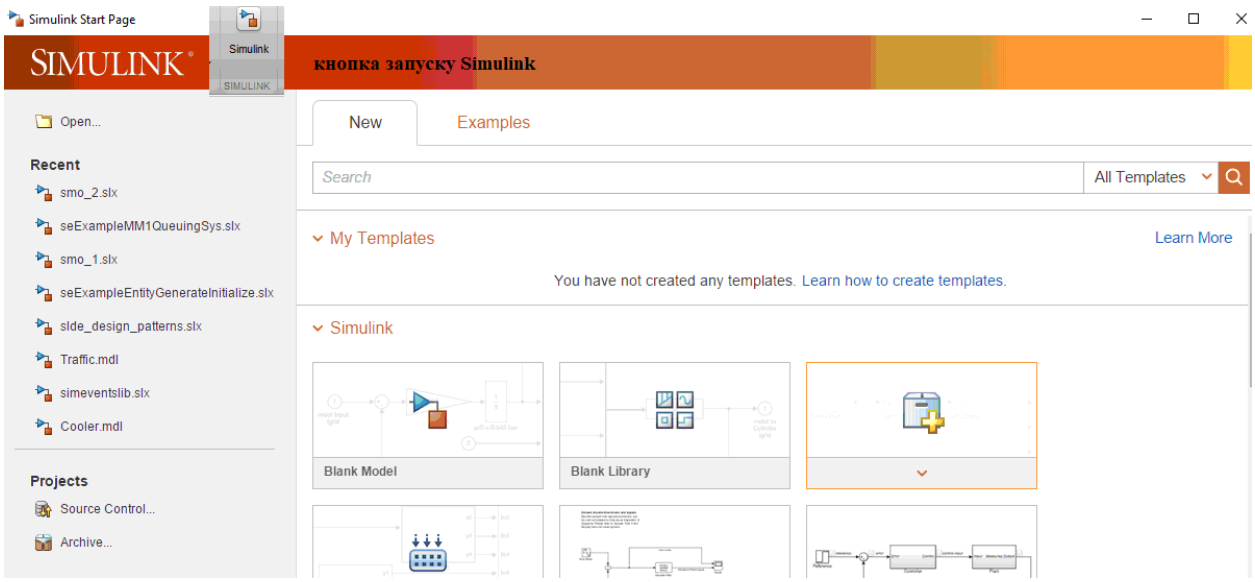



Рис. 7.1. Кнопка запуску та стартова сторінка Simulink

Для створення моделі необхідно відкрити Blank Model та Blank Library (бланк бібліотек можна відкрити з головного меню бланку моделі) (рис. 7.2).

Модель створюється шляхом встановлення та поєднання блоків з бібліотек на бланку моделі. Блок можна встановити двома шляхами – або перетягти з бібліотеки, або натисканням на праву клавішу миші відкрити випадаюче меню і вибрати пункт Add block to model. Блоки поєднуються шляхом з'єднання виходів і входів під час утримання лівої клавіші миші.

Кожний блок необхідно налаштувати, вікно налаштування відкривається подвійним натисканням на ліву клавішу миші (для уникнення помилки невизначеності даних кожний блок має налаштування за замовчуванням). Задати значення змінних можна двома способами:

– задати числове значення у відповідному полі блоку налаштування;
– задати значення у робочому вікні Workspace, а у вікні налаштування задати ім'я змінної.

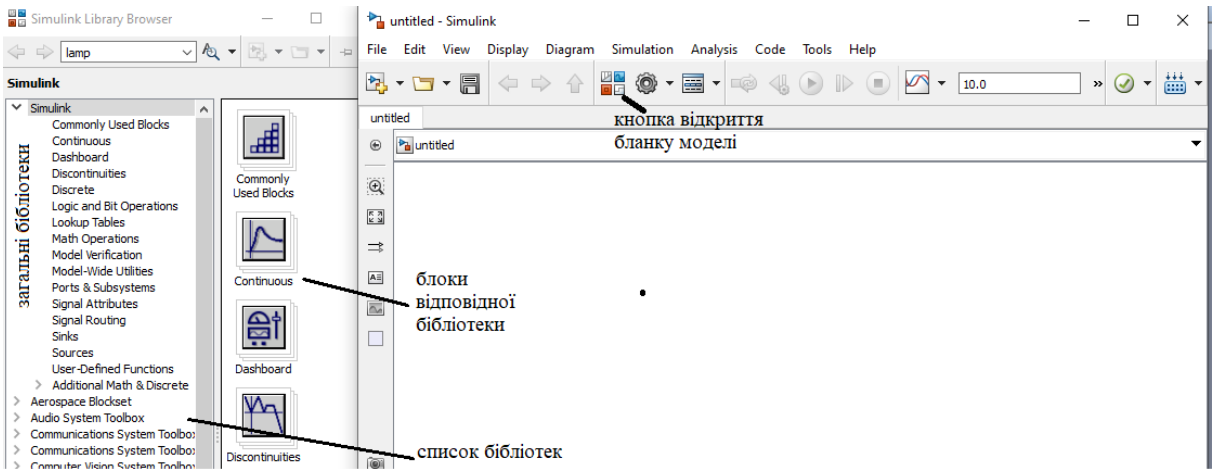



Рис. 7.2. Бланк бібліотек та бланк моделі

Призначення блоку та правила налаштування наведені у верхній частині блоку.

Бібліотеки та блоки розташовані у алфавітному порядку.

Розглянемо основні загальні бібліотеки, які необхідні для опису математичних виразів та виведення результатів.

***Math Operations – бібліотека математичних операцій***

До складу бібліотеки входять блоки для задання математичних операцій та елементарних алгебраїчних та тригонометричних функцій (рис.7.3).

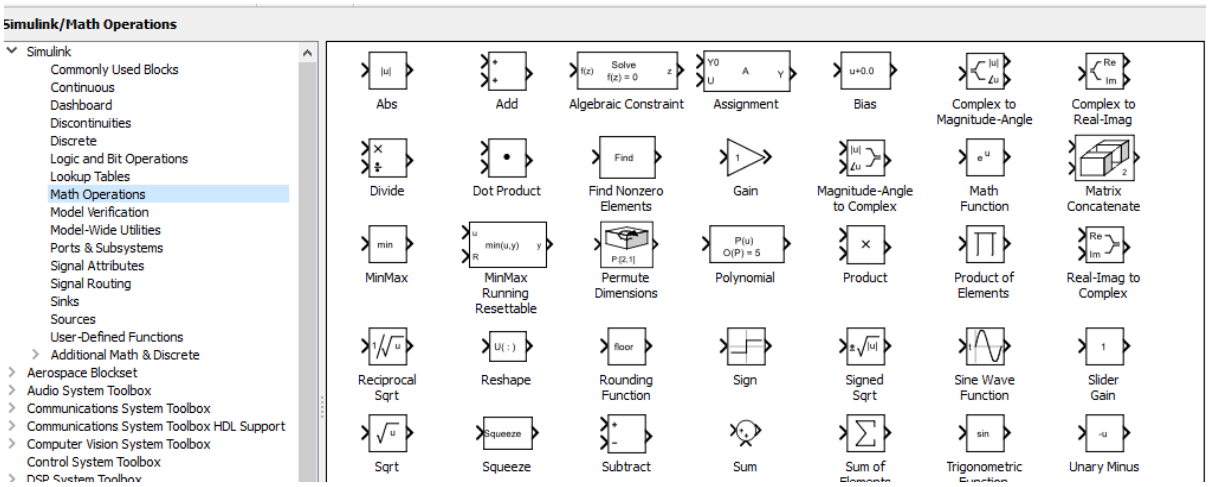



Рис. 7.3. Блоки бібліотеки математичних функцій

Необхідно відзначити, що для операції добутку передбачена матрична (Product) та елементна (Dot Product) операція. Для множення та ділення декількох значень можна скористатись блоком Product of Elements. Для множення на число можна скористатись операцією підсилення (Gain).

Блок Algebraic Constraint реалізує метод простих інтеграцій, його можна застосовувати для розв'язання систем лінійних, нелінійних рівнянь та систем. Параметром налаштування є початкове наближення кореня.

*Зауваження.* Для задач лінійної алгебри, роботи з поліномами та додаткові операції знаходяться в бібліотеці DSP System Toolbox – Math Functions.

***Sinks – бібліотека виводу***

Бібліотека містить блоки для різних способів виводу результатів: на дисплей, на графік у реальному часі (Scope) та XYграфік, у файл або робочий простір. Також можна призупинити процес симуляції, блок Terminator використовують для закриття блоків, виходи яких не з'єднані з іншими блоками, що дозволяє уникати попереджень (рис. 7.4)

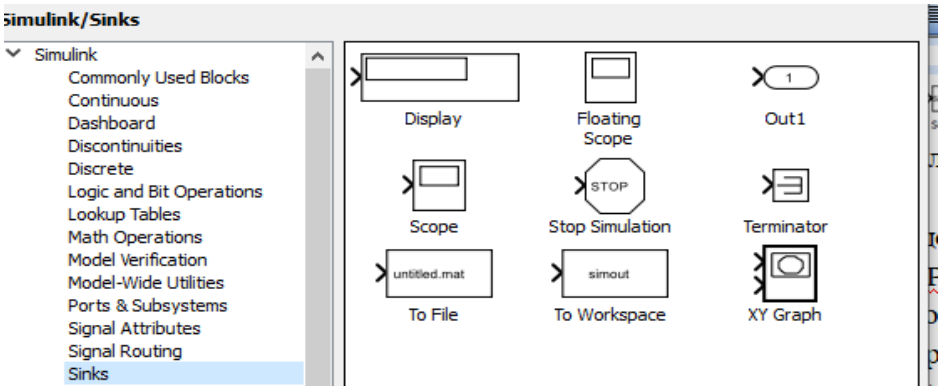



Рис. 7.4. Блоки бібліотеки Sinks

***Sources – бібліотека генерації сигналів***

Бібліотека включає джерела сигналів, такі як «генератор імпульсних/синусоїдальних сигналів», «генератор випадкових чисел», «генератор пилкоподібних сигналів» і т.д. Для задання числового значення використовують блок Constant, згенерувати неперервну послідовність значень можна за допомогою блоку Clock, а дискретну – Digital Clock (рис. 7.5.)

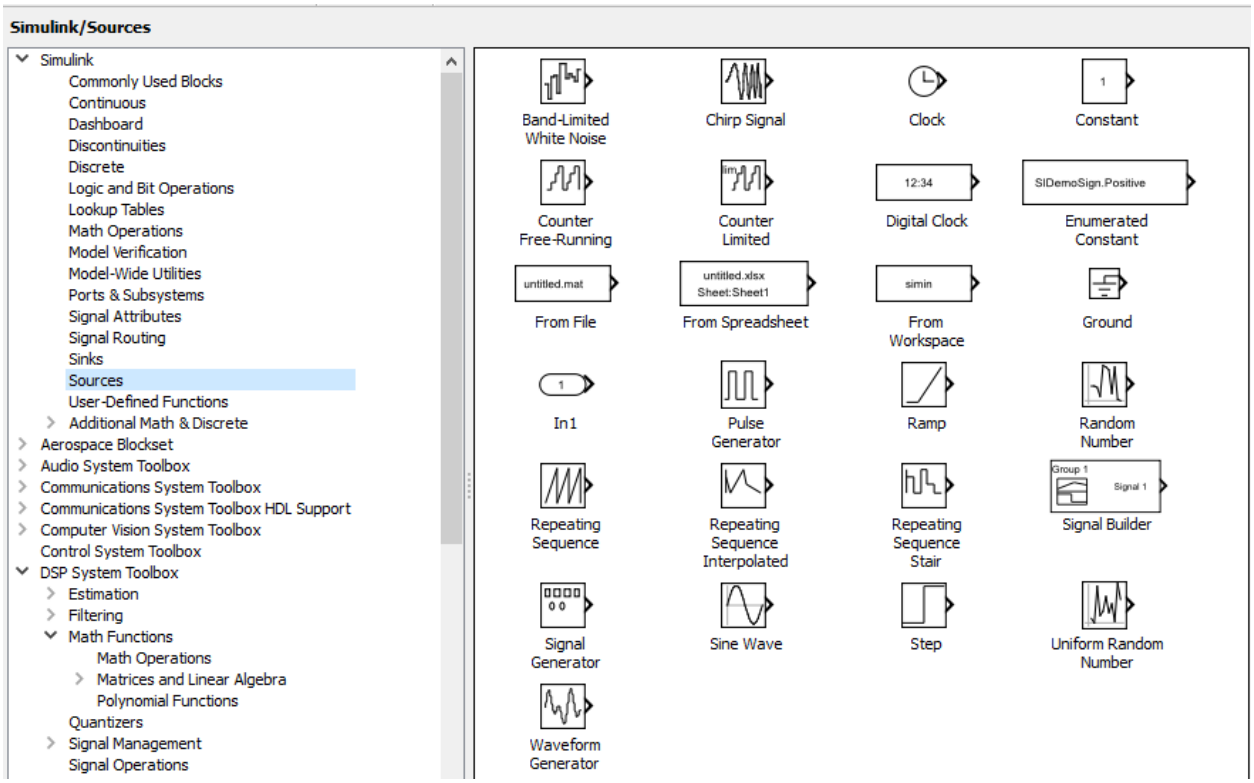



Рис. 7.5. Блоки бібліотеки Sources

Для створення складних моделей корисними є такі бібліотеки:

– Ports&Subsystems – бібліотека створення підпрограм та складних структур, наприклад, розгалуженого процесу;
– User-Defined Functions – задання функцій. Так функцію можна описати, скориставшись блоком Fcn, блок Matlab Function дозволяє завантажити функцію з М-файлу.

***Порядок виконання***

1. Розв’язати систему лінійних рівнянь: $\begin{cases} 2{,}7x_1 + 3{,}3x_2 + 1{,}3x_3 = 2{,}1 \\ 3{,}5x_1 - 1{,}7x_2 + 2{,}8x_3 = 1{,}7 \\ 4{,}1x_1 + 5{,}8x_2 - 1{,}7x_3 = 0{,}8. \end{cases}$

Для розв’язання систем лінійних рівнянь в середовищі Matlab вбудований набір чисельних методів: DSP System Toolbox/Math Functions/Matrices and Linear Algebra/Linear System Solvers. Скористаємось методом LU-розкладу, блок реалізації якого представлений на (рис. 7.6) і має два вхідних параметри: A – квадратна матриця коефіцієнтів системи; B – вектор стовпчик правої частини.

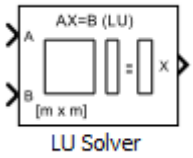



Рис. 7.6. Блок реалізації методу LU-розкладу

Можна задати вхідні параметри, безпосередньо записавши їх у блок Constant, але якщо матриця великої розмірності, то такий спосіб незручний, то створимо відповідні змінні безпосередньо у робочому просторі Workspace (рис. 7.7).

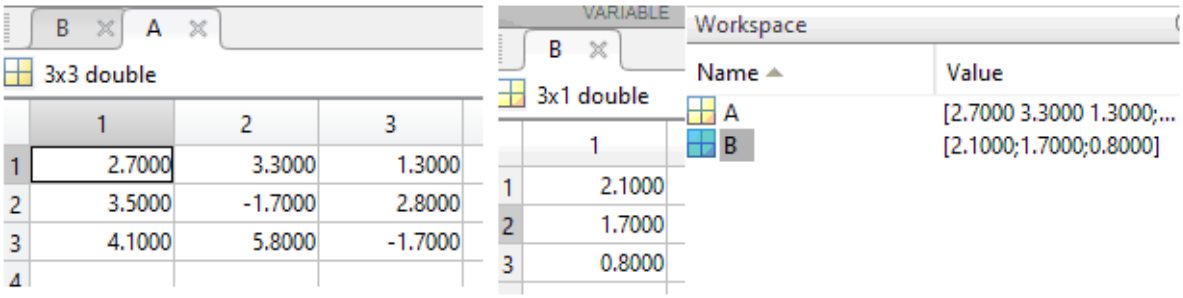



Рис. 7.7. Завдання системи лінійних рівнянь

Створимо вікно моделі і відкриємо браузер бібліотек. За зазначеним вище шляхом відкриємо бібліотеку чисельних методів лінійної алгебри і встановимо блок методу LU-розкладу. З бібліотеки Sources встановимо два блоки Constant для задання матриці коефіцієнтів та правої частини системи. У вікні налаштування в полі Constant Value поставимо відповідні імена змінних. Для виводу результату з бібліотеки Sinks встановимо блок Display. Поєднаємо блоки зв'язками і запустимо програму на виконання (рис. 7.8.).

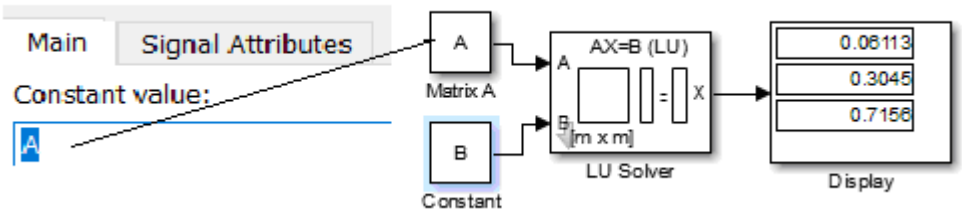



Рис. 7.8. Розв'язання системи лінійних рівнянь

У вікні блоку Display отримаємо значення невідомих $x_{1,2,3}$. Збережемо програму на диску.

2. Нехай задано функцію: $y = 2arctgx - a \cdot x + 2$, $a = 3$.

2.1. Обчислити значення функції в точці $x = 1.5$. Відкриємо нове вікно моделі і складемо програму, що описує нашу функцію, для задання значення $x$ скористаємось блоком Constant (бібліотека Sources). За допомогою блоків бібліотеки Math Operations запишемо функцію. Для обчислення суми/різниці скористаємось блоком Sum. У полі List of signs вікна налаштування задаємо послідовність дій і для більшої зручності надаємо блоку форму прямокутника. Для виводу результату встановимо блок Display і запустимо програму на виконання (рис. 7.9). Збережемо програму на диску.

2.2. Прирівнявши $y = 0$, отримаємо нелінійне рівняння, яке необхідно розв'язати методом простих інтеграцій (блок Algebraic Constraint бібліотеки Math Operations). Для вирішення задачі скористаємось попередньою моделлю. Значення аргументу невідомо, тому видалимо блок, який задає значення $x$. Між описом виразу та блоком виводу Display вставимо блок Algebraic Constraint. У вікні налаштування блоку задаємо початкове наближення кореня. На виході цього блоку маємо поточне значення шуканого кореня, отож з'єднаємо вихід блоку з початком опису виразу. Запустимо програму на виконання (рис. 7.10.)

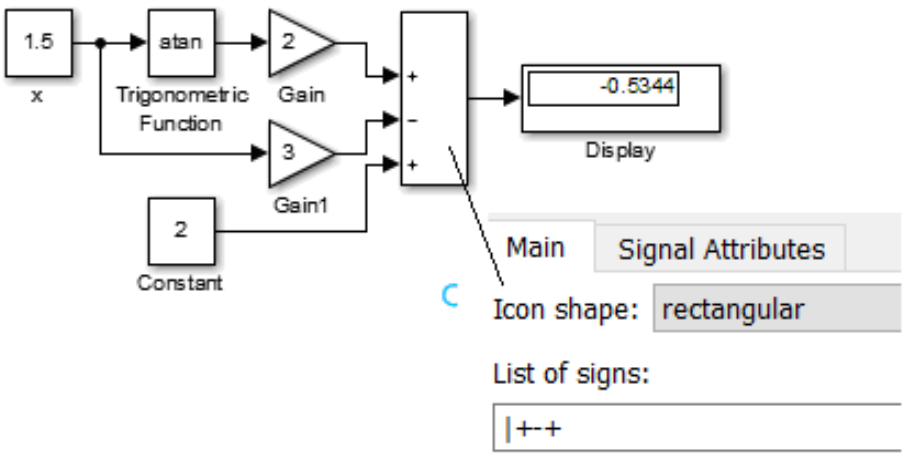



Рис. 7.9. Обчислення значення функції у заданій точці

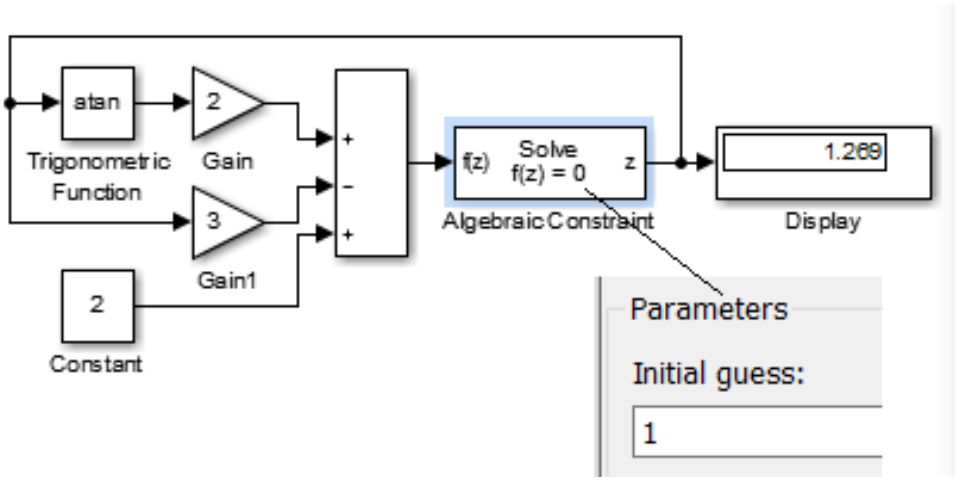



Рис. 7.10. Розв’язання нелінійного рівняння

2.3. Проведемо дослідження поведінки функції за різних значень параметра $a$. Знову за основу візьмемо першу модель, для задання значень аргументу $x$ скористаємось годинником. Візьмемо значення параметра рівним 1, 2, 4, 6, для цього у вікні налаштування відповідного блоку Gain запишемо всі можливі значення як вектор. Результат виведемо на осцилограф Scope. В головному меню моделі виберемо пункт налаштування параметрів моделі Model Configuration Parameters і на вкладенці Solver встановимо значення аргумента від –5 до 5 (рис. 7.11).

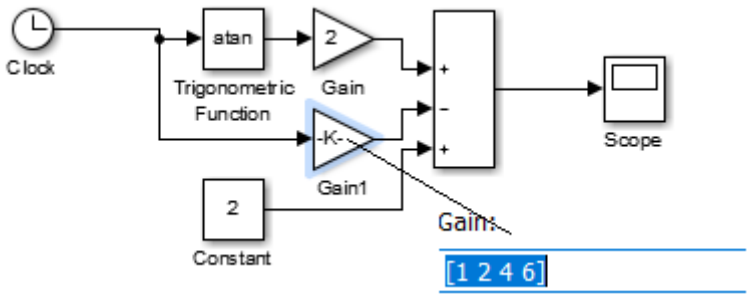



Рис. 7.11. Побудова графіка функції

Запустимо програму на виконання, отримаємо графіки функції при різних значень параметра, за допомогою інструментів налаштування вставимо легенду (рис. 7.12).

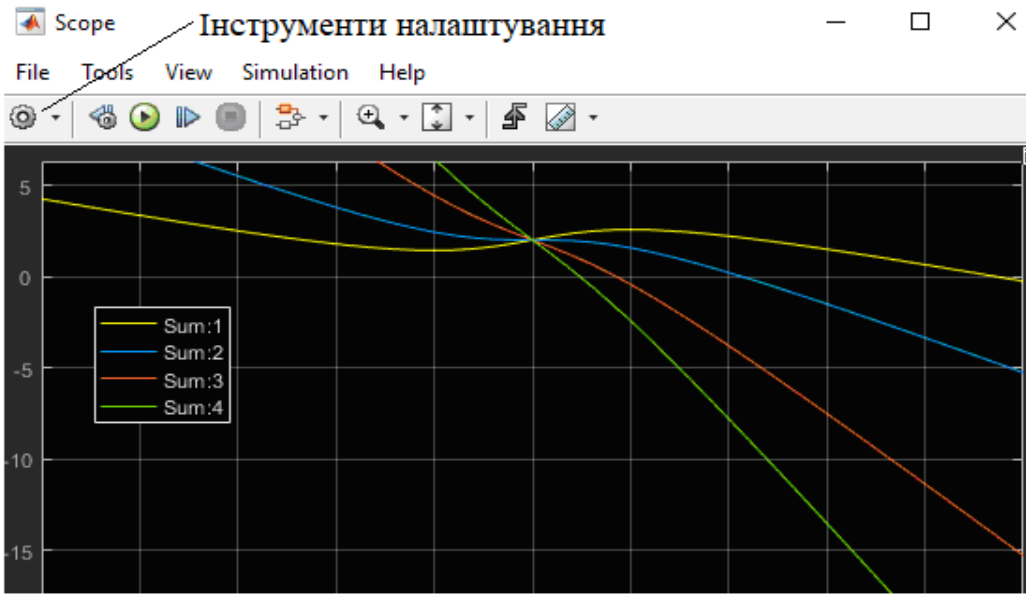



Рис. 7.12. Графік функції за різних значень параметра

З графіка видно, що кут нахилу графіка функції до осі OX змінюється прямопропорційно до зміни значення параметру.

3. Розв'язати систему нелінійних рівнянь: $\begin{cases}\sin(x+y)-1.2x=0.1\\ x^2+y^2=1.\end{cases}$

Для застосування методу простих інтеграцій для знаходження розв'язку системи необхідно задати початкові значення для шуканих змінних $x$ та $y$, від якості початкових значень значною мірою залежить збіжність методу. В середовищі Matlab є функція для побудови графіка неявно заданої функції, що спрощує нашу задачу:

ezplot(fun,[xmin,xmax,ymin,ymax]).

Параметр fun необхідно привести до вигляду $f(x,y)=0$.

Друге рівняння системи задає коло з центром у початку координат радіусом одиниця, на підставі чого діапазон зміни шуканих параметрів можна встановити від -1,2 до 1,2.

Побудуємо графік спочатку функції, що задає перше рівняння і, утримуючи поточне графічне вікно за допомогою команди hold, побудуємо графік для другого рівняння (рис. 7.13):

```
ezplot('sin(x+y)-1.2*x-0.1',[-1.2 1.2]), hold
ezplot('x^2+y^2-1',[-1.2 1.2])
```

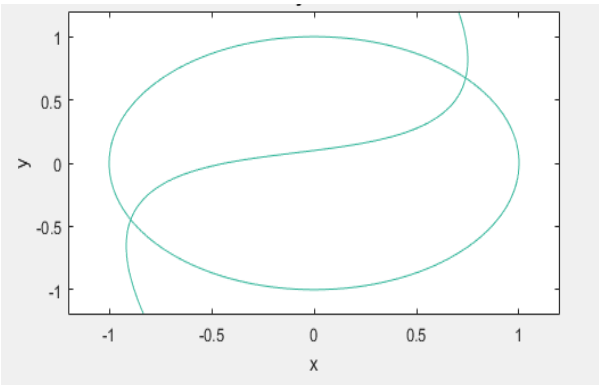



Рис. 7.13. Графік неявно заданих функцій

Будемо шукати додатній корінь, в якості початкових наближень візьмемо наступні значення: $x = 0.7$, $y = 0.7$.

Для розв'язання системи використаємо два блоки Algebraic Constraint, де перший блок буде знаходити змінну $x$ з першого рівняння, другий – змінну $y$ з другого рівняння (рис. 7.14).

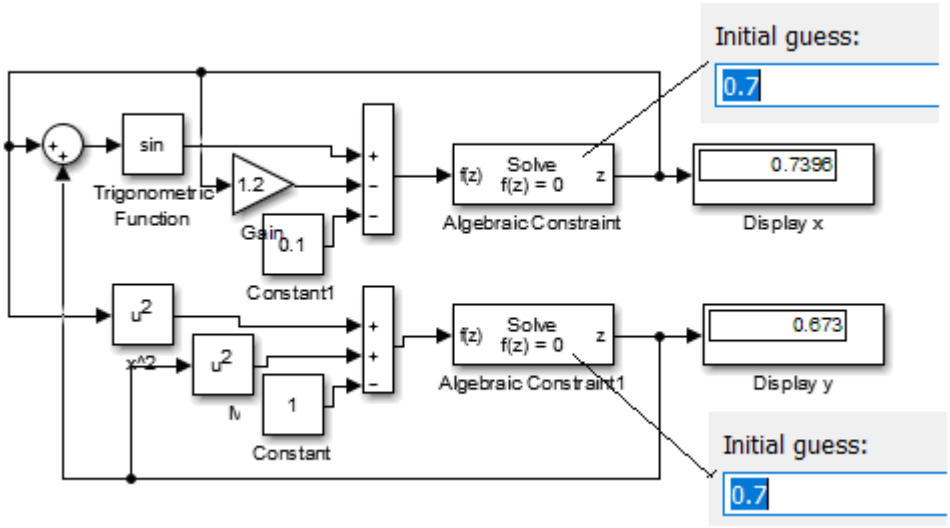



Рис. 7.14. Розв'язання системи нелінійних рівнянь

Після запуску програми у віконцях блоків Display отримаємо значення шуканих змінних.

## Лабораторна робота № 8

## Тема: Побудови та дослідження динамічних імітаційних моделей

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження простих динамічних імітаційних моделей, що пов'язано зі складанням та розв'язкою диференціальних рівнянь.

***Завдання для самостійного виконання***

Засобами Simulink скласти та знайти розв'язок диференціальних рівнянь першого та вищого порядків.

***Варіант*** № 1

1. $x \cdot y' + x^2 + x \cdot y - y = 0,\ y(0) = 1,\ x = [0,3].$

2. $y'' - 2y' = x^2 - 1,\ y(1) = -1.6,\ y'(1) = -3/4,\ x = [1,3].$

***Варіант*** № 2

1. $y \cdot y' = y^2 + x \cdot y',\ y(0) = 0.1,\ x = [0,3].$

2. $y'' + 2y' = 3e^x,\ y(0.3) = -1,\ y(0.3) = 5.8,\ x = [0.3,1].$

***Варіант*** № 3

1. $x^2 \cdot y' - 2x \cdot y = 3y,\ y(0) = 0.1,\ x = [0,3].$

2. $y'' + y' = 3x^2,\ y(1) = -1,\ y(1) = 2,\ x = [1,2].$

***Варіант*** № 4

1. $y' + y = x \cdot y^3,\ y(0) = 0.5,\ x = [0,3].$

2. $y'' = -\frac{y'}{x} + \frac{y}{x^2} + 1,\ y(3) = 6,\ y'(3) = 3,\ x = [3,6].$

***Варіант*** № 5

1. $x - \frac{y}{y'} = \frac{2}{y},\ y(1) = 1,\ x = [1,3].$

2. $y'' - y = e^x,\ y(0) = 0,\ y'(0) = 0.5,\ x = [0,1].$

***Теоретичні відомості***

Для розв'язання диференціальних рівнянь в середовище Matlab вбудована досить широка бібліотека спеціалізованих чисельних методів та реалізовано декілька способів розв'язання. Вибір методу залежить від різних факторів, це питання розглядалося в курсі дисципліни «Чисельні методи».

Згадаємо, що основною операцією розв'язання диференціальних рівнянь є інтегрування, яке виконується з точністю до константи, тому будемо розглядати задачу Коші, коли, крім рівняння, задані і початкові умови. Також варто згадати, що розв'язкою диференціального рівняння

є функція, якщо задача вирішується за допомогою чисельних методів, то результатом є її табличне представлення, яке доцільно представляти графічно.

Для застосування чисельних методів рівняння необхідно привести до нормальної форми, де ліворуч стоїть лише похідна, праворуч – вираз, якому вона дорівнює: $y = f(x, y)$.

Диференціальне рівняння вищих порядків при приведенні до нормальної форми приводиться до системи рівнянь, наприклад, рівняння $y'' + y' = x \cdot y$ буде переписано у вигляді наступної системи:

$y1 = y'$

$y2 = x \cdot y - y1$.

Отримати розв'язок звичайного диференціального рівняння можна як в у чисельному, так і в символьному вигляді.

***Символьний розв'язок звичайного диференціального рівняння***

dsolve(eqn1,eqn2, ...) – функція для отримання розв'язку звичайного диференціального рівняння в символьному вигляді, де eqn – диференціальне рівняння та початкові умови, задані як рядок або символьний вираз типу: Diff(x) == вираз. Для задання функції використовується подвійний символ, дорівнює, наприклад, розв'язання рівняння $y' = -a \cdot x$:

```
syms x(t) a
dsolve(diff(x) == -a*x) returns
ans = C1/exp(a*t)
```

Якщо розв'язується задача Коші і треба задати початкові змінні та ім'я незалежної змінної, то звернення до функції буде мати вигляд:

```
x = dsolve(diff(x) == -a*x, x(0) == 1, 's') returns
x = 1/exp(a*s)
```

Наступний приклад демонструє розв'язок нелінійного рівняння $(y')^2 + y^2 = 1$ за початкової умови $y(0) = 0$, коли рівняння задано символьним рядком:

```
y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0')
y =
cosh(t*1i + (pi*1i)/2)
cosh(t*1i - (pi*1i)/2)
```

За допомогою функції dsolve можна розв'язувати звичайні диференціальні рівняння вищих порядків, але їх необхідно привести до нормальної форми.

***Чисельний розв'язок звичайного диференціального рівняння***

Знайти розв'язок диференціальної задачі чисельно можна наступними способами:

- в режимі програмування, застосувавши спеціальні функції;
- за допомогою спеціального вбудованого інструменту dee (differential equation editor);
- засобами Simulink.

***В режимі програмування***

Для розв'язання звичайних диференціальних рівнянь першого порядку Matlab реалізовано різні методи. Найбільш популярними є:

ode23 – однокрокові явні методи Рунге-Кутта 2-го та 4-го порядків;

ode45 – однокрокові явні методи Рунге-Кутта 4-го та 5-го порядків;

ode113 – метод Адамса.

Загальна схема їх застосування має вигляд:

[T,y]=solver(fh, tspan, y0),

де $T$ – незалежна змінна; $y$ – значення функції у відповідній точці $T$; *fh* – вказівник на функцію, описує диференціальне рівняння, *tspan* – діапазон зміни незалежної змінної, $y0$ – початкове наближення.

**Приклади**

1. Розв'язати задачу Коші: $y' = 2t$, $t \in [0, 5]$, $y(0) = 0$.

**Розв'язок**:

```
tspan = [0 5];  y0 = 0;
[t,y] = ode45(@(t,y) 2*t, tspan, y0);
```

2. Знайти розв'язок рівняння ван дер Поля: $y'' - \mu(1 - y^2)y' + y = 0$, $t \in [0, 20]$, $y(0) = 0$, $y'(0) = 2$.

**Розв'язок**. Так як це рівняння 2-го порядку, то спочатку його треба привести до нормальної форми:

$$y_1' = y_2$$

$$y_2' = \mu(1 - y_1^2)y_2 - y_1.$$

Складемо підпрограму-функцію для опису отриманої системи:

```
function dydt = vdp1(t,y)
%VDP1  Evaluate the van der Pol ODEs for mu = 1
dydt = [y(2); (1-y(1)^2)*y(2)-y(1)];
end
```

Розв’яжемо рівняння і побудуємо графік:

```
[t,y] = ode45(@vdp1,[0 20],[2; 0]);
plot(t,y(:,1),’-o’,t,y(:,2),’-o’)
```

***За допомогою редактора dee***

Виклик редактора здійснюється за допомогою команди dee у командному вікні Matlab, з’явиться редактор dee. Подвійним клацанням лівої клавіші миші на верхній прямокутник відкривається вікно налаштування редактора (рис. 8.1).

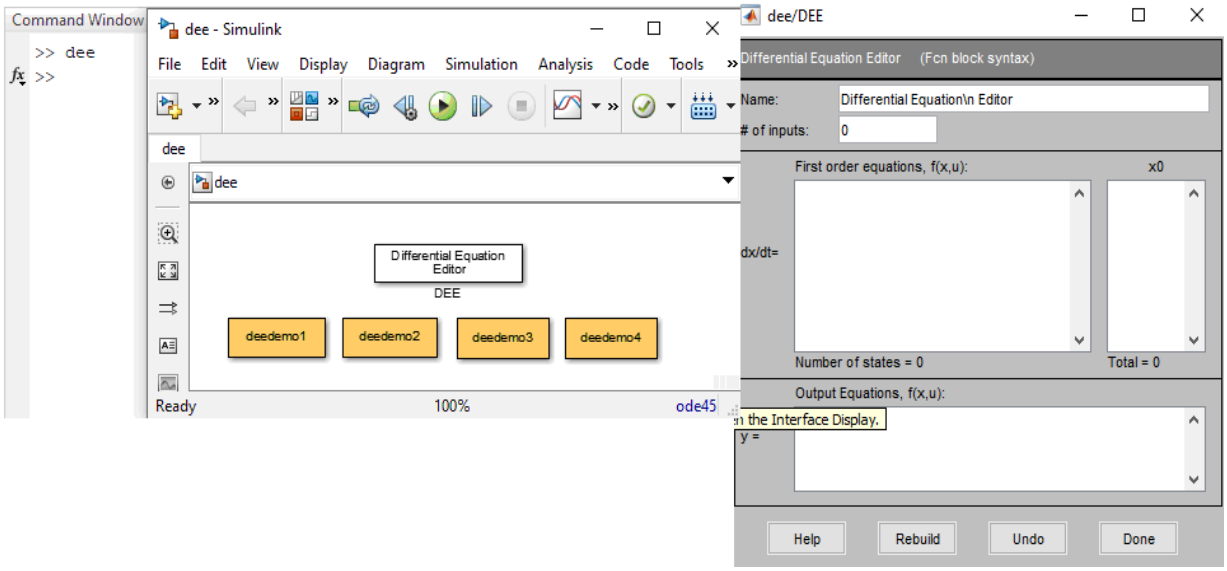



Рис. 8.1. Редактор Differential Equation Editor

У вікні налаштування необхідно задати ім’я задачі, якщо дані надходять ззовні, то у вікні of inputs необхідно зазначити їх кількість, всі зовнішні змінні визначаються як масив з ідентифікатором *u*(), незалежна змінна – *t*, залежна – *x*. У лівому вікні *dx/dt* задається рівняння або система рівнянь приведена до нормально вигляду, у правому вікні *x*0 – відповідні початкові умови. У нижньому вікні *y*= – вихідні змінні. Прикладами застосування редактора знаходяться під жовтими прямокутниками редактора.

Щоб скористатись редактором, верхній білий прямокутник треба перетягти на бланк моделі.

***Засобами Simulink***

В бібліотеках Commonly Used Blocks та Continuous є блок Integrator, що реалізує операцію чисельного інтегрування. Параметром налаштування блоку є значення початкової умови (рис. 8.2).

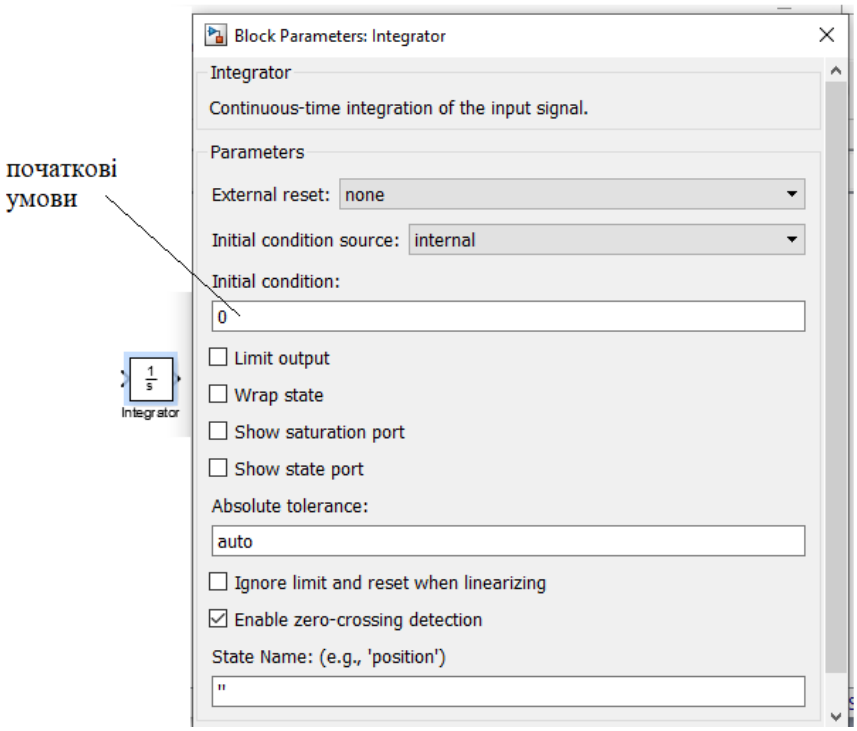



Рис. 8.2. Блок інтегрування

Якщо диференціальне рівняння записане у вигляді:

$$y^{(n)} = f\left(x, y, y', ..., y^{(n-1)}\right),$$

то розв'язок полягає у *n*-кратному інтегруванні його правої частини.

***Порядок виконання***

1. Розв'язати задачу Коші: $\frac{dy}{dx} = \cos(x+y) + \frac{3}{2}(x-y),\ x \in (0,20),$ $y(0) = 0.$

1.1. Розв'язання у режимі програмування. Складемо підпрограму функцію, яка описує праву частину рівняння:

```
function dydx = fun_8(x,y)
```

```
%Лабораторна №8
dydx=cos(x+y)+3/2.*(x-y);
end
```

Для розв’язання застосуємо метод Рунге-Кутта та представимо розв’язок у вигляді графіка (рис. 8.3):

```
[x y]=ode45(@fun_8,[0 20],0);
plot(x,y)
```

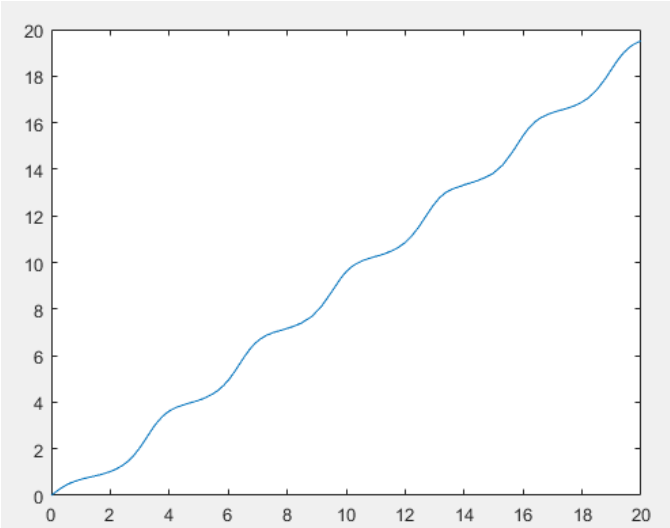



Рис. 8.3. Графічне представлення розв’язку

1.2. Розв’язання за допомогою додатка dee (рис. 8.4). Створимо бланк моделі, у командному вікні відкриємо додаток dee і перетягнемо блок DEE1 і на бланк моделі. Для виводу результату на графік встановимо блок Scope. Подвійним кліком відкриємо блок DEE1 і задаємо рівняння (необхідно пам’ятати, що незалежна змінна позначається як *t*, а шукані функції – масив *x*, також арифметичні операції задаються як матричні без точки), початкову умову та вихідну змінну. Для завершення процесу натиснемо Rebuild і закриємо додаток Done.

У блоку DEE1 з’явиться вихід, який з’єднаємо з входом блоку Scope. Задамо час симуляції рівний 20 і запустимо програму на виконання, у вікні Scope з’явиться графік шуканої функції $y(x)$. У блоку DEE1 з’явиться вихід, який з’єднаємо з входом блоку Scope.

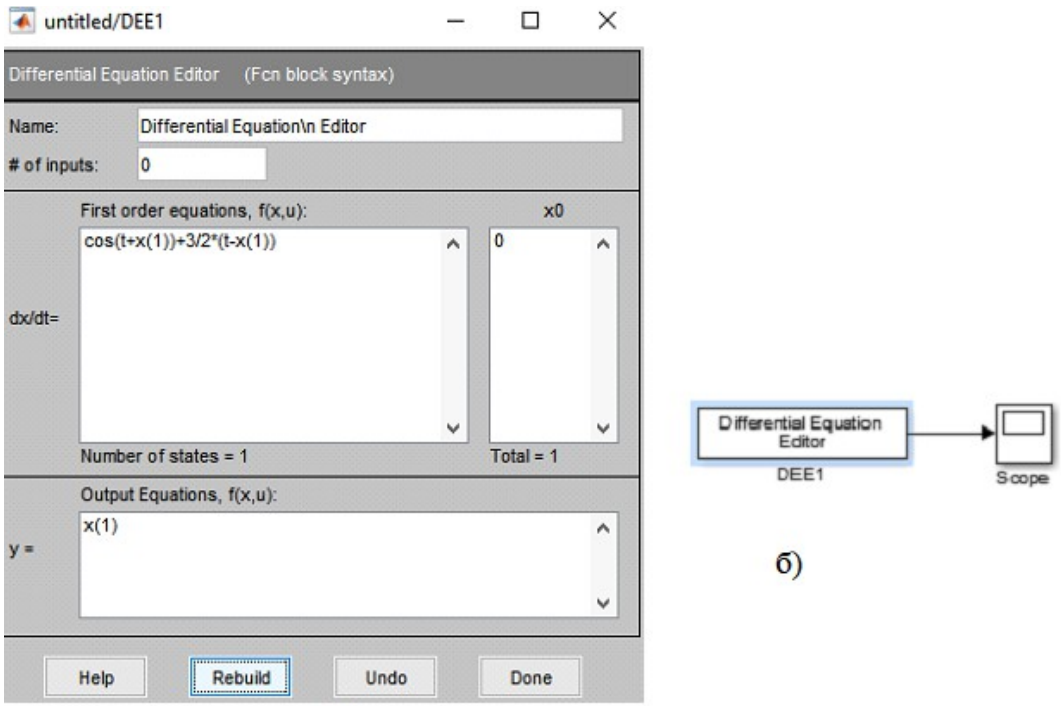



Рис. 8.4. Розв’язання за допомогою додатку dee

1.3. Розв’язання засобами Simulink. Створимо новий бланк моделі. Складати дану задачу з блоків зручніше, починаючи з кінця. Оскільки розв’язок диференціального рівняння отримується шляхом інтегрування, то з бібліотеки Continuous візьмемо блок Integrator, після нього встановимо блок Scope і з’єднаємо їх. Якщо на інтегратор подається сума двох доданків, то перед інтегратором встановимо блок суми, задаємо вигляд кожного доданка, де значення незалежної змінної $x$ можна задати як годинник або лінійним сигналом Ramp, значення $y$ знімаємо на виході блоку Integrator (рис. 8.5).

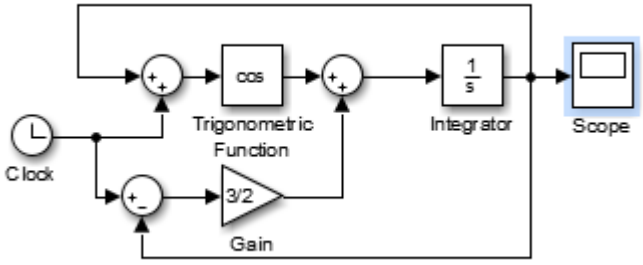



Рис. 8.5. Simulink-модель розв’язання диференціального рівняння

Аналогічно розв’яжемо рівняння другого порядку, але необхідно пам’ятати, що для застосування стандартних функцій та блоку dee його необхідно привести до нормальної форми.

На (рис. 8.6.) наведена програма розв’язання звичайного диференціального рівняння 2-го порядку. На виході першого інтегратора отримуємо значення похідної і відповідно початкова умова дорівнює 1, на виході другого – значення функції і відповідна початкова умова дорівнює 0.

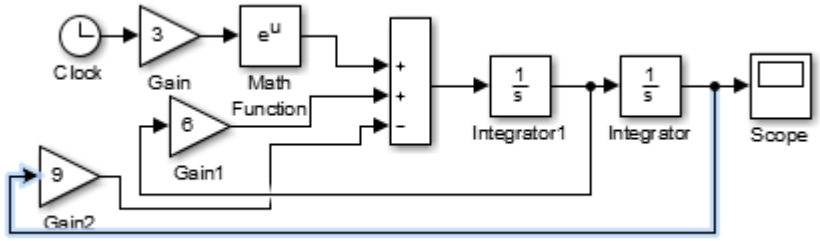



Рис. 8.6. Розв’язання звичайного диференціального рівняння 2-го порядку

## Лабораторна робота № 9
## Тема: Побудова та дослідження складних динамічних імітаційних моделей

***Мета:*** ознайомити студентів та надати практичних навичок, що пов’язано з складанням та розв’язком систем звичайних диференціальних рівнянь. Розглянути приклад ускладнення моделі.

***Завдання для самостійного виконання***

Засобами Simulink скласти та знайти розв’язок системи звичайних диференціальних рівнянь. Скласти план експерименту та провести дослідження поведінки системи відносно параметрів.

1. Модель притягуючої точки. Точка, масою $m$, рухається в плоскому середовищі, що опирається руху під дією притягуючого нерухомого центра. Оскільки середовище опирається руху, то енергія точки зменшується і вона з часом повинна впасти на притягуючий центр. На точку діє дві сили : $F_T = -w^2 mr$ – сила тяжіння до центра; $Q = -kmv$ – сила опору середовищу. Також розглянемо випадки, коли на точку діє зовнішня сила $F$ і коли точка має двигун з силою тяги $S = k_1 mv$ (рис. 9.1).

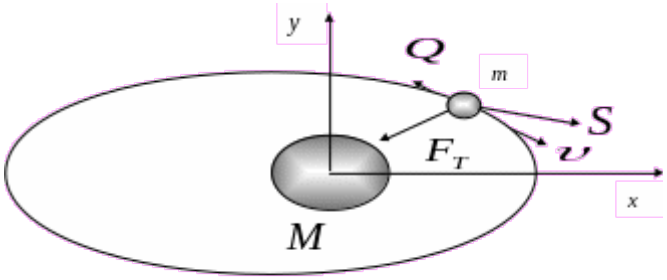



Рис. 4.1

Рис. 9.1. Рух точки навколо притягуючого центра

Знайшовши проекції на координатні осі всіх сил, отримаємо наступні системи рівнянь:

1.1 $\ddot{x} = -w^2 x - k\dot{x}$

$\ddot{y} = -w^2 y - k\dot{y}$ випадок відсутності зовнішнього впливу;

1.2 $\ddot{x} = -w^2 x - k\dot{x} + F_x$

$\ddot{y} = -w^2 y - k\dot{y} + F_y$ на точку діє зовнішня сила;

1.3 $\ddot{x} = -w^2 x - (k_1 - k)\dot{x} + F_x$

$\ddot{y} = -w^2 y - (k_1 - k)\dot{y} + F_y$ на точку діє зовнішня та внутрішня сили.

Початкові дані:

$\dot{x}(0) = 0,\ x(0) = x_0,\ \dot{y}(0) = y_0,\ y(0) = 0$

$k = 0.1,\ k_1 = 2k,\ w = 1.$

Скласти план експерименту та дослідити траєкторію руху за різних співвідношень зовнішньої сили ($F_x = F_y, F_x \neq F_y$ при цьому значення сил змінюються) та параметра $k_1$. Зробити висновки.

2. Модель подвійного маятника (рис. 9.2):

$\ddot{X} = -w^2(1 + 2\mu)X + \mu w^2 x,$

$\ddot{x} = -w^2 x + w^2 X,$

Початкові дані:

$\dot{X}(0) = 0,\ X(0) = X_0,\ \dot{x}(0) = x(0) = 0,$

$w = 1,\ \mu = 0.01 = m/M,\ L = l$

Провести дослідження за різних співвідношень мас. Зробити висновки.

***Теоретичні відомості***

Необхідні теоретичні відомості викладені у матеріалах лабораторної роботи 8.

***Порядок виконання***

1. Створити Simulink-модель.

2. Скласти план експерименту. Виконати прогони моделі згідно з планом.

3. Зробити висновки.

## Лабораторна робота № 10
## Тема: Побудова та дослідження динамічних імітаційних моделей на прикладі реальних систем

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження більш складних динамічних імітаційних моделей на прикладі реальних систем, закріпити навички розв'язання систем звичайних диференціальних рівнянь та навчитись створювати підпрограми у Simulink-моделях.

***Завдання для самостійного виконання***

Засобами мови Simulink побудувати та дослідити модель «метелика Лоренца», що демонструє хаотичну поведінку. Дослідити поведінку системи за зміною значень параметрів, побудувати графіки:

$$\begin{cases} x' = \sigma(y - x) \\ y' = x(r - z) - y \\ z' = xy - bz \end{cases}$$

Початкові умови: $x(0) = y(0) = z(0) = 1$.

Вхідні дані: $\sigma = 10$, $r = 28$, $b = 8/3$.

***Теоретичні відомості***

Необхідні теоретичні відомості викладені у матеріалах лабораторної роботи 8.

***Порядок виконання***

Порядок виконання аналогічний до лабораторної роботи 8.

## Лабораторна робота № 11
## Тема: Побудова та дослідження двомасної системи

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження більш складних динамічних імітаційних моделей на прикладі реальних систем, закріпити навички розв'язання систем звичайних диференціальних рівнянь та навчитись створювати підпрограми у Simulink-моделях.

***Завдання для самостійного виконання***

Модель вимушених коливань. Для дослідження впливу основних параметрів екіпажу на вертикальні коливання використовують спрощену модель із двома ступенями свободи, в якій дві маси пов'язані пружними та дисипативними зв'язками (рис. 11.1). Така модель описує вертикальні коливання рейкових екіпажів з двоярусним підвішуванням: магістральних локомотивів (електровозів та тепловозів) та пасажирських вагонів:

$m_1 z_1'' + b_1 z' + b_2\left(z_1' - z_2'\right) + c_1 z_1 + c_2\left(z_1 - z_2\right) = b_1\eta' + c_1\eta$

$m_2 z_2'' + b_2\left(z_2' - z_1'\right) + c_2\left(z_2 - z_1\right) = 0$, де

$m_1$ – обресорена маса візка;

$m_2$ – маса кузова, приведена до одного візка;

$c_1$, $b_1$ – жорсткість та демпфування у першому ярусі підвішування;

$c_2$, $b_2$ – жорсткість та демпфування у другому ярусі підвішування;

$\eta(t)$ – збурення з боку шляху;

$z_i, z_i', z_i''$ – узагальнені координати та їх похідні за часом:

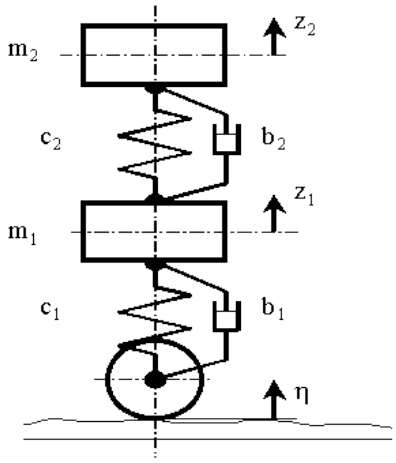



Рис. 11.1. Схема двомісної системи

В якості збурень використаємо модель нерівності, що є сумою напівхвилі синусоїди частотою $\omega$ і трьох напівхвиль синусоїди частотою $3\omega$, укладені на довжині рейкової ланки L.

$\eta(t) = |A_1 \sin(\omega t) + A_2 \sin(3\omega t)|$, $\omega = \frac{\pi}{L} V$, $V$ – швидкість руху.

Амплітуди нерівностей $A_1$, $A_2$ вибираються залежно від типу та стану шляху.

Для моделювання взяти наступні початкові дані:

- амплітуда першої гармоніки нерівностей $A_1 = 0.005$ м;
- амплітуда другої гармоніки нерівностей $A_2 = 0.002$ м;
- довжина рейки $L = 25$ м;
- маса першого тіла $m_1 = 8.82$ т;
- маса другого тіла $m_1 = 25.8$ т;
- жорсткості $c_1 = 7000$ кН/м, $c_2 = 2600$ кН/м;
- демпфування $b_1 = 60$ кН·с/м, $b_2 = 125$ кН·с/м.

***Теоретичні відомості***

Необхідні теоретичні відомості викладені у матеріалах лабораторної роботи 8.

***Порядок виконання***

Порядок виконання аналогічний до лабораторної роботи №10.

## Лабораторна робота № 12

## Тема: Побудова та дослідження моделей дискретних систем

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження моделей дискретних систем, закріпити навички побудови схематичних моделей на основі мереж Петрі.

***Завдання для самостійного виконання***

В якості об’єкта моделювання розглянути світлофор. Побудувати:

– мережу Петрі об’єкта;

– на основі мережі Петрі побудувати імітаційну модель об’єкта засобами діаграми Chart бібліотеки Stateflow.

***Теоретичні відомості***

Бібліотека Stateflow дозволяє створювати дискретні моделі на основі станів, де графічний блок Chart призначений для моделювання систем, які представляють F-схеми: класичний (Classic), автомати Мілі

(Mealy) або Мура (Moor). Щоб скористатись діаграмою, необхідно відкрити вікно моделі, відкрити бібліотеку Stateflow, перетягти блок Chart у вікно моделі або командою sfnew() у командному вікні, наприклад: sfnew('-Mealy','MyModel').

Подвійний клік відкриває вікно діаграми, у якому на лівій панелі розташовані графічні компоненти для побудови моделі (рис. 12.1).

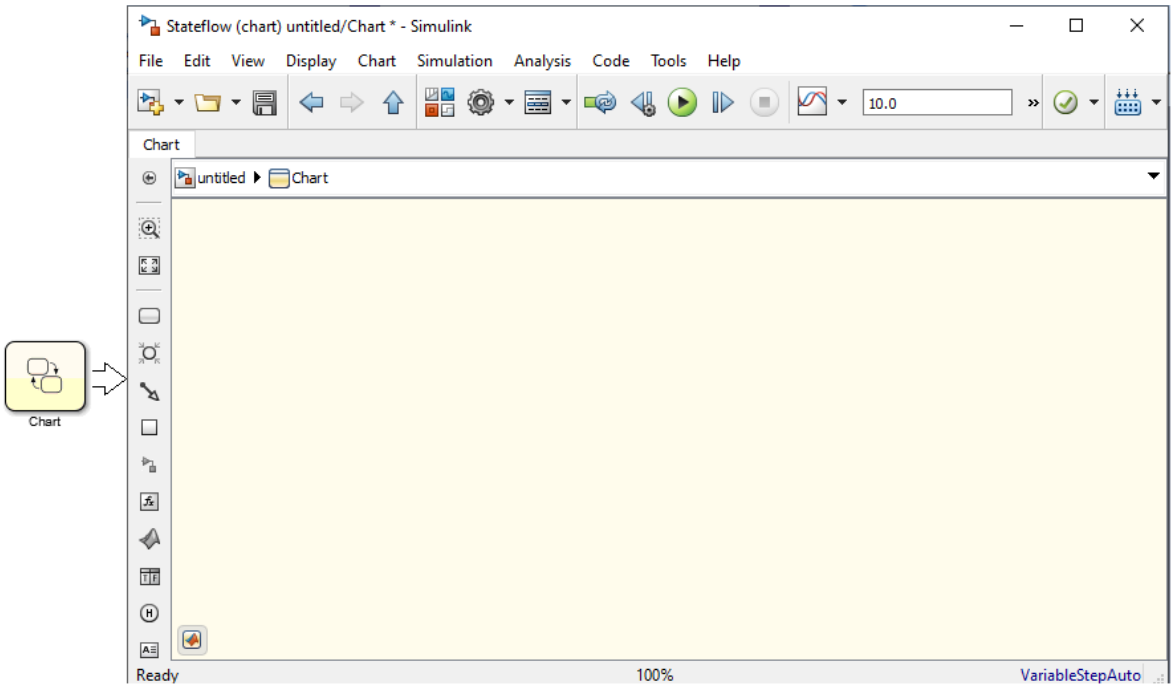



Рис. 12.1. Вікно діаграми Chart

Для отримання додаткової інформації див. sfnew. Виберіть один із наступних типів кінцевого автомата:

- Класичний — тип машини за замовчуванням. Надає повний набір семантики для діаграм MATLAB і діаграм C.
- Mealy – тип машини, у якій вихід є функцією входів і стану.
- Мур – тип машини, в якій вихід є функцією стану.

Stateflow Chart діаграма складається з набору графічних (стани, переходи, з'єднання) та неграфічних (події, дані, програмні коди) компонентів (рис. 12.2.)

*Стан* (State) – режим, у якому модельована система перебуває деякий час, протягом якого вона поводиться однаково. Наприклад, система може бути в одному з двох станів: працездатному і непрацездатному. Кожен стан має свого батька і може мати нащадків (стани нижчого рівня). Якщо стан є єдиним, його батьком є сама SF-

діаграма, звана також кореневої діаграмою. Отже діаграма може складатися з вкладених один одного станів.

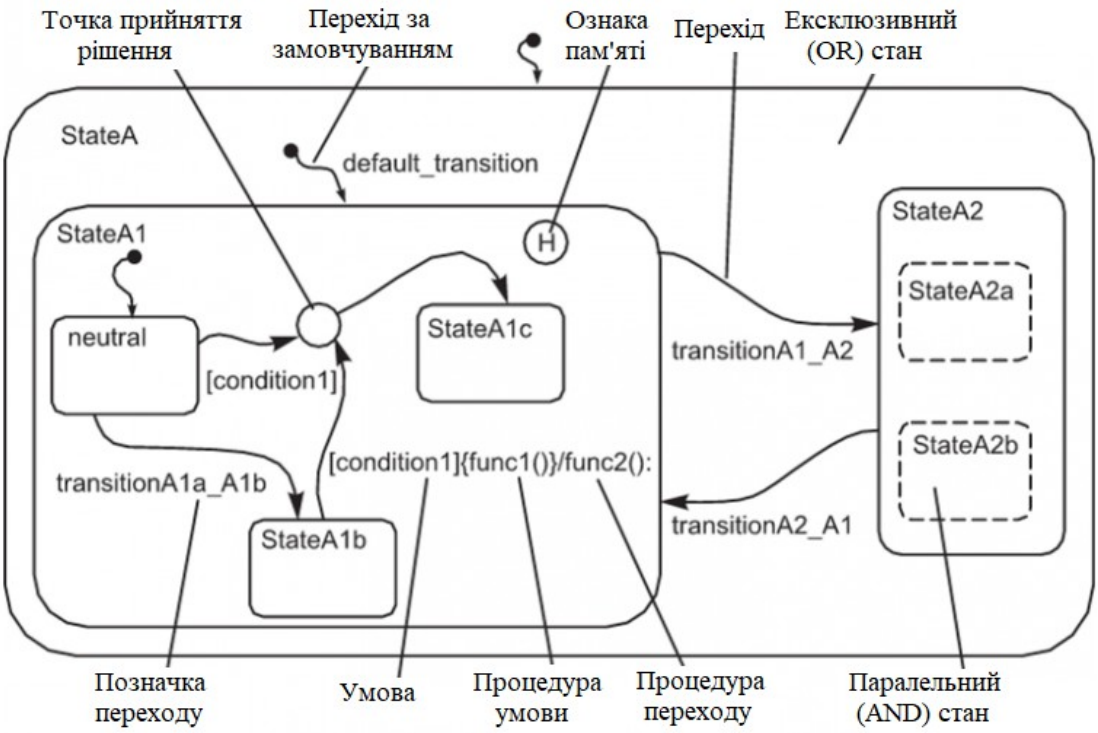



Рис. 12.2. Компоненти діаграми Chart

Стани можуть бути нейтральними (neutral) та зайнятими (engaged). Є два типи станів: паралельні, що існують одночасно (AND) і ті, що взаємно виключають один одного (OR). Наприклад, взаємно виключають один одного стани вимикача «Включено» і «Вимкнено».

Кожен стан має ознаку пам’яті, чи свою хронологію (history), що забезпечує визначення майбутнього переходу до іншого стану з урахуванням інформації про минуле системи. Ознака пам’яті – це функція, що має найвищий пріоритет у виконанні.

*Перехід* (Transition) – зміна стану, що зазвичай викликається деякою подією. Переходи показуються лініями зі стрілками, що вказують напрямок переходу. Найчастіше перехід свідчить про зміну стану системи. Переходи не мають своєї кнопки на панелі інструментів. Вони створюються, коли ви, вказавши курсором миші (при натиснутій лівій кнопці) об’єкт, від якого починається стрілка переходу, рухаєте курсор до іншого об’єкта. Переходи мають мітки, що описують обставини або умови, за яких відбувається перехід від одного стану до

іншого. Наприклад, мітка clutch_engaged супроводжує перехід від нейтрального стану до зайнятого.

Для зазначення альтернативних шляхів переходу систем з одного стану до іншого служать ознаки альтернативи (connective junction). Це графічні об'єкти у вигляді чорного кола, зафарбованого всередині червоним кольором, що мають стрілку переходу. Застосування таких об'єктів унеможливлює перетинання переходів і спрощує побудову SF-діаграм.

*Подія* (Event) – дія, що відбувається в системі або поза нею. Наприклад, події можуть бути пов'язані з фактами виявлення відмови або відновлення елементів системи. Події керують виконанням діаграми Stateflow і можуть запустити перехід, або може запустити дію. Подія не є графічним об'єктом, для візуалізації події можна використовувати позначки переходів, пов'язані. Кожна подія має бути визначена за допомогою тієї чи іншої умови, записаної як логічний вираз.

Події мають властивості. Головною є властивість видимості події. Залежно від значення властивості видимості події можуть бути наступних типів:

• локальні, тобто видимі лише у межах заданої SF-діаграми;

• вхідні – передані в SF-діаграму з моделі Simulink;

• вихідні – передані з SF-діаграми в модель Simulink;

• експортовані – передані із SF- або Simulink-моделі у зовнішню програму;

• імпортовані – отримані із зовнішніх програм.

Редактор SF-програм має меню Add, за допомогою якого можна задати тип події та вказати його властивості.

*Дія* (Action) – це результат виконання будь-якої частини діаграми. Дія не є графічним об'єктом. Для опису дії є спеціальна мова процедур – Action Language. При цьому дія може бути задана як виклик функції, завданням певної події або переходу тощо.

*Дані* (data) у SF-моделі є числовими значеннями. Дані не є графічними об'єктами та безпосередньо на SF-діаграмі не вказуються. Вони можуть створюватися на будь-якому рівні ієрархії моделі та мають властивості.

***Опис компонентів***

*Стан*. Усередині прямокутника, що означає стан, вказуються мітки. Мітки станів починаються з імені стану. Крім того, мітка стану

може мати символ / (слеш) та одне або кілька ключових слів. За наявності символу / (слеш) ключове слово Entry може бути відсутнім. Нижче наведена структура та ключові слова (табл.12.1), які визначають різні типи дій, пов'язаних із станом:

*name*/
entry:*entry actions*
during:*during actions*
exit:*exit actions*
bind:*data* and *events*
on *event_or_message_name*:*on event_or_message_name actions*

*Таблиця 12.1*

**Ключові слова програмування станів**

| **Ключове слово** | **Параметри, що задаються** |
|---|---|
| немає | ідентифікатор стану – унікальне посилання на стан із необов'язковою косою рискою |
| entry or en | entry actions – дії, які виконуються під час входу в результаті переходу до цього стану |
| during or du | during actions – дії під час активності стану |
| exit or ex | exit actions – дії під час виходу зі стану |
| bind | data and *events* – прив'язує вказані дані або події до цього стану. Прив'язані дані можуть бути змінені та транслюватися лише цим станом або його дочірніми елементами, але можуть бути прочитані іншими станами |
| on | *event_or_message_name* – подія або повідомлення<br>*event_or_message_name actions* – дії, що виконуються, коли стан активний і відбувається зазначена подія або присутнє повідомлення |

Після введення ключового слова, яке визначає тип дії, необхідно вказати, які саме дії виконуватимуться. Складові дії кожного типу поділяються символом точка з комою. Після кожного ключового слова ставиться двокрапка.

Розглянемо приклад, наведений на (рис. 12.3).

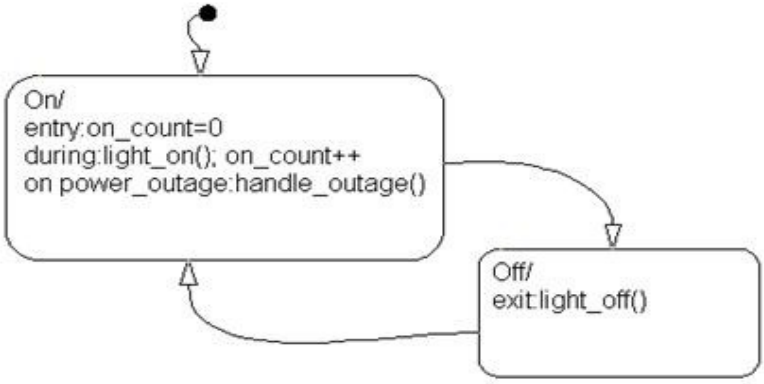



Рис. 12.3. Приклад опису станів перемикача

У цьому прикладі імена станів – On і Off. Далі наведені такі дії.

Entry Action (Дія під час входу в стан). Стан On діє під час входу on_count=0 означає, що значення on_count встановлюється 0, коли виконується вхід у стан On.

During Action (Дія під час активності стану). Стан On має дві During дії: light_on() та on_count++, які виконуються під час дії стану On.

Exit Action (Дія під час виходу зі стану). Стан Off має дію під час виходу light_off(), яка виконується, коли виконується вихід із стану Off.

On Event_Name Action (Дія під час події Event_Name). Стан On має дію, що коли подія power_outage відбудеться, дія handle_outage() виконається.

Переглянути або змінити властивості стану можна, натиснувши на нього правою кнопкою миші і вибравши пункт Properties.

*Перехід.* Перехід відбувається у разі настання події, але з урахуванням істинності умови, яка записується на місті мітки переходу, якщо її визначено. Визначення імені події необов’язкове. Якщо ім’я події не вказано, перехід відбудеться у разі настання будь-якої події. Складові події визначаються за допомогою використання логічного оператора АБО (|). У прикладі, наведеному на (рис. 12.4) у разі настання події E відбувається перехід із стану On у стан Off, якщо при цьому буде істинною умова [off_count==0].

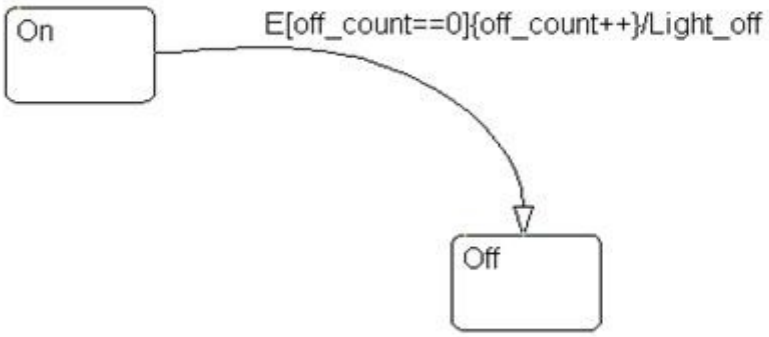



Рис. 12.4. Перехід між станами моделі перемикача

Дії переходу позначаються символом « \ ». У наведеному прикладі, якщо умова [off_count==0] істинна і стан Off досягнуто, здійснюється дія переходу Light_off.

Можливість спрацьовування переходу наведена у табл.12.2.

*Таблиця 12.2*

**Умови спрацьовування переходу**

| Мітка переходу | Перехід став можливим, якщо… |
|---|---|
| Подія | Подія сталася |
| Подія та умова | Подія відбулася і умова істинна |
| Умова | Будь-яка подія відбулася і умова істинна |
| Дія | Будь-яка подія сталася |
| Не визначена | Будь-яка подія відбулася |

Переходи можуть бути між станами одного та різних рівнів ієрархії (рис. 12.5), підключені до з’єднань (рис. 12.6, *а, б*), бути циклічними (рис. 12.7), безумовними.

Безумовні переходи переважно використовуються для визначення, який послідовний (АБО) стан має стати активним, коли є неоднозначність між двома або більше АБО-станами. Безумовні переходи мають об’єкт-адресат, але вони не мають об’єкта-джерела.

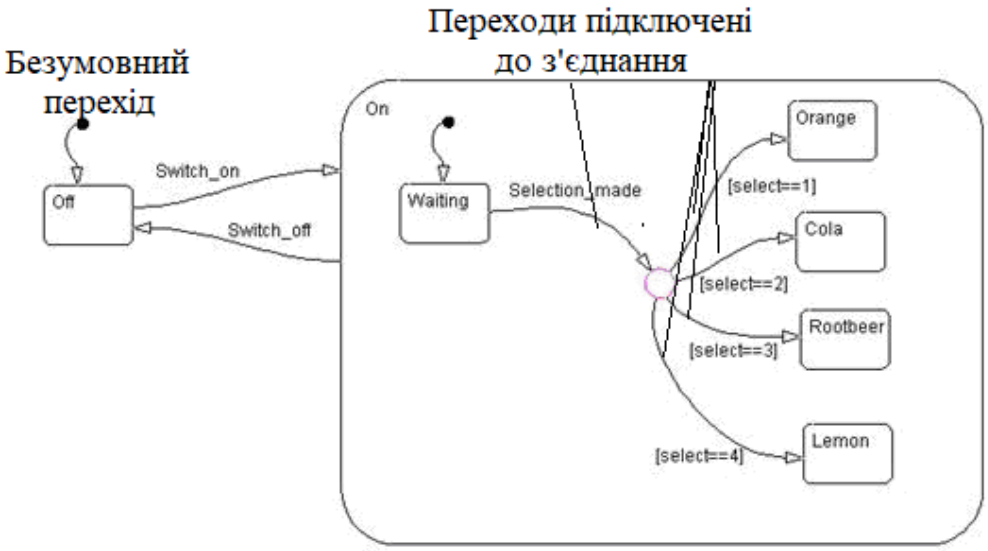



Рис. 12.5. Типи переходів (автомат з продажу напоїв)

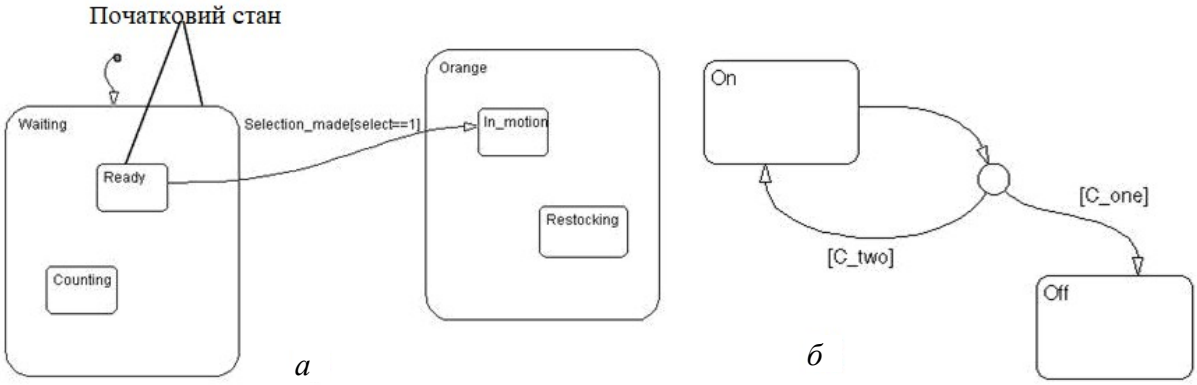



Рис. 12.6. Типи переходів: *а* – між підстанами; *б* – циклічний перехід

За рахунок з’єднання можна створювати циклічний процес for (рис. 12.7).

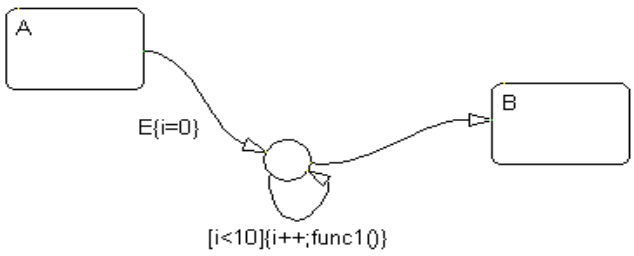



Рис. 12.7. Цикл for

Нехай у стані *A* сталася подія *E*. Перехід від стану до *A* до стану дійсний, якщо умова на шляху переходу істинна. Перший сегмент переходу немає умови, але має дію умови. Дія умови {i=0} виконується.

Умова циклічного переходу [i<10] стає істинною і дії умови {i++;func1()} виконуються. Дії умови виконуються доти, доки умова [i<10] стане хибною (виконується цикл for для значень і від 0 до 9, викликаючи кожного разу функцію func1()). Далі виконується вихід із циклу та перехід у стан B.

Безумовний перехід може підключатись і до переходу. На (рис. 12.8) показана Stateflow-діаграма моделі 8-бітного аналого-цифрового перетворювача (АЦП). Розглянемо випадок, коли стан Sensor.Low активний і подія UPDATE сталася. Внутрішній перехід від Sensor до підключення дійсний. Наступний сегмент переходу має дію умови {start_adc()}, що ініціює читання з АЦП. Цикл на другому підключенні повторюється доти, доки істинна умова [adc_busy()]. Цей циклічний перехід використовується для подання затримки, необхідної для читання з АЦП. Затримка може бути представлена і у вигляді окремого стану, але код став би менш ефективний. Дія умови наступного сегмента переходу {sensorValue=read_adc()} встановлює нове значення, що зчитується з АЦП змінної sensorValue. Заключний сегмент переходу визначається значенням змінної sensorvalue. Якщо [sensorValue <100] істинно, адресат – це стан Sensor.Low. Якщо [sensorValue >200] істинно, адресат – це стан Sensor.High. Інакше адресат – це стан Sensor.Normal.

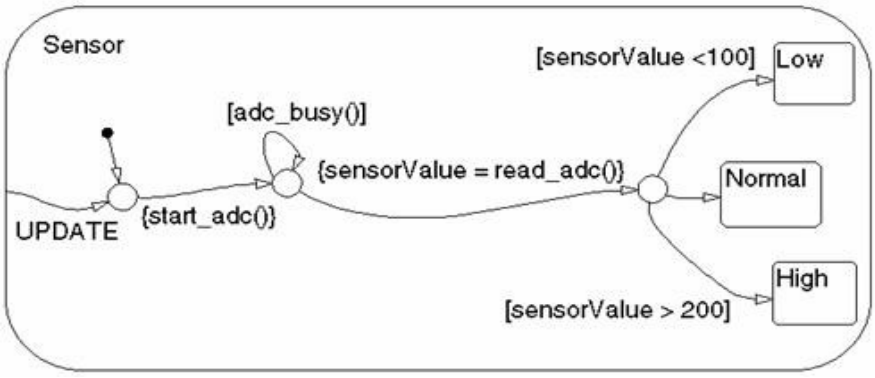
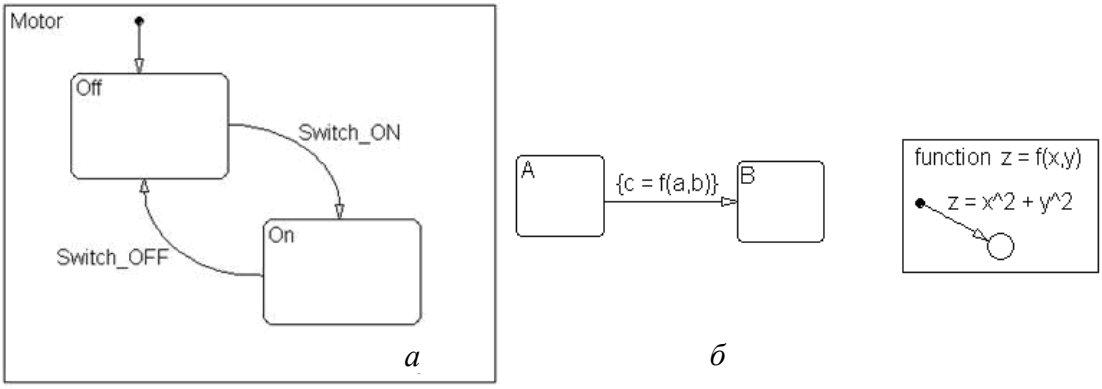



Рис. 12.8. Модель 8-бітного аналого-цифрового перетворювача

З'єднання, що має пам'ять (позначка H), запам'ятовує попередній активний стан.

*Блоки* використовуються для покращення графічного представлення діаграми і дозволяють згрупувати об'єкти під одним іменем (рис. 12.9, *а*).

*Графічні функції* (прямокутник з позначкою *fx*) – це функції, визначені графом переходів (рис. 12.9 *б*). Наявність графічних функцій забезпечує зручність і підвищує виразність мови дій. В прикладі функція z = f(x,y) викликається дією умови під час переходу стану A до стану B.




*а* *б*

Рис. 12.9. Приклади: *а* – блоку; *б* – графічної функції

*Управління діаграмою* може виконуватись за часом (в термінах часу) або на основі неявних подій.

Часова логіка управляє виконанням графіка термінах часу. У станах активності та переходах можна використовувати два типи тимчасової логіки:

Щоб задати поведінку діаграми Stateflow на основі логіки часу, використовуються оператори в on станах дії:

оператор(n, T),

де *n* – ціле число > 0;

- параметр *T* може приймати значення: *E* – подія; tick, sec, msec, usec – такти, секунди, мілі- та мікросекунди;
- параметр *C* може приймати значення:

*Опис операторів.*

– after(n, T) – повертає true, якщо n разів відбулася подія E або діаграма активувалась, або пройшов заданий час.

Приклади.

1. on after(3,E): disp('ON'); – відображає стан на 3-му запуску події E.
2. after(7,tick)[temp > 98.6] – перехід відбувається, коли стан в 7-й раз став активним, але за умови temp > 98.6.

3. on after(12.3,sec):temp = LOW; – якщо стан активний на протязі 12.3 с., змінна temp приймає значення LOW.

– at(n,T) – повертає true, якщо подія відбулась точно n раз або точно пройшов заданий час. Запис подібний до оператора after().

– before(n,T) – повертає true, якщо подія відбулась менше, ніж n разів або якщо час менше встановленого.

– every(n,E) – повертає true при кожній n-й появі події або n-му значенні часу.

**Приклад**

on every(12.3,sec): temp = temp+5; – значення temp збільшується на 5 кожні 12.3 с.

– temporalCount(T) – повертає кількість разів настання події або час, коли стан стає активним.

**Приклад**

en,du: M(temporalCount(tick)+1) = u; – зберігає вхідні значення u в масиві M, коли стан стає активним.

– elapsed(sec) – повертає відрізок часу до активації події (еквівалентно temporalCount sec ()).

– et – альтернативне виконання elapsed(sec).

**Приклад**

E{disp(et);} – діаграма оброблює подію E, відображається попередній час до активації стану.

– count(C), де C – вираз, що приймає значення true або false – повертає кількість разів активації стану, якщо вираз приймає значення true. Якщо вираз приймає значення false або асоційований стан стає неактивним, значення лічильника скидається.

**Приклад**

en,du: y = count(x>5); – підраховує кількість спрацьовування діаграми, починаючи з виконання умови x > 5.

– duration(C [,T]) – повертає відрізок часу, який пройшов, починаючи з виконання умови C.

**Приклад**

1. [duration(x>=0) > 0.1] – перехід із стану, коли x>=0 на протязі часу > 0.1 с.

2. en,du: y = duration(x>5,msec); – зберігає кількість мілісекунд, починаючи з виконання умови x>5,msec і активації стану.

**Приклад**

Розглянемо систему (Enabled Subsystem), яка представлена Stateflow-діаграмою, що складається з двох станів *A* і *B* та управляється зовнішнім сигналом (Signal Elitor) (рис. 12.10).

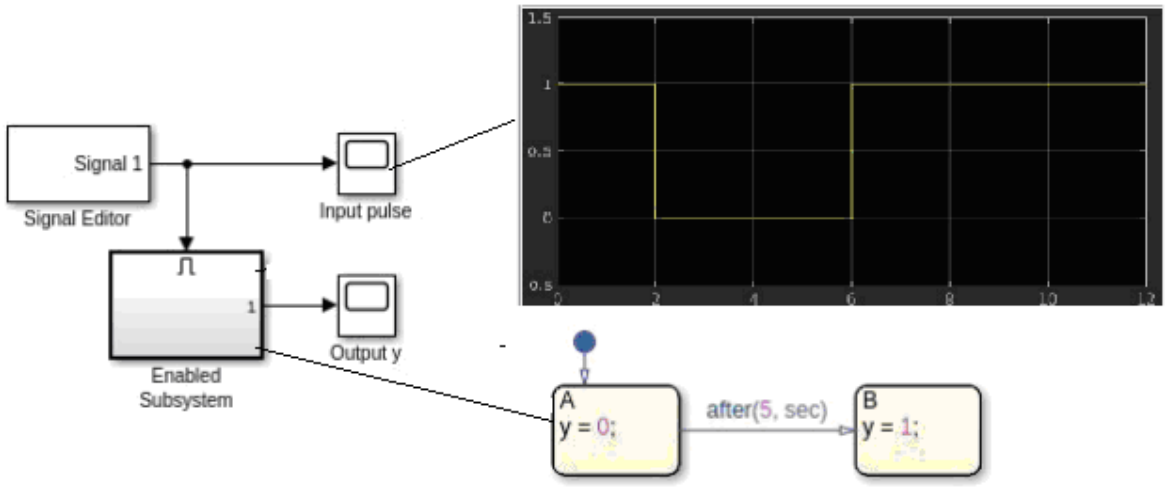



Рис. 12.10. Модель системи

Блок Signal Editor (Simulink) представляє імпульсний сигнал, що надає вхідному сигналу діаграми Stateflow наступні характеристики:

- сигнал включає підсистему t = 0;
- сигнал відключає підсистему t = 2;
- сигнал повторно включає підсистему t = 6.

Діаграма містить перехід, який використовує оператор after().

Коли вхідний сигнал включає підсистему (час t = 0), стан *A* стає активним. Коли система активна, час збільшується. Для 2 <t <6 минулий час залишається замороженим у 2 секунди, тому що система відключена. З просуванням часу його значення знову починає збільшуватись (рис. 12.11).

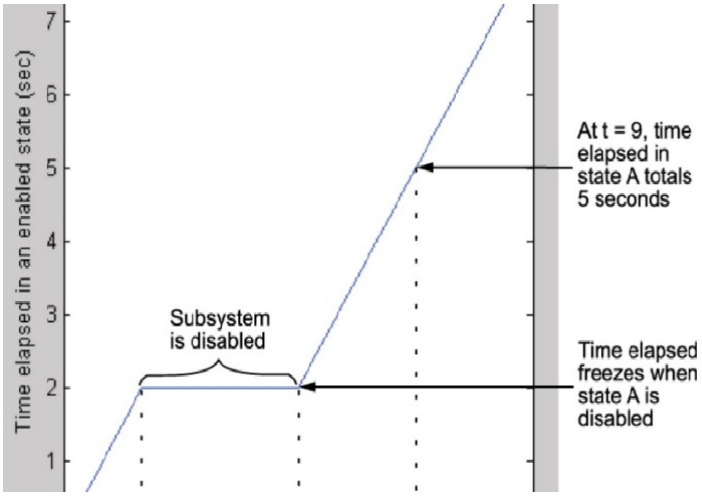



Рис. 12.11. Графік зміни часу

Implicit events є вбудованими подіями, що відбуваються під час виконання діаграми, коли:

- діаграма прокидається;
- діаграма входить у стан і стан стає активним;
- діаграма виходить із стану і стан стає неактивним;
- діаграма надає значення внутрішньому об’єкта даних.

Ці події неявні, тому що не задаються або ініціюються явно. Неявні події є дочірніми елементами діаграми та відображаються тільки у діаграмі вищого рівня.

Для управління на основі неявних подій використовуються наступні оператори:

– change((data_name) – генерує неявну локальну подію, коли діаграма встановлює значення data_name;

– enter(state_name) – генерує неявну локальну подію, коли стан state_name стає активним;

– exit(state_name) – генерує неявну локальну подію, коли стан state_name стає неактивним.

**Приклад**

Ключове слово tick визначає неявну подію, згенеровану, коли діаграма «прокидається» в симуляції дискретного часу. В цій діаграмі Fan і Heater паралельні (AND) стани. Кожен стан має кілька підходів, On і Off. Спочатку стани Fan.Off і Heater.Off активні. Щоразу, коли діаграма активна, генерується tick подія Heater.On. Точно так само четвертий tick ініціює перехід від Fan.Off до Fan.On (рис. 12.12).

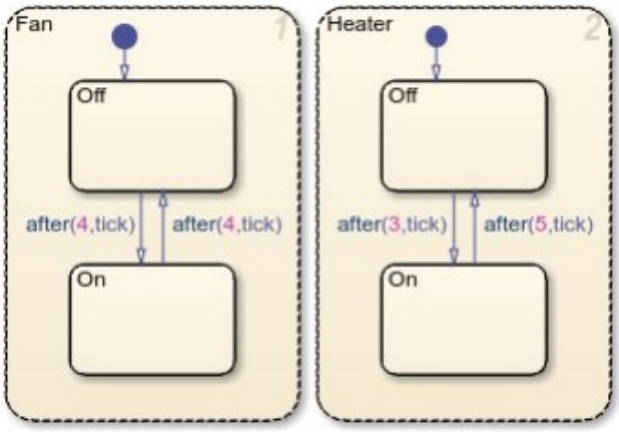



Рис. 12.12. Приклад управління на основі неявних похід

Після створення моделі необхідно її налаштування, зберегти та запустити на виконання. Інструменти налаштування діаграми знаходяться в головному пункті меню Chart.

***Порядок виконання***

На Землі погода буває лише трьох типів: дощ, сонячно чи сніг. Якщо сьогодні сонячний день, то завтра буде дощ чи сніг із однаковою ймовірністю. Якщо сьогодні дощ (або сніг), то половина шансів за те, що така сама погода буде завтра. Якщо ж відбувається зміна, то лише у половині випадків вона призводить до сонячного дня.

**Розв'язок**

1. Побудуємо мережу Петрі для даної задачі. Визначимо стани та переходи системи, яка визначає погоду (табл.12.3).

*Таблиця 12.3*

**Стани та переходи системи**

| Стан | Призначення | Перехід | Умова | Дія |
|---|---|---|---|---|
| $P_1$ | сонячно – *sunny* | $T_1$ | $p \le 0.5$, | $P_1 \Rightarrow P_2$ |
| | | | $p > 0.5$, | $P_1 \Rightarrow P_3$ |
| $P_2$ | дощ – *rain* | $T_2$ | $p \le 0.5$ | $P_2 \Rightarrow P_2$ |
| | | | $0.5 < p \le 0.75$ | $P_2 \Rightarrow P_1$ |
| | | | $p > 0.75$ | $P_2 \Rightarrow P_3$ |
| $P_3$ | сніг – *snow* | $T_3$ | $p > 0.5$ | $P_3 \Rightarrow P_3$ |
| | | | $0.5 < p \le 0.75$ | $P_3 \Rightarrow P_1$ |
| | | | $p > 0.75$ | $P_3 \Rightarrow P_2$ |

Відповідна мережа Петрі представлена на рис. 12.13.

2. На основі мережі Петрі побудуємо Stateflow-діаграму (рис. 12.14).
3. Початковий стан – sunny, що свідчить про наявність переходу за замовчанням (Default transition) графічного об'єкта до стану sunny. Цей перехід супроводжується дією переходу (Transition action)/nrain=0; nsunny = 0; nsnow=0. Ця дія встановлює в нуль лічильники кількості дощових, сонячних днів та днів, коли йде сніг.

Під час входу цей стан виконує дія nsunny++, тобто кількість сонячних днів зростає на одиницю.

$T_2$

Рис. 12.13. Мережа Петрі системи

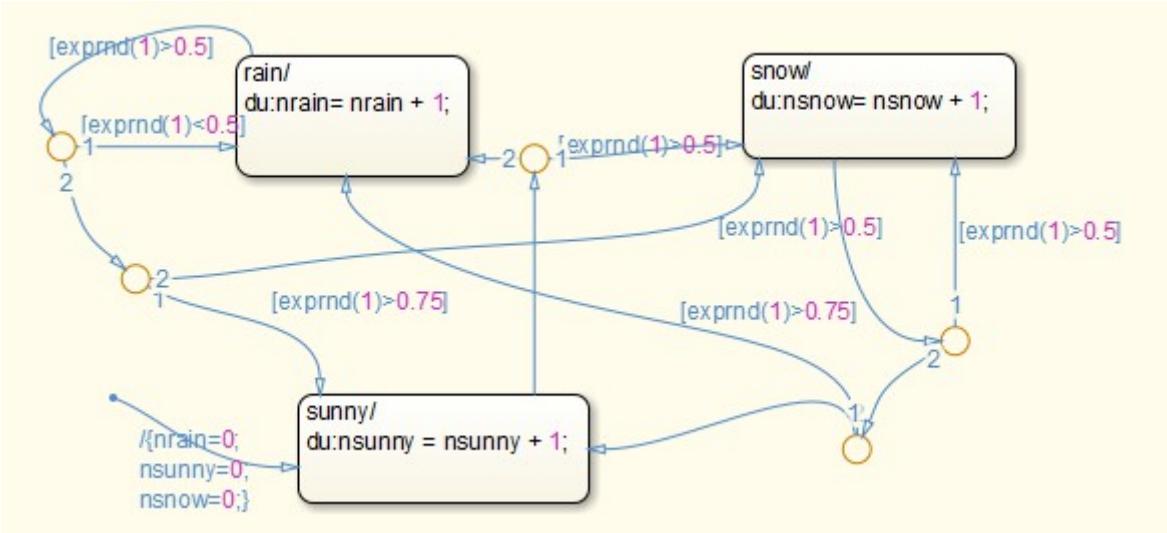



Рис. 12.14. Stateflow-діаграма системи

Наступна подія event переводить діаграму в підключення Connective Junction, що з'єднується, звідки з ймовірністю 0,5 діаграма переходить в стан snow і з ймовірністю 0,5 – в стан rain. Імовірнісний перехід заснований на використанні умови exprnd(1) > 0.5, де exprnd(1) – стандартна функція, що генерує випадкове число з діапазону (0,1) з експоненціальним законом розподілу і математичним сподіванням, рівним 1.

3. Запам'ятати програму на диску, встановити час моделювання, рівний 700. З пункту Simulation – Stateflow Animation вибрати повільний режим відтворення – Slow та запустити на виконання. Активні стани та спрацьовуючі переходи будуть виділятись жирним, що дозволить відслідкувати зміну погоди.

## Лабораторна робота № 13
## Тема: Побудова та дослідження дискретних імітаційних моделей на прикладі реальних систем

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження дискретних імітаційних моделей на прикладі реальних систем, закріпити та розширити навички застосування діаграми Chart бібліотеки Stateflow.

***Завдання для самостійного виконання***

В якості об’єкта дослідження розглянути систему охолодження (підігріву), що складається з 2-х охолоджуючих (підігріваючих) пристроїв, за рахунок чого підтримує температуру у заданому діапазоні. Якщо температура знаходиться у заданих межах, то працює один пристрій, коли температура починає збільшуватись (зменшуватись), то вмикається другий пристрій. Система працює під дією зовнішнього імпульсного сигналу. Представити результати роботи: системи зміну станів, системи графік зміни температури.

***Теоретичні відомості***

Основні теоретичні відомості викладені у завданні №12.

Діаграми Stateflow можуть взаємодіяти з іншими блоками та об’єктами у моделі Simulink за допомогою:

– обміну даними через вхідні та вихідні з’єднання;

– обміну (імпорту або збереження) з робочим простором Workspace.

Щоб визначити вхідні або вихідні дані, у пункті верхнього меню діаграми Chart виберіть Add Inputs&Outputs і відповідно Data Input або Data Output from Simulink. Відкриється вікно налаштування даних (рис. 13.1), де необхідно визначити їх основні параметри:

- Name – ім’я;
- Scope – область застосування:
  - Local – дані визначені тільки в поточній діаграмі;
  - Constant – постійне значення, доступне лише для читання, видиме батьківському об’єкту Stateflow і його дочернім об’єктам;
  - Parameter – константа, значення якої визначено в Workspace або отримано з параметра блоку Simulink, який визначається та ініціалізується в батьківській замаскованій підсистемі. Об’єкт даних Stateflow повинен мати те саме ім’я, що і змінна MATLAB, запис словника даних Simulink або параметр Simulink.
  - Input – вхідний аргумент функції, якщо батьківська функція – графічна функція, таблиця істинності або функція MATLAB. У протилежному випадку модель Simulink надає дані діаграми через вхідний порт блоку Stateflow;

- Output – вихідне значення функції, якщо батьківська функція є графічною функцією, таблицею істинності або функцією MATLAB. У протилежному випадку діаграма надає дані моделі Simulink через вихідний порт на блоці Stateflow;
- Data Store Memory – об'єкт даних, який зв'язується із сховищем даних Simulink, є глобальною змінною. Усі блоки в моделі можуть отримати доступ до неї;
- Temporary – дані, які зберігаються тільки під час виконання функцій. Визначаються лише для графічних функцій, таблиць істинності чи функцій MATLAB.

– Port – індекс порту, зв'язаного з об'єктом даних. Ця властивість застосовується лише для вхідних та вихідних даних.

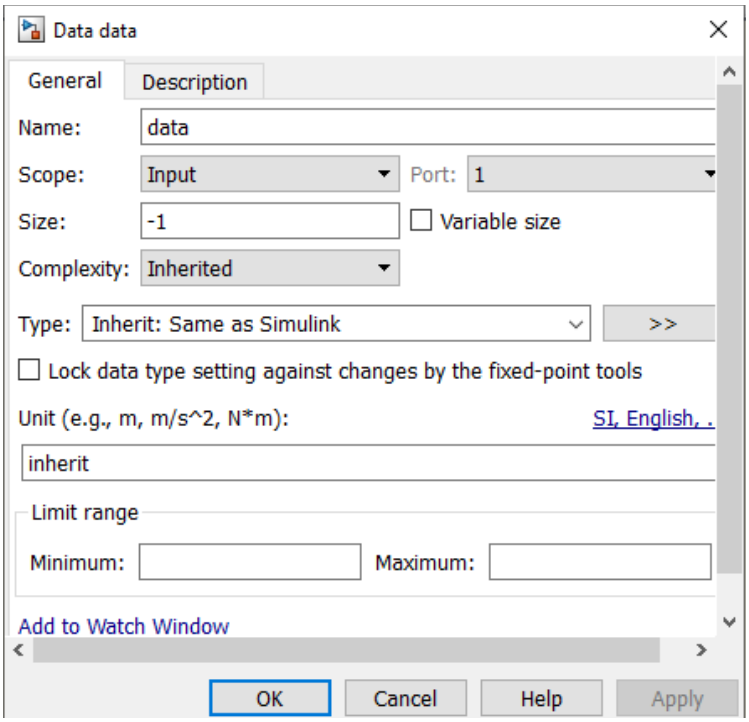



Рис. 13.1. Вікно налаштування даних

– Complexity – складність вказує, чи приймає об'єкт даних складні значення:
  - Off – дані не приймають складні значення;
  - On – дані приймають комплексні значення;
  - Inherited – об'єкт даних успадковує налаштування складності від блоку Simulink.
– Type – тип даних;

– Limit range – встановлення границь зміни значень.

Ініціювати дії Stateflow-діаграми можна за допомогою подій, які поділяють на:

– вхідні – транслюється в діаграму Stateflow іззовні діаграми;
– вихідні – події, які відбувається в діаграмі Stateflow, але транслюється в блок Simulink;
– локальні – події, які можна створити в будь-якому місці діаграми Stateflow, але видимі лише в батьківському об’єкті та його нащадках.
– Event Input

Для додання події в пункті головного меню діаграми Chart виберіть Add Inputs&Outputs і відповідно Event Input або Event Output from Simulink, відкриється діалогове вікно для налаштування події (рис. 13.2).

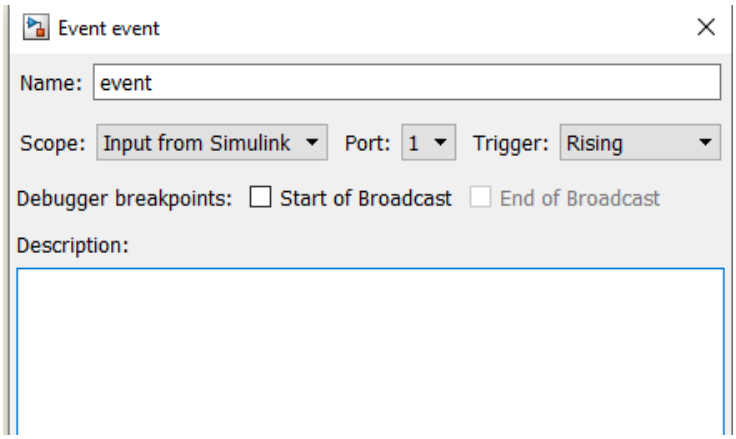



Рис. 13.2. Діалогове вікно налаштування події

Параметри налаштування Name, Scope, Port подібні до параметрів даних. Параметр Trigger визначає тип сигналу, який запускає подію введення або виведення:

– Rising – за зростанням, діаграма активується, коли сигнал керування змінюється з нуля або негативного значення на позитивне значення;

– Falling – за спаданням, діаграма активується, коли сигнал керування змінюється з позитивного значення на нуль або негативне значення;

– Either – за зростанням або спаданням, діаграма активується, коли сигнал керування перетинає нуль під час зміни в будь-якому напрямку.

Переглянути та редагувати дані та події можна у вікні Explorer, яке можна відкрити, клацнувши правою кнопкою миші на діаграмі і вибрати відповідний пункт з випадаючого меню (рис. 13.3).

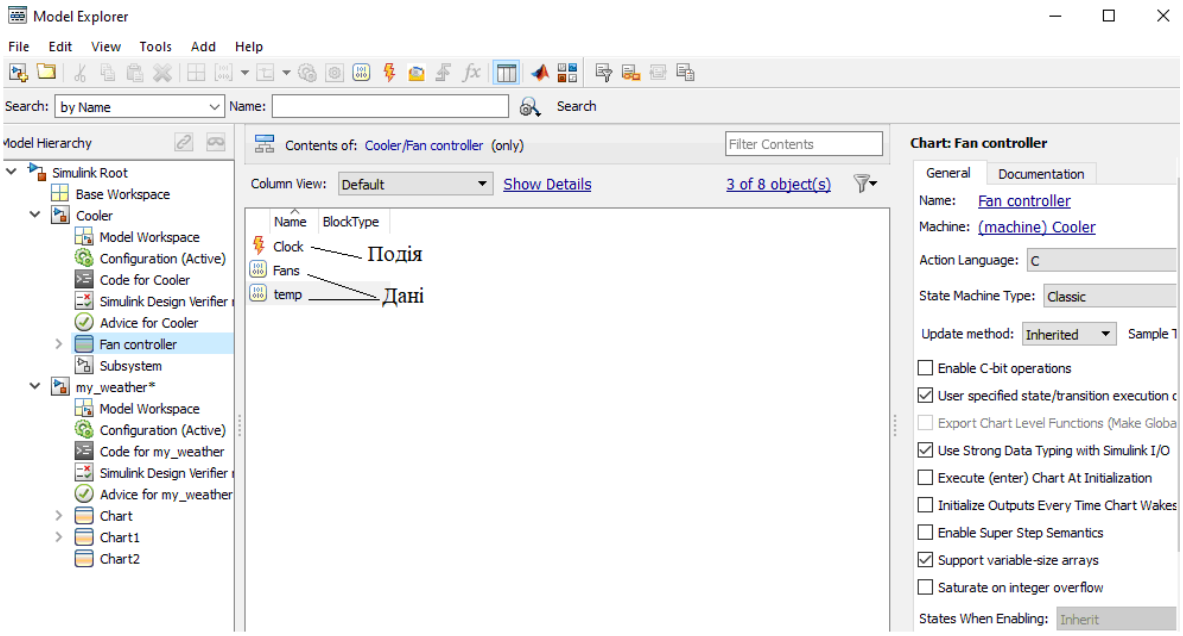



Рис. 13.3. Вікно Explorer (провідника) діаграми Stateflow

На панелі контенту відображається перелік всіх подій та даних діаграми, на правій панелі – параметри відповідних даних чи подій.

Подію введення використовують для активації діаграми, коли модель потребує регулярного або періодичного виконання діаграми.

***Порядок виконання***

Порядок виконання роботи:

– представити логіку моделі у вигляді мережі Петрі. Система може знаходитись у виключеному та активному станах. У активному стані залежно від поточного значення температури може працювати або одна або дві установки. Під час досягнення верхньої границі температури система вимикається з метою економії ресурсів, а потім знову включається для підтримки необхідного температурного режиму;

– самостійно створити діаграму Stateflow для управління устаткуванням;

– створити модель, додавши вхідний управляючий сигнал та елементи виводу результатів роботи.

## Лабораторна робота № 14
## Тема: Побудова та дослідження імітаційних моделей систем масового обслуговування

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження імітаційних моделей систем масового обслуговування з використанням бібліотеки Simulink SimEvents.

***Завдання для самостійного виконання***

Засобами бібліотеки Simulink SimEvents створити модель транспортної мережі та дослідити час очікування транспортних засобів на перехресті. Параметри інтенсивності руху та ймовірності розподілу на перехрестях підібрати самостійно.

Варіант №1

Варіант №2

Варіант №3

Варіант №4

Варіант №5

***Теоретичні відомості***

Теоретичні відомості побудови та дослідження математичних та імітаційних моделей засобами бібліотеки Simulink SimEvent, викладені в навчальному посібнику [6], але в останніх версіях є деякі зміни.

*Генерація та ініціалізація подій*

Для генерації подій застосовується блок Entity Generator. За замовчуванням метод генерації – Time-based. За цим методом блок генерує сутності, використовуючи час між генераціями, зазначений у параметрі Period, з вхідного сигналу або статистичного розподілу. Блок також може створювати сутності з урахуванням подій – Event-based.

Для налаштування дій під час генерації сутності або виходу її з блоку на вкладці Event action у полі Generate action необхідно задати відповідний код.

Приклад генерації періодичних подій наведений на рис. 14.1. На вкладці Entity Generation в полі Time Source встановлено Dialog.

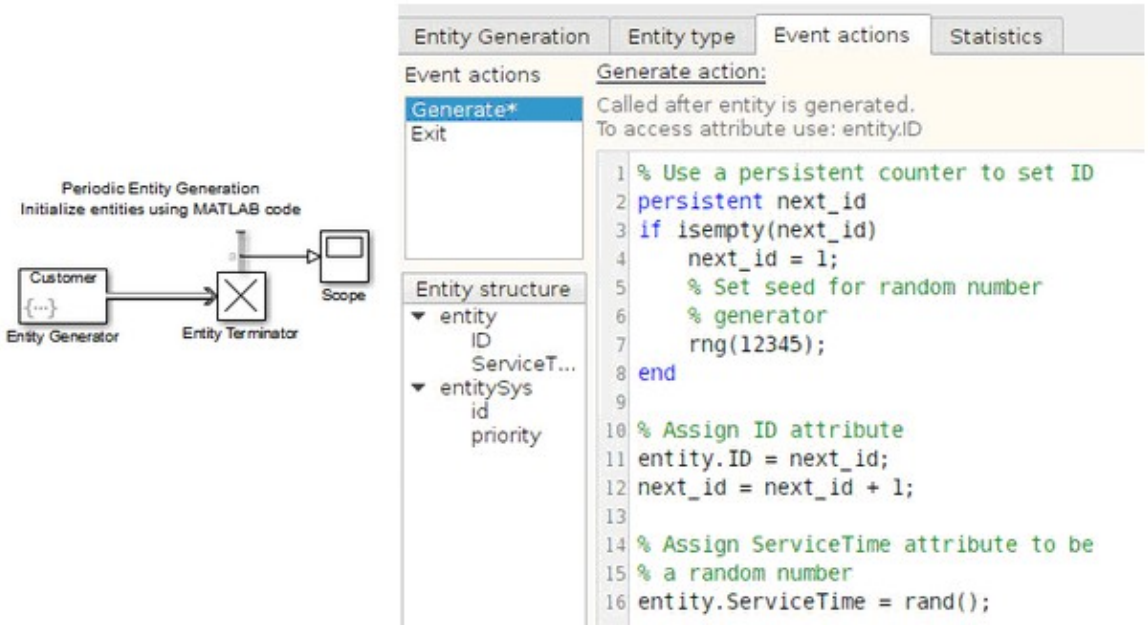



Рис. 14.1. Генерація періодичних подій

Приклад генерації подій за експоненціальним законом розподілу наведений на рис.14.1 та рис. 14.2.

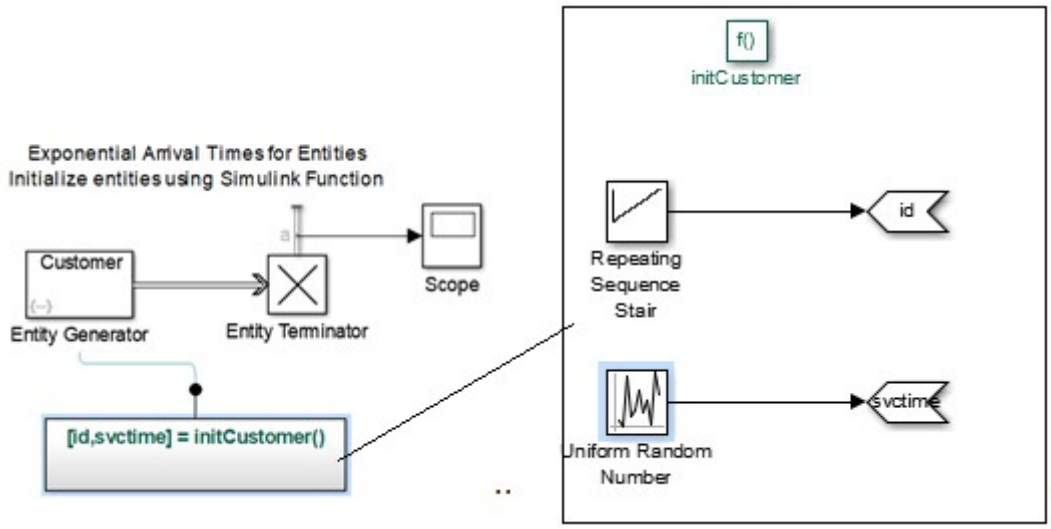



Рис. 14.2. Simulink модель генерації подій за експоненціальним законом

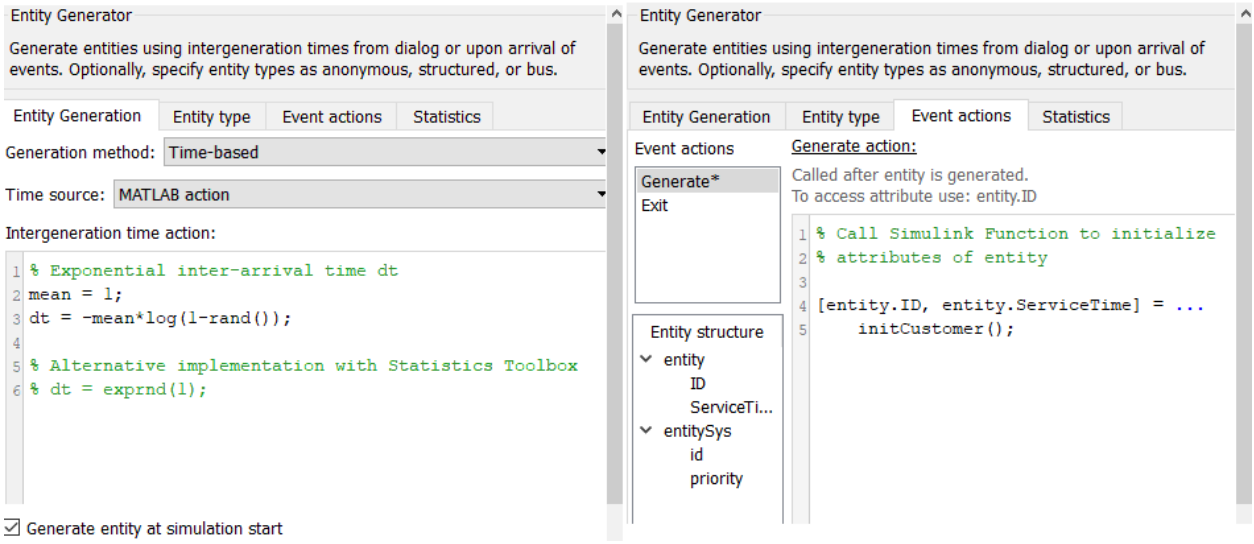



Рис. 14.3. Налаштування генератора подій за експоненціальним законом

Блок серверу також може налаштовуватись на режими обслуговування та типи сутностей. Докладніше це розглянемо на прикладі.

Для моделювання систем масового обслуговування можна використовувати загальні шаблони SimEvents Common Design Patterns.

***Порядок виконання***

Нехай задана транспортна мережа, що має два входи (коричневі кружечки), два виходи (зелені кружечки) та перехрестя (сині кружечки). Швидкість в'їзду транспортних засобів у мережу представлена пуасонівськими процесами зі швидкостями 0.5 для В'їзду_1 і 0.15 для В'їзду_2. Швидкість обслуговування представляє час, який транспортні засоби проводять на кожному перехресті, який виводиться з

експоненційного розподілу і математичним сподіванням, рівним 1. Значення поряд зі стрілочками є ймовірністю вибору маршруту для транспортних засобів на перехресті (рис. 14.4). Створити імітаційну модель мережі та дослідити час очікування на перехресті.

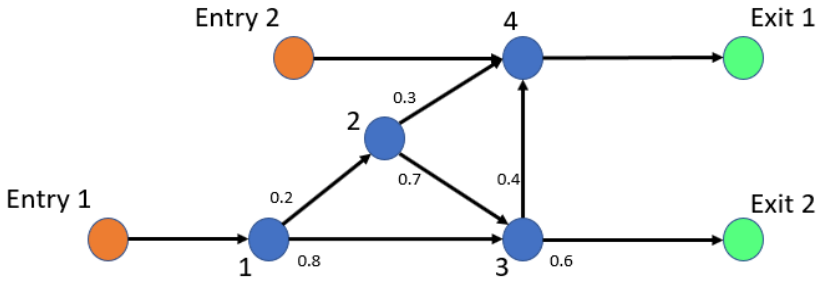



Рис. 14.4. Схема транспортної мережі

*Розв'язок*. Для представлення мережі руху транспортних засобів для створення моделі застосуємо блоки Entity Generator, Entity Server, Entity Queue, Entity Input Switch, Entity Output Switch и Entity Terminator. Загальний вигляд моделі представлений на (рис. 14.5).

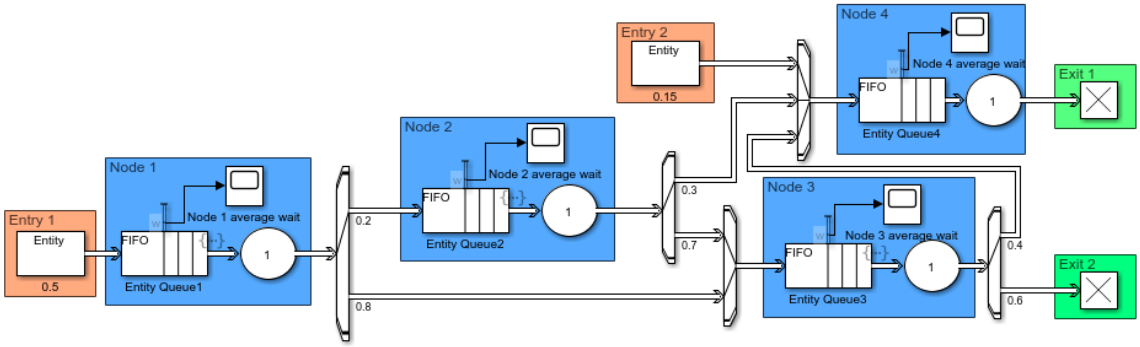



Рис. 14.5. Імітаційна модель транспортної мережі

Коричневі квадратики (входи) задані блоками Entity Generator, які налаштовані на пуасонівський потік (рис. 14.6).

Зелені квадратики (виходи) задані блоками Entity Terminator, що поглинають сутності, які вийшли із системи.

Перехрестя представлення синіми квадратиками, де кожний квадратик є підсистемою, яка складається з черги – Entity Queue, серверу – Entity Server та блоку Scope для графічного представлення часу очікування у черзі. На рис. 14.7, рис. 14.8. представлені вікна налаштування блоків для першого перехрестя.

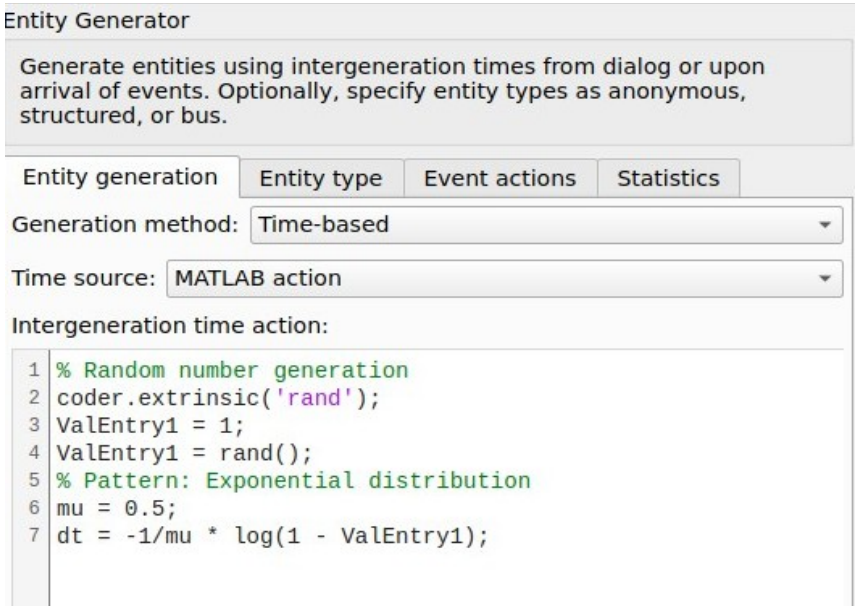



Рис. 14.6. Налаштування генератора на пуасонівський потік з математичним сподіванням 0,5

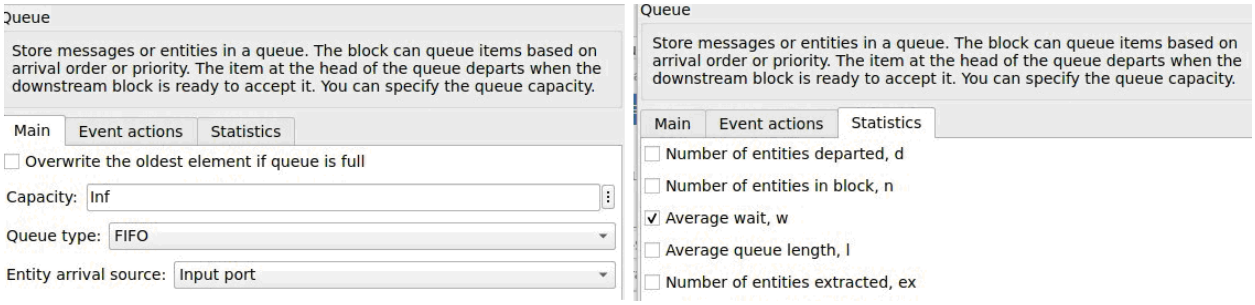



Рис. 14.7. Вікно налаштування блоку черги

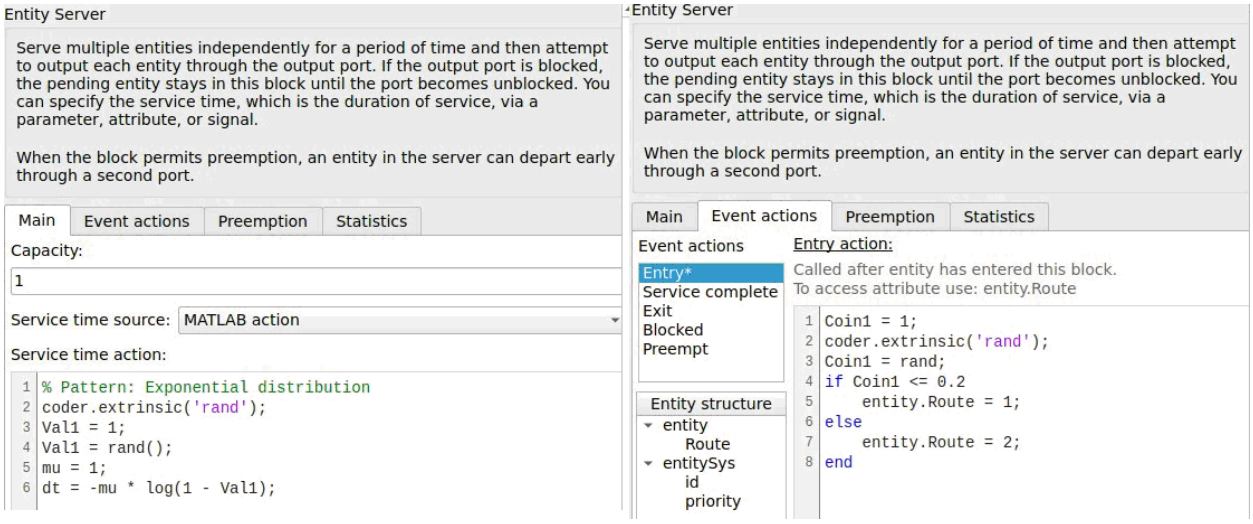



Рис. 14.8. Вікно налаштування блоку серверу

Блоки Entity Input Switch, Entity Output Switch, застосовуються для злиття або розділення потоків сутностей.

Швидкість прибуття транспортних засобів для кожного вузла розраховується за цією формулою:

$$\lambda_i = r_i + \sum_{j=1}^{N} \theta_{ji} \lambda_j ,$$

де $r_i$ – швидкість зовнішніх надходжень в *i*-му вузлі;

$j = \overline{1, N}$ – загальна кількість стрілочок для *i*-го вузла;

$\theta_{ji}$ – ймовірність переходу з *i*-го вузла в *j*-й вузол;

$\lambda_j$ – загальна швидкість надходження транспортних засобів у *j*-й вузол.

В матричній формі рівняння приймає вигляд: $\lambda = R(I - \theta)^{-1}$, де:

$R = [0.5 \quad 0 \quad 0 \quad 0.15]$ – вектор зовнішніх надходжень в кожному вузлі;

$I$ – одинична матриця;

$\theta = \begin{pmatrix} 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0.4 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ – матриця маршрутизації.

Для кожного вузла розраховується середня швидкість прибуття: $\lambda = [0.5 \quad 0.1 \quad 0.47 \quad 0.15]$. Кожен вузол веде себе як незалежна черга M/M/1, а середній час очікування для кожного сайту розраховується за формулою:

$$p_i = \frac{\lambda_i}{1 - \lambda_i} .$$

Середній час очікування для кожного вузла: $p = [1 \quad 0.11 \quad 0.88 \quad 0.58]$.

На графіках можна подивитись час очікування у кожному вузлі.

## Лабораторна робота № 15

## Тема: Побудова та дослідження гібридних моделей

***Мета:*** ознайомити студентів та надати практичних навичок з побудови та дослідження імітаційних моделей складних систем, які представляються гібридними схемами з використанням різних бібліотек Simulink.

***Завдання для самостійного виконання***

Засобами Simulink створити модель неперервно-дискретної (гібридної) системи на прикладі стрічкового конвеєра із змінною

швидкістю, де показано, як об’єднуються неперервний час та дискретна подія.

***Теоретичні відомості***

Теоретичні відомості наведені в лабораторних завданнях 12,13, 14.

***Порядок виконання***

Для реалізації моделі використовуються блоки з SimEvents і Stateflow. Імітаційна модель представлена на (рис. 15.1).

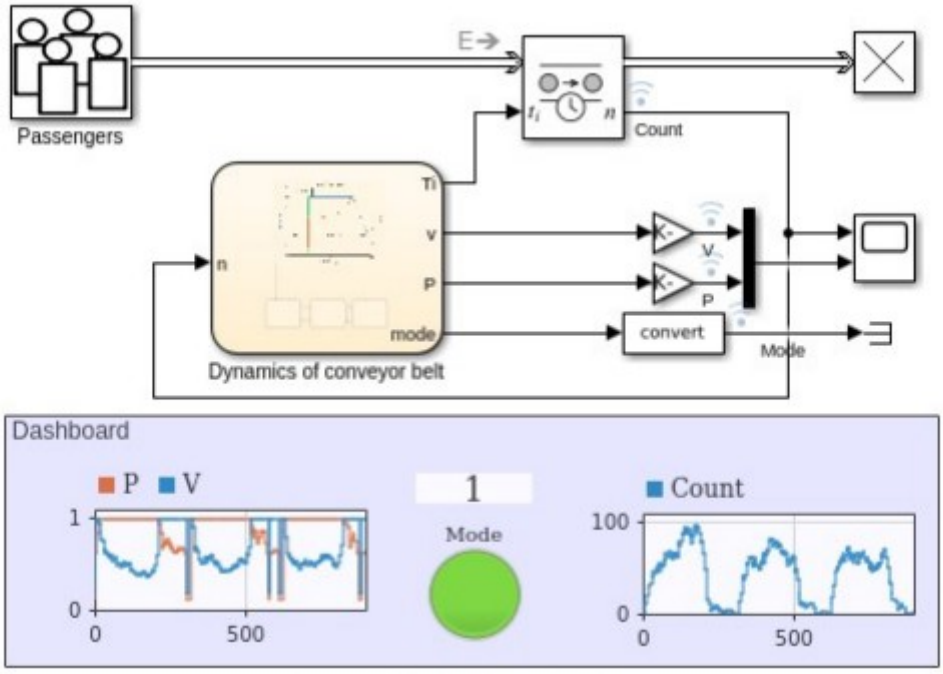



Рис. 15.1. Імітаційна модель конвеєра

На конвеєр надходять комплектуючі, які утворюють пуассонівський потік. Вихід є послідовністю сутностей SimEvents відповідних комплектуючих, які просуваються у стрічковий конвеєр. Розподіл часу надходження ($\Delta t$) пуасонівського процесу, де частота надходження моделюється у блоці Entity Generator протягом години пік, нормального навантаження та вільного часу. Частота надходження змінюється згодом як

$$\lambda(t)=\begin{cases}2 & \text{mod}(t,300)\in[0,180]\ \text{пікове навантаження}\\ 0.5 & \text{mod}(t,300)\in[180,240]\ \text{нормальне навантаження}\\ 0.1 & \text{mod}(t,300)\in[240,300]\ \text{вільний}\end{cases}$$

Запрограмуємо блок Entity Generator, записавши на вкладці Entity Generation наступний код:

```
persistent rngInit;
```

```
if isempty(rngInit)
   seed = 12345;
   rng(seed);
   rngInit = true;
end

% Pattern: Exponential distribution persistent time;
if isempty(time)
   time = 0;
end
if time < 180
   lambda = 2;
elseif time < 240
   lambda = .5;
elseif time < 300
   lambda = .1;
else
   time = 0;
   lambda = 2;
end
% Use inverse transform sampling to
% generate dt according to the exponential
% distribution
dt = -1/lambda * log(1 - rand());
time = time + dt;
```

Транспортна затримка сутності (Entity Transport Delay) зберігає комплектуючі на стрічковому конвеєрі доти, доки вони не прибувають в інший блок, на основі затримки, що обчислюється діаграмою Stateflo (довжина черги – inf).

Діаграма Stateflow моделює динаміку стрічкового конвеєра змінної швидкості (рис. 15.2).

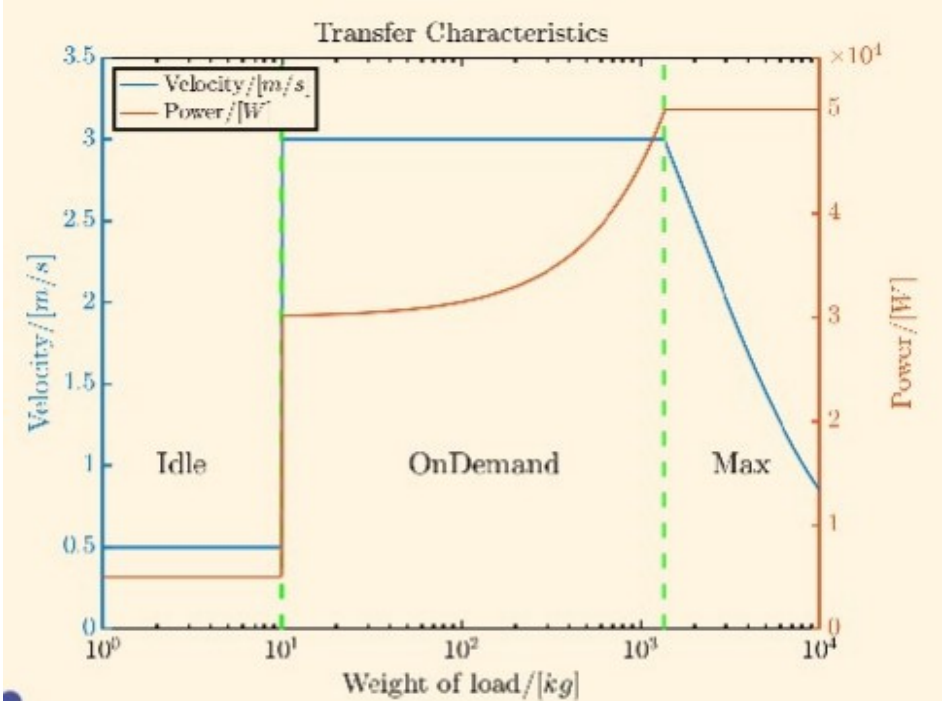



Рис. 15.2. Графік залежності швидкості та потужності від ваги

У діаграмі швидкість та потужність побудовані за логарифмічною шкалою завантаження. Стрічковий конвеєр має три режими (рис. 15.3):

– неактивний (Idle) – вага завантаження мала. Стрічка забезпечує низьку швидкість, щоб зберегти енергію;

– нормальний (OnDemand) – робочий режим, який забезпечує оптимальну швидкість стрічки та пропускної спроможності. Потужність пропорційно збільшується із вагою завантаження. Mode;

– максимальний (Max) – режим максимальної потужності. Вага завантаження є занадто великою для стрічкового конвеєра, щоб забезпечити оптимальну швидкість. Стрічковий конвеєр діє за максимальної можливої швидкості, яка не перевищує максимальну потужність.

На рис. 15.4 представлено вікно налаштування параметрів діаграми.

Для побудови графіків задані два масштабних коефіцієнта Gain_V = 1/3, Gain_P =1/50000.

Для утилізації сутностей застосовуються блоки утилізації та накопичувач. Для надходження комплектуючих у термінатор застосовується блок convert (перетворення типів даних Data Type conversation – Floor).

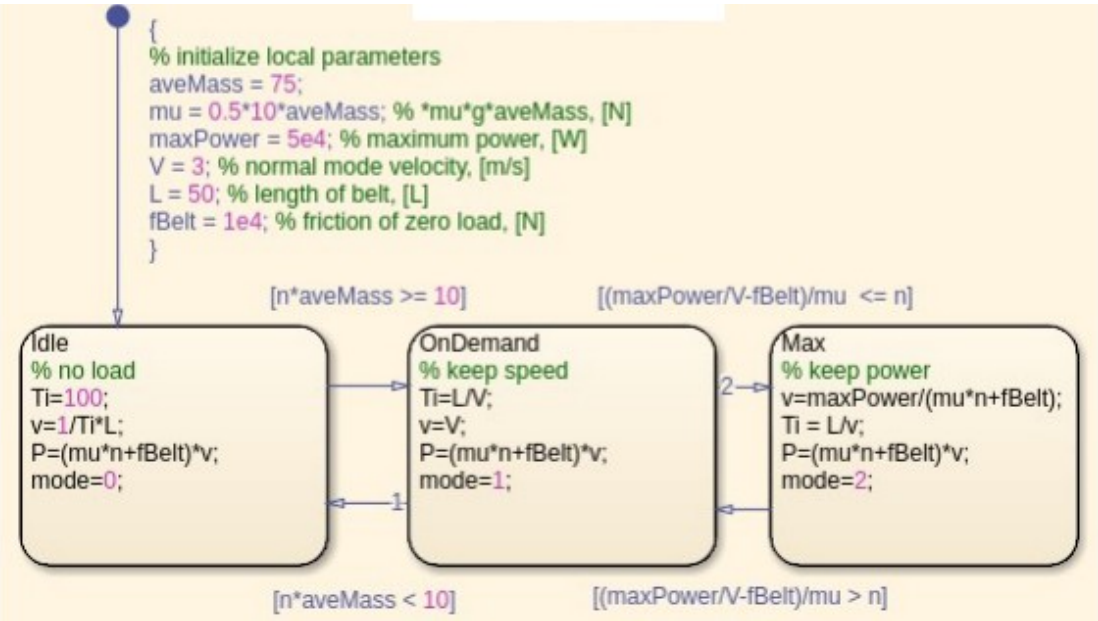



Рис. 15.3. Діаграма Stateflow

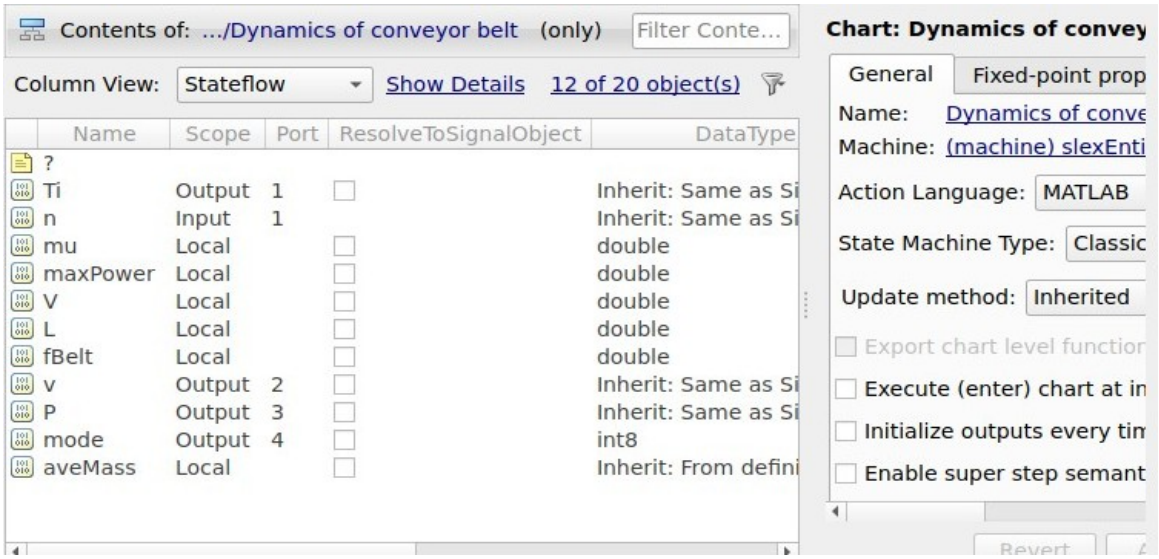



Рис. 15.4. Вікно налаштування параметрів діаграми Stateflow

Три цикли роботи спостерігаються на відрізку часу 900. Кожен цикл має період 300, який вирівнюється з періодом частоти надходження. Головний графік показує кількість комплектуючих на стрічковому конвеєрі залежно від часу, і нижній графік показує швидкість та ступінь стрічкового конвеєра. Швидкість та ступінь нормовані для кращої візуалізації (рис. 15.5). Кількість пасажирів, порівняно з часом симуляції. 2. (Синя) швидкість та ступінь (червона) порівняно з часом симуляції).

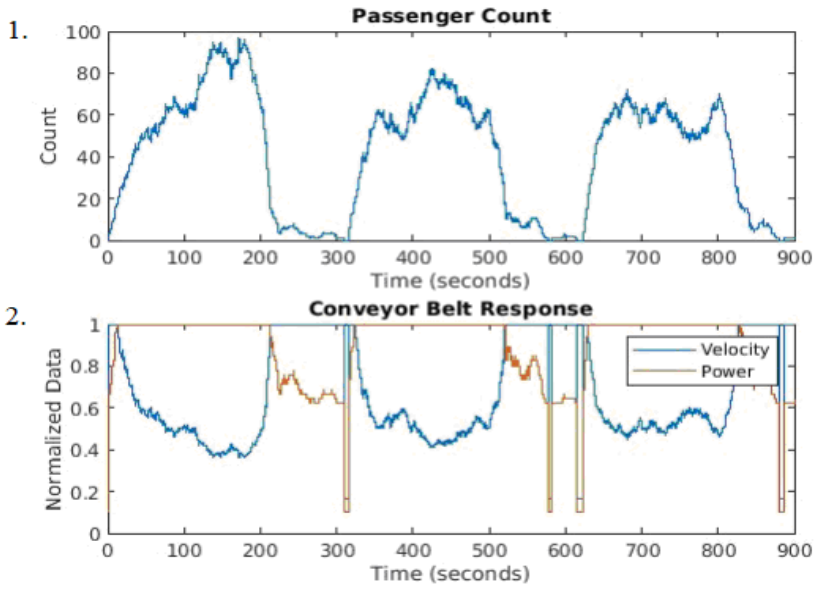



Рис. 15.5. Графіки результатів моделювання конвеєра

Перші дві третини кожного періоду відповідають годині пік, і кількість комплектуючих на стрічковому конвеєрі суттєво збільшується. Отже, стрічковий конвеєр входить в режим Max, який характеризується максимальною вихідною потужністю зі швидкістю, яка обернено пропорційна кількості комплектуючих. В останню третину кожного періоду конвеєр знаходиться в нормальному режимі роботи, що супроводжується вільною годиною. Кількість комплектуючих зменшується і навіть стає нульовою протягом деякого часу. Потім конвеєр діє в OnDemand та Неробочих режимах відповідно. У режимі OnDemand швидкість заблокована до значення за замовчуванням, і потужність пропорційна кількості пасажирів. У Неробочому режимі швидкість і потужність мають низькі значення, щоб зменшувати споживання енергії. Загалом стрічковий конвеєр діє згідно з завантаженням аеропорту.

Модель можна переглянути в додатку Matlab, задавши у командному вікні:

openExample(‘simulink/VariableSpeedConveyorBeltExample’).

# СПИСОК ЛІТЕРАТУРИ


1. Горда О.В. Конспект лекцій з дисципліни «Чисельні методи». Частина 1 / О.В. Горда, І.І. Назаренко. – Київ : КНУБА.
2. Горда О.В. Конспект лекцій з дисципліни «Чисельні методи». Частина 2. / О.В. Горда, І.І. Назаренко – Київ : КНУБА.
3. Горда О.В. Методичні вказівки до курсової роботи з дисципліни «Моделювання систем». – Київ : КНУБА.
4. Горда О.В. Методичні вказівки до виконання практичих занять з дисципліни «Чисельні методи». Частина 1. – Київ : КНУБА.
5. Горда О.В. Математичне та імітаційне моделювання систем масового обслуговування : навчальний посібник / О.В. Горда, В.М. Михайленко. – Київ : КНУБА, 2019. – 216 с.
6. Горда Е.В. Основы работы в среде Matlab : учебное пособие / О.В. Горда, В.М. Михайленко. – Київ : КНУБА, 2015. – 260 с.
7. Сайт середовища Matlab. Документація та приклади. https://matlab.mathworks.com/.

**ДЛЯ НОТАТОК**

Навчально-методичне видання

# МОДЕЛЮВАННЯ СИСТЕМ

Методичні вказівки
до виконання лабораторних робіт
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальностей F3 (122) «Комп'ютерні науки»,
F6 (126) «Інформаційні системи і технології»,
A5 (015.39) «Професійна освіта (Цифрові технології)»

Укладач **Горда** Олена Володимирівна

Випусковий редактор *Л.С. Тавлуй*
Комп'ютерне верстання *Т.І. Кукарєвої*

Підписано до друку 07.09. 2025. Формат 60 × 84 1/ 16
Ум. друк. арк. 6,04. Обл.-вид. арк. 6,5.
Електронний документ. Вид. № 63/III–25.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.