МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

А.Д. Кожухівський, Г.І. Гайдур, О.А. Кожухівська,
В.В. Марченко, С.О. Алексенко

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ СИСТЕМ ТА ПРОЦЕСІВ КІБЕРБЕЗПЕКИ В СЕРЕДОВИЩІ MATLAB

ПРАКТИКУМ

Київ - 2020

УДК 004.942 (076)
І - 52

Рекомендовано до друку Вченою радою Державного університету телекомунікацій, протокол № від 2020 р.

Р е ц е н з е н т и :

1.Головний конструктор АТ "Інститут інформаційних технологій", професор кафедри безпеки інформаційних систем і технологій Національного університету ім. В.Н. Каразіна, м. Харків, д.т.н., професор І. Д. Горбенко;

2. Завідувач кафедри програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича, д. фіз.-мат. наук, професор С. Е. Остапов;

3.Завідувач спеціальної кафедри №5 Інституту спеціального зв'язку та захисту інформації Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», д.т.н., доцент І. Ю. Субач.

І - 52 Імітаційне моделювання систем та процесів кібербезпеки в середовищі MATLAB: Практикум [Електронний ресурс] (Для студентів техн. спец. вищ. навч. закл.) / [А.Д. Кожухівський, Г.І. Гайдур, О.А. Кожухівська, В.В. Марченко, С.О. Алексенко]; М-во освіти і науки України, Державний університет телекомунікацій. – К: ДУТ, 2020. – 78 с.

У посібнику розглядаються основні методи побудови імітаційних моделей з використанням системи MATLAB. Подана велика кількість практичного матеріалу і прикладів, які дозволяють створювати власні імітаційні моделі, що розв'язують широкий спектр прикладних задач, використовуючи пакет MATLAB.

Для студентів напряму „Комп`ютерні науки" усіх спеціальностей.

Ключові слова: Імітаційне моделювання, MATLAB, SIMULINK.

УДК 004.942 (076)

Н а в ч а л ь н е в и д а н н я

В авторській редакції

ВСТУП

Моделювання – це спосіб дослідження будь-яких явищ, процесів або об’єктів шляхом побудови та аналізу їх моделей. У широкому розумінні моделювання є однією з основних категорій теорії пізнання і чи не єдиним науково обґрунтованим методом наукових досліджень систем і процесів будь-якої природи в багатьох сферах людської діяльності.

Складні за внутрішніми зв’язками і великі за кількістю елементів системи економічно важко піддаються прямим способам моделювання і часто для побудови і вивчення використовують імітаційні методи моделювання.

Імітаційне моделювання – це метод конструювання моделі системи та проведення експериментів на моделі. Суттєвими особливостями цього виду моделювання є опис структури системи, що моделюється, застосування засобів відтворення функціонування (поведінки) системи на моделі, відображення властивостей середовища, в якому функціонує досліджувана система.

Метод імітаційного моделювання з успіхом застосовують у таких областях, як автоматизація проектування, організація роботи обчислювальних комплексів, організація роботи транспорту, сфера обслуговування, аналіз різних сторін діяльності людини (охорона оточуючого середовища, керування водними ресурсами, екологічні проблеми, енергетика і т.п.), автоматизоване керування виробничими та іншими процесами.

Наведений у практикумі матеріал має практичну спрямованість на оволодіння методами імітаційного моделювання систем та процесів при підготовці до практичних занять, лабораторних і контрольних робіт.

У практикумі наведені основні методи побудови імітаційних моделей з використанням системи MATLAB. Велика кількість практичного матеріалу і прикладів, що представлені в практикумі, дозволяють створювати власні імітаційні моделі і розв’язувати широкий спектр прикладних задач з використанням пакету MATLAB.

Для програмного продукту, розглянутого у посібнику, наведені варіанти завдань до виконання контрольних робіт і курсових робіт.

1 Інструментарій імітаційного моделювання Simulink

Simulink – інтерактивний інструмент для моделювання, імітації і аналізу динамічних систем. Він дає можливість будувати графічні блок-схеми, імітувати динамічні системи, досліджувати роботоздатність систем і удосконалювати проекти. **Simulink** повністю інтегрований з **MATLAB**, забезпечуючи швидкий доступ до широкого спектру інструментів аналізу і проектування. **Simulink** також інтегрується з **Stateflow** для моделювання поведінки, що викликана подіями. Ці переваги роблять **Simulink** найбільш популярним інструментом для проектування систем керування і комунікації, цифрової обробки і інших додатків моделювання.

1.1 Загальні відомості

Программа **Simulink** являється додатком **до** пакету **MATLAB**. При моделюванні з використанням **Simulink** реалізується принцип візуального програмування, у відповідності з яким користувач на екрані із бібліотеки стандартних блоків створює модель пристрою і здійснює розрахунки. При цьому, на відміну від класичних способів моделювання, користувачеві не треба досконало вивчати мови програмування і чисельні методи математики, достатньо загальних знань, що необхідні при роботі на комп'ютері, і, природньо, знань тієї предметної області, у якій він працює.

Simulink являється достатньо самостійним інструментом **MATLAB** і при роботі з ним зовсім не потрібно знати сам **MATLAB** і інші його додатки. З іншої сторони, доступ до функцій **MATLAB** і інших його інструментів залишається відкритим і їх можна використовувати в **Simulink.** Частина пакетів, що входять у склад **MATLAB,** мають інструменти, які вбудовуються в **Simulink** (наприклад, **LTI-Viewer** додатки **Control System Toolbox –** пакету для розробки систем керування). Є також додаткові бібліотеки блоків для різних областей застосування (наприклад, **Power System Blockset** – моделювання електротехнічних пристроїв, **Digital Signal Processing Blockset** – набір блоків для розробки цифрових пристроїв і т. д).

При роботі з **Simulink** користувач має можливість модернізувати бібліотечні блоки, створювати свої власні, а також складати нові бібліотеки блоків.

При моделюванні користувач може вибирати метод розв'язку диференційних рівнянь, а також спосіб змінювання модельного часу (з

фіксованим або змінним кроком). На протязі моделювання є можливість слідкувати за процесами, що відбуваються в системі. Для цього використовуються спеціальні пристрої спостереження, які входять в склад бібліотеки **Simulink**. Результати моделювання можуть бути представлені у вигляді графіків або таблиць.

Перевага **Simulink** полягає також у тому, що він дозволяє поповнювати бібліотеки блоків за допомогою підпрограм, які написані як на мові **MATLAB,** так і на мовах **C + +, Fortran** і **Ada**.

1.2 Створення моделі

1.2.1 Постановка задачі і початок створення моделі

Вирішення будь-якої проблеми в **Simulink** повинне починатися з постановки задачі. Чим більш глибоко продумана постановка задачі, тим більша імовірність успішного її вирішення. В ході постановки задачі необхідно оцінити, наскільки сутність задачі відповідає можливості пакету **Simulink** і які компоненти цього пакету можуть бути використані для побудови моделі. Для створення моделі у середовищі **Simulink** необхідно послідовно виконати ряд дій:

1) Створити новий файл моделі за допомогою команди **File/New/Model** або використовуючи кнопку  на панелі інструментів (тут і дальше за допомогою символу “/”, указані пункти меню програми, які необхідно послідовно вибрати для виконання указаної дії). Заново створене вікно моделі показане на рис.1.1.

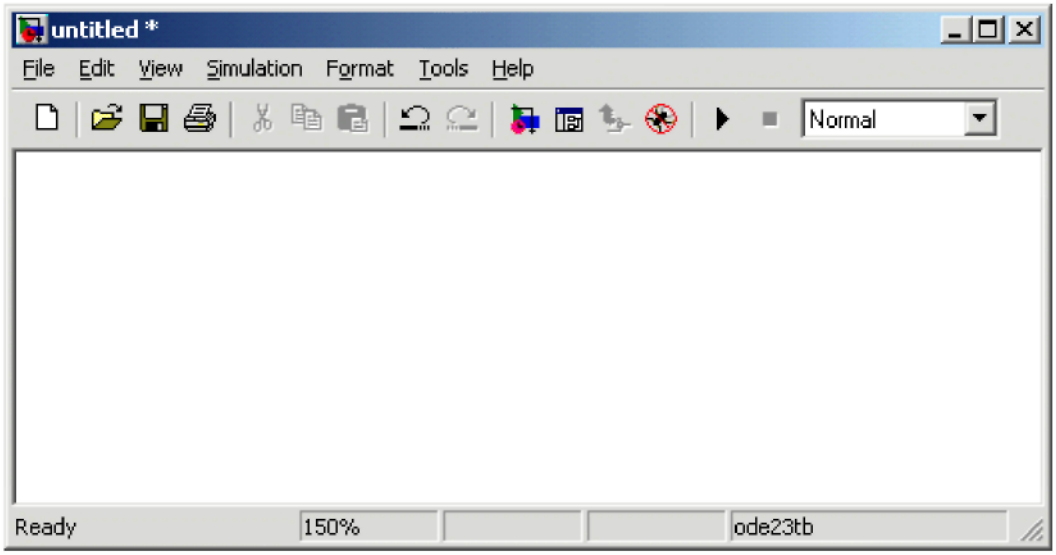

Рис 1.1. Пусте вікно моделі

2) Розташувати блоки у вікні моделі. Для цього необхідно відкрити відповідний розділ бібліотеки (наприклад, **Sources** - Джерела). Дальше, указавши курсором на необхідний блок і натиснувши на ліву клавішу мишки – «перетягнути» блок у створене вікно. Клавішу мишки необхідно ***тримати натиснутою***. На рис. 1.2 показане вікно моделі, що містить блоки.

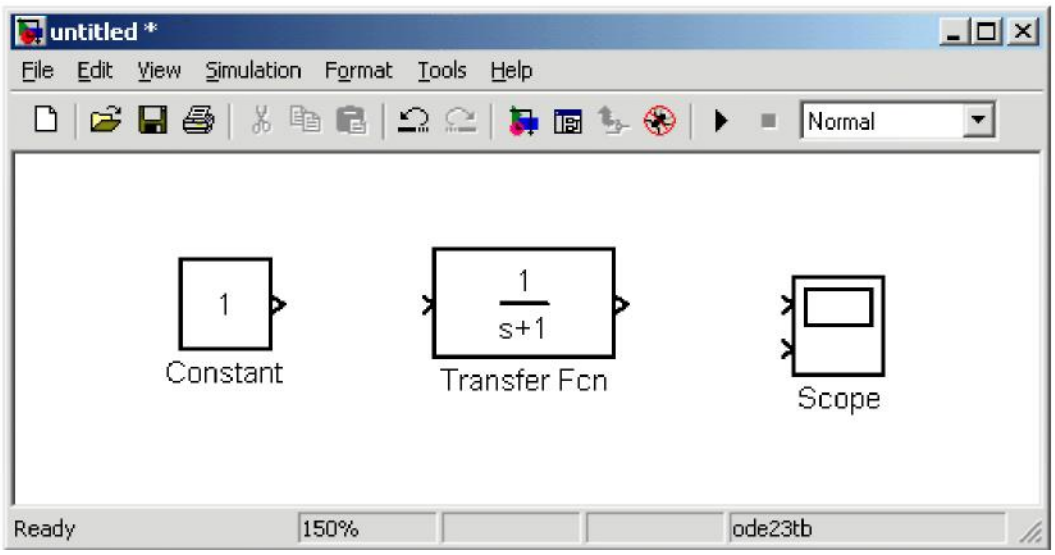

Рис 1.2. Вікно моделі, що містить блоки

Для видалення блоку необхідно вибрати блок (указати курсором на його зображення і натиснути ліву кнопку мишки), а потім натиснути клавішу **Delete** на клавіатурі.

Для зміни розмірів блоку необхідно вибрати блок, установити курсор в один із кутів блоку і, натиснувши ліву кнопку мишки, змінити розмір блоку (курсор при цьому перетвориться в двонаправлену стрілку).

3) Дальше, якщо це вимагається, необхідно змінити параметри блоку, які установлені програмою «за замовчуванням». Для цього необхідно двічі клацнути лівою кнопкою мишки, указавши курсором на зображення блоку. Відкриється вікно редактування параметрів даного блоку. При задаванні числових параметрів необхідно мати на увазі, що у якості десяткового роздільника повинна використовуватися крапка, а не кома. Після внесення змін необхідно закрити вікно кнопкою **OK**. На рис. 1.3 у якості прикладу показаний блок, який моделює передатну функцію, і вікно редактування параметрів даного блоку.

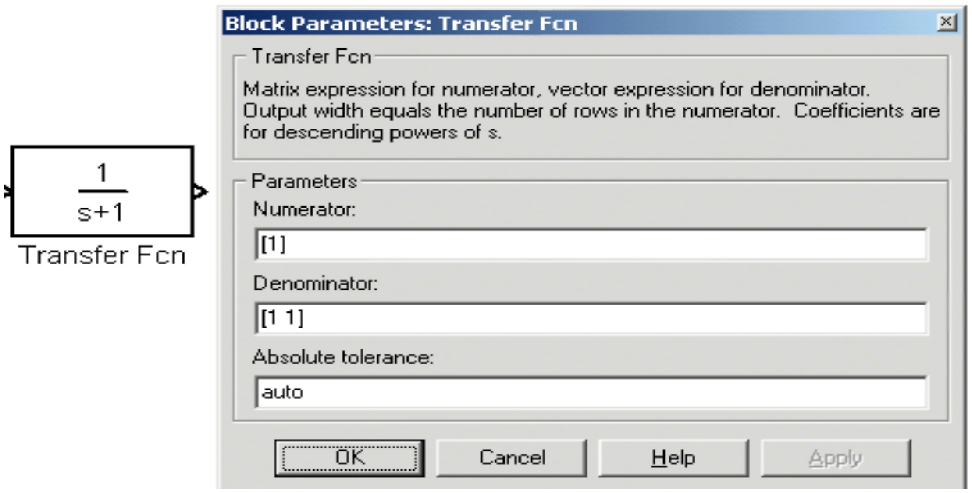

Рис 1.3. Блок, який моделює передатну функцію, і вікно редактування параметрів даного блоку

Після встановлення на схемі всіх блоків із необхідних бібліотек треба виконати з'єднання елементів схеми. Для з'єднання блоків необхідно указати курсором на «вихід» блоку, а потім, натиснувши і не опускаючи ліву клавішу мишки, провести лінію до входу іншого блоку. Після чого відпустити клавішу. У випадку правильного з'єднання, зображення стрілки на вході блоку змінить колір. Для створення точки розгалуження в з'єднуючій лінії необхідно підвести курсор до передбачуваного вузла і, натиснувши *праву* клавішу мишки, протягнути лінію.

4) Після створення розрахункової схеми необхідно зберегти її у вигляді файлу на диску, вибравши пункт меню **File/Save As...** у вікні схеми і указавши папку і ім'я файлу. Необхідно мати на увазі, що ім'я файлу не повинне перевищувати

32 символи, повинне починатися з букви і не може містити символи кирилиці і спеціальні символи. Ця ж вимога відноситься і до шляху файлу (до тих папок, у яких зберігається файл). При наступному редактуванні схеми можна користуватися пунктом меню **Fille/Save**. При повторних запусках програми **Simulink** завантаження схеми здійснюється за допомогою меню **File/Open...** у вікні оглядача бібліотеки або із основного вікна **MATLAB.**

Для видалення лінії необхідно вибрати лінію (так само, як це виконується для блоку) і натиснути клавішу **Delete** на клавіатурі. Схема моделі, в якій виконані з'єднання між блоками, показана на рис. 1.4.

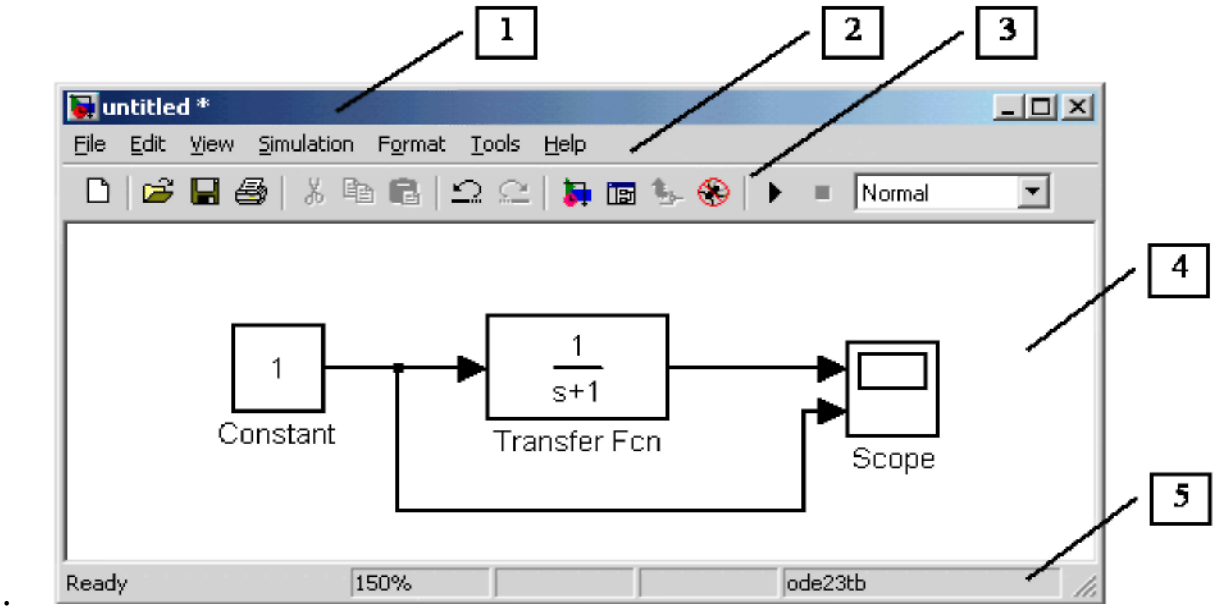

Рис 1.4. Схема моделі

1.2.2 Вікно моделі

Вікно моделі містить наступні елементи (див. рис. 1.4):

1. Заголовок із назвою вікна (Заново створеному вікну присвоюється ім'я **Untitled** з відповідним номером).
2. Меню з командами **File**, **Edit**, **View і** т.д.
3. Панель інструментів.
4. Вікно для створення схеми моделі.
5. Рядок стану, що містить інформацію про поточний стан моделі.

Меню вікна містить команди для редактування моделі, її настроювання і керування процесом розрахунку, роботи з файлами і т.п.:

- **File (Файл)** —Робота з файлами моделей.
- **Edit (Редактування)** — Змінювання моделі і пошук блоків.

- **View (Вид)** — Керування виглядом елементів інтерфейсу.
- **Simulation (Моделювання)** — Задавання настроювань для моделювання і керування процесом розрахунку.
- **Format (Форматування)** — Змінювання зовнішнього вигляду блоків і моделі в цілому.
- **Tools (Інструментальні засоби)** — Застосування спеціальних засобів для роботи з моделлю (налагоджувальник, лінійний аналіз і т.п.)
- **Help (Довідка)** — Виведення вікон довідкової системи.

Для роботи з моделлю можна також використовувати кнопки на панелі інструментів (рис. 1.5).

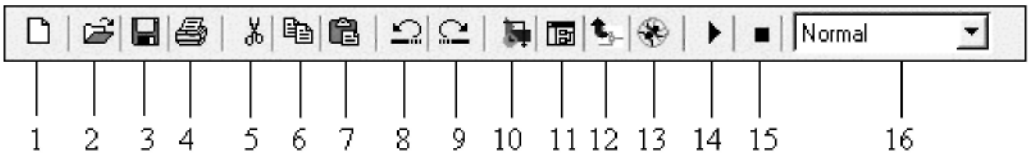

Рис. 1.5. Панель інструментів вікна моделі

Кнопки панелі інструментів мають наступні призначення:

1. New Model – Відкрити нове (пусте) вікно моделі.

2. Open Model - Відкрити існуючий **mdl**-файл.
3. Save Model – Зберегти **mdl**-файл на диску.
4. Print Model – Виведення на друк блок-діаграми моделі.
5. Cut – Вирізати виділену частину моделі в буфер проміжного зберігання.
6. Copy – Скопіювати виділену частину моделі в буфер проміжного зберігання.
7. Paste – Вставити у вікно моделі вмістиме буферу проміжного зберігання.
8. Undo – Відмінити попередню операцію редактування.
9. Redo – Відновити результат відміненої операції редактування.
10. Library Browser – Відкрити вікно оглядача бібліотек.
11. Toggle Model Browser - Відкрити вікно оглядача моделі.
12. Go to parent system – Перехід із підсистеми в систему вищого рівня ієрархії («батьківську систему»). Команда доступна тільки якщо відкрита підсистема.
13. Debug – Запуск налагоджувальника моделі.

14. Start/Pause/Continue Simulation - Запуск моделі на виконання (команда **Start**); після запуску моделі на зображенні кнопки виводиться символ , і їй відповідає уже команда **Pause** (Призупинити моделювання); для відновлення моделювання необхідно клацнути по тій же кнопці, так як в режимі паузи їй відповідає команда **Continue** (Продовжити).

15. Stop –Закінчити моделювання. Кнопка стає доступною після початку моделювання, а також після виконання команди **Pause**.

16. Normal/Accelerator - Звичайний/Прискорений режим розрахунку. Інструмент доступний, якщо встановлений додаток **Simulink Performance Tool**.

В нижній частині вікна моделі знаходиться рядок стану, в якому відображаються короткі коментарі до кнопок панелі інструментів, а також до пунктів меню, коли вказівник мишки знаходиться над відповідним елементом інтерфейсу. Це ж текстове поле використовується і для індікації стану **Simulink**: **Ready** (Готовий) або **Running** (Виконання).

В рядку стану відображаються також:

• масштаб відображення блок-діаграми (в процентах, початкове значення дорівнює 100%),
• індикатор степені закінченості сеансу моделювання (з'являється після запуску моделі),
• поточне значення модельного часу (виводиться також тільки після запуску моделі),
• використовуваний алгоритм розрахунку станів моделі (метод розв'язку).

1.3 Основні прийоми підготовки і редактування моделі

1. 3.1 Добавлення текстових надписів

Для підвищення наочності моделі зручно використовувати текстові надписи. Для створення надпису необхідно указати мишкою місце надпису і два рази клацнути лівою клавішею мишки. Після цього з'явиться прямокутна рамка з курсором введення. Аналогічним чином можна змінити і підписи під блоками моделей. На рис. 1.6 показані текстовий надпис і змінювання надпису в блоці передатної функції. Необхідно мати на увазі, що версія програми, яка розглядається (**Simulink 4**), не адаптована до використання шрифтів кирилиці, і застосування їх може мати самі різні наслідки: відображення надписів в нечитаємому вигляді, обрізка надписів, повідомлення про помилки, а також неможливість відкрити модель після її збереження. Тому застосування надписів на російській мові для поточної версії **Simulink** зовсім не бажане.

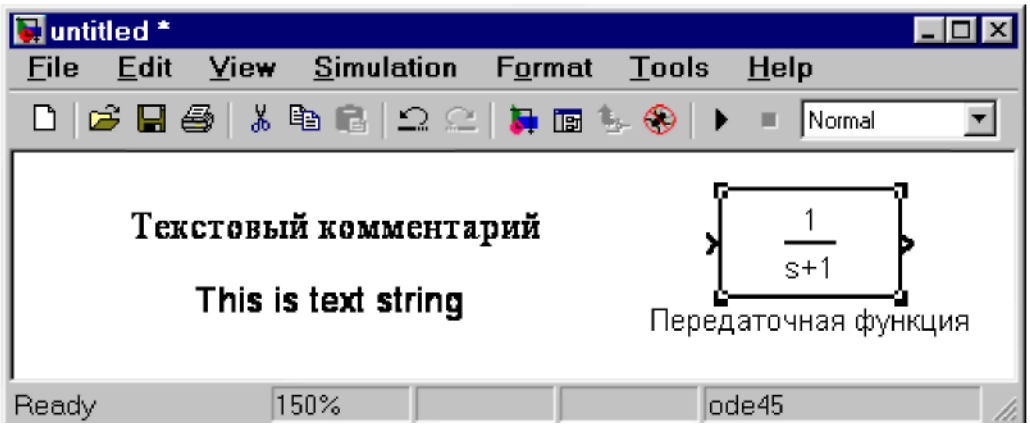

Рис 1. 6. Текстовий надпис і змінювання надписів в Transfer Function

1.3.2 Виділення об'єктів

Для виконання будь-якої дії з елементом моделі (блоком, з'єднуючою лінією, надписом) цей елемент необхідно спочатку виділити.

Виділення об'єктів простіше за все здійснюється мишкою. Для цього необхідно встановити курсор мишки на потрібному об'єкті і клацнути лівою клавішою. Відбудеться виділення об'єкту. При цьому з'являться маркери по кутам об'єкта (див. рис. 1.6). Можна також виділити декілька об'єктів. Для цього необхідно встановити курсор мишки поблизу групи об'єктів, натиснути ліву клавішу мишки і, не відпускаючи її, почати переміщувати мишку. З'явиться пунктирна рамка, розміри якої будуть змінюватися при переміщенні мишки. Усі охоплені рамкою об'єкти виділяються. Виділити усі об'єкти також можна, використовуючи команду **Edit/Select All**.

Для стирання виділеного об'єкту можна визвати команду **Clear** із меню **Edit** або із контекстного меню (рис. 1.7). Контекстне меню дуже зручне тим, що для будь-якого об'єкту воно виводить перелік команд, які доступні в даному випадку.

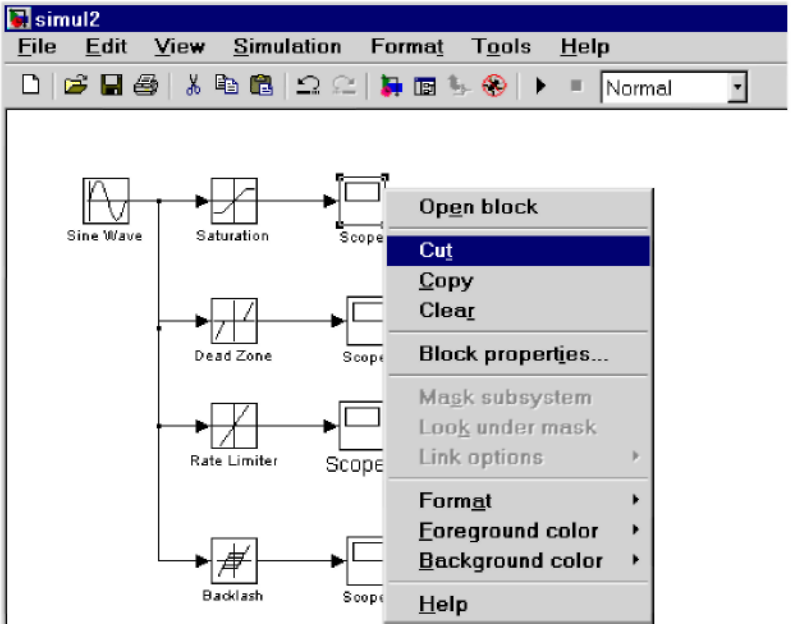

Рис. 1. 7. Контекстне меню

Для відновлення об’єкту у вікні моделі необхідно клацнути лівою кнопкою мишки в передбачуваному місці розташування об’єкту. Після цього виконання команди **Paste** із меню **File** вікна **Simulink** або із контекстного меню розміщує об’єкт (блок), що зберігається в буфері, в задане місце.

Необхідно врахувати, що команда **Clear** стирає блок безповоротно, тобто без розміщення його в буфері обміну. Однак цю операцію можна відмінити командою меню **File / Undo** вікна **Simulink**.

1.3.3 Вставлення блоків і їх з’єднання

Вставлення блоків за допомогою браузера бібліотек **Simulink** ми уже достатньо детально розглянули на прикладах. Відмітимо також, що для перенесення блоків, їх копіювання і розмноження доцільно використовувати буфер обміну. Дуже результативним є підхід, коли користувач для створення своєї моделі використовує раніше створену модель, наприклад, із налагоджених демонстраційних прикладів, яких багато в пакеті **Simulink**.

Для під’єднання нових блоків необхідні нові з’єднання. Вони також легко виконуються за допомогою мишки. Прийоми уведення нових блоків і їх з’єднань виконуються дуже просто. При цьому, прийоми редактування нагадують роботу з популярними графічними редакторами, яку легко освоюють навіть діти.

Тим не менше, корисно відмітити найважливіші прийоми здійснення з’єднань. Блоки моделей звичайно мають входи і виходи. Як правило, вихід якого-небудь блоку під’єднується до входу наступного блоку і т. д. Для цього курсор мишки установлюється на виході блоку, від якого повинне виходити з’єднання. При цьому курсор перетворюється в великий хрестик із тонких ліній. Тримаючи натиснутою ліву кнопку мишки, необхідно плавно перетягнути курсор до входу наступного блоку (рис. 1.8), де курсор мишки набуває вигляду хрестика із тонких здвоєних ліній.

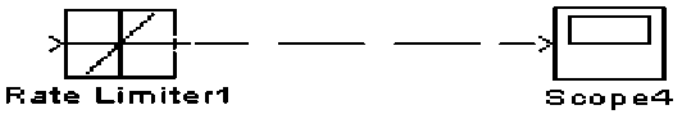

Рис. 1. 8. Початок з’єднання блоку джерела живлення з блоком осцилографа

Добившись протягування лінії до входу наступного блоку, необхідно відпустити ліву кнопку мишки. З’єднання буде закінчене і в кінці його з’явиться жирна стрілка. Клацанням мишки можна виділити з’єднання, ознакою чого будуть чорні прямокутники, що розташовані у вузлових точках з’єднуючої лінії (рис. 1.9).

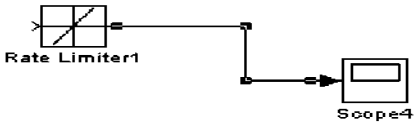

Рис. 1. 9. Виділене з’єднання

Інколи буває необхідно зробити петлю з’єднуючої лінії в ту або іншу сторону. Для цього необхідно захопити потрібну частину лінії і провести її в необхідну сторону, переміщуючи мишку з натиснутою лівою кнопкою. Рис. 1.10 пояснює цей процес.

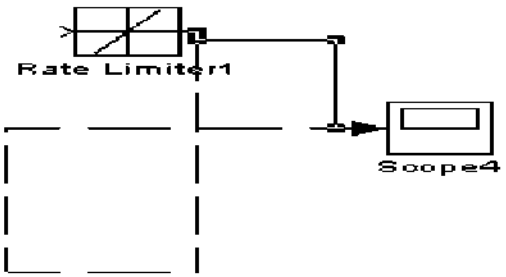

Рис. 1.10. Початок створення петлі лінії з'єднання

Створення петлі лінії закінчується відпусканням лівої кнопки мишки. Лінія, що отримана таким чином, показана на рис. 1.11.

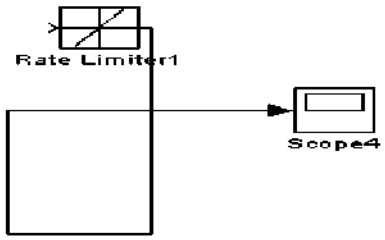

Рис. 1.11. Готова петля з'єднання

Особливо треба відмітити можливість задавання ліній з'єднань, що мають нахил, при натиснутій клавіші **Shift**. Приклад такого з'єднання наведений на рис. 1.12.

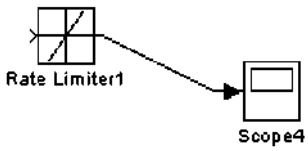

Рис. 1.12. Приклад з'єднання з нахиленою лінією

1.3.4 Створення відведення лінії

Часто виникає необхідність зробити відведення від уже створеної лінії. Приклад створення такого відведення показаний на рис. 1.13. Відмітимо, що при натиснутій клавіші **Shift** відведення здійснюється лініями, що мають нахил.

В прикладі, що наведений на рис. 1.13, використана модель інтегратора, який під'єднується до виходу джерела прямокутних імпульсів. Щоб можна

було спостерігати осцилограми як на виході джерела, так і на виході інтегратора, в схему був включений блок мультиплексора сигналів **Мих з** двома входами. Щоб підключити нижній вхід до уже задіяного виходу джерела, нам і необхідно було створити відведення лінії.

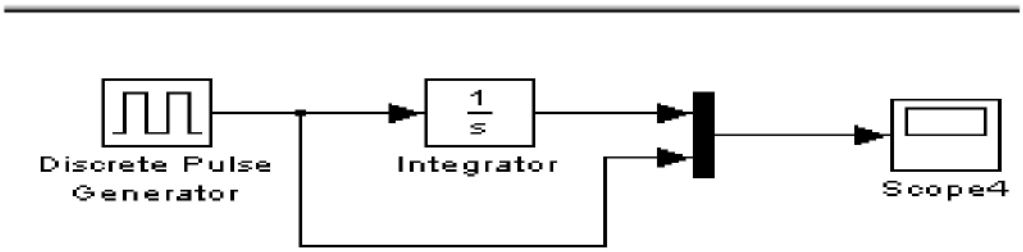

Рис. 1.13. Приклад моделі з відведенням лінії

Тепер можна запустити цю модель і подивитися, які сигнали діють на виходах джерела і інтегратора. Результат запуску представлений на рис. 1.14.

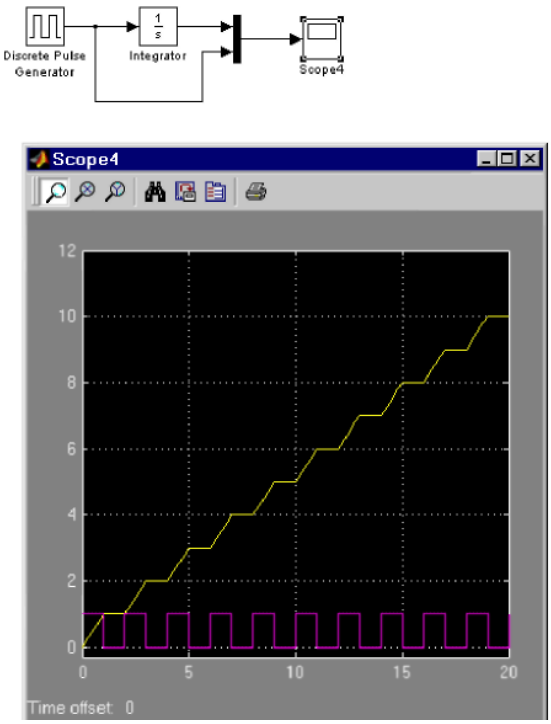

Рис. 1.14. Результат моделювання

Неважко переконатися у тому, що сигнал на виході інтегратора представляє собою лінію, що східчасто наростає. Коли на виході генератора буде високий (умовно) рівень напруги, на виході інтегратора сигнал лінійно наростає. Коли рівень напруги на генераторі дорівнює 0, сигнал на виході інтегратора залишається незмінним. Такий характер процесу, зрозуміло, добре

знайомий спеціалістам, проте початківцям, що вивчать електронні (і не тільки) системи, цей приклад дає наочну ілюстрацію роботи інтегратора.

1.3.5 Видалення з'єднань

Для видалення з'єднуючої лінії достатньо виділити її і виконати команду **Clear** або **Cut**.

1.3.6 Зміна розмірів блоків

Simulink має розширені можливості редактування блок-схем. Так, блоки у вікні редактування можна не тільки переміщувати за допомогою мишки, але і змінювати за розмірами. Для цього блок виділяється, після чого курсор мишки необхідно установити на кільця по кутам блоку. Як тільки курсор мишки перетвориться в двонаправлену діагональну стрілку, можна буде при натиснутій лівій кнопці розтягувати блоки по діагоналі, збільшуючи або зменшуючи їх розміри (рис. 1.15)

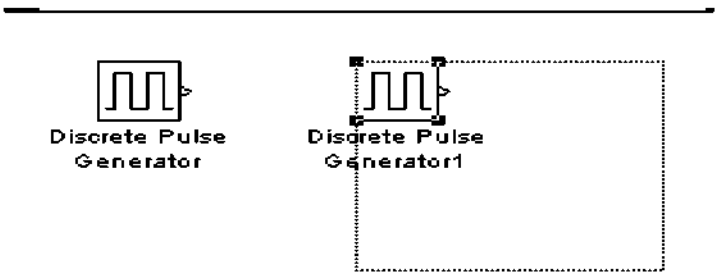

Рис. 1.15. Розтягування блоку

Збільшений за розмірами блок показаний на рис. 1.16. Звернимо увагу на те, що розтягується тільки графічне зображення (піктограма) блоку, а розміри його назви у вигляді текстового надпису не змінюються.

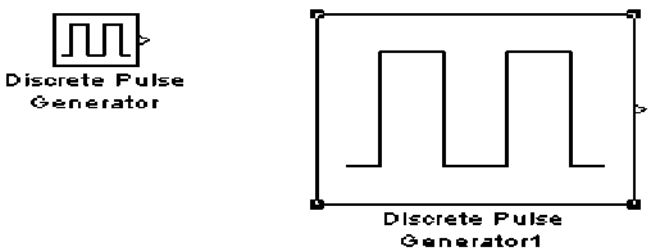

Рис. 1. 16. Приклад розтягування блоку

1.3.7 Переміщення блоків і вставлення блоків у з'єднання

Блок, що приймає участь у з'єднанні, можна переміщувати у вікні моделі, виділивши його і перетягуючи, як звичайно, мишкою. При цьому з'єднання не переривається, а просто скорочується або збільшується по довжині. В довге з'єднання можна вставити новий блок, не руйнуючи його і не виконуючи складних маніпуляцій. Рис. 1.17 показує вставлення блоку диференціюючого пристрою між джерелом синусоїдного сигналу і осцилографом.

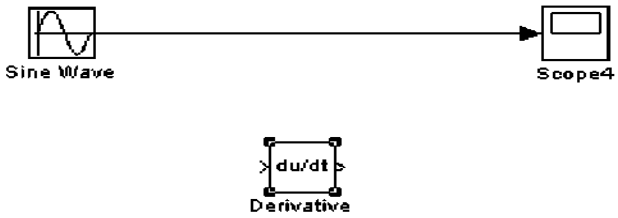

Рис. 1.17. Початкове з'єднання і блок для вставлення

Результат вставлення диференціюючого пристрою в з'єднання між джерелом і осцилографом показаний на рис. 1.18.

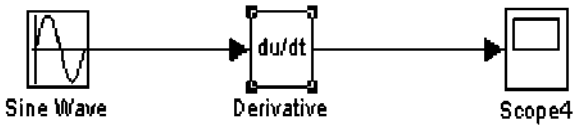

Рис. 1.18. Приклад вставлення блоку в з'єднання

Проте аналогічне просте вставлення можливе для блоків, що мають один вхід і один вихід, які включаються у з'єднання. Спроба вставити таким чином мультиплексор буде безуспішною, оскільки він має два входи і не стикується із розриваємим з'єднанням. Щоб вставити мультиплексор, необхідно видалити з'єднання між диференціюючим пристроєм і входом осцилографа. Для цього з'єднання виділяється і виконується команда Edit/Clear. Після цього мультиплексор переміщується в потрібне місце і з'єднання створюються заново.

1.3.8 Моделювання диференціюючого пристрою

Отже, ми фактично створили модель диференціюючого пристрою і можемо подивитися, що відбувається при диференціюванні одного

синусоїдного сигналу. Результат запуску створеної моделі представлений на рис.1.19.

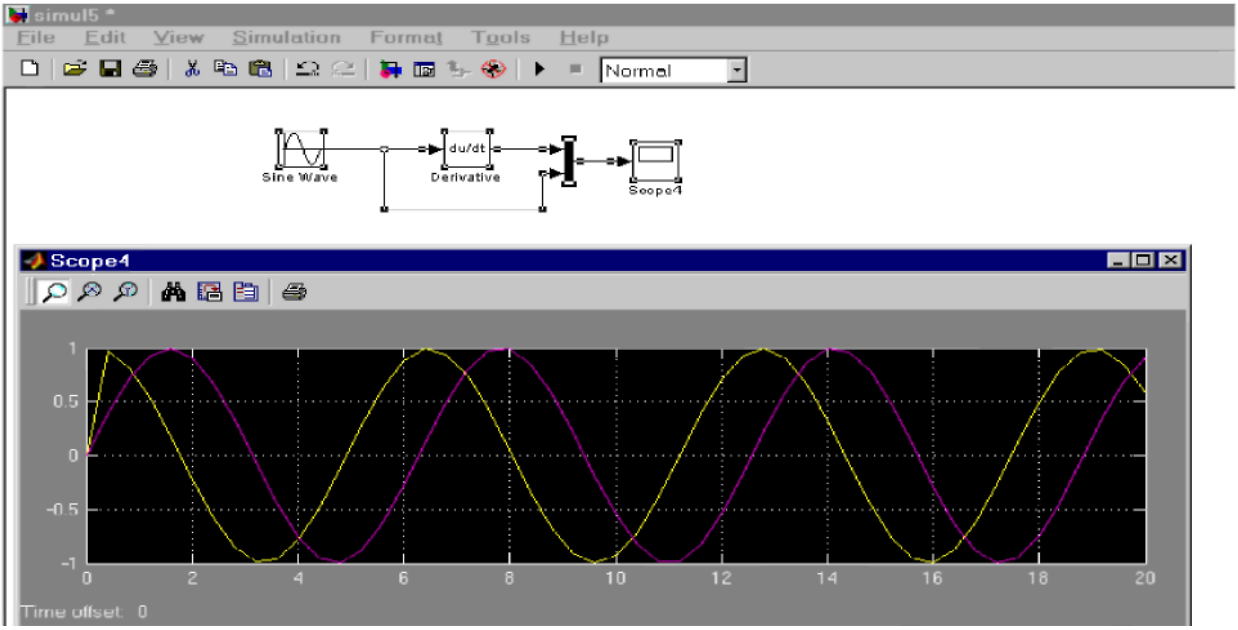

Рис. 1.19. Моделювання диференціюючого пристрою

Уважно придивившись до осцилограм, ми бачимо, що при вхідному синусоїдному сигналі вихідний сигнал являється косинусоїдою. Це повністю відповідає математичним співвідношенням для даного випадку (як відомо, похідна sin(x) є cos(x)).

Проте на самому початку процесу диференціювання добре видно ваду роботи моделі – при t = 0 похідна дорівнює не 1, а 0.

Це зв’язано з тим, що процес починається при нульових початкових умовах. Проте достатньо швидко ситуація виправляється, і в подальшому вихідний сигнал стає косинусоїдним. Таким чином, диференціюючий пристрій можна використовувати для точного зсуву на 90° гармонічного сигналу будь-якої частоти.

Звернимо увагу, що, на відміну від блоку інтегратора (рис. 1.14), блок цифрового диференціатора немає параметрів, що настроюються. Його вікно параметрів, яке показане на рис. 1.19, являється чисто інформаційним.

1.3.9 Команди Undo і Redo у вікні моделі

Велику допомогу у редактуванні подає команда **Undo** – відміна останньої операції. Вона підтримує більше ста різноманітних операцій, включаючи операції добавлення і стирання ліній. Цю команду можна визвати за допомогою

кнопки панелі інструментів вікна моделі або із меню **Edit**. Для відновлення відміненої операції служить команда **Redo.**

1.3.10 Форматування об'єктів

В меню **Format** (так само як і в контекстному меню, що викликається натисканням правої клавіші мишки на об'єкті) знаходиться набір команд форматування блоків.

Команди форматування діляться на декілька груп:

1. Змінювання відображення надписів:

• **Font** —Форматування шрифту надписів і текстових блоків.
• **Text alignment** —Вирівнювання тексту в текстових надписах.
• **Flip name** —Переміщення підпису блоку.
• **Show/Hide name** —Відображення або хования підпису блоку.

2. Змінювання кольорів відображення блоків:

• **Foreground color** — Вибір кольору ліній для виділених блоків.
• **Background color** — Вибір кольору фону для виділених блоків.
• **Screen color** — Вибір кольору фону для усього вікна моделі.

3.Змінювання положення блоку і його вигляду:

• **Flip block –** Зеркальне відображення відносно вертикальної вісі симетрії.
• **Rotate block** – Поворот блоку на 90^0 за годинниковою стрілкою.
• **Show drop shadow** — Показ тіні від блоку.
• **Show port labels** — Показ міток портів.

4. Інші установки:

• **Library link display** — Показ зв'язків з бібліотеками.
• **Sample time colors** —Вибір кольору блоку індікації часу.
• **Wide nonscalar lines** —Збільшення (зменшення) ширини нескалярних ліній.
• **Signal dimensions** — Показ розмірності сигналів.
• **Port data types** — Показ даних про типи портів.
• **Storage class** — Клас пам'яті (параметр, що установлюється при роботі **Real-Time Workshop)**.
• **Execution order** — Виведення порядкового номеру блоку в послідовності виконання.

Команда **Format/Font** виводить вікно з установками шрифту для текстових надписів (рис. 1.20).

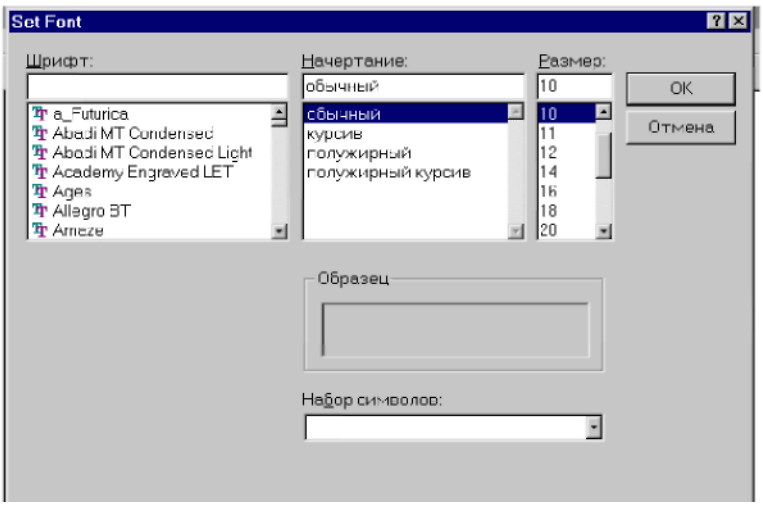

Рис. 1. 20. Вікно вибору шрифту

Нарешті, на рис. 1.21 наочно показана дія ряду операцій форматування іншого виду простої моделі, яка була описана трохи вище. Крім того, на цьому рисунку масштаб моделі збільшений удвічі за допомогою команди **View/Zoom In**.

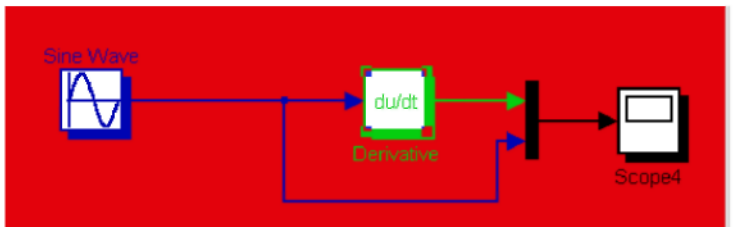

Рис. 1. 21. Вигляд моделі після операцій форматування

Цей рисунок демонструє можливості кольорового оформлення блоків, виділення нескалярних ліній з’єднання, виведення даних про тип портів і уведення вказівок на порядок виконання блоків на протязі моделювання.

1.4 Встановлення параметрів розрахунку і його виконання

Перед виконанням розрахунків необхідно попередньо задати параметри розрахунку. Задавання параметрів розрахунку виконується в панелі керування меню **Simulation/Parameters**. Вид панелі керування наведений на рис.1.22.

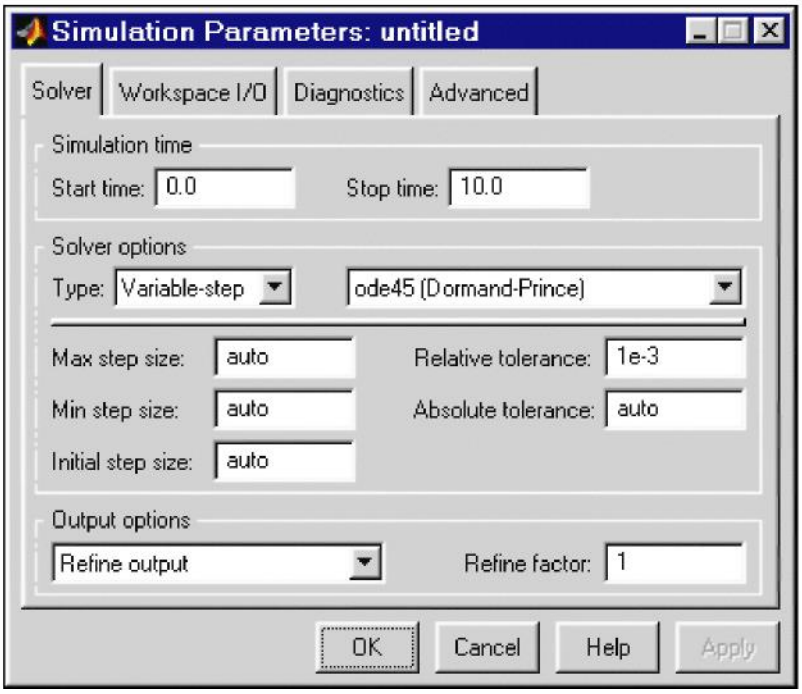

Рис 1.22. Панель керування

Вікно настроювання параметрів розрахунку має 4 вкладки:

• **Solver** (**Розрахунок**) –Встановлення параметрів розрахунку моделі.
• **Workspace I/O** (**Уведення/виведення даних в робочу область**) – Встановлення параметрів обміну даними з робочою областю **MATLAB**.
• **Diagnostics** (**Діагностика**) – Вибір параметрів діагностичного режиму.
• **Advanced** (**Додатково**) –Встановлення додаткових параметрів. Встановлення параметрів розрахунку моделі виконується за допомогою елементів керування, які знаходяться на вкладці **Solver**. Ці елементи поділені на три групи (рис. 1.22):
• **Simulation time** (Інтервал моделювання, або, іншими словами, час розрахунку)**,**
• **Solver options** (Параметри розрахунку),
• **Output options** (Параметри виведення).

1.4.1 Встановлення параметрів розрахунку моделі

1.4.1.1 Simulation time (Інтервал моделювання або час розрахунку)

Час розрахунку задається указуванням початкового (**Start time**) і кінцевого (**Stop time**) значень часу розрахунку.Початковий час, як правило, задається рівним нулю. Величина кінцевого часу задається користувачом, виходячи із умов розв’язуваної задачі.

1.4.1.2 Solver options (Параметри розрахунку)

При виборі параметрів розрахунку необхідно указувати спосіб моделювання (**Type**) і метод розрахунку нового стану системи. Для параметру **Type** доступні два варіанти – з фіксованим (**Fixed-step**) або із змінним (**Variable-step**) кроком. Як правило, **Variable-step** використовується для моделювання неперервних систем, а **Fixed-step** – для дискретних.

Список методів розрахунку нового стану системи вміщує декілька варіантів. Перший варіант (**discrete**) використовується для розрахунку дискретних систем. Решта методів використовуються для розрахунку неперервних систем. Ці методи різні для змінного (**Variable-step**) і для фіксованого (**Fixed-step**) кроків часу, проте, по суті, представляють собою процедури розв'язку систем диференційних рівнянь. Детальніший опис кожного із методів розрахунку станів системи розглядається у вбудованій довідковій системі **MATLAB**.

Нижче двох списків **Type**, що розкриваються, знаходиться область, вмістиме якої змінюється в залежності від вибраного способу змінювання модельного часу. При виборі **Fixed-step** в даній області з'являється текстове поле **Fixed-step size** (величина фіксованого кроку), що дозволяє указувати величину кроку моделювання (див. рис. 1.23).

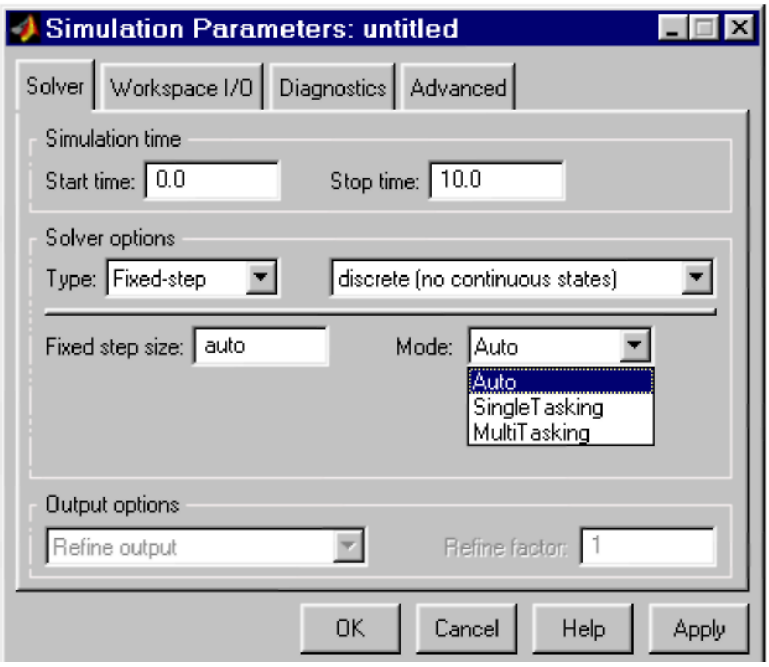

Рис 1.23. Вкладка Solver при виборі фіксованого кроку розрахунку

Величина кроку моделювання за замовчуванням встановлюється системою автоматично (**auto**). Необхідна величина кроку може бути введена замість значення **auto** або **у** формі числа, або у вигляді обчислювального виразу

(те ж саме відноситься і до всіх параметрів, що установлюються системою автоматично).

При виборі **Fixed-step** необхідно також задати режим розрахунку (**Mode**).

Для параметру **Mode** доступні три варіанти:

• **MultiTasking** (Багатозадачний) – необхідно використовувати, якщо в моделі присутні паралельно працюючі підсистеми, і результат роботи моделі залежить від часових параметрів цих підсистем. Режим дозволяє виявити невідповідність швидкості і дискретності сигналів, які передаються блоками друг другу.
• **SingleTasking** (Однозадачний) –використовується для тих моделей, в яких недостатньо строга синхронізація роботи окремих складових не впливає на кінцевий результат моделювання.
• **Auto** (Автоматичний вибір режиму) –дозволяє **Simulink** автоматично встановлювати режим **MultiTasking** для тих моделей, в яких використовуються блоки з різними швидкостями передачі сигналів, і режим **SingleTasking** для моделей, в яких містяться блоки, що оперують одинаковими швидкостями.

При виборі **Variable-step** в її області з'являються поля для установки трьох параметрів:

• **Max step size** – максимальний крок розрахунку. За замовчуванням він установлюється автоматично (**auto**) і його значення в цьому випадку дорівнює (**SfopTime** – **StartTime**)**/50.** Досить часто це значення виявляється дуже великим, і графіки, що спостерігаються, представляють собою ломані (а не плавні) лінії. В цьому випадку величину максимального кроку розрахунку необхідно задавати явним образом.
• **Min step size** – мінімальний крок розрахунку.
• **Initial step size –**початкове значення кроку моделювання.

При моделюванні неперервних систем з використанням змінного кроку необхідно указувати точність обчислень: відносну (**Relative tolerance**) і абсолютну **(Absolute tolerance**). За замовчуванням вони дорівнюють відповідно $\mathbf{10^{-3}}$ і **auto**.

1.4.1.3 Output options (Параметри виведення)

В нижній частині вкладки **Solver** задаються настройки параметрів виведення вихідних сигналів системи, що моделюється (**Output options**). Для даного параметру можливий вибір одного із трьох варіантів:

• **Refine output** (Скоректоване виведення) – дозволяє змінювати дискретність регістрації модельного часу і тих сигналів, які зберігаються в робочій області **MATLAB** за допомогою блоку **To Workspace**. Установка

величини дискретності виконується в рядку редактування **Refine factor,** який розташований справа. За замовчуванням значення **Refine factor** дорівнює 1, це означає, що регістрація відбувається з кроком Dt = 1 (тобто для кожного значення модельного часу). Якщо задати **Refine factor** рівним 2, тоді буде регіструватися кожне друге значення сигналів, 3 – кожне третє і т. д. Параметр **Refine factor** може приймати тільки цілі додатні значення.

• **Produce additional output** (Додаткове виведення) – забезпечує додаткову регістрацію параметрів моделі в задані моменти часу; їх значення уводяться в рядку редактування (в цьому випадку він називається **Output times**) у вигляді списку в квадратних дужках. При використанні цього варіанту базовий крок регістрації (Dt) дорівнює 1. Значення часу в списку **Output times** можуть бути дробовими числами і мати будь-яку точність.
• **Produce specified output only** (Формувати тільки задане виведення) – установлює виведення параметрів моделі тільки в задані моменти часу, які указуються в полі **Output times** (Моменти часу виведення).

1.4.2 Встановлення параметрів обміну з робочою областю

Елементи, які дозволяють керувати введенням і виведенням в робочу область **MATLAB** проміжних даних і результатів моделювання, розташовані на вкладці **Workspace I/O** (рис. 1.24).

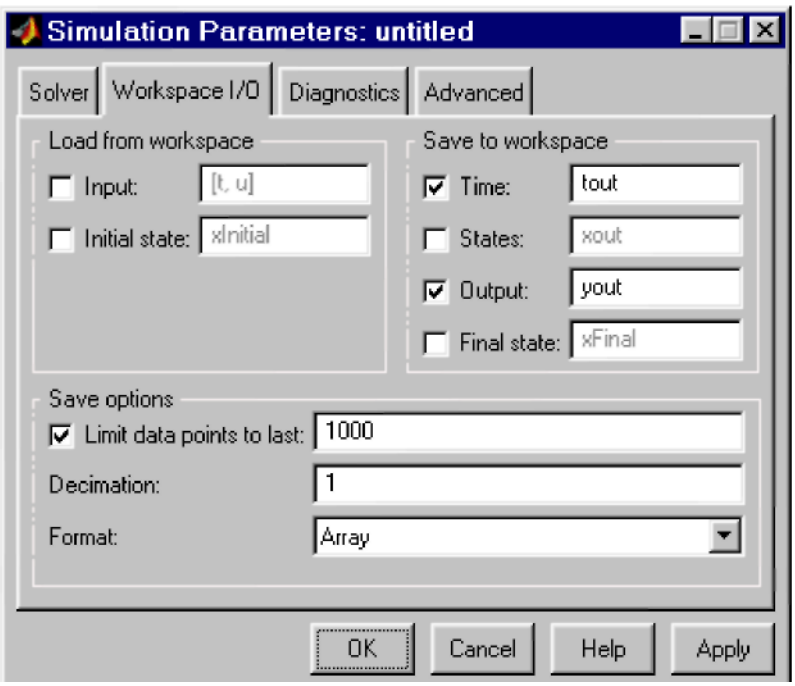

Рис 1.24. Вкладка Workspace I/O діалогового вікна установки параметрів моделювання

Елементи вкладки поділені на 3 поля:

• **Load from workspace** (Завантажити із робочої області). Якщо прапорець **Input** (Вхідні дані) встановлений, тоді в розташованому справа текстовому полі можна ввести формат даних, які будуть відраховуватися із робочої області **MATLAB**. Встановлення прапорця **Initial State** (Початковий стан) дозволяє увести в зв'язаному з ним текстовому полі ім'я змінної, що містить параметри початкового стану моделі. Дані, які указані в полях **Input** і **Initial State**, передаються у виконувану модель через одного або більше блоків **In** (із розділу бібліотеки **Sources**).
• **Save to workspace** (Записати в робочу область) –Дозволяє установити режим виведення значень сигналів в робочу область **MATLAB** і задати їх імена.
• **Save options** (Параметри запису) – Задає кількість рядків при передачі змінних в робочу область. Якщо прапорець **Limit rows to last** установлений, тоді в полі введення можна указати кількість рядків, що передаються (відлік рядків відбувається від моменту закінчення розрахунку). Якщо прапорець не установлений, тоді передаються всі дані. Параметр **Decimation** (Виключення) задає крок запису змінних в робочу область (аналогічно параметру **Refine factor** вкладки **Solver**). Параметр **Format** (формат даних) задає формат передаваємих в робочу область даних. Доступні формати **Array** (Масив), **Structure** (Структура**), Structure With Time** (Структура з додатковим полем – «час»).

1.4.3 Встановлення параметрів діагностування моделі

Вкладка **Diagnostics** (рис.1.25) дозволяє змінювати перелік діагностичних повідомлень, які виводяться **Simulink** в командному вікні **MATLAB**, а також установлювати додаткові параметри діагностики моделі.

Повідомлення про помилки або проблемні ситуації, що виявлені **Simulink** на протязі моделювання і вимагають втручання розробника, виводяться в командному вікні **MATLAB**. Початковий перелік таких ситуацій і вид реакції на них приведений в списку на вкладці **Diagnostics**. Розробник може указати вид реакції на кожний із них, використовуючи групу перемикачів в полі **Action** (вони стають доступні, якщо в списку вибрана одна із подій):

• **None** –ігнорувати,
• **Warning** – видати попередження і продовжити моделювання,
• **Error** – видати повідомлення про помилку і зупинити процес моделювання.Вибраний вид реакції відображається в списку рядом з назвою події.

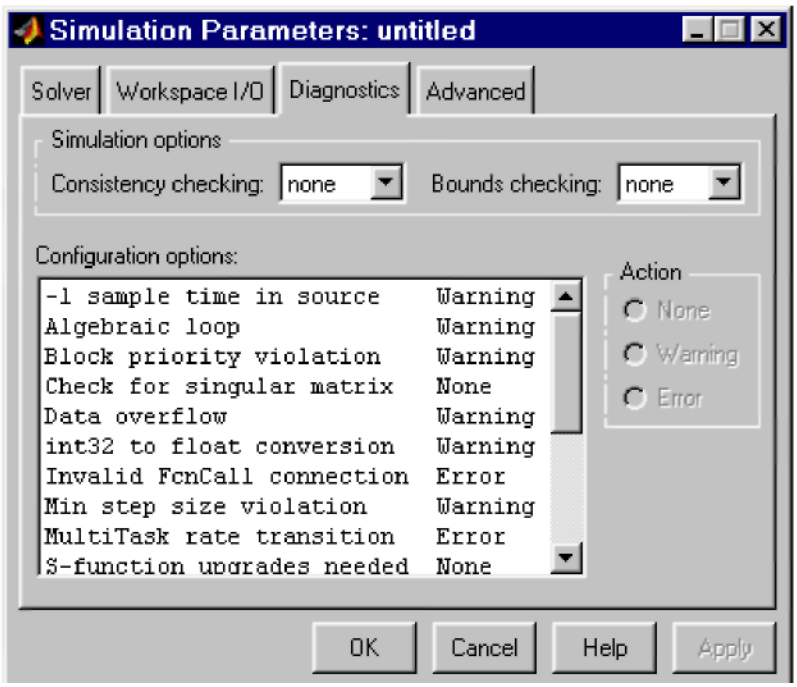

Рис 1.25. Вкладка Diagnostics вікна встановлення параметрів моделювання

1.4.4 Виконання розрахунку

Запуск розрахунку виконується за допомогою вибору пункту меню **Simulation/Start** або інструменту ▶ на панелі інструментів. Процес розрахунку можна завершити дотерміново, вибравши пункт меню **Simulation/Stop** або інструмент ■. Розрахунок також можна зупинити (**Simulation/Pause)** і потім продовжити (**Simulation/Continue)**.

1.5 Огляд бібліотеки блоків Simulink

1.5.1 Sources – джерела сигналів

1.5.1.1 Джерело постійного сигналу Constant

Призначення:

Задає постійний за рівнем сигнал.

Параметри:

1. **Constant value** –Постійна величина.

2. **Interpret vector parameters as 1-D –** Інтерпретувати вектор параметрів як одновимірний (при установленому прапорці). Даний параметр зустрічається

у більшості блоків бібліотеки **Simulink**. В подальшому він розглядатися не буде.

Значення константи може бути дійсним або комплексним числом, що обчислює вирази, вектори або матриці.

Рис. 1.26. ілюструє застосування цього джерела і вимірювання його вихідного сигналу за допомогою цифрового індикатора **Display**.

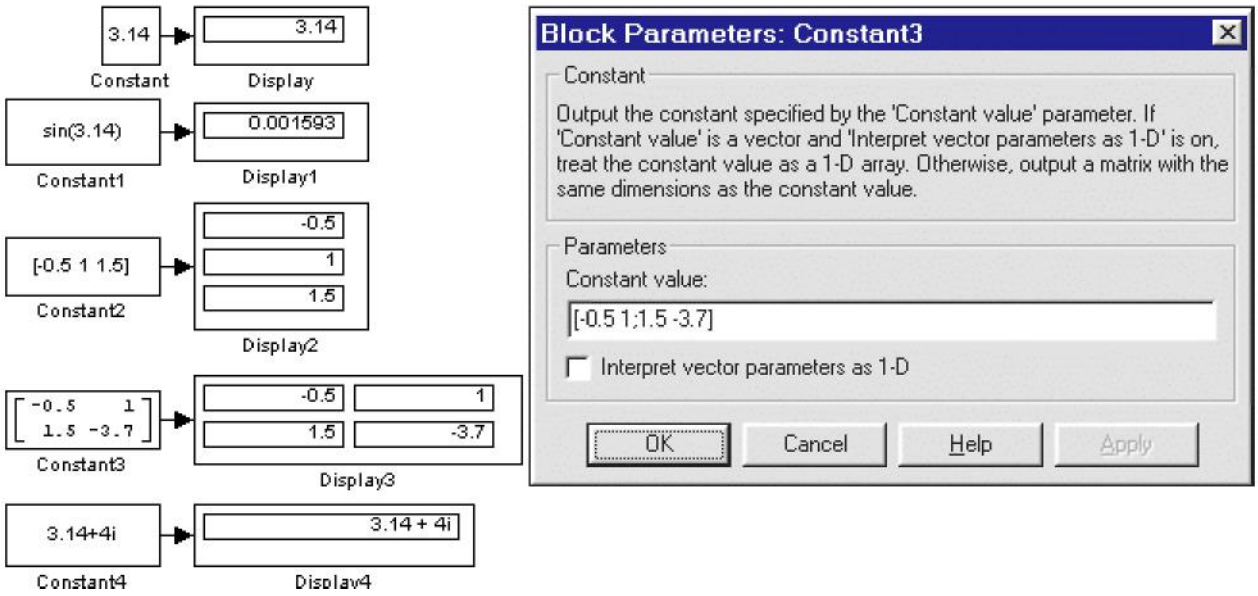

Рис. 1.26. Джерело постійного впливу Constant

1.5.1.2 Генератор ступінчастого сигналу Step

Призначення:

Формує ступінчастий сигнал.

Параметри:

1. **Step time** –Час початку перепаду сигналу (с).
2. **Initial value** –Початкове значення сигналу.
3. **Final value** –Кінцеве значення сигналу.

Перепад може бути як в більшу сторону (кінцеве значення більше чим початкове), так і в меншу (кінцеве значення менше чим початкове). Значення початкового і кінцевого рівнів можуть бути не тільки додатніми, але і від'ємними (наприклад, змінювання сигналу з рівня –5 до рівня –3).

На рис. 1.27. показане використання генератора ступінчастого сигналу.

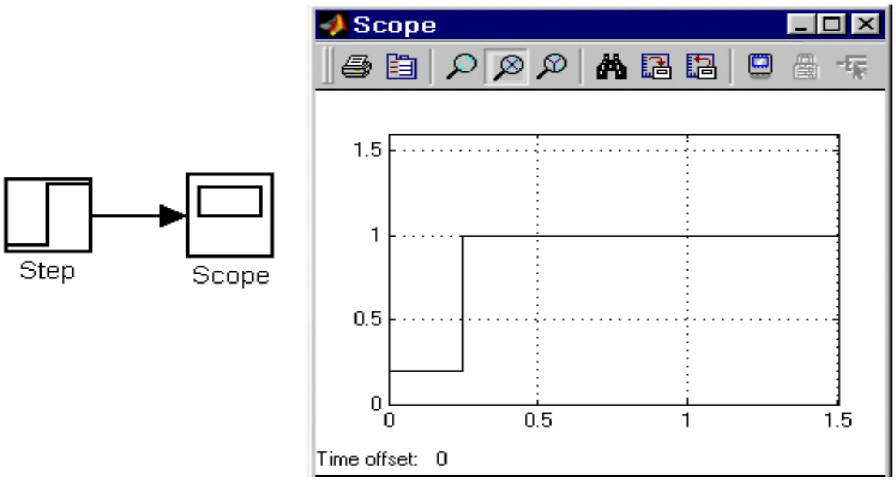

Рис. 1. 27. Блок Step

1.5.1.3. Генератор сигналів Signal Generator

Призначення:

Формує один із чотирьох видів періодичних сигналів:

1. **sine** –Синусоїдний сигнал.
2. **square** – Прямокутний сигнал.
3. **sawtooth** – Пилкоподібний сигнал.
4. **random** –Випадковий сигнал. *Параметри*:

1. **Wave form** – Вид сигналу.
2. **Amplitude** – Амплітуда сигналу.
3. **Frequency** – Частота (рад/с).
4. **Units** –Одиниці вимірювання частоти. Може приймати два значення:

• **Hertz –** Гц.
• **rad/sec –** рад/с.

На рис. 1.28. показане застосування цього джерела при моделюванні прямокутного сигналу.

1.5.1.4 Джерело випадкового сигналу з рівномірним розприділенням Uniform Random Number

Призначення:

Формування випадкового сигналу з рівномірним розприділенням.

Параметри:

1. **Minimum** – Мінімальний рівень сигналу.
2. **Maximum** – Максимальний рівень сигналу.
3. **Initial seed** – Початкове значення.

Приклад використання блоку і графік його вихідного сигналу представлені на рис. 1.29.

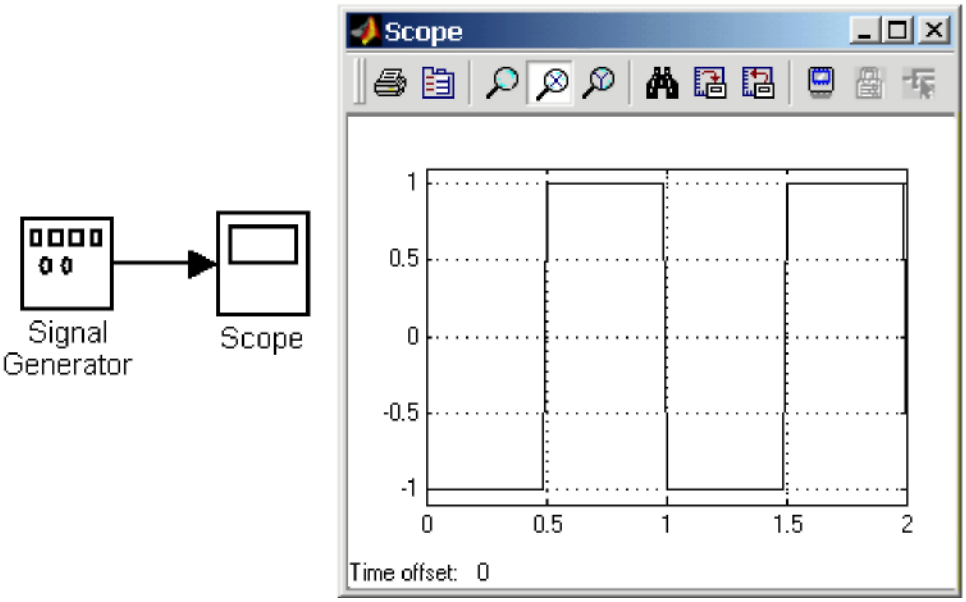

Рис. 1. 28. Блок генератора сигналів

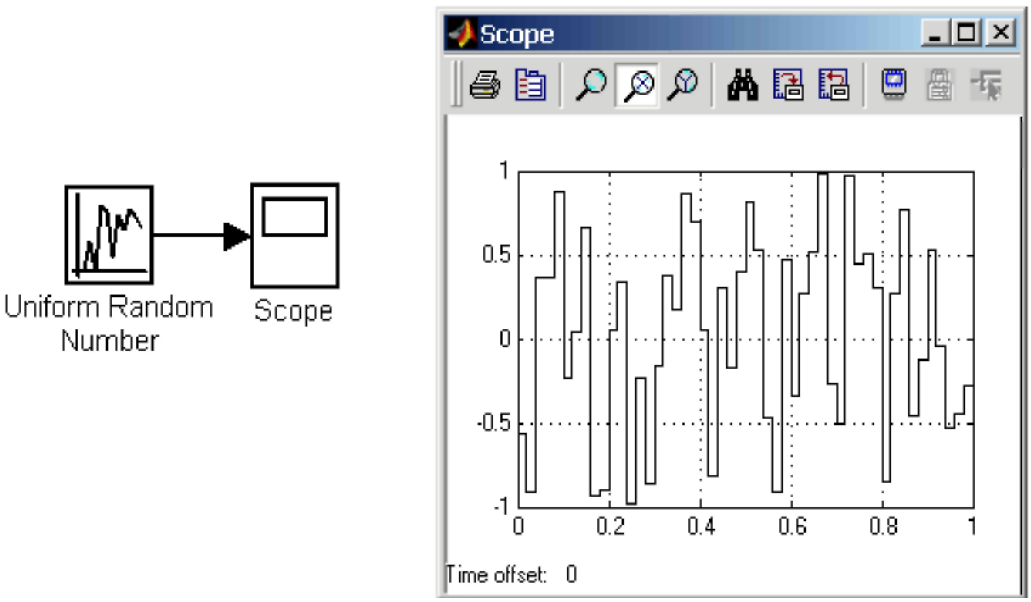

Рис. 1. 29. Джерело випадкового сигналу з рівномірним розприділенням

1.5.1.5 Джерело випадкового сигналу з нормальним розприділенням Random Number

Призначення:

Формування випадкового сигналу з нормальним розприділенням рівня сигналу (рис.1.30).

Параметри:

1. **Mean** – Середнє значення сигналу
2. **Variance** – Дисперсія (середньоквадратичне відхилення).
3. **Initial seed** –Початкове значення.

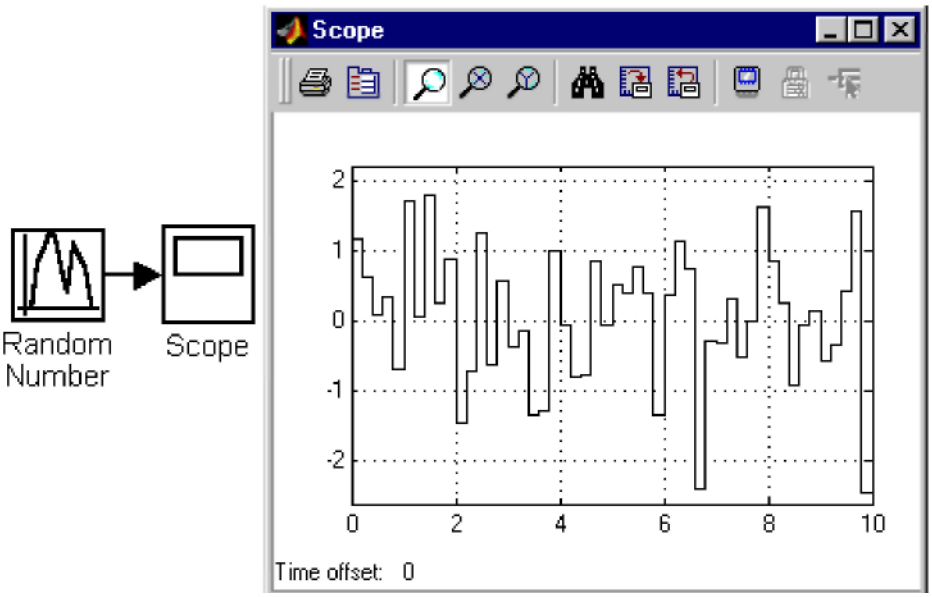

Рис. 1. 30. Джерело випадкового сигналу з нормальним роз приділенням

1.5.1.6 Джерело імпульсного сигналу Pulse Generator

Призначення:

Формування прямокутних імпульсів (рис. 1.31.)

Параметри:

1. Pulse Type – Спосіб формування сигналу. Може приймати два значення:

- **Time-based** –За поточним часом.
- **Sample-based** – За величиною модельного часу і кількістю розрахункових кроків.

2. Amplitude – Амплітуда.
3. Period – Період. Задається в секундах для **Time-based Pulse Type** або в кроках модельного часу для **Sample-based Pulse Type**.
4. Pulse width – Ширина імпульсів. Задається в процентах по відношенню до періоду для **Time-based Pulse Type** або в кроках модельного часу для **Sample-based Pulse Type**.
5. Phase delay – Фазова затримка. Задається в секундах для **Time-based Pulse**

Type або в кроках модельного часу для **Sample-based Pulse Type.**
6. Sample time –Крок модельного часу . Задається для **Sample-based Pulse Type**.

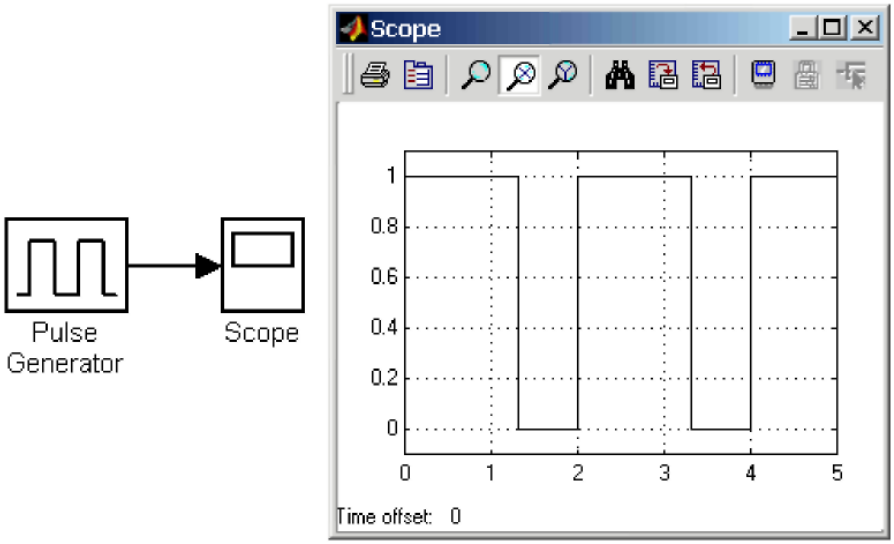

Рис. 1. 31. Джерело прямокутних імпульсів

1.5.1.7 Джерело часового сигналу Clock

Призначення:

Формує сигнал, величина якого на кожному кроці розрахунку дорівнює поточному часу моделювання .

Параметри:

1. **Decimation** –Крок, з яким оновлюється показ часу на зображенні джерела (в тому випадку, якщо встановлений прапорець параметру **Display time**). Параметр задається як кількість кроків розрахунку. Наприклад, якщо крок розрахунку моделі у вікні діалогу **Simulation parameters в**становлений рівним **0,01** с, а параметр **Decimation** блоку **Clock** заданий рівним **1000**, тоді оновлення показів часу буде відбуватися кожні **10** с модельного часу (рис. 1.32).

2. **Display time** –Відображення значення часу в блоці джерела. На рис. 1.32 показаний приклад роботи даного джерела.

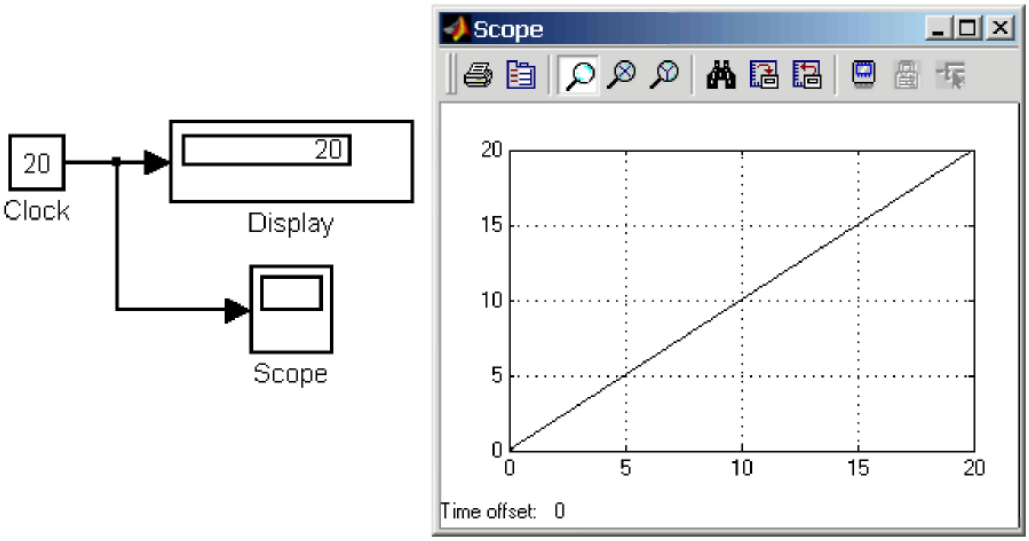

Рис. 1. 32. Джерело часового сигналу

1.5.1.8 Цифрове джерело часу Digital Clock

Призначення:

Формує дискретний часовий сигнал.

Параметр:

Sample time –Крок модельного часу (с). На рис. 1.33 показана робота джерела **Digital Clock**

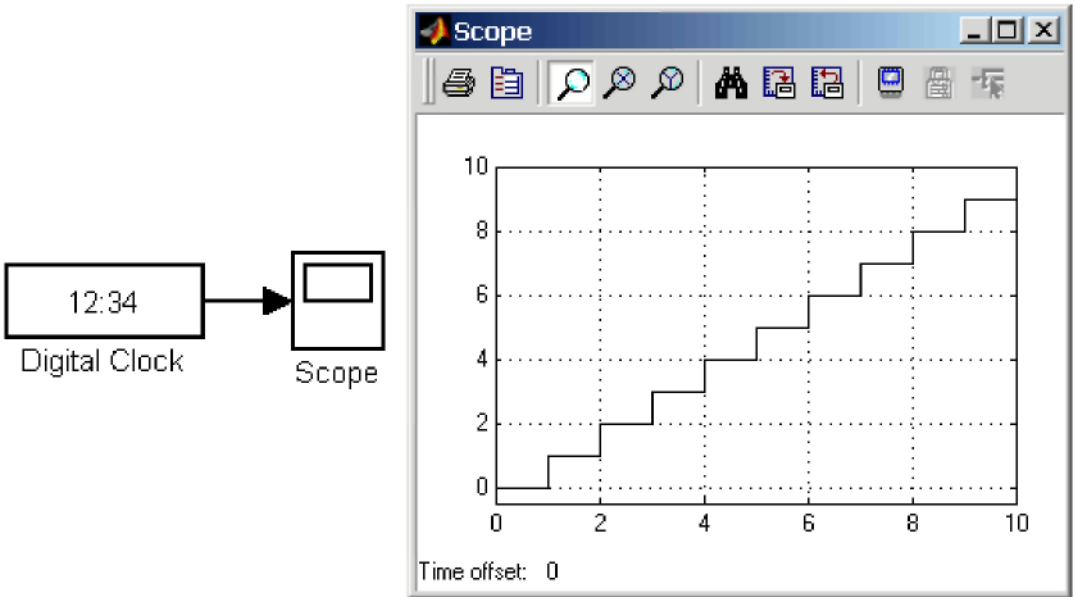

Рис. 1. 33. Цифрове джерело часового сигналу

1.5.2 Sinks – приймачі сигналів

1.5.2.1 Осцилограф Scope

Призначення:

Будує графіки досліджуваних сигналів в функції часу. Дозволяє спостерігати за змінами сигналів в процесі моделювання.

Зображення блоку і вікно для перегляду графіків показані на рис. 1.34.

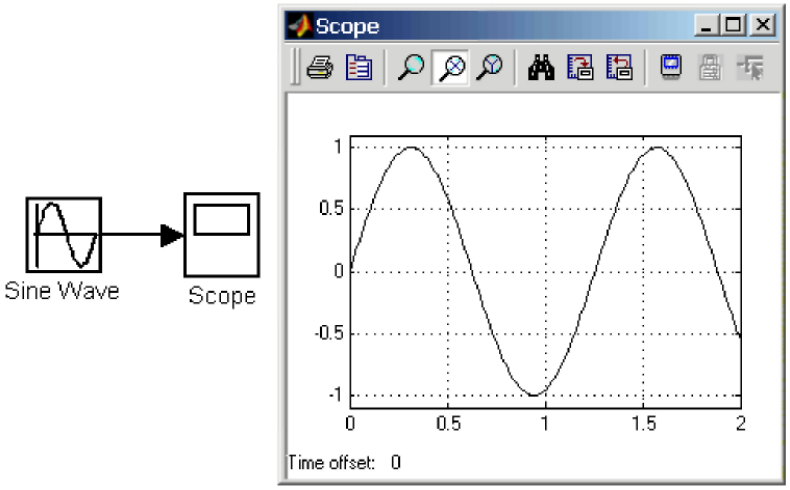

Рис. 1. 34. Осцилограф Scope

Для того щоб відкрити вікно перегляду сигналів, необхідно виконати подвійне клацання лівою клавішою мишки на зображенні блоку. Це можна зробити на будь-якому етапі розрахунку (як до початку розрахунку, так і після нього, а також під час розрахунку). В тому випадку, якщо на вхід блоку поступає векторний сигнал, тоді крива для кожного елементу вектора будується окремим кольором.

Настроювання вікна осцилографу виконується за допомогою панелей інструментів (рис. 1.35).

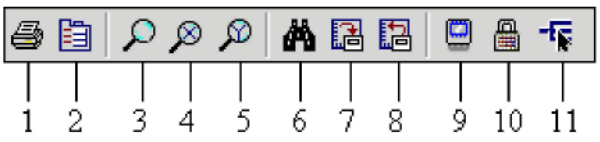

Рис. 1. 35. Панель інструментів блоку Scope

Панель інструментів має 11 кнопок:

1. **Print** – друк вмістимого вікна осцилографа.

2. **Parameters** - доступ до вікна настроювання параметрів.
3. **Zoom** – збільшення масштабу по обом вісям.
4. **Zoom X-axis** – збільшення масштабу по горизонтальній вісі.
5. **Zoom Y-axis** - збільшення масштабу по вертикальній вісі.
6. **Autoscale** - автоматична установка масштабів по обом вісям.
7. **Save current axes settings** – збереження поточних настроювань вікна.
8. **Restore saved axes settings** - установка раніше збережених настроювань вікна.
9. **Floating scope** - переведення осцилографа в *«вільний»* режим.
10. **Lock/Unlock axes selection** – закріпити/розірвати зв'язки між поточноюкоординатною системою вікна і відображаємим сигналом. Інструмент доступний, якщо включений режим **Floating scope**.

11. **Signal selection** - вибір сигналів для відображення. Інструмент доступний, якщо включений режим **Floating scope**.

Змінювання масштабів відображаємих графіків можна виконувати декількома способами:

1. Натиснути відповідну кнопку(, або)і клацнути один раз лівою клавішою мишки в потрібному місці графіка. Відбудеться 2,5 кратне збільшення масштабу.

2. Натиснути відповідну кнопку (, або) і, натиснувши ліву клавішу мишки, за допомогою динамічної рамки або відрізку вказати область графіка для збільшеного зображення. Рис. 1.36 пояснює цей процес.

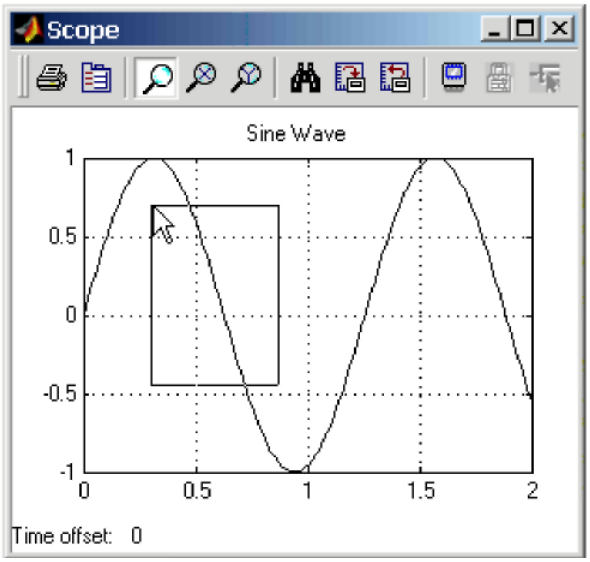

Рис. 1. 36. Збільшення масштабу графіка.

3. Клацнути правою клавішою мишки у вікні графіків і вибрати комаду **Axes properties…** в контекстному меню. Відкриється вікно властивостей

графіка, в якому за допомогою параметрів **Y-min** і **Y-max** можна вказати граничні значення вертикальної вісі. В цьому ж вікні можна вказати заголовок графіка (**Title**), замінивши вираз **%<SignalLabel>** в рядку введення. Вікно властивостей показане на рис. 1.37.

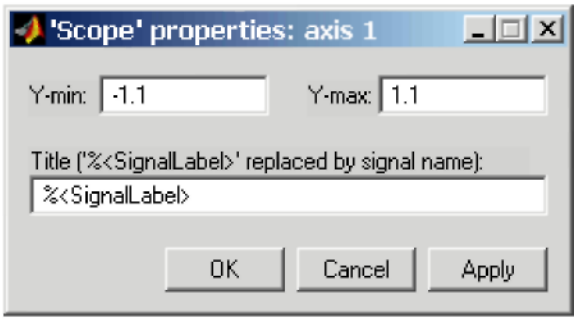

Рис. 1. 37. Вікно властивостей графіка.

Параметри:

Параметри блоку установлюються у вікні **'Scope' parameters**, яке відкривається за допомогою інструменту  (**Parameters**) панелі інструментів. Вікно параметрів має дві вкладки:

General – загальні параметри.

Data history - параметри збереження сигналів в робочій області **MATLAB.**

Вкладка загальних параметрів показана на рис. 1.38.

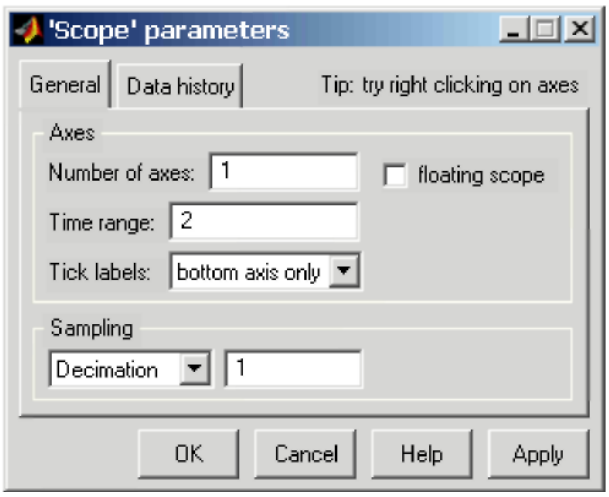

Рис. 1. 38. Вкладка загальних параметрів General.

На вкладці **General** задаються наступні параметри:

1. Number of axes - число входів (систем координат) осцилографа. При змінюванні цього параметру на зображенні блоку появляються додаткові вхідні порти.

2. Time range - величина часового інтервалу, для якого відображаються графіки. Якщо час розрахунку моделі перевищує задане параметром **Time range**, тоді виведення графіка відбувається порціями, при цьому інтервал відображення кожної порції графіка рівний заданому значенню **Time range**.

3. Tick labels - виведення/приховування вісів і міток вісів. Може приймати три значення (вибираються зі списку):

all - підписи для всіх вісів,

none – відсутність всіх вісів і підписів до них,

bottom axis only - підписи горизонтальної вісі тільки для нижнього графіка.

4. Sampling - установка параметрів виведення графіків у вікні. Задає режим виведення розрахункових точок на екран. При виборі **Decimation** кратність виведення установлюється числом, що задає крок розрахункових точок, які виводяться. На рис. 1.39 і 1.40 показані графіки синусоїдних сигналів, які розраховані з фіксованим кроком **0.1** с. На рис. 1.39 у вікні блоку **Scope** виводиться кожна розрахункова точка (параметр **Decimation** дорівнює 1). На рис. 1.40 показано виведення кожного другого значення (параметр **Decimation** дорівнює 2). Маркерами на графіках відмічені розрахункові точки.

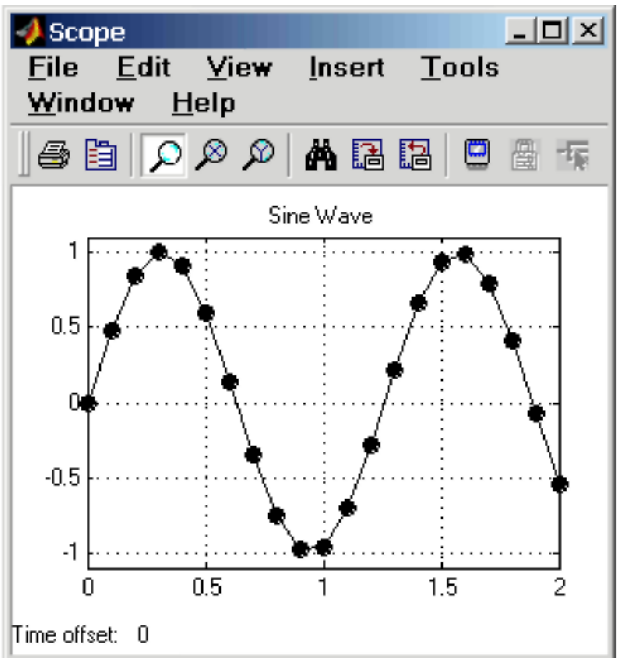

Рис. 1. 39. Відображення синусоїдного сигналу (Decimation = 1).

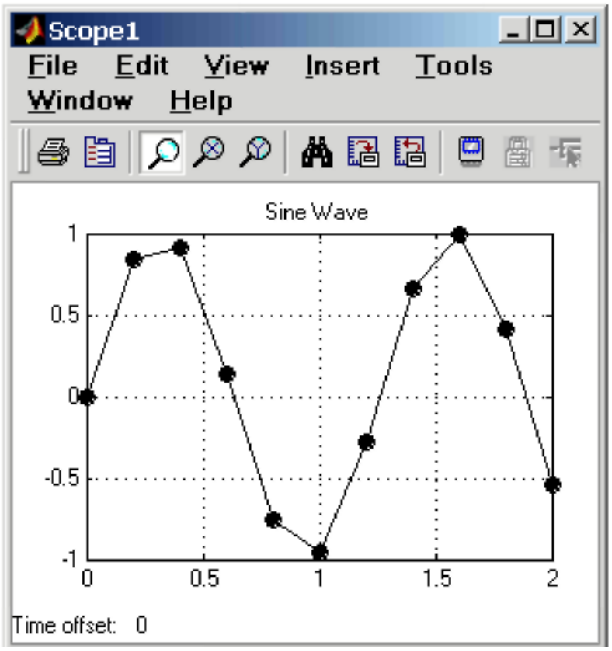

Рис. 1. 40. Відображення синусоїдного сигналу (Decimation = 2).

В тому випадку, якщо режим виведення розрахункових точок задається як **Sample time**, тоді його числове значення визначає інтервал квантування при відображенні сигналу. На рис. 1.41 показаний графік синусоїдного сигналу для випадку, коли значення параметру **Sample time** дорівнює **0,1**.

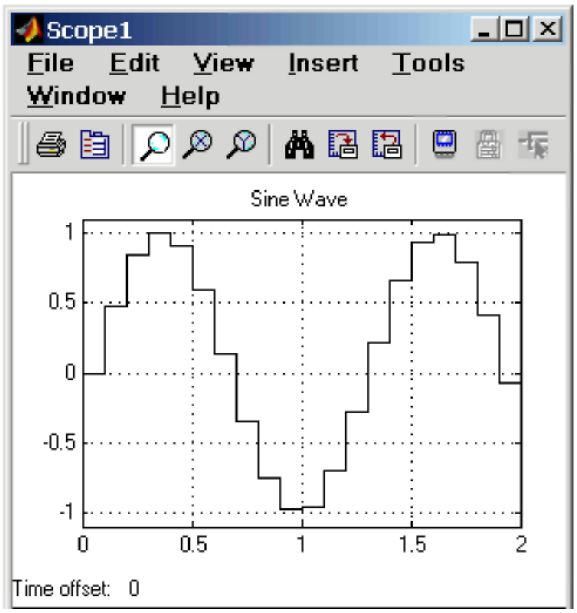

Рис. 1. 41. Відображення синусоїдного сигналу (Sample time = 0,1).

5. floating scope – переведення осцилографа в «*вільний*» режим (при установленому прапорці). На вкладці **Data history** (рис. 1. 42) задаються наступні параметри:

1. **Limit data points to last** – максимальна кількість відображаємих розрахункових точок графіка. При перевищенні цього числа початкова частина графіка обрізається. В тому випадку, якщо прапорець параметру **Limit data points to last** не встановлений, тоді **Simulink** автоматично збільшить значення цього параметру для відображення всіх розрахункових точок.

2. **Save data to workspace** –збереження значень сигналів в робочій області **MATLAB**.

3. **Variable name – і**м'я змінної для збереження сигналів в робочій області **MATLAB**.

4. **Format –** формат даних при збереженні в робочій області **MATLAB**. Може приймати значення:

• **Array** – масив,

• **Structure** – структура,

• **Structure with time** – структура з додатковим полем «час».

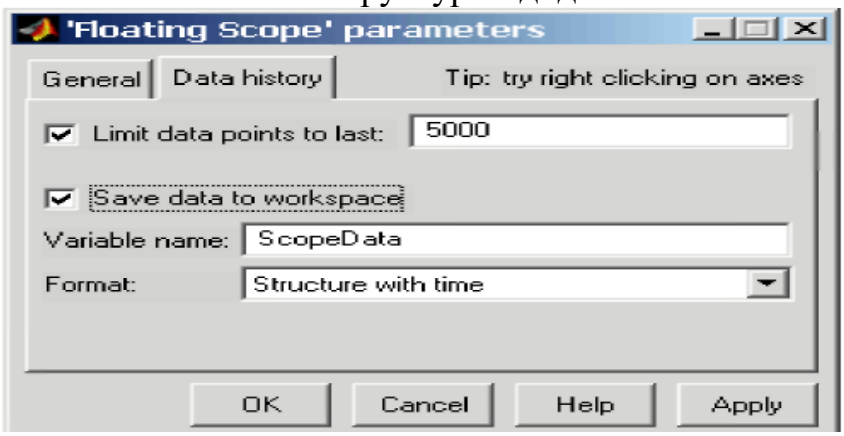

Рис. 1. 42. Вкладка Data history.

1.5.2.2 Графопобудовник XY Graph

Призначення:

Будує графік одного сигналу в функції іншого (графік виду **Y(X)**).

Параметри:

x-min – Мінімальне значення сигналу по вісі **X**.

x-max – Максимальне значення сигналу по вісі **X.**

y-min – Мінімальне значення сигналу по вісі **Y**.

y-max – Максимальне значення сигналу по вісі **Y**

Sample time –крок модельного часу.

Блок має два входи. Верхній вхід призначений для подачі сигналу, який являється аргументом (**X**), нижній – для подачі значень функції (**Y**).

На рис. 1. 43, в якості прикладу використання графопобудовника, показана побудова фазової траєкторії коливальної ланки.

Графопобудовник можна використовувати і для побудови часових залежностей. Для цього на перший вхід необхідно подати часовий сигнал з виходу блока **Clock**. Приклад такого використання графопобудовника показаний на рис. 1. 44.

1.5.2.3 Цифровий дисплей Display

Призначення:

Відображає значення сигналу у вигляді числа.

Параметри:

1. Format – формат відображення даних. Параметр **Format** може приймати наступні значення:

• **short –** 5 значущих десяткових цифр.

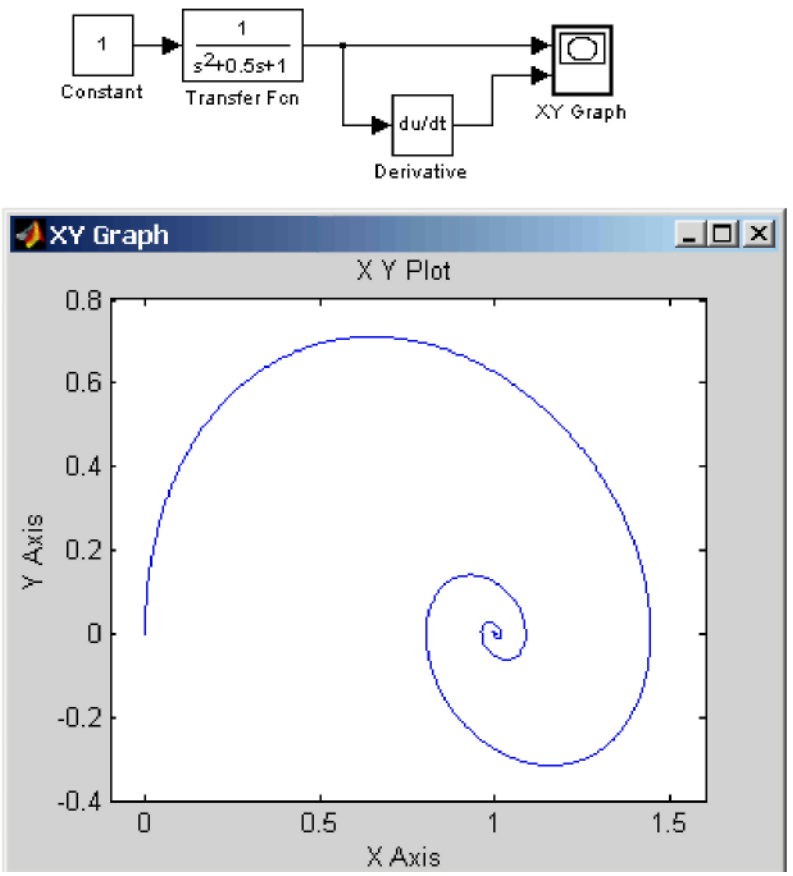

Рис. 1. 43. Приклад використання графопобудовника XY Graph.

- **long –** 15 значущих десяткових цифр.
- **short_e –** 5 значущих десяткових цифр і 3 символи степені десяти.
- **long_e –** 15 значущих десяткових цифр і 3 символи степені десяти.
- **bank –** «*грошовий*» формат. Формат з фіксованою точкою і двома десятковими цифрами в дробовій частині числа.

2. Decimation – кратність відображення вхідного сигналу. При **Decimation = 1** відображається кожне значення вхідного сигналу, при **Decimation = 2** – кожне друге значення, при **Decimation = 3** – кожне третє значення і т.д.

3. Sample time –крок модельного часу. Визначає дискретність відображення даних.

4. Floating display (прапорець) – переведення блоку в «*вільний*» режим. В даному режимі вхідний порт блоку відсутній, а вибір сигналу для відображення виконується клацанням лівої клавіші мишки на відповідній лінії зв'язку. В цьому режимі для параметру розрахунку **Signal storage reuse** повинне бути установлене значення **off** (вкладка **Advanced** у вікні діалогу **Simulation parameters…**). На рис. 1. 45 показано застосування блоку **Display** з використанням різних варіантів параметру **Format**.

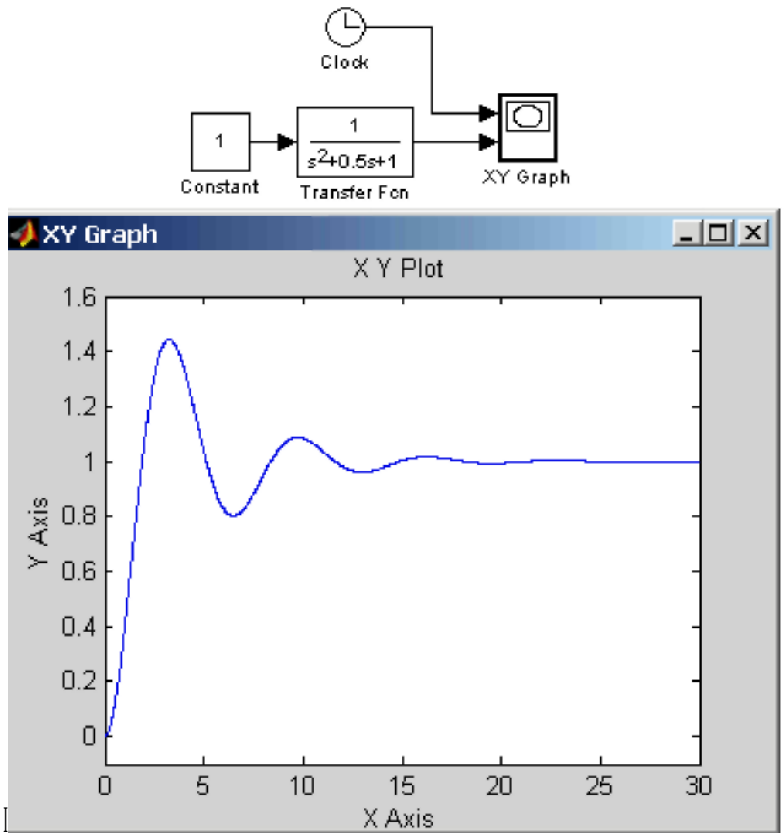

Рис. 1. 44. Приклад використання блоку XY Graph для відображення часових залежностей.

Блок **Display** може використовуватися для відображення не тільки скалярних сигналів, але також векторних, матричних і комплексних. Рис. 1. 46 ілюструє це. Якщо всі відображаємі значення не можуть поміститися у вікні блоку, в правому нижньому куті блоку появляється символ, що указує на необхідність збільшити розміри блоку (див. блок **Display4** на рис. 1. 46).

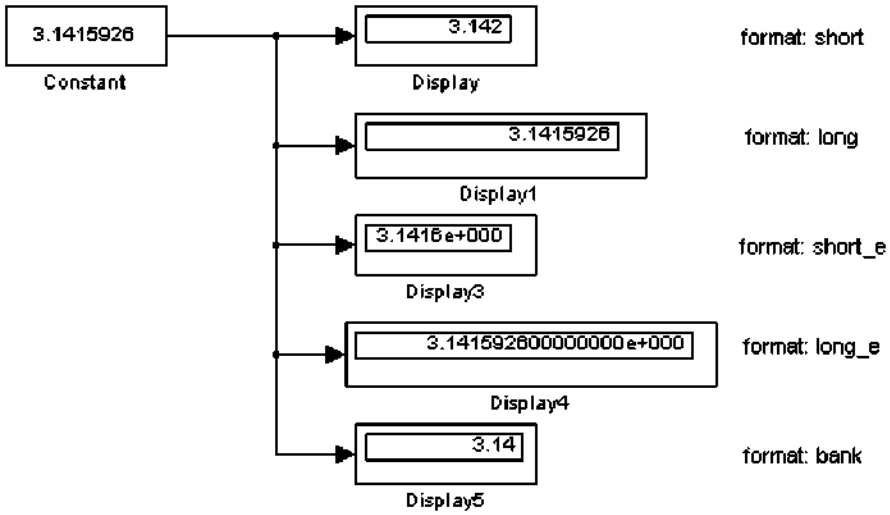

Рис. 1. 45. Застосування блоку Display з використанням різних варіантів параметру Format.

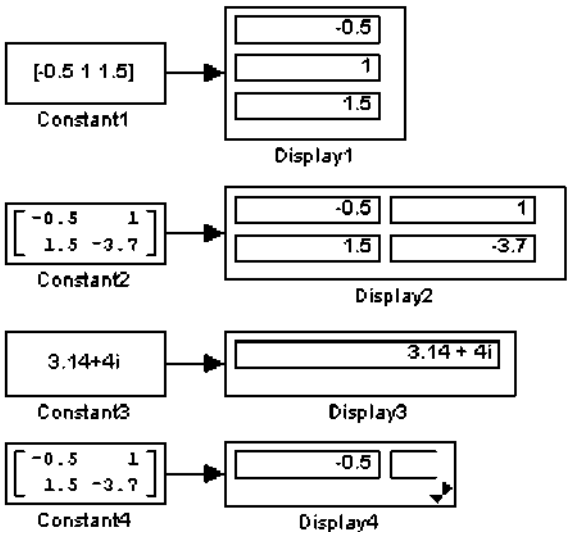

Рис. 1. 46. Застосування блоку Display для відображення векторних, матричних і комплексних сигналів.

1.5.2.4 Блок зупинки моделювання Stop Simulation

Призначення:

Забезпечує закінчення розрахунку, якщо вхідний сигнал блоку становиться нерівним нулю.

Параметри:

Немає.

При подачі на вхід блоку ненульового сигналу **Simulink** виконує поточний крок розрахунку, а потім зупиняє моделювання. Якщо на вхід блоку поданий векторний сигнал, тоді для зупинки розрахунку достатньо, щоб один елемент вектора став ненульовим. На рис. 1. 47 показаний приклад використання даного блоку. В прикладі зупинка розрахунку відбувається, якщо вихідний сигнал у **Transfer Function** становиться великим **або** рівним **0,99**.

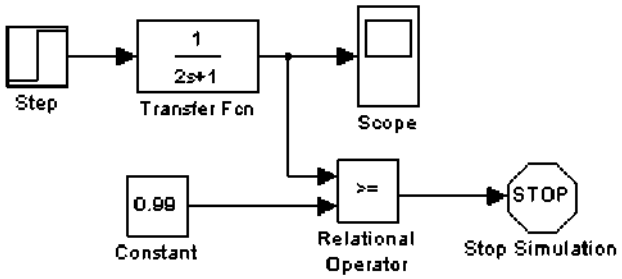

Рис. 1. 47. Застосування блоку Stop Simulation

1.5.3 Continuous – аналогові блоки

1.5.3.1 Блок Memory

Призначення:

Виконує затримку вхідного сигналу на один часовий такт.

Параметри:

- **Initial condition** –початкове значення вихідного сигналу.
- **Inherit sample time** (прапорець) – наслідувати крок модельного часу. Якщо цей прапорець установлений, тоді блок **Memory** використовує крок модельного часу (**Sample time**) такий же, як і в попередньому блоці.

На рис. 1. 48 показаний приклад використання блоку **Memory** для затримки дискретного сигналу на один часовий такт.

1.5.3.2 Блок фіксованої затримки сигналу Transport Delay

Призначення:

Забезпечує затримку вхідного сигналу на заданий час. *Параметри:*

1. **Time Delay** —Час затримки сигналу (невід'ємне значення).
2. **Initial input** —Початкове значення вихідного сигналу.

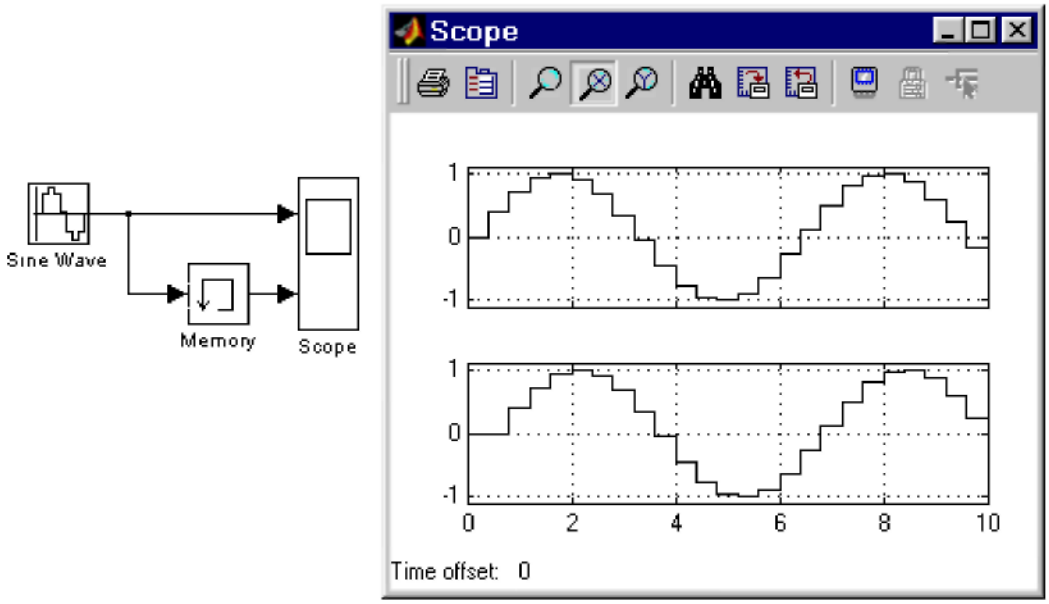

Рис. 1. 48. Застосування блоку для затримки сигналу на один часовий такт

3. Buffer size — Розмір пам'яті, що виділяється для зберігання затриманого сигналу. Задається в байтах числом, кратним восьми (за замовчуванням -1024).

4. Pade order (for linearization) — Порядок ряду Паде, який використовується при апроксимації вихідного сигналу. Задається цілим додатнім числом.

При виконанні моделювання значення сигналу і відповідаючий йому модельний час зберігаються у внутрішньому буфері блоку **Transport Delay**. По закінченню часу затримки значення сигналу витягується із буферу і передається на вихід блоку. В тому випадку, якщо кроки модельного часу не співпадають із значеннями моментів часу для записаного в буфер сигналу, блок **Transport Delay** виконує апроксимацію вихідного сигналу.

В тому випадку, якщо початкового значення об'єму пам'яті буферу не хватить для зберігання затриманого сигналу, **Simulink** автоматично виділить додаткову пам'ять. Після закінчення моделювання в командному вікні **MATLAB** появиться повідомлення з указуванням необхідного розміру буфера.

На рис. 1.49 показаний приклад використання блоку **Transport Delay** для затримки прямокутного сигналу на **0,5** с.

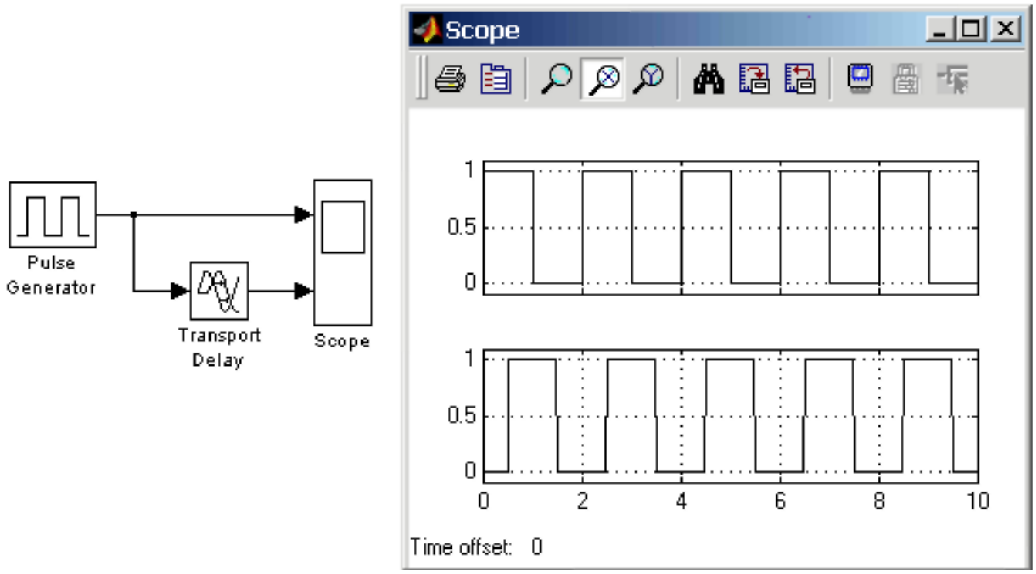

Рис. 1. 49. Приклад використання блоку Transport Delay для затримки сигналу.

1.5.4 Discrete – дискретні блоки

1.5.4.1 Блок одиничної дискретної затримки Unit Delay

Призначення:

Виконує затримку вхідного сигналу на один крок модельного часу.

Параметри:

1. **Initial condition** –Початкове значення для вихідного сигналу.
2. **Sample time** –Крок модельного часу.

Вхідний сигнал блоку може бути як скалярним, так і векторним. При векторному сигналі затримка виконується для кожного елементу вектора. Блок підтримує роботу з комплексними і дійсними сигналами.

На рис. 1.50 показаний приклад використання блоку для затримки дискретного сигналу на один часовий крок, рівний **0,1 с.**

1.5.4.2 Блок дискретного інтегратора Discrete-Time Integrator

Призначення:

Блок використовується для виконання операції інтегрування в дискретних системах.

Параметри:

1. **Integration method** – Метод чисельного інтегрування:

• **Forward Euler** – Прямий метод Ейлера.

Метод використовує апроксимацію T/(z-1) передатної функції 1/s.

Вихідний сигнал блоку розраховується за виразом:

y(k) = y(k–1) + T*u(k–1),

y – вихідний сигнал інтегратора,

u – вхідний сигнал інтегратора,

T –крок дискретизації,

k – номер кроку моделювання.

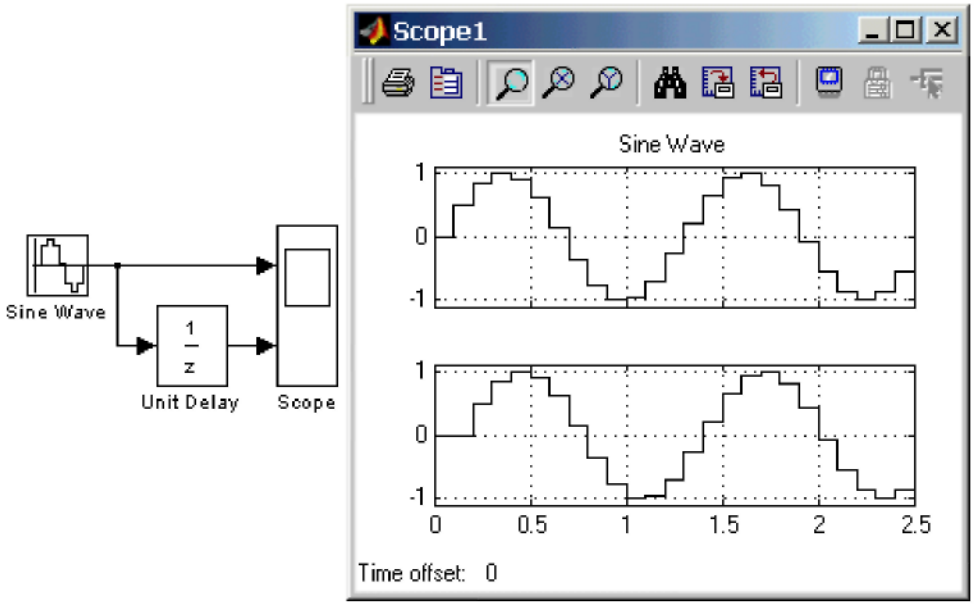

Рис. 1. 50. Приклад використання блоку Unit Delay

• **Backward Euler** – Обернений метод Ейлера.

Метод використовує апроксимацію **T*z/(z–1)** передатної функції **1/s.** Вихідний сигнал блоку розраховується за виразом:

y(k) = y(k–1) + T*u(k).

• **Trapeziodal** – Метод трапецій.

Метод використовує апроксимацію (T/2)*(z+1)/(z–1) передатної функції **1/s.** Вихідний сигнал блоку розраховується за виразом:

x(k) = y(k–1) + (T/2) *u(k–1).

2. Sample time – Крок дискретизації за часом.

Решта параметрів дискретного інтегратора ті ж, що і у блока аналогового інтегратора **Integrator** (бібліотека **Continuous**).

На рис. 1. 51 показаний приклад, який демонструє всі три способи чисельного інтегрування блоку **Discrete-Time Integrator**. Як видно із рисунку, зображення блоку змінюється в залежності від вибраного методу інтегрування.

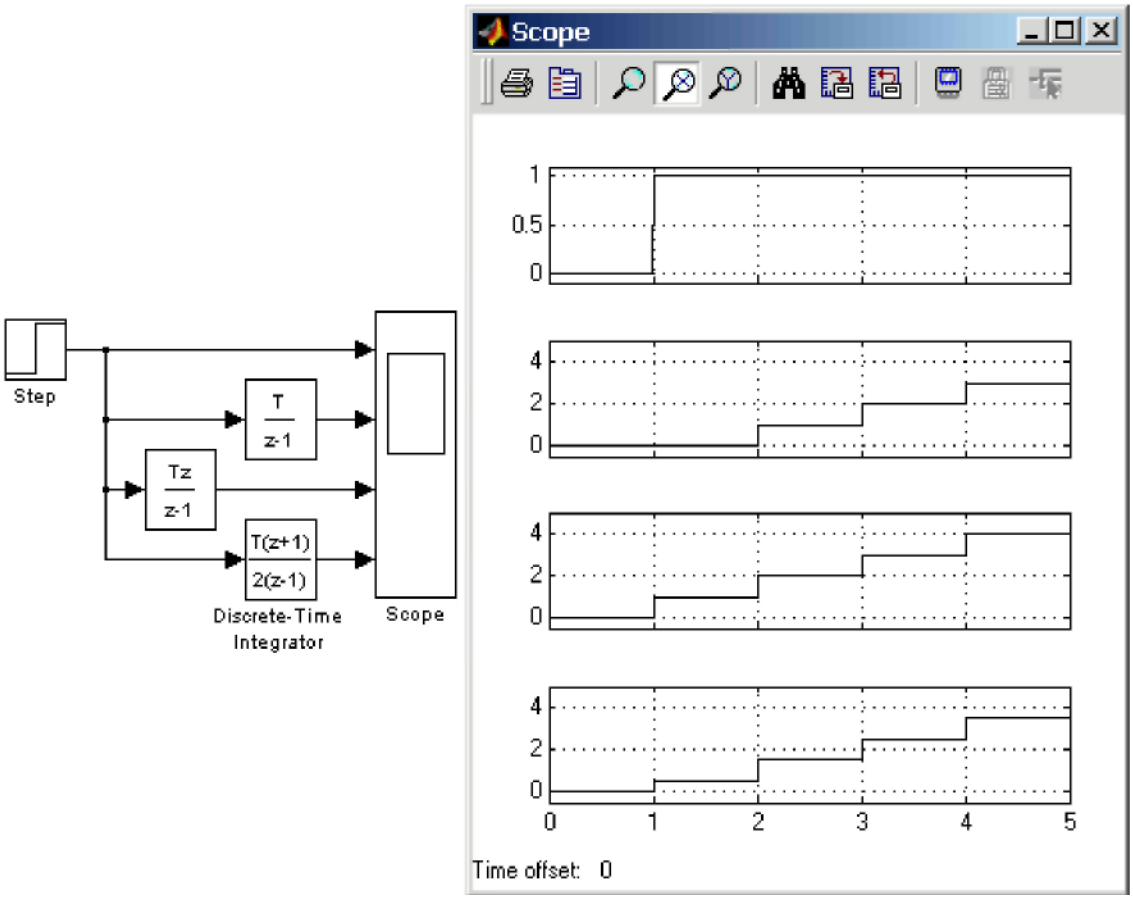

Рис. 1. 51. Виконання інтегрування блоками Discrete-Time Integrator, що реалізують різні чисельні методи.

1.5.5 Nonlinear - нелінійні блоки

1.5.5.1 Блок перемикача Switch

Призначення:

Виконує перемикання вхідних сигналів по сигналу керування.

Параметри:

Threshold – Поріг керуючого сигналу.

Блок працює наступним чином.

Якщо сигнал керування, який подається на середній вхід, менше, чим величина порогового значення **Threshold**, тоді на вихід блоку проходить сигнал з першого (верхнього) входу. Якщо сигнал керування перевищує порогове значення, тоді на вихід блоку буде поступати сигнал з другого (нижнього) входу.

На рис. 1. 52 показаний приклад роботи блоку **Switch**. В тому випадку, коли сигнал на керуючому вході ключа дорівнює **1**, на вихід блоку проходить гармонічний сигнал, якщо ж керуючий сигнал дорівнює нулю, тоді на вихід проходить сигнал нульового рівня від блоку **Ground**. Порогове значення керуючого сигналу задане рівним **0,5**.

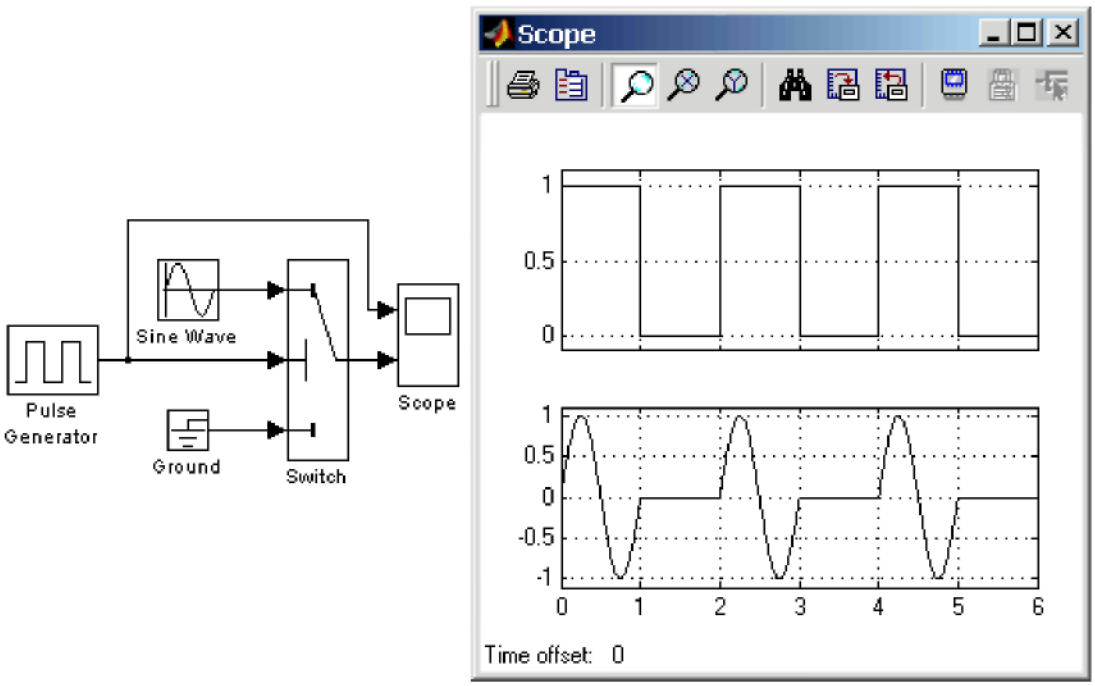

Рис. 1. 52. Застосування перемикача Switch

1.5.5.2 Блок багатовходового перемикача Multiport Switch

Призначення:

Виконує перемикання вхідних сигналів по сигналу керування, що задає номер активного вхідного порту.

Параметри:

Number of inputs –Кількість входів.

Блок багатовходового перемикача **Multiport Switch** пропускає на вихід сигнал з того вхідного порту, номер якого дорівнює поточному значенню

керуючого сигналу. Якщо керуючий сигнал не являється сигналом цілого типу, тоді блок **Multiport Switch** проводить відкидання дробової частини числа, при цьому в командному вікні **Matlab** появляється попереджаюче повідомлення.

На рис. 1. 53 показаний приклад роботи блоку **Multiport Switch**. Керуючий сигнал перемикача має три рівні і формується за допомогою блоків **Constant**, **Step**, **Step1** і **Sum**. На вихід блоку **Multiport Switch,** в залежності від рівня вхідного сигналу**,** проходять гармонічні сигнали, які мають різні частоти.

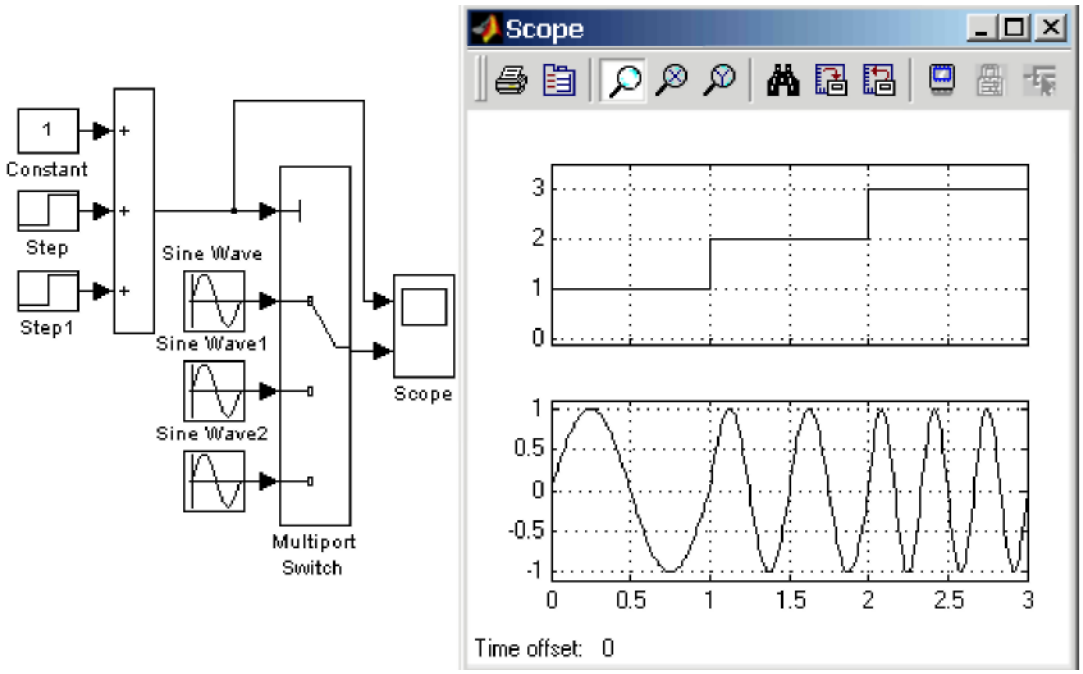

Рис. 1. 53. Застосування перемикача Multiport Switch.

Кількість входів блоку **Multiport Switch** можна задати рівним **1**. В цьому випадку на вхід блоку необхідно подати векторний сигнал, а сам блок буде пропускати на вихід той елемент вектора, номер якого співпаде з рівнем керуючого сигналу.

1.5.5.3 Блок ручного перемикача Manual Switch

Призначення:

Виконує перемикання вхідних сигналів по команді користувача.

Параметри:

Немає.

Командою на перемикання являється подвійне клацання лівою клавішою мишки на зображенні блоку. При цьому зображення блоку змінюється, показуючи, який вхідний сигнал в даний момент проходить на вихід блоку.

Перемикання блоку можна виконувати як до початку моделювання, так і в процесі розрахунку.

На рис. 1. 54 показаний приклад використання блоку **Manual Switch**.

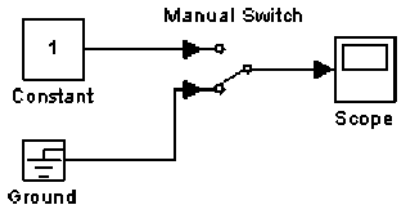

Рис. 1. 54. Приклад використання блоку Manual Switch.

1.5.6 Math - блоки математичних операцій

1.5.6.1 Блок обчислення модуля Abs

Призначення:

Виконує обчислення абсолютного значення величини сигналу.

Параметри:

Saturate on integer overflow (прапорець) - Подавлювати переповнення цілого. При установленому прапорці обмеження сигналів цілого типу виконується коректно.

Приклад використання блоку **Abs**, який обчислює модуль поточного значення синусоїдного сигналу, показаний на рис. 1. 55.

Блок **Abs** може використовуватися також для обчислення модуля сигналу комплексного типу. На рис. 1.56 показаний приклад обчислення модуля комплексного сигналу виду:

$$u = cos(w\,t) + i\,sin\,(w\,t)$$

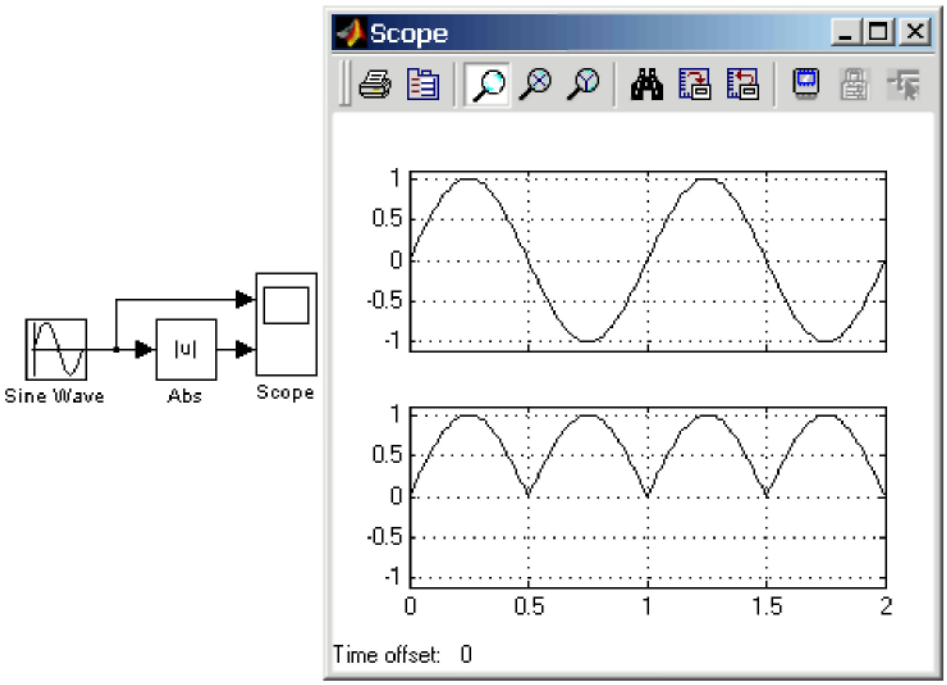

Рис. 1. 55. Приклад використання блоку Abs

Модуль цього сигналу, як і можна було чекати, дорівнює **1** для будь-якого моменту часу.

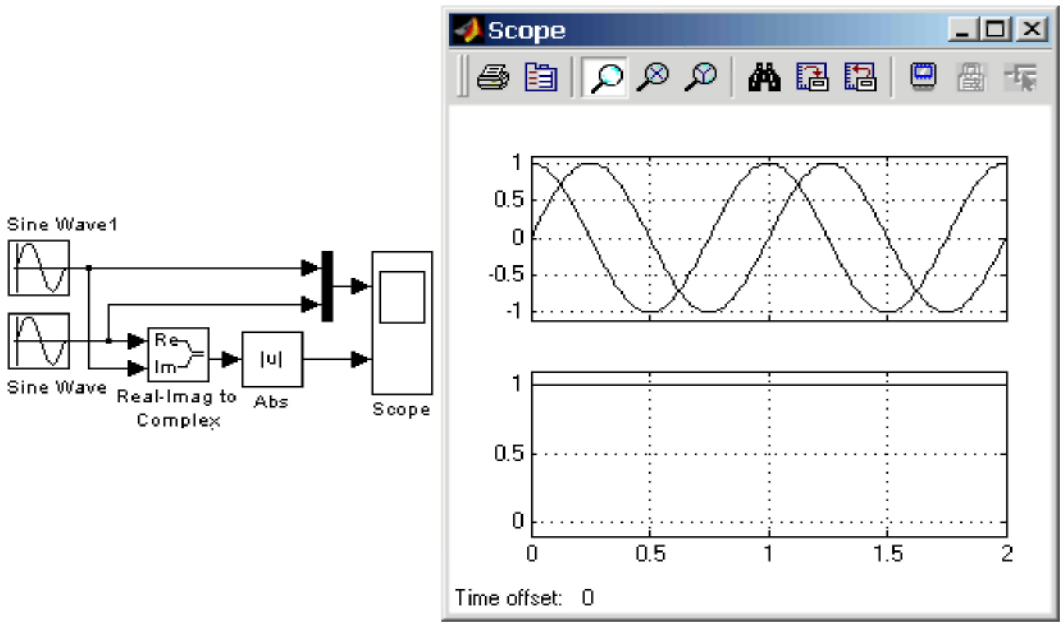

Рис. 1. 56. Приклад використання блоку Abs для обчислення модуля комплексного сигналу

1.5.6.2 Блок обчислення суми Sum

Призначення:

Виконує обчислення суми поточних значень сигналів. *Параметри:*

1. Icon shape – Форма блоку. Вибирається із списку.

• **round –**коло,
• **rectangular –** прямокутник.
2. List of sign – Список знаків. В списку можна використовувати наступні знаки: **+ (**плюс), - (мінус) і **|** (роздільник знаків).
3. Saturate on integer overflow (прапорець) – Подавляти переповнення цілого. При установленому прапорці обмеження сигналів цілого типу виконується коректно.

Кількість входів і операція (додавання або віднімання) визначаються списком знаків параметру **List of sign**, при цьому мітки входів позначаються відповідними знаками. В параметрі **List of sign** можна також указати число входів блоку. В цьому випадку всі входи будуть підсумовуючими.

Якщо кількість входів блоку перевищує три, тоді зручніше використовувати блок **Sum** прямокутної форми.

Блок може використовуватися для підсумовування скалярних, векторних або матричних сигналів. Типи сигналів, які підсумовуються, повинні співпадати. Не можна, наприклад, подати на один і той же підсумовуючий блок сигнали цілого і дійсного типів.

Якщо кількість входів блоку більше, чим один, тоді блок виконує поелементні операції над векторними і матричними сигналами. При цьому кількість елементів в матриці або векторі повинна бути однаковою.

Якщо в якості списку знаків указати цифру **1** (один вхід), тоді блок можна використовувати для визначення суми вектора.

Приклади використання блоку **Sum** показані на рис. 1. 57.

1.5.6.3 Блок перемноження Product

Призначення:

Виконує обчислення добутку поточних значень сигналів.

Параметри:

1. Number of inputs – Кількість входів. Може задаватися як число або як список знаків. В списку знаків можна використати знаки «×» (перемножити) і «/» (розділити).

2. Multiplication – Спосіб виконання операції. Може приймати значення (із списку):

• **Element-wise –** Поелементний.
• **Matrix –** Матричний.

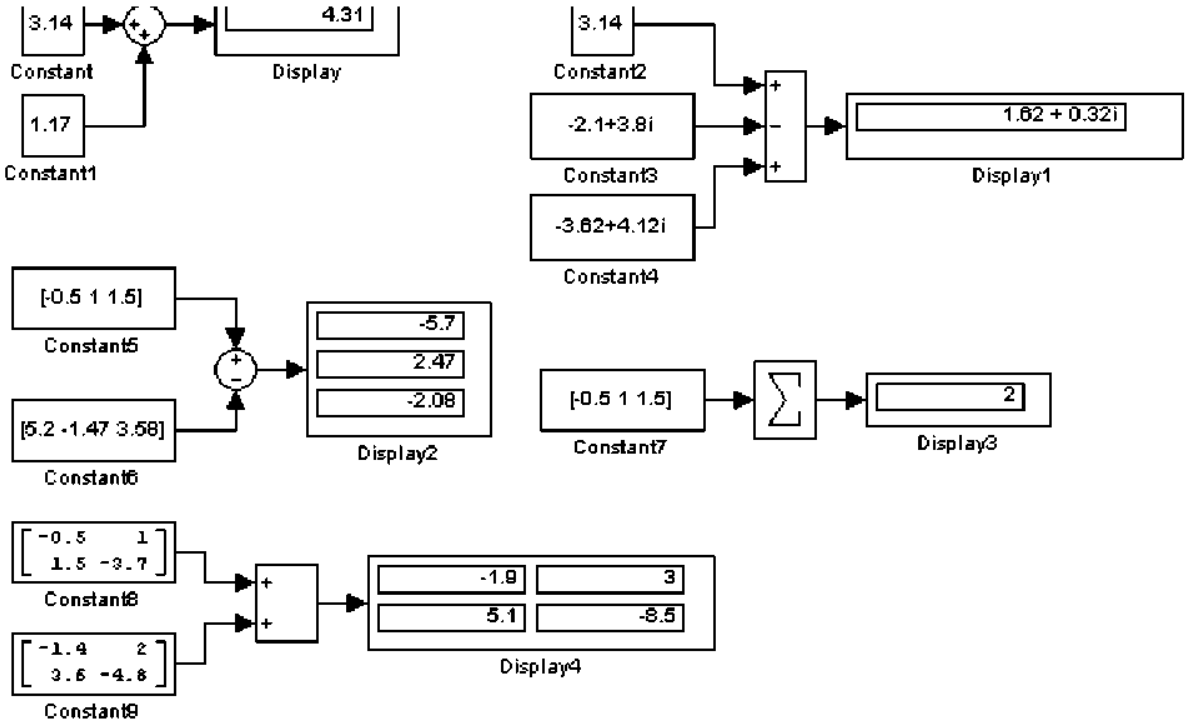

Рис. 1. 57. Приклади використання блоку Sum

3. Saturate on integer overflow (прапорець) – Подавлювати переповнення цілого. При установленому прапорці обмеження сигналів цілого типу виконується коректно.

Якщо параметр **Number of inputs** заданий списком, що включає, крім знаків множення, також знаки ділення, тоді мітки входів будуть позначені символами відповідних операцій.

Блок може використовуватися для операцій множення або ділення скалярних, векторних або матричних сигналів. Типи вхідних сигналів блоку повинні співпадати. Якщо в якості кількості входів указати цифру **1** (один вхід), тоді блок можна використовувати для визначення добутку елементів вектора.

Приклади використання блоку **Product** при виконанні скалярних і поелементних операцій показані на рис. 1. 58.

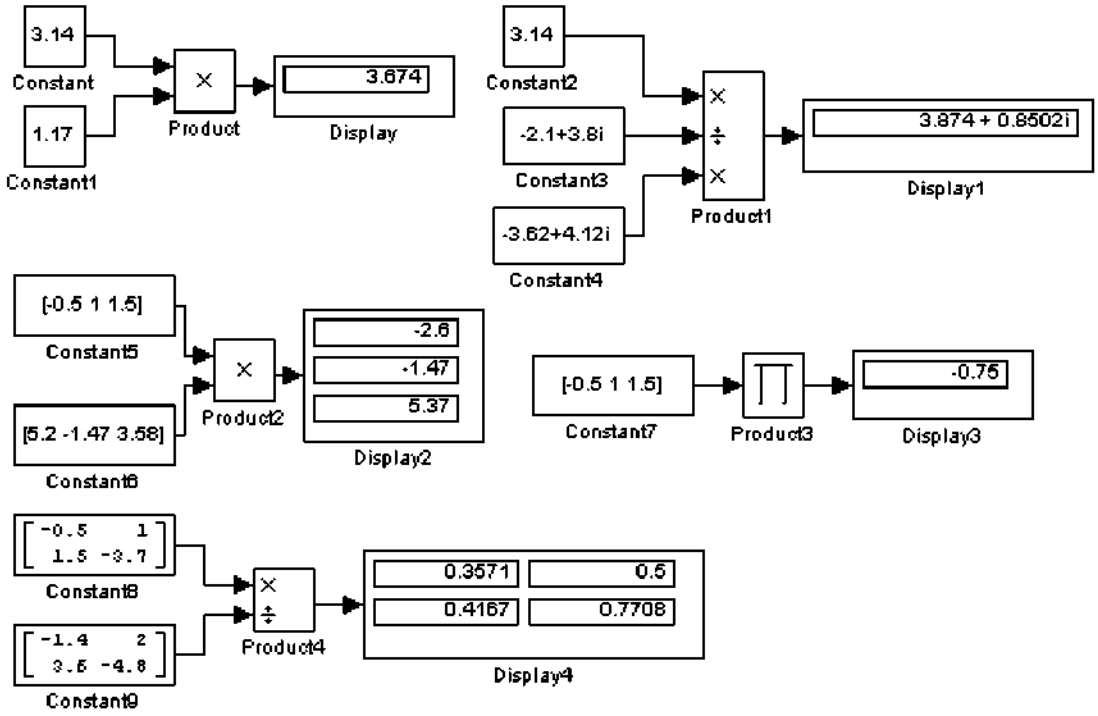

Рис. 1. 58. Приклади використання блоку Product при виконанні скалярних і поелементних операцій

При виконанні матричних операцій необхідно додержуватися правил їх виконання. Наприклад, при перемноженні двох матриць необхідно, щоб кількість рядків першої матриці дорівнювала кількості стовбців другої матриці. Приклади використання блоку **Product** при виконанні матричних операцій показані на рис. 1.59. В прикладі показані операції формування оберненої матриці, ділення матриць, а також перемноження матриць.

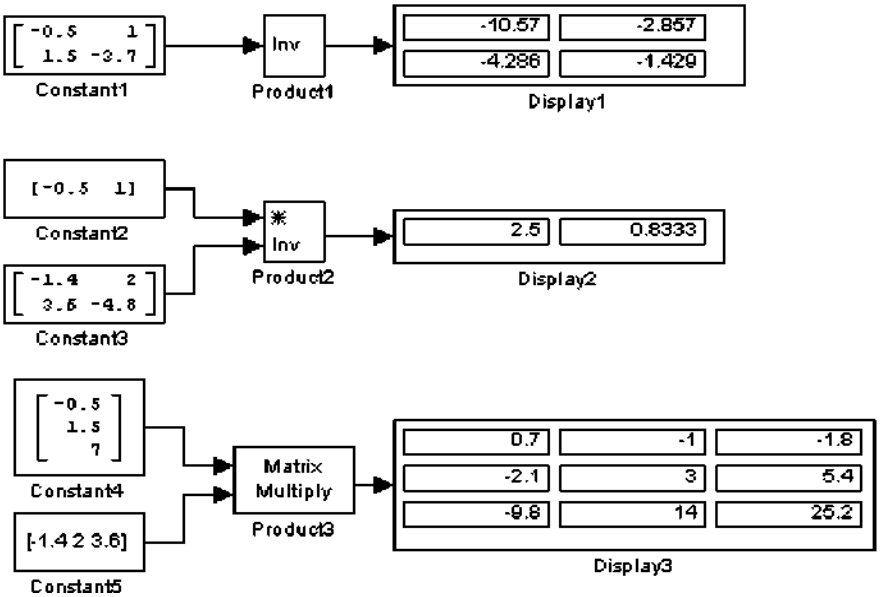

Рис. 1. 59. Приклади використання блоку Product при виконанні матричних операцій

1.5.6.4 Блок визначення мінімального або максимального значення MinMax

Призначення:

Визначає максимальне або мінімальне значення із всіх сигналів, які поступають на його входи.

Параметри:

1. Function – Вихідний параметр. Вибирається із списку:

- **min** – Мінімальне значення.
- **max** – Максимальне значення.

2. Number of input ports –Кількість вхідних портів.

Вхідні сигнали блоку можуть бути скалярними або векторними. Блок визначає максимальне або мінімальне значення із всіх скалярних сигналів, які поступають на його входи. Якщо вхідні сигнали являються векторними, тоді блок виконує поелементну операцію пошуку мінімального або максимального значення. В цьому випадку розмірності векторів повинні співпадати. Якщо кількість вхідних портів блоку задана рівною **1**, тоді блок може використовуватися для знаходження мінімального або максимального значення у вхідному векторі.

Приклади використання блоку **MinMax** показані на рис. 1.60.

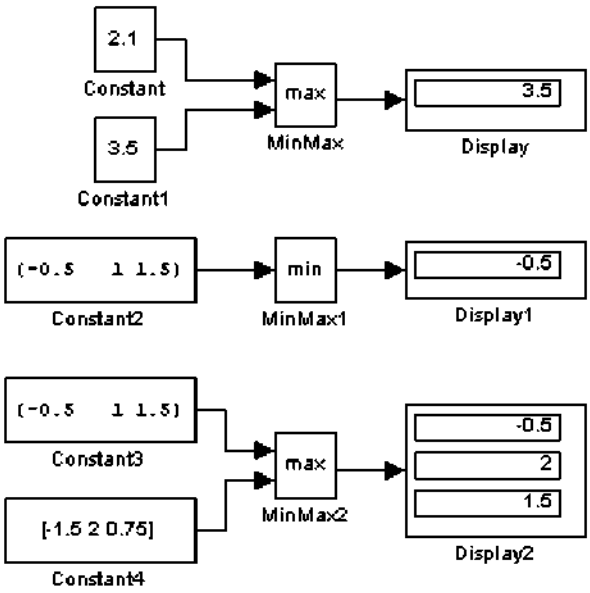

Рис. 1. 60. Приклади використання блоку MinMax

1.5.6.5 Блок заокруглення числового значення Rounding Function

Призначення:

Виконує операцію заокруглення числового значення.

Параметри:

Function – Спосіб заокруглення (вибирається із списку):

- **floor** –Заокруглення до найближчого меншого цілого.
- **ceil** – Заокруглення до найближчого більшого цілого.
- **round** – Заокруглення до найближчого цілого.
- **fix** – Заокруглення відкиданням дробової частини.

Вхідні сигнали блоку можуть бути скалярними, векторними або матричними дійсного і комплексного типу. При векторному або матричному вхідному сигналі блок виконує поелементні операції.

Вихідний сигнал блоку буде мати тип **double** або **single**.

Приклади використання блоку **Rounding Function** показані на рис. 1.61.

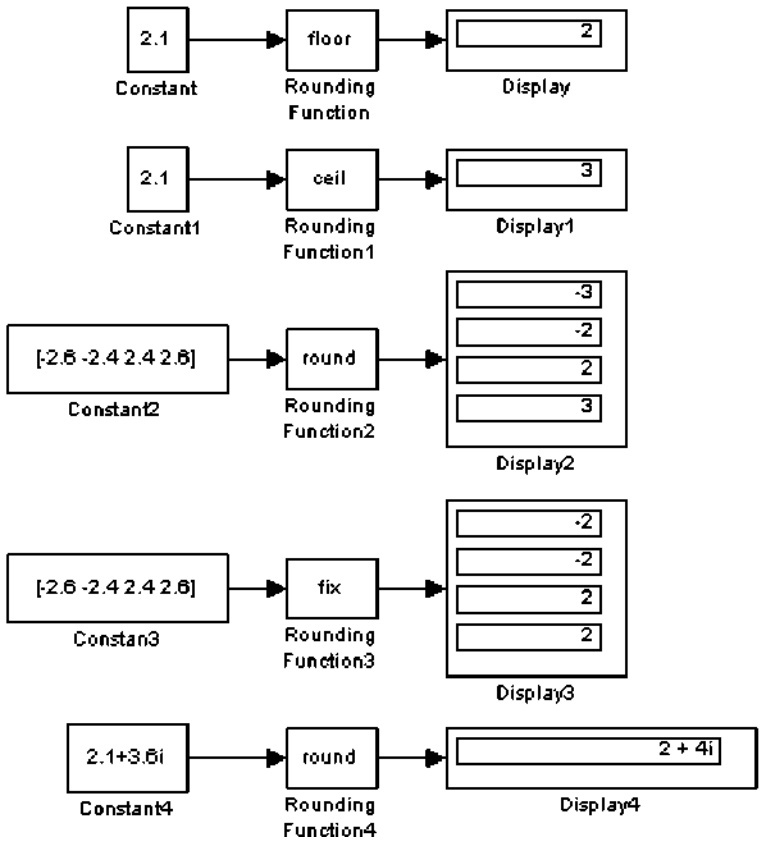

Рис. 1. 61. Приклади використання блоку Rounding Function

1.5.6.6 Блок логічних операцій Logical Operation

Призначення:

Реалізує одну із базових логічних операцій.

Параметри:

1. **Operator –** Вид реалізуємої логічної операції (вибирається із списку):

- **AND** – Логічне множення (операція **І**).
- **OR** – Логічне додавання (операція **АБО**).
- **NAND** - Операція **І - НІ**.
- **NOR** - Операция **АБО – НІ**
- **XOR** – Виключне **АБО** (операція додавання по модулю **2**).
- **NOT** - Логічне заперечення (**НІ**).

2. **Number of input ports –** Кількість вхідних портів.

Вихідним сигналом блоку являється **1**, якщо результат обчислення логічної операції є **“ІСТИНА”,** і **0,** якщо результат - **“ХИБНІСТЬ ”**.

Вхідні сигнали блоку можуть бути скалярними, векторними або матричними. Якщо вхідні сигнали - вектори або матриці, тоді блок виконує поелементну логічну операцію, при цьому розмірність вхідних сигналів повинна співпадати. Якщо частина вхідних сигналів - вектори або матриці, а інша частина вхідних сигналів - скаляри, тоді блок виконує логічну операцію для скалярних вхідних сигналів і кожного елементу векторних або матричних сигналів. Розмірність вихідного сигналу в цьому випадку буде визначатися розмірністю векторних або матричних вхідних сигналів.

При виконанні логічної операції заперечення блок буде мати лише один вхідний порт.

Вхідні сигнали можуть бути як дійсного, так і логічного типу (**boolean**).

Приклади використання блоку **Logical Operation** показані на рис. 1. 62.

1.5.6.7 Блок комбінаторної логіки Gombinatorical Logic

Призначення:

Перетворює вхідні сигнали у відповідності з таблицею істинності.

Параметри:

Truth table - Таблиця істинності.

Блок **Combinatorical Logic** забезпечує перетворення вхідного сигналу у відповідності з правилами, які визначаються таблицею істинності. Таблиця істинності представляє собою список можливих вихідних значень блоку. Такий опис роботи пристроїв прийнятий в теорії кінцевих автоматів. Число рядків в таблиці істинності визначається співвідношенням:

number of rows = *2^* (number of inputs)

де

number of inputs - число вхідних сигналів,

number of rows - число рядків таблиці істинності.

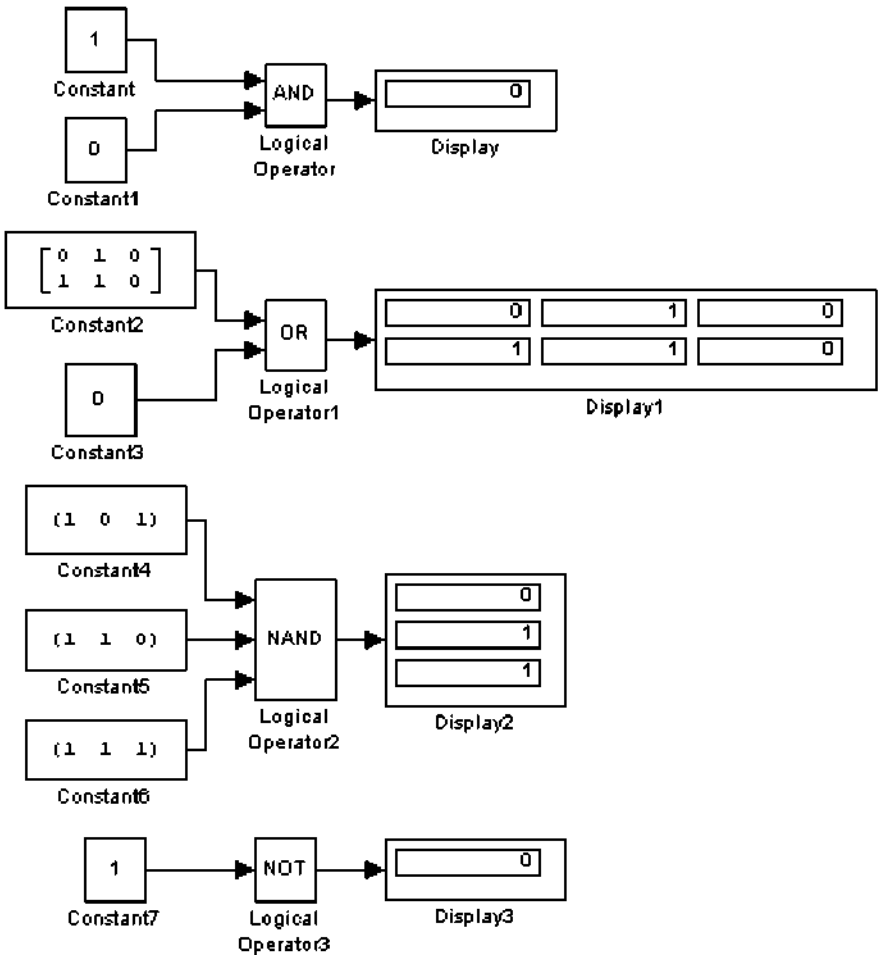

Рис. 1. 62. Приклади використання блоку Logical Operation

Вхідні сигнали при складанні таблиці істинності вважаються заданими. Вони визначають індекс (номер) рядка, в який записуються вихідні значення блоку. Індекс кожного рядка визначається виразом:

row index = 1 + u(m)*2° + u(m-1j * 2^1 + ... + u(1)*2^{m-1}

де

row index – індекс рядка,

m – кількість вхідних сигналів (елементів у вхідному векторі),

u(1) - перший вхідний сигнал (перший елемент вхідного вектору),

u(m) – останній вхідний сигнал (останній елемент вхідного вектору).

u(m) – останній вхідний сигнал (останній елемент вхідного вектору).

Наприклад, у випадку операції логічного **I** (**AND**) для двох операндів вираз, що визначає індекс рядка, буде виглядати наступним образом:

row index = 1 + u(2) * 2° +u(1)* 2^1

Нижче приведений приклад формування таблиці істинності операції логічного **I** (**AND**) для двох операндів (табл. 1.1):

Таблиця 1.1

Вход 2	Вход 1	Выражение для индекса строки	Значение индекса строки	Таблица истинности (Выход)
0	0	$1+0\cdot 2^0+0\cdot 2^1$	1	0
1	0	$1+1\cdot 2^0+0\cdot 2^1$	2	0
0	1	$1+0\cdot 2^0+1\cdot 2^1$	3	0
1	1	$1+1\cdot 2^0+1\cdot 2^1$	4	1

На рис. 1.63 показаний приклад реалізації операції логічного **I** за допомогою блоку **Combinatorical Logic**. Параметр блоку **Truth table** заданий виразом **[0;0;0;1]**.

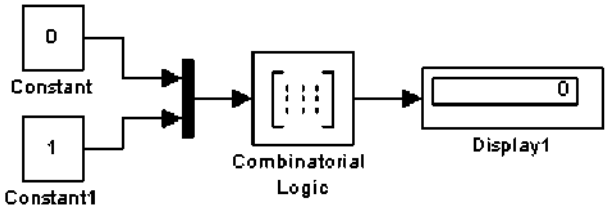

Рис. 1. 63. Приклад використання блоку Combinatorical Logic

1.5.7 Signal&Systems - блоки перетворення сигналів і допоміжні блоки

1.5.7.1 Мультиплексор (змішувач) Мих

Призначення:

Об’єднує вхідні сигнали у вектор.

Параметри:

1. **Number of Inputs** –Кількість входів.
2. **Display option** – Спосіб відображення. Вибирається із списку:

• **bar** – Вертикальний вузький прямокутник чорного кольору.
• **signals** – Прямокутник з білим фоном і відображенням міток вхідних сигналів.
• **none** – Прямокутник з білим фоном без відображення міток вхідних сигналів.

Вхідні сигнали блоку можуть бути скалярними і (або) векторними.

Якщо серед вхідних сигналів є вектори, тоді кількість входів можна задавати як вектор указуванням числа елементів кожного вектора. Наприклад, вираз **[2 3 1]** визначає три вхідних сигнали, перший сигнал – вектор із двох елементів, другий сигнал – вектор із трьох елементів, і останній сигнал – скаляр. В тому випадку, якщо розмірність вхідного вектора не співпадає з указаною в параметрі **Number of Inputs**, після початку розрахунку **Simulink** видасть повідомлення про помилку. Розмірність вхідного вектора можна задавати як **-1** (мінус одиниця). В цьому випадку розмірність вхідного вектора може бути будь-якою.

Параметр **Number of Inputs** можна задавати також у вигляді списку міток сигналів, наприклад: **Vector1, Vector2, Scalar.** В цьому випадку мітки сигналів будуть відображатися рядом з відповідними з’єднуючими лініями.

Сигнали, що подаються на входи блоку, повинні бути одного типу (дійсного або комплексного).

Приклади використання блоку **M** показані на рис. 1.64.

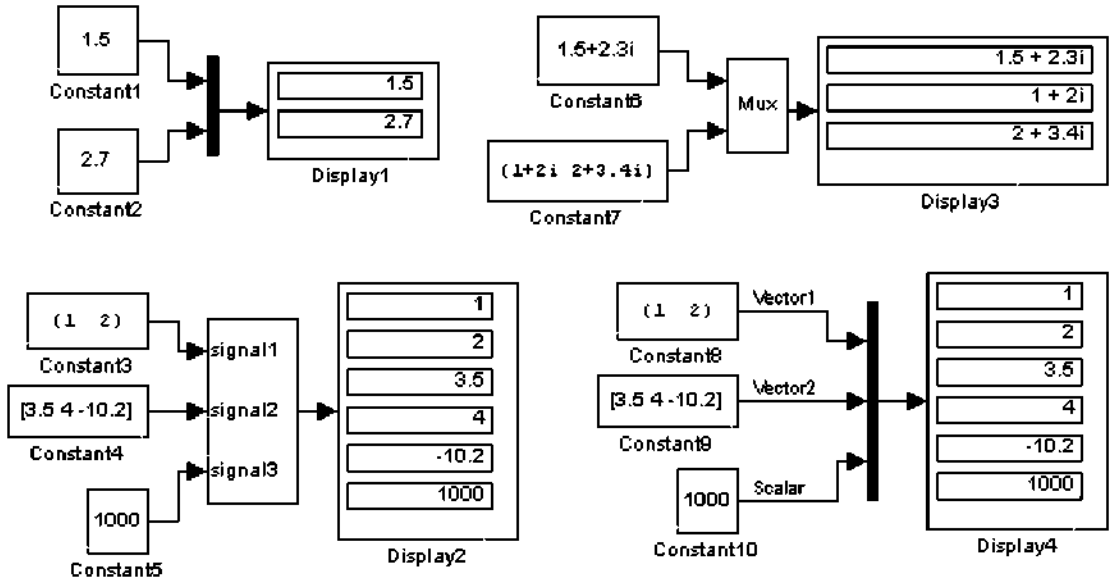

Рис. 1. 64. Приклади використання блоку Mux

1.5.7.2 Демультиплексор (розподільник) Demux

Призначення:

Розподіляє вхідний векторний сигнал на окремі складові.

Параметри:

1. **Number of Outputs** –Кількість виходів.
2. **Bus Selection Mode** (прапорець) – Режим розподілення векторних сигналів. Вхідним сигналом в звичайному режимі являється вектор, що сформований будь-яким способом. Вихідними сигналами являються скаляри або вектори, кількість яких і розмірність визначається параметром **Number of Outputs** і розмірністю вхідного вектора.

Якщо кількість виходів **P** (значення параметру **Number of Outputs**) дорівнює розмірності вхідного сигналу **N**, тоді блок виконує розподілення вхідного вектора на окремі елементи.

Якщо кількість виходів **P** менше, чим розмірність вхідного сигналу **N**, тоді розмірність перших **P-1** вихідних сигналів дорівнює відношенню **N/P**, заокругленому до найближчого більшого числа, а розмірність останнього вихідного сигналу дорівнює різниці між розмірністю вхідного сигналу і сумою розмірностей перших **P-1** виходів. Наприклад, якщо розмірність вхідного сигналу дорівнює восьми, а кількість виходів дорівнює трьом, тоді перші два вихідних вектори будуть мати розмірність **ceil(8/3) = 3**, а останній вихідний вектор буде мати розмірність **8-(3+3) = 2**.

Параметр **Number of Outputs** може бути заданий також за допомогою вектора, що визначає розмірність кожного вихідного сигналу. Наприклад, вираз **[2 3 1]** визначає три вихідних сигнали, перший сигнал – вектор із двох

елементів, другий сигнал - вектор із трьох елементів, і останній сигнал – скаляр. Розмірність можна також задавати як **-1** (мінус одиниця). В цьому випадку розмірність відповідного вихідного сигналу визначається як різниця між розмірністю вхідного вектора і сумою розмірностей заданих вихідних сигналів. Наприклад, якщо розмірність вхідного вектора дорівнює шести, а параметр **Number of Outputs** заданий вираом [**1 -1 3]**, тоді другий вихідний сигнал буде мати розмірність **6 - (3 +1)=2.**

Приклади використання блоку **Demux** показані на рис. 1.65.

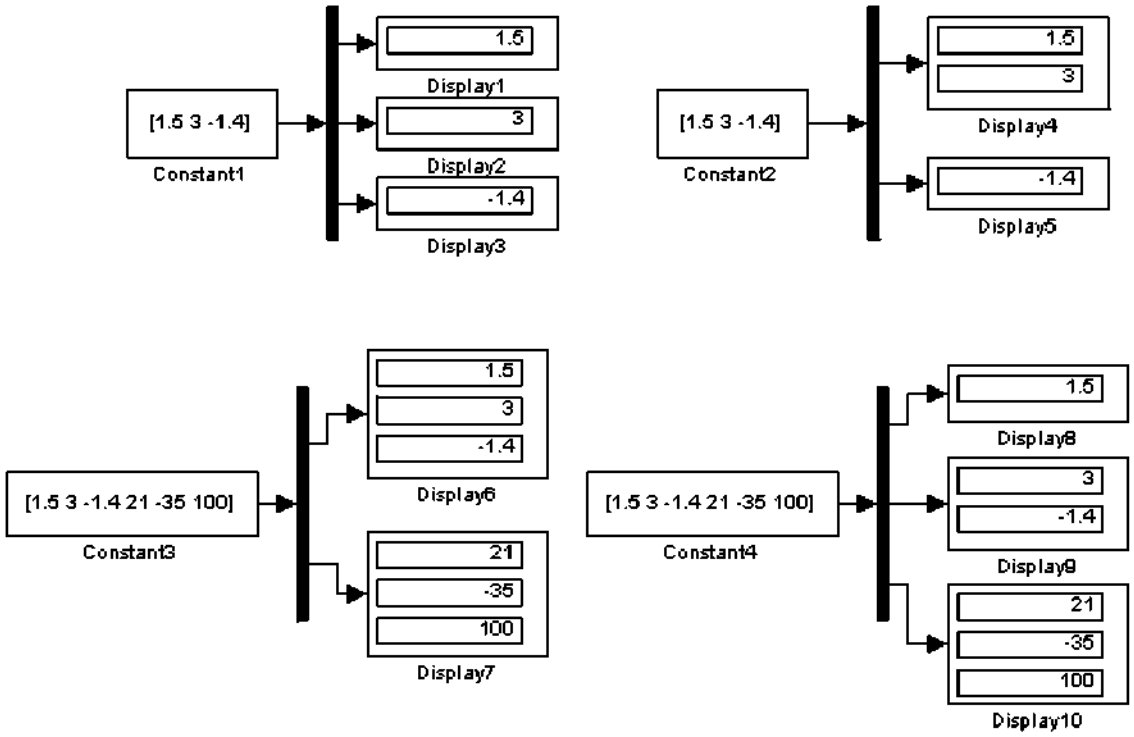

Рис. 1. 65. Приклади використання блоку Demux

В режимі **Bus Selection Mode** блок **Demux** функціонує не з окремими елементами векторів, а із векторними сигналами в цілому. Вхідний сигнал в цьому режимі повинен бути сформований блоком **Mux** або іншим блоком **Demux**. Параметр **Number of Outputs** в цьому випадку задається в виді скаляру, який визначає кількість вихідних сигналів, або у виді вектора, кожний елемент якого визначає кількість векторних сигналів в даному вихідному сигналі. Наприклад, при вхідному сигналі, що складається із трьох векторів, параметр **Number of Outputs**, який заданий вектором **[2 1]**, визначить два вихідних сигнали, перший із яких буде вміщувати два векторних сигнали, а другий – один.

Приклади використання блоку **Demux** в режимі **Bus Selection Mode** показані на рис. 1. 66.

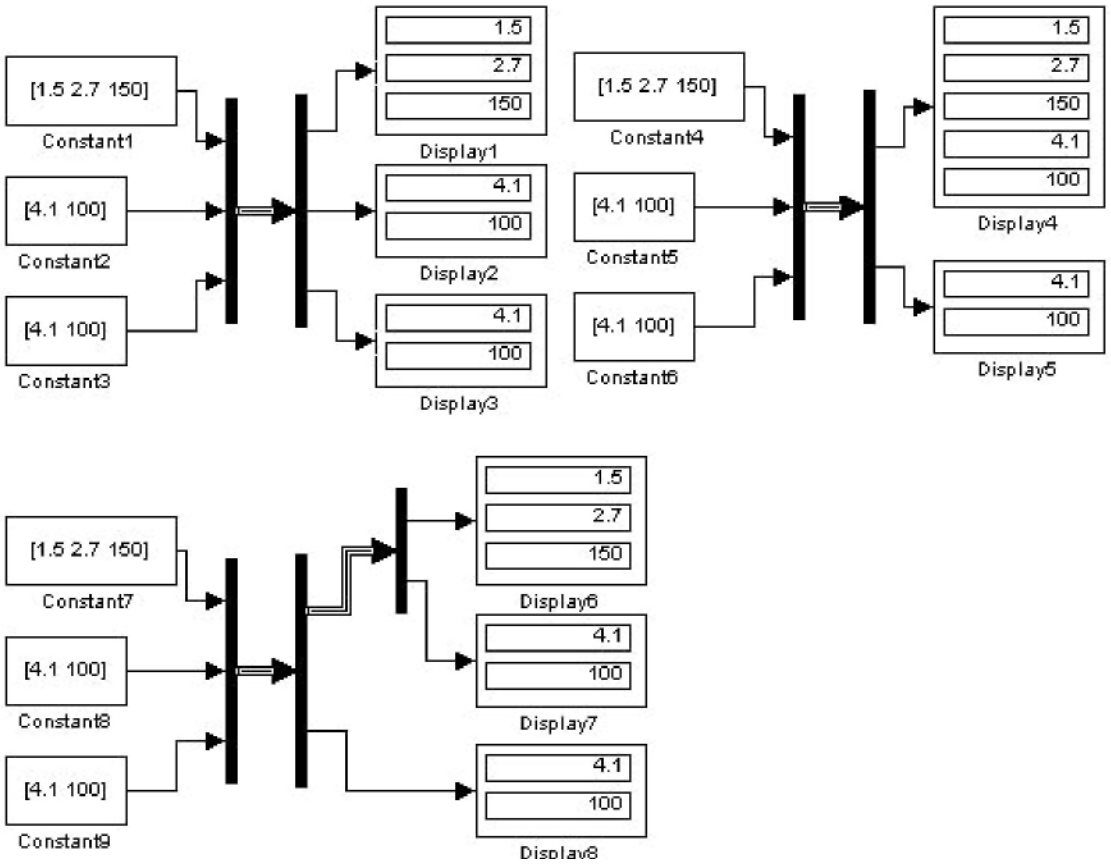

Рис. 1. 66. Приклади використання блоку Demux в режимі Bus Selection Mode

1.5.7.3 Блок створення загальної області пам'яті Data Store Memory

Призначення:

Блок створює поіменовану область пам'яті для зберігання даних.

Параметри:

1. **Data store name** – Ім'я області пам'яті.
2. **Initial value** –Початкове значення.
3. **Interpret vector parameters as 1-D** (прапорець) – Інтерпретувати вектор параметрів даних як одновимірний вектор.

Блок використовується сумісно з блоками **Data Store Write** (запис даних) і **Data Store Read** (зчитування даних).

Параметр **Initial value** задає не тільки початкове значення сигналу, але і його розмірність. Наприклад, якщо початкове значення сигналу задане матрицею **[0 1; 2 3]**, тоді сигнал, що зберігається, повинен бути матрицею **2×2**.

Якщо блок **Data Store Memory** розташований в моделі верхнього рівня, тоді задану ним область пам’яті можна використовувати як в самій моделі, так і у всіх підсистемах нижнього рівня ієрархії. Якщо блок **Data Store Memory** розташований в підсистемі, тоді задану ним область пам’яті можна використовувати в даній підсистемі і у всіх підсистемах нижнього рівня ієрархії.

Блок функціонує з дійсними сигналами типу **double**.

Приклад використання блоку **Data Store Memory** сумісно **з** блоками **Data Store Write** і **Data Store Read** показаний на рис. 1.67.

1.5.7.4 Блок запису даних в загальну область пам’яті Data Store Write

Призначення:

Блок записує дані в поіменовану область пам’яті.

Параметри:

1. **Data store name** – Ім’я області пам’яті.
2. **Sample time –**Крок модельного часу.

Операція запису виконується для значення сигналу, який отриманий на попередньому кроці розрахунку.

В моделі можуть використовуватися декілька блоків **Data Store Write**, які виконують запис в одну область пам’яті. Проте, якщо запис відбувається на одному і тому кроці розрахунку, тоді результат буде непередбачуваним.

Приклад використання блоку **Data Store Write** сумісно з блоками **Data Store Memory** і **Data Store Read** показаний на рис. 1.67.

1.5.7.5 Блок зчитування даних із загальної області пам’яті Data Store Read

Призначення:

Блок зчитує дані із поіменованої області пам’яті.

Параметри:

1. **Data store name** – Ім’я області пам’яті.
2. **Sample time –**Крок модельного часу.

Операція зчитування виконується на кожному кроці розрахунку.

В моделі можуть використовуватися декілька блоків **Data Store Read**, які виконують зчитування даних із одної і тої ж самої області пам'яті. Приклад використання блоку **Data Store Read** сумісно з блоками **Data Store Memory** і **Data Store Write** показаний на рис. 5.7.4. В прикладі використовується тригерна підсистема, яка виконує обчислення по передньому фронту керуючого сигналу. Таким чином, запис значень в загальну область пам'яті відбувається тільки в моменти змінювання керуючого сигналу в додатньому напрямку. В інші моменти часу значення даних в області пам'яті не змінюється.

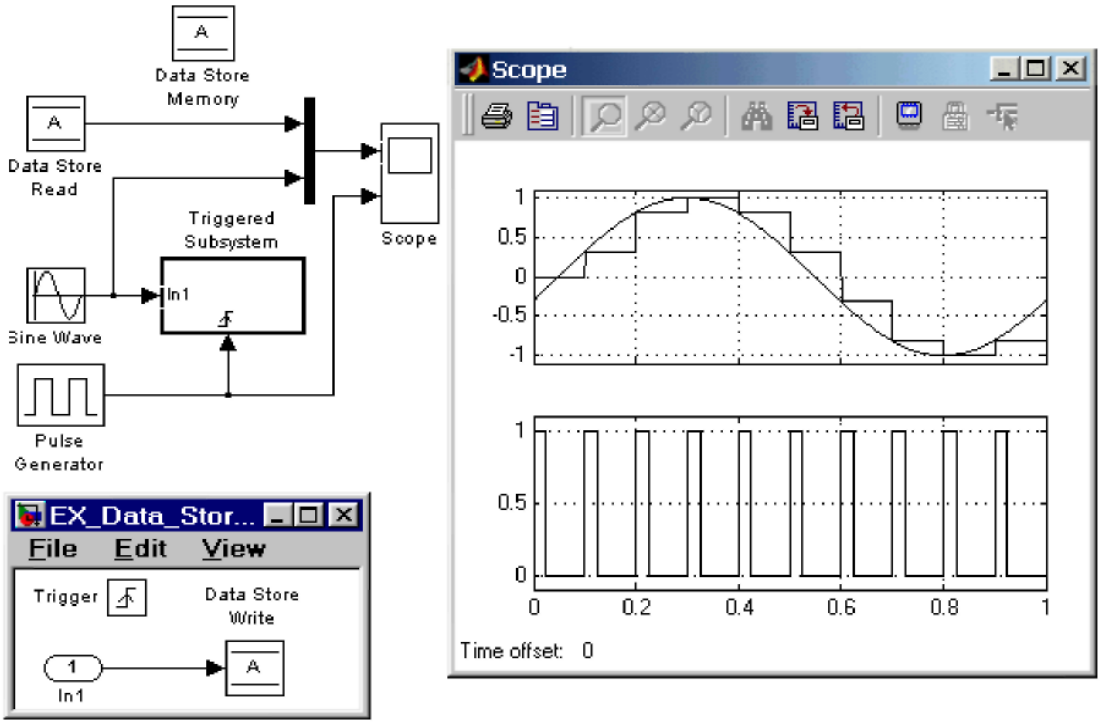

Рис. 1. 67. Використання блоків Data Store Memory, Data Store Write і Data Store Read.

1.5.8 Subsystem – підсистеми.

Підсистема – це фрагмент **Simulink**-моделі, який оформлений у вигляді окремого блоку. Використання підсистем при упорядкуванні моделі має наступні позитивні сторони:

1. Зменшує кількість одночасно відображаємих блоків на екрані,що полегшує сприйняття моделі (в ідеалі модель повністю повинна відображатися на екрані монітора).

2. Дозволяє створювати і налагоджувати фрагменти моделі окремо один від одного, що підвищує технологічність створення моделі.
3. Дозволяє створювати власні бібліотеки.

4. Дає можливість синхронізації паралельно функціонуючих підсистем.
5. Дозволяє включати в модель власні довідкові засоби.
6. Дає можливість зв'язувати підсистему з яким-небудь **m**-файлом, забезпечуючи запуск цього файла при відкритті підсистеми (нестандартне відкриття підсистеми).

Використання підсистем і механізму їх блоків дозволяє створювати блоки, які не уступають стандартнмм за своїм оформленням (власне вікно параметрів блоку, піктограма, довідка і т.п.).

Кількість підсистем в моделі необмежена, крім того, підсистеми можуть включати в себе інші підсистеми. Рівень вкладеності підсистем друг в друга також необмежений.

Зв'язок підсистеми з моделлю (або підсистемою верхнього рівня ієрархії) виконується за допомогою вхідних (блок **Inport** бібліотеки **Sources**) і вихідних (блок **Outport** бібліотеки **Sinks**) портів. Додавання в підсистему вхідного або вихідного порту приводить до появи на зображенні підсистеми мітки порту, за допомогою якої зовнішні сигнали передаються в середину підсистеми або виводяться в основну модель. Перейменування блоків **Inport** або **Outport** **до**зволяє змінювати відображаємі на піктограмі підсистеми зі стандартних (**In** і **Out**) на ті, які необхідні користувачу.

Підсистеми можуть бути віртуальними (**Subsystem**) і монолітними (**Atomic Subsystem**). Відмінність цих видів підсистем полягає в порядку виконання блоків під час розрахунку. Якщо підсистема являється віртуальною, тоді **Simulink** ігнорує наявність границь, які відокремлюють таку підсистему від моделі при визначенні порядку розрахунку блоків. Іншими словами, у віртуальній системі спочатку можуть бути розраховані вихідні сигнали декількох блоків, потім виконаний розрахунок блоків в основній моделі, а потім знову виконаний розрахунок блоків, які входять в підсистему. Монолітна підсистема вважається єдиним (неподільним) блоком і **Simulink** виконує розрахунок всіх блоків в такій підсистемі, не перемикаючись на розрахунки інших блоків в основній моделі. Зображення монолітної підсистеми має більш товсту рамку в порівнянні з віртуальною підсистемою.

Підсистеми можуть бути також керованими або некерованими. Керовані підсистеми завжди являються монолітними. Керовані підсистеми мають додаткові (керуючі) входи, на які поступають сигнали, що активізують дану підсистему. Керовані входи розташовані зверху або знизу підсистеми. Коли керована підсистема активізована – вона виконує обчислення. В тому випадку, коли керована підсистема пасивна, вона не виконує обчислення, а значення сигналів на її виходах визначаються настроюваннями вихідних портів (рис.1.68).

Для створення в моделі підсистеми можна скористатися двома способами:

1. Скопіювати необхідну підсистему із бібліотеки **Subsystem** в модель.
2. Виділити за допомогою мишки необхідний фрагмент моделі і виконати команду **Create Subsystem** із меню **Edit** вікна моделі. Виділений фрагмент буде поміщений в підсистему, а входи і виходи підсистеми будуть забезпечені відповідними портами. Даний спосіб дозволяє створювати віртуальну некеровану підсистему. В подальшому, якщо це необхідно, можна зробити підсистему монолітною, змінивши її параметри, або керованою, добавивши керуючий елемент із потрібної підсистеми, яка знаходиться в бібліотеці. Відмінити групування блоків в підсистему можна командою **Undo**.

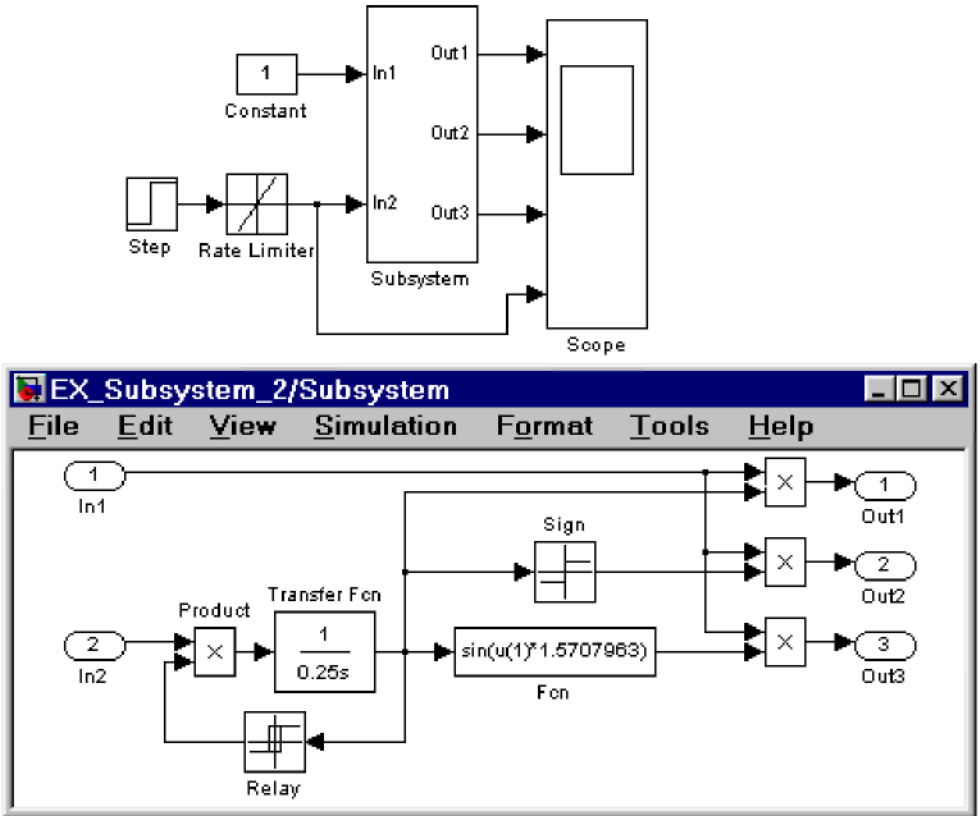

Рис. 1. 68. Модель, яка використовує підсистему

1.5.9 Маскування підсистем

1.5.9.1 Загальні відомості

Механізм маскування підсистем дозволяє оформити підсистему як повноцінний бібліотечний блок, тобто забезпечити підсистему власним вікном параметрів, піктограмою, довідковою системою і т. п.

Маскування підсистем дає користувачу наступні переваги:

1. Розширяє можливості користувача по керуванню параметрами моделі.
2. Дозволяє створювати більш зрозумілий інтерфейс підсистеми.
3. Підвищує наочність блок-діаграми.
4. Розширяє можливості побудови складних моделей.
5. Підвищує захищеність моделі від несанкціонованої модифікації.

Для виконання маскування наявної підсистеми необхідно попередньо виконати наступні дії:

1. Визначити, які параметри підсистеми повинні задаватися користувачем в майбутньому вікні параметрів. Задати ці параметри в підсистемі за допомогою ідентифікаторів (імен).
2. Визначити, яким чином параметр повинен задаватися в вікні діалогу (за допомогою рядка введення, вибором із списку, що розкривається, або установленням прапорця).
3. Розробити ескіз піктограми блоку.
4. Створити коментарі (довідку) по використанню підсистеми.

Маскування підсистеми виконується за домогою **Mask Editor**

(редактор маски). Для запуску редактора маски необхідно виділити підсистему, яка буде маскуватися, і виконати команду **Mask Subsystem…** із меню **Edit**. Можна також скористатися контекстним меню. Після запуску **Mask Editor** на екран буде виведене вікно редактора (рис. 1.69), яке має три вкладки:

- **Icon** (Піктограма),
- **Initialization** (Ініціалізація)**,**
- **Documentation** (Документація).

Перша із вкладок забезпечує створення піктограмми підсистеми, друга – дає можливість створити вікно діалогу для введення параметрів і третя – дозволяє ввести опис блоку і створити його довідку.

В верхній частині всіх вкладок є поле **Mask Type**, за допомогою якого можна задати ім’я блоку. В нижній частині вікна є п’ять кнопок керування редактором:

1. **OK** – Зберегти внесені зміни і закрити вікно.
2. **Cancel** – Відмінити внесені зміни і закрити вікно.
3. **Unmask** – Зняти маску з підсистеми. До закриття файлу моделі маску можна поновити, скориставшись командою **Edit Mask…** із меню **Edit**.
4. **Help** – Відкрити вікно довідки редактора маски.
5. **Apply** –Зберегти внесені зміни без закриття вікна редактора. Повторний визов редактора маски для уже маскованої підсистеми

здійснюється командою **Edit Mask…** із меню **Edit** (або аналогічною командою із контекстного меню).

Після того, як маскування системи буде виконане, подвійне клацання на її зображенні буде відкривати вікно параметрів підсистеми, а не вікно моделі.

Відкрити саму підсистему (вікно моделі) для редактування або перегляду можна командою **Look under mask** із меню **Edit** або контекстного меню.

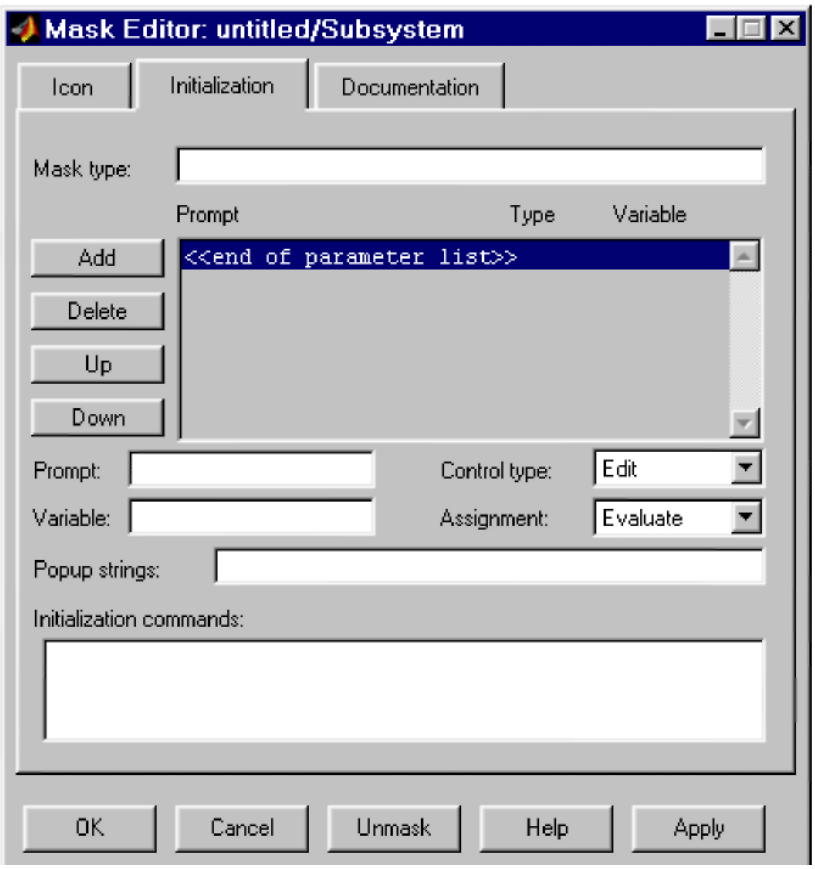

Рис. 1. 69. Вікно редактора маски Mask Editor

1.5.9.2 Створення вікна параметрів

Вікно параметрів створюється за допомогою вкладки **Initialization** (Ініціалізація) редактора маски. Для створення поля введення параметру з його описом необхідно виконати наступні дії:

1. Натиснути кнопку **Add** (Додати).

2. Ввести опис параметру в полі **Prompt** (Підказування). В якості опису параметра звичайно використовується його назва в вигляді тексту, наприклад, «**Gain**», «**Constant value**» і т. п.

3. Указати ідентифікатор параметру в полі **Variable** (Змінна). Природньо, що це повинен бути один із тих ідентифікаторів, який використовувався при задаванні параметрів блоків усередині підсистеми (хоча це і необов'язково, оскільки параметр може бути використаний і для модифікації самого вікна діалогу). Всі змінні, ідентифікатори яких задані на вкладці **Initialization**, розміщуються в **Mask Workspace** – локальну робочу область маски і являються

доступними тільки усередині підсистеми.

4. Вибрати тип елемента інтерфейсу, що задає параметр, із списку **Control Type**:

• **Edit** – Редактуєме поле введення.

• **Checkbox** – Прапорець.

• **Popup** – Список, що розкривається. В цьому випадку в графі **Popup Strings** (Елементи списку) необхідно ввести елементи списку, які розділені символом вертикальної риси. Наприклад, вираз **alpha|beta|gamma** задасть список із трьох елементів: **alpha**, **beta** і **gamma**.

5. Вибрати формат параметру із списку **Assiggment**:

• **Evaluate** –Обчислюваний. Вибирається, якщо параметр повинен мати числове значення. В дане поле можна буде ввести вираз у відповідності з правилами мови **MATLAB**. Формат **Evaluate** дозволяє також використовувати числову форму значення змінної в тому випадку, якщо тип елемента інтерфейсу вибраний в вигляді прапорця або списку, що розкривається. Так, наприклад, для списку, що розкривається, **alpha|beta|gamma** значення зв'язаної зі списком змінної буде дорівнювати одиниці, якщо в списку вибране **alpha**, двом, якщо в списку вибране **beta**, і трьом, якщо в списку вибране **gamma**. Для елемента інтерфейсу **Checkbox** обчислювані значення будуть дорівнювати **1** (при установленому прапорці) і **0** (при знятому прапорці).

• **Literal** – Текстовий. Вибирається, якщо параметр повинен бути рядком символів.

6. Ввести команди ініціалізації в графі **Initialization commands**. Команди ініціалізації представляють собою звичайні команди на мові **MATLAB** і можуть включати оператори і **m**-функції. Такі команди задають змінні, які будуть знаходитися в робочій області маскованої підсистеми. Ці змінні доступні усередині підсистеми і можуть бути використані в якості параметрів блоків, що входять в склад підсистеми, а також для створення піктограми підсистеми. Команди ініціалізації виконуються в наступних випадках:

- При відкритті вікна моделі.
- При запуску моделі на виконання.
- При виконанні команди **Edit/Update diagram**.
- При обертанні блоку маскованої підсистеми (в цьому випадку команди ініціалізації забезпечують перерисування піктограми).
- При автоматичному змінюванні піктограми, що залежить від параметрів блоку.

В якості прикладу маскованої підсистеми розглянемо функціональний генератор. Схема моделі генератора показана на рис. 1. 70.

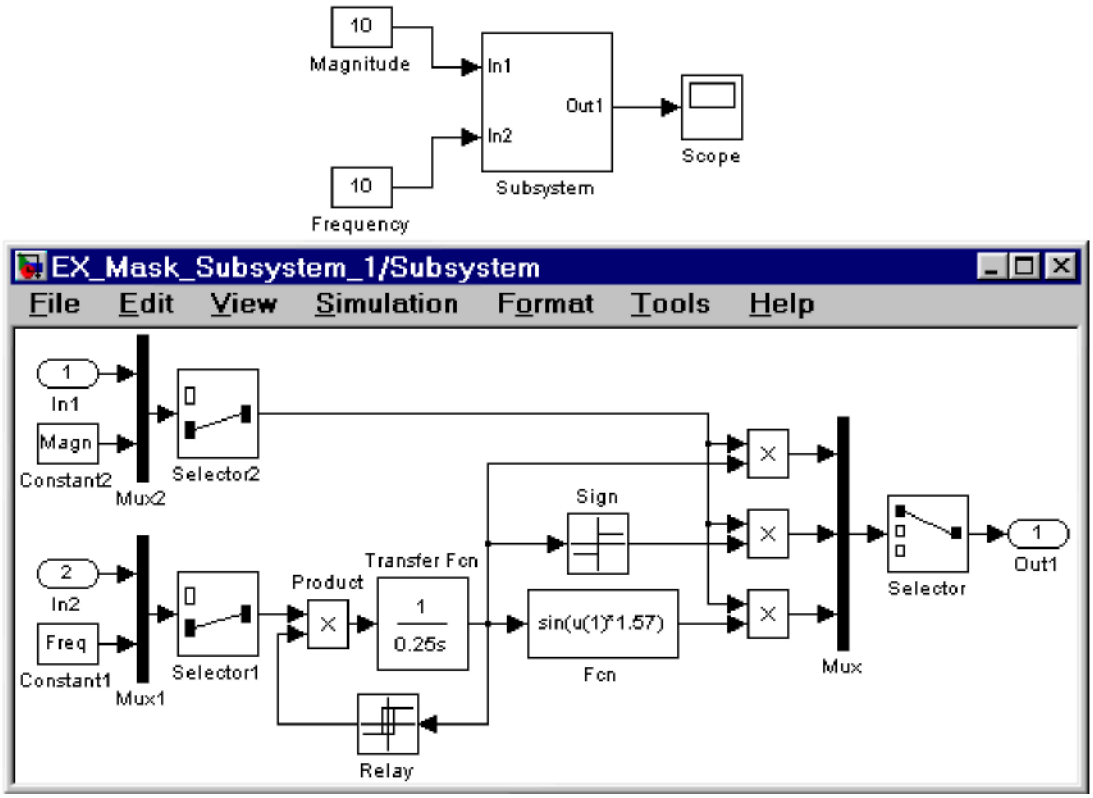

Рис. 1. 70. Функціональний генератор.

Модель генератора володіє наступними можливостями:

1. Значення амплітуди і частоти сигналу можуть задаватися або як параметри генератора в його вікні діалогу, або від зовнішніх джерел через вхідні порти.
2. Форма вихідного сигналу генератора (трикутник, прямокутник або синусоїда) задається у вікні діалогу.

Вигляд вікна діалогу, створеного за допомогою редактора маски, показаний на рис. 1. 71.

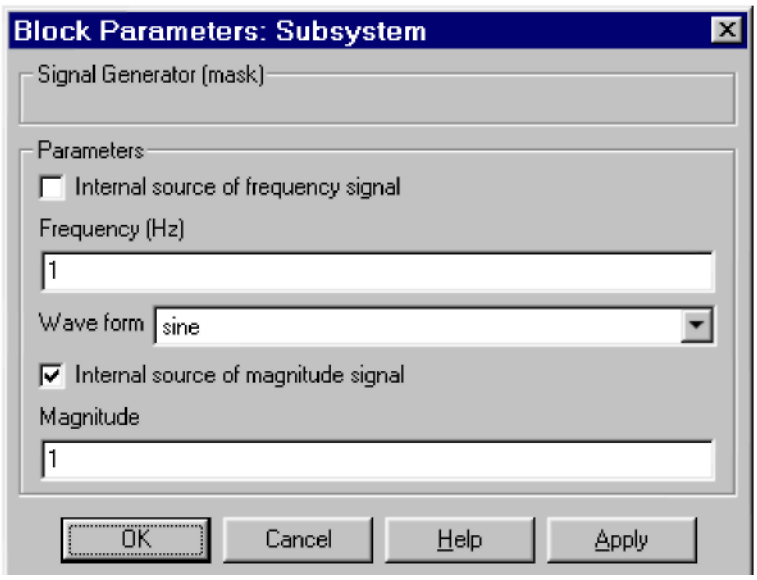

Рис. 1. 71. Вікно параметрів генератора

Назва параметру, ідентифікатор зв’язаної з ним змінної, тип элемента інтерфейсу і формат параметра приведені в таблиці 1.2.

Таблиця 1. 2

N	Prompt	Variable	Control Type	Assigggment	Назначение
1	Internal source of frequency signal	Internal_freq	Checkbox	Evaluate	Задает тип источника сигнала задания на частоту: внутренний или внешний.
2	Frequency (Hz)	Freq	Edit	Evaluate	Задает величину задания на частоту внутреннего источника
3	Wave form	Wave_form	Popup	Evaluate	Задает форму выходного сигнала: треугольник, прямоугольник или синусоида
4	Internal source of magnitude signal	Internal_magn	Checkbox	Evaluate	Задает тип источника сигнала задания на амплитуду: внутренний или внешний.
5	Magnitude	Magn	Edit	Evaluate	Задает величину задания на амплитуду внутреннего источника

Вікно редактора маски з відкритою вкладкою **Initialization**, в якому створене вікно параметрів генератора, показане на рис. 1. 72.

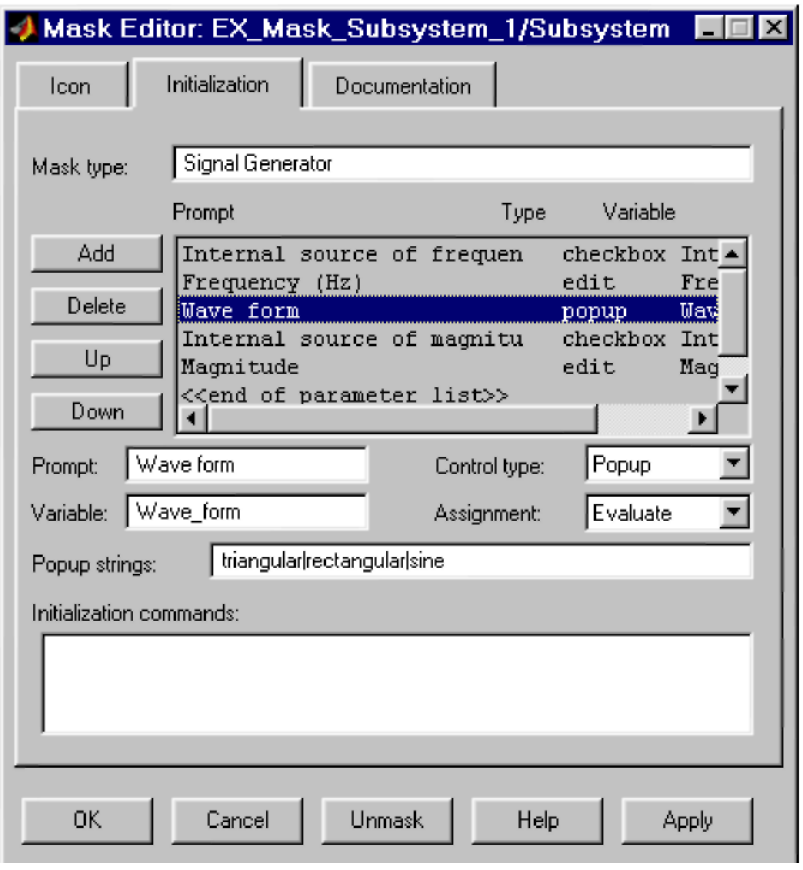

Рис. 1. 72. Вікно редактора маски на етапі створення вікна параметрів

Вибір типу джерел задавання на частоту (внутрішній або зовнішній) здійснюється за допомогою блоку **Selector1** (див. рис. 1. 70). Для цього значення параметра **Elements** блоку **Selector1** задане як **[(Internal_freq+1)]**. Таким чином, якщо прапорець параметра **Internal source of frequency signal** установлений, тоді числове значення змінної **Internal_freq** дорівнює **1** і на вихід селектора проходить сигнал від внутрішнього джерела, якщо ж прапорець знятий, тоді на вихід селектора проходить сигнал від вхідного порта системи (тобто від зовнішнього по відношенню до генератора джерела). Аналогічним чином за допомогою змінної **Internal_magn** виконується вибір джерела сигналу задавання на амплітуду.

Вибір форми вихідного сигналу виконується також за допомогою блоку **Selector**. Трикутний, прямокутний і синусоїдний сигнали об'єднуються в вектор за допомогою блоку **Mux**, а потім, в залежності від числового значення змінної **Wave_form**, блок **Selector** виконує вибір необхідного елемента вхідного вектора. Значення параметра **Elements** блоку **Selector** задане як **[Wave_form]**. Таким чином, якщо, наприклад, параметр генератора **Wave form** має значення **Sine**, тоді числове значення змінної **Wave_form** дорівнює трьом, і, отже, на вихід селектора проходить третій елемент вхідного вектора, тобто синусоїдний сигнал.

1.6 Курсове проектування з дисципліни «Імітаційне моделювання економічних процесів»

1.6.1 Мета курсового проектування і приклади варіантів завдань

В процесі роботи над курсовим проектом необхідно розробити і дослідити імітаційну модель конкретного економічного процесу або системи. В якості програмного середовища моделювання в даному навчальному посібнику пропонується використати інструментарій **Simulink** в складі широко відомої системи **MATLAB**.

Кожна модель містить параметри, що варіюються, і спостерігаємі змінні, які визначаються згідно варіанту завдання. Після побудови і налагоджування імітаційної моделі виконується підбір параметрів розприділення спостерігаємих змінних: будуються гістограми відносних частот і підбирається вид функцій розприділень.

Приблизні варіанти завдань:

1. Банківська система з двома касами. Черговий відвідувач вибирає касу, у якій найменша черга. Модель зупиняється у випадку закінчення моделюємого часу або при перевищенні довжини однієї із черг. **Змінні, що варіюються:** середній час обслуговування кліента для кожного касира, максимальна довжина черги. **Змінні, що спостерігаються:** процент простоювання кожного касира, середня довжина кожної черги.
2. Автозаправочна станція, що реалізує три види бензину. Для кожного виду задається імовірність його використання. Модель зупиняється при витраченні одного із видів бензину. **Змінні, що варіюються:** запаси кожного виду бензину, імовірності використання кожного із видів. **Змінні, що спостерігаються:** валовий прибуток, нереалізовані залишки.
3. Лінія із складання комп'ютерів, які складаються із п'яти компонентів. Для кожного компонента задається період поступлення, який являється випадковим числом. Модель зупиняється при закінченні часу моделювання. Якість компонентів вважати необмеженою. **Змінні, що варіюються:** період поступлення кожного із компонентів, час складання комп'ютера. **Змінні, що спостерігаються:** кількість зібраних комп'ютерів за одиницю часу.
4. Аеропорт на дев'ять літаків. Задаються середнє значення інтервалів часу між прилітаючими і відлітаючими літаками. Кількість літаків, що чекають посадки, обмежена. Модель зупиняється у випадку неможливості прийняти черговий літак. **Змінні, що варіюються: і**нтервали часу між прилітаючими і відлітаючими літаками, кількість літаків, що чекають посадки. **Змінні, що спостерігаються:** середній час чекання посадки, середнє число літаків на посадковій смузі.
5. Процес подачі заяв в приймальну комісію. Заяви подаються на два факультети. Для кожного факультету визначається прохідний бал. Кожна заява супроводжується сумою балів, які були набрані в результаті тестування. В процесі моделювання необхідно врахувати нерівномірність кількості заяв, що подаються, за часом. **Змінні, що варіюються:** прохідний бал для кожного факультету, середня кількість балів вступників. **Змінні, що спостерігаються:** кількість поданих заяв на кожний факультет.
6. Страхова компанія. Необхідно промоделювати два потоки інформації: заяву на отримання страхових полісів і заяву на виплату страховок. Передбачити наявність часових інтервалів, які відносяться до розгляду заяви і видачі страхових сум. **Змінні, що варіюються:** середній інтервал подачі заяв, середній інтервал сум, що виплачуються, імовірність виникнення нещасливого випадку. **Змінні, що спостерігаються:** сума страхових надходжень, сума страхових виплат.
7. Проектування WEB-сайтів. Необхідно опрацювати потік заявок на проектування WEB-сайтів. Вартість проектування в кожному випадку являється випадковою величиною. Необхідно установити зв'язок між вартістю проектування і часом розробки сайту. Проектні роботи виконують дві групи розробників**. Змінні, що варіюються: і**нтервал надходження заявок, середня вартість проектування сайту. **Змінні, що спостерігаються:** отримуваний

валовий прибуток, кількість заказів.
8. Магазин «Хот-догів». Магазин продає два види «хот-догів», які відрізняються розмірами і вартістю. Покупці купують продукцію і випадковим чином вибирають розмір «хот-дога». Модель зупиняється при закінченні запасу одного із видів**. Змінні, що варіюються:** середній інтервал покупок, запас кожного із видів продукції. **Змінні, що спостерігаються:** залишок нерозпроданих запасів, сумарний прибуток.
9. Маршрутне таксі. Необхідно промоделювати роботу маршрутного таксі, що виконує рух по кільцю з чотирма зупинками. Для кожної зупинки генерується потік пасажирів. **Змінні, що варіюються:** кількість місць в маршрутному таксі, швидкість маршрутки, середня кількість пасажирів на зупинці. **Змінні, що спостерігаються:** сумарний прибуток, середній процент заповнення маршрутки.
10. Підтримка програм 1С: Підприємство. Фірма займається розробкою власних конфігурацій і продажою типових рішень. Вартість робіт по виконанню нетипового рішення вище, чим типового, але і витраченого часу також більше. Потік заявок на виконання робіт повинен передбачити процедуру співвідношення кожної заявки до того або іншого рішення. **Змінні, що варіюються:** середній інтервал між заявками, середня вартість кожного із рішень. **Змінні, що спостерігаються:** валовий прибуток, сумарні витрати.

1.6.2 Зміст пояснюючої записки курсової роботи

Пояснююча записка курсової роботи вміщує наступні основні розділи:

- ***Вступ***. Даний розділ вміщує матеріал, який сприяє розумінню актуальності виконаної роботи. Коротко описуються причини необхідності побудови імітаційної моделі економічної системи або процесу і фіксуються бажані цілі моделювання.
- ***Постановка задачі.*** В цьому розділі пояснюючої записки детально представляється задача моделювання, а також структура вхідних і вихідних даних. Крім того, в цьому ж розділі даються пояснення про наявні ресурси для розв'язування указаної задачі (технічні засоби і програмні інструментарії).
- ***Блок-схема розробленої моделі***. Цей розділ являється одним із основних розділів курсового проекту в плані представлення результатів побудови імітаційної моделі. Результуюча модель повинна представлятися у вигляді ієрархічної структури. При цьому окремі блок-схеми представляють собою закінчені і повністю налагоджені описи функціонально виділених підсистем моделюємої економічної системи або процесу. Всі елементи блок-схем повинні супроводжуватися необхідними поясненнями.
- ***Часові діаграми***. Даний розділ служить для ілюстрації результатів прогонів імітаційної моделі. Формуємі часові діаграми повинні вміщувати дані про змінювання модельного часу, спостерігаємих змінних, змінних, що варіюються,

а також змінних, які носять пояснюючий характер. При необхідності, в даний розділ можуть бути включені допоміжні часові діаграми яких-небудь локальних дільниць імітаційної моделі.

- ***Опис функціонування моделі.*** Тут міститься пояснюючий текст детального опису функціонування розробленої імітаційної моделі з указанням ролі кожного застосовуваного елемента блок-схеми. Даний опис повинен обов'язково містити посилання на розроблені блок-схеми і часові діаграми.
- ***Опис результатів дослідження моделі.*** В даному розділі викладаються результати множини прогонов побудованої і налагодженої імітаційної моделі. Отримані результати представляються в вигляді гістограм відносних частот і функцій розприділень спостерігаємих змінних.
- ***Висновок.*** Розділ пояснюючої записки, в якому формулюються висновки, основані на отриманих результатах дослідження моделі, і даються рекомендації про оптимальне сполучення параметрів, що варіюються.

СПИСОК ЛІТЕРАТУРИ

1. Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н. MATLAB 7. – СПб.: БХВ-Петербург, 2005. – 1104 с.
2. Наместников А.М. Разработка имитационных моделей в среде MATLAB. – Ульяновск, УлГТУ, 2004.
3. Ануфриев И.Е. Самоучитель MatLab 5.3/6.x.- СПб.: БХВ-Петербург, 2002.- 736 с.
4. Дьяконов В.П. Matlab 6/6/1/6/5+Simulink 4/5/ Основы программирования: Руководство пользователя.- М.: Солон-Пресс, 2002.- 768 с.
5. Мэтьюз Д., Финк К. Численные методы. Использование Matlab (3-е издание).- СПб.: Вильямс, 2001.- 720 с.
6. Чен К., Джиблин П., Ирвинг А. Matlab в математических исследованиях.- М.: Мир, 2001.- 346 с.
7. Черных И.В. Simulink:среда создания инженерных приложений.- М.:Диалог-МИФИ, 2004.- 496 с.
8. Kwon Y.W. The Finite Element Method using MATLAB.-Boca Raton a.o.: CRC Press, 1997.- 519 p.