МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Донецький національний технічний університет

Методичні вказівки
до лабораторних занять з дисципліни

“МОДЕЛЮВАННЯ ЕЛЕКТРОМЕХАНІЧНИХ СИСТЕМ”

(для студентів спеціальності 7.0922.08)

Розділ
“ОСНОВИ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ”

Донецьк 2007

УДК 681-332(07)

Методичний посібник до лабораторних робіт з дисципліни “Моделювання електромеханічних систем” (для студентів спеціальності 7.0922.08). Розділ “Основи математичного моделювання” / Сост.: О.І. Толочко, Г.С. Чекавський, О.В. Песковатська, П.І. Розкаряка. - Донецьк: ДонНТУ, 2006. – 101 с.

В роботі наведені відомості про програму структурного моделювання *Simulink* пакета *MATLAB* та основні прийоми роботи в цьому середовищі; викладені засоби математичного опису лінійних та нелінійних, неперервних та дискретних систем і методика подання їх у вигляді структурних моделей, засоби створення вхідних сигналів та фіксації вихідних сигналів; описані основні бібліотечні *Simulink*-блоки і методика створення власних блоків у вигляді незамаскованих і замаскованих підсистем; приведені завдання і методичні рекомендації до виконання 6 лабораторних робіт, спрямованих на придбання навичок структурного математичного моделювання основних типів динамічних систем. Усі розділи забезпечені переліком контрольних питань і завдань.

У розробці та постановці лабораторних робіт приймали участь студенти В.М. Майборода і А.В. Бабенко.

Укладачі: О.І. Толочко, проф.
Г.С. Чекавський, доц.
О.В. Песковатська, асист.
П.І. Розкаряка, асист.

Відповідальний за випуск П.Х.Коцегуб, проф.

Рецензент
зав. кафедри СПУ С.С. Старостін, доц.

ВВЕДЕННЯ

Математичне моделювання систем електропривода є важливим етапом їх проектування. На цьому етапі виконується аналіз динамічних властивостей електроприводу з точки зору відповідності технологічним вимогам, уточнюється структура системи керування, типи регуляторів, їх параметри. При моделюванні можна безбоязно досліджувати поведінку системи в аварійних ситуаціях, що неможливо на лабораторних і тим паче на діючих промислових установках. Іноді моделювання проводять для того, щоб оцінити коректність прийнятих при математичному опису системи спрощень**.**

Математичне моделювання на ЕОМ зводиться до розв‘язання системи диференційних або різницевих рівнянь, що описують динамічні властивості досліджуваного об‘єкта, в результаті чого отримують графіки перехідних процесів об‘єкта в характерних режимах його роботи.

При дослідженні систем автоматичного регулювання і, зокрема, систем керування електроприводами вихідним матеріалом для моделювання дуже часто є структурна схема, що уявляє собою графічну інтерпретацію математичного опису системи. В структурну схему, крім неперервних та (або) дискретних динамічних ланок можуть входити арифметичні ланки, ”типові нелінійності” та нелінійності довільного вигляду, задані аналітично або графічно, різні ключі, логічні елементи та т.і.

Перехідні процеси можна розраховувати як за допомогою програм, написаних будь-якою алгоритмічною мовою, що потребує від дослідника достатньо високої кваліфікації в галузі програмування та обчислювальної математики, так і за допомогою спеціалізованого програмного забезпечення, що дозволяє користувачу задавати моделі у вигляді математичних рівнянь або у вигляді структурних схем, обирати методи розв‘язання диференційних рівнянь та їх параметри в діалоговому режимі та отримувати результати у зручній формі.

Одним із найзручніших програмних засобів структурного математичного моделювання на теперішній час є додаток *Simulink* пакета *MATLAB* фірми *Mathwork* [1-8].

Основою для розробки моделей в *Simulink* є бібліотеки блоків, з котрих складаються структурні схеми САР, що повинні бути дослідженими. Розрахунок перехідних процесів може бути виконаний за допомогою відповідних операцій *Simulink*-меню або в програмному режимі (з використанням функцій пакета *MATLAB*).

Метою даного лабораторного практикуму є придбання практичних навичок математичного моделювання лінійних та нелінійних, неперервних та дискретних електромеханічних систем у названому вище програмному середовищу.

Перша частина практикуму присвячена основам математичного моделювання, друга – моделюванню електромеханічних систем на базі двигунів постійного струму, третя – на базі двигунів змінного струму.

Автори методичних вказівок вважають, що читач обізнаний з основами *Windows*-технології та має навички роботи і програмування в пакеті *MATLAB*.

У часи, передбачені для самостійної роботи студентів, вони повинні скласти звіт з попередньої лабораторної роботи та підготуватися до наступної шляхом проробления відповідного лекційного матеріалу, рекомендованої літератури та методичних вказівок.

Для перевірки готовності до виконання лабораторної роботи викладач може провести тестове контрольне опитування.

Під час виконання лабораторної роботи студенти збирають розроблені при підготовці до неї математичні моделі, розробляють програмні файли для їхньої ініціалізації та для досліджень у відповідності з планом модельного експерименту і фіксують потрібні результати.

Звіт з лабораторної роботи повинен отримувати таку інформацію:

1) назва роботи;
2) завдання згідно з варіантом;
3) схеми моделей з параметрами блоків;
4) тексти програмних модулів;
5) результати досліджень.

1 Лабораторна робота №1

ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ ПРОГРАМИ СТРУКТУРНОГО МОДЕЛЮВАННЯ *Simulink* ПАКЕТА *MATLAB*

1.1 Запуск *Simulink*. Перелік бібліотек та демонстрацій

Simulink можна запустити з командної строки пакета *MATLAB* однойменною командою

```
>>simulink
```

де **>>** – запрошення *MATLAB* до вводу команди, або спеціальною кнопкою на панелі інструментів, що показана на рис. 1.1.

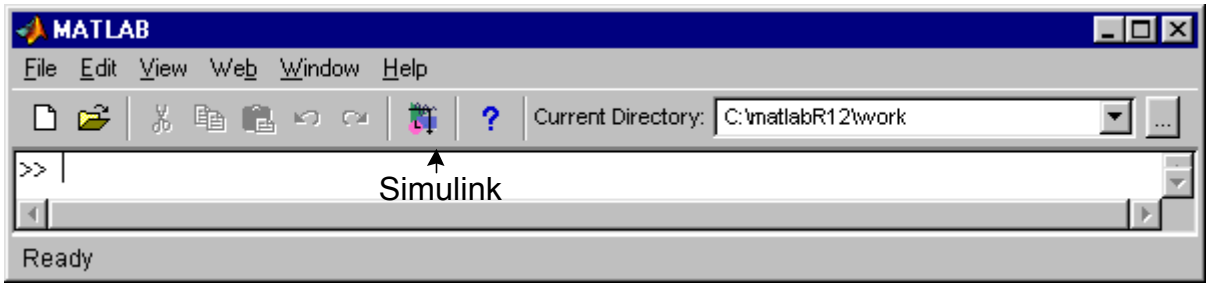


Рис. 1.1. Командне вікно системи *MATLAB*

При цьому на екрані відкривається вікно з бібліотеками блоків, представлене на рис. 1.2, яке має назву *Simulink Library Browser* (*Навігатор Simulink-Бібліотек*).

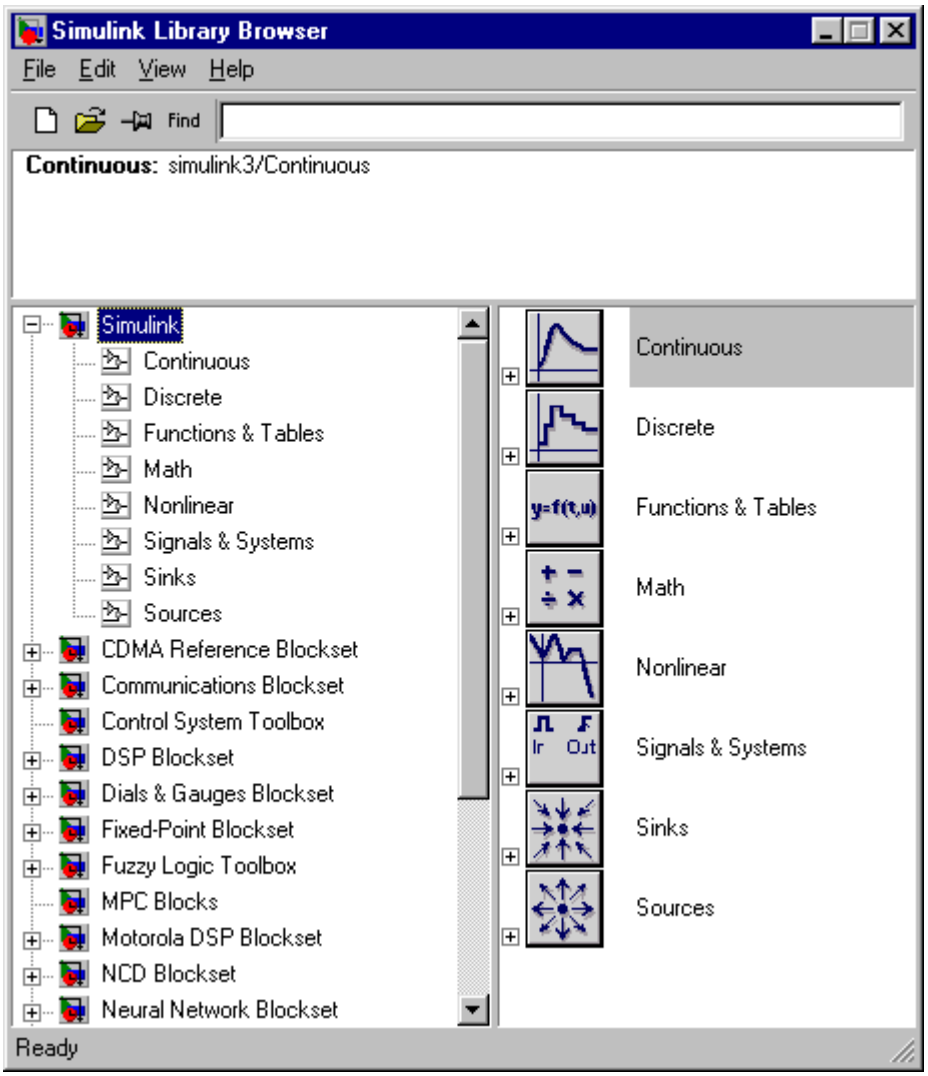


Рис. 1.2. Вікно *Simulink Library Browser*

Кожну бібліотеку можна розкрити подвійним щигликом миші на піктограмі (іконці) бібліотеки в правій частині вікна або вибором назви бібліотеки в лівій частині вікна. При цьому піктограми блоків обраної бібліотеки заміщають піктограми бібліотек.

Щигликом правою кнопкою миші на імені бібліотеки через *popup-menu* (меню, що випадає), можна відкрити будь-яку бібліотеку у вигляді, звичному для користувачів більш старих версій (*MATLAB* 5.1 та *Simulink* 3).

Simulink зі старим інтерфейсом можна також відкрити з командного рядка командою

```
>>simulink3
```

При цьому відкривається вікно, зображене на рис. 1.3.

З цього вікна кожну бібліотеку можна розкрити подвійним щигликом миші по її піктограмі. Вікна бібліотек будуть показані нижче (при розгляді їхніх блоків).

До основних *Simulink*-бібліотек відносяться ті, що розташовані у верхньому рядку вікна *Simulink 3*, тобто

Source – джерела вхідних сигналів;
Sinks – блоки, що реєструють;
Continues – неперервні лінійні динамічні ланки;
Discrete – дискретні лінійні динамічні ланки;
Math – математична бібліотека;
Functions & Tables – функції та таблиці;
Nonlinear – нелінійні блоки;
Signals & Systems – сигнали та системи;
Subsystems – підсистеми.

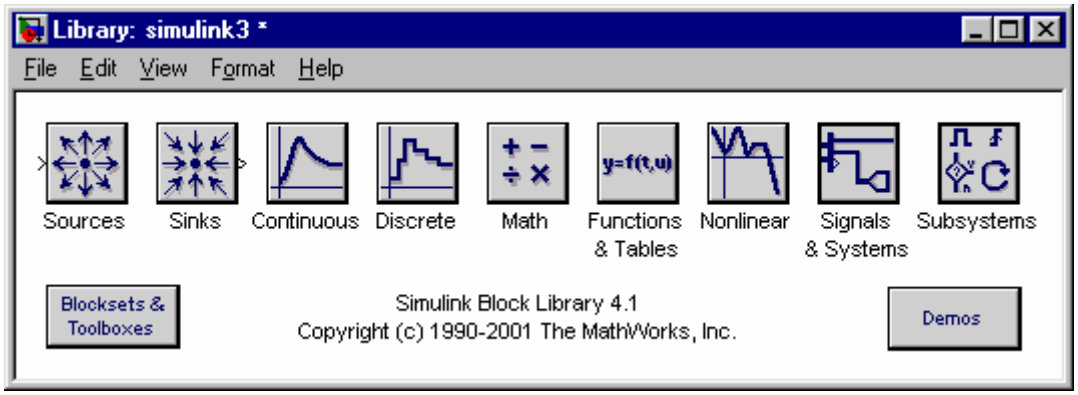


Рис. 1.3. Вікно бібліотек блоків *Simulink* 4.1

Розділ *Blocksets & Toolboxes* являє собою сукупність бібліотек установок і інструментів, зміст якої (не повний) показано на рис. 1.4.

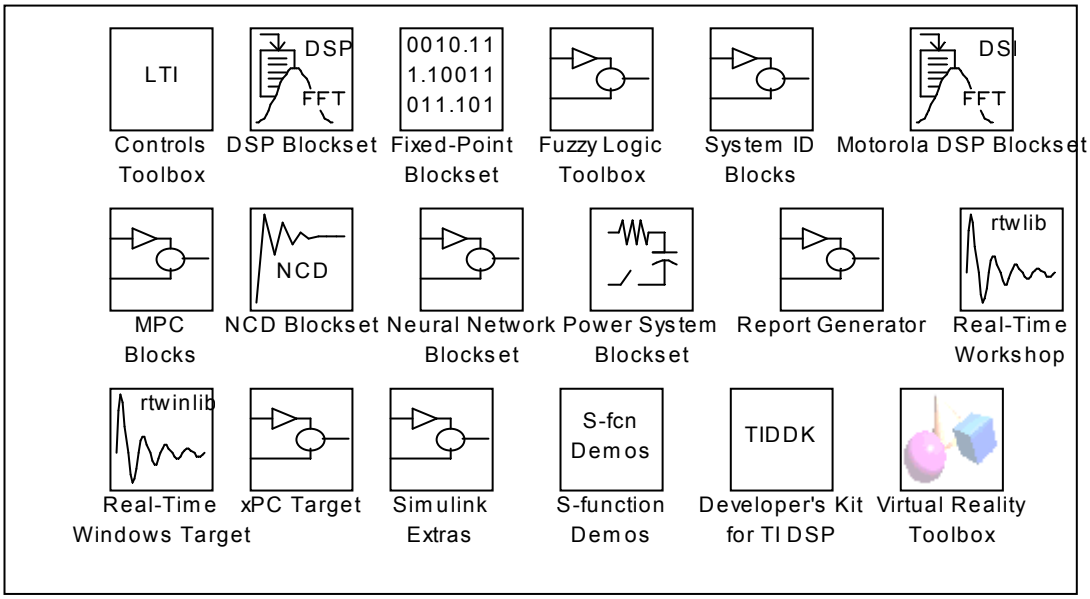


Рис. 1.4. Вікно бібліотек *"Blocksets & Toolboxes"*

До цієї сукупності належать:

Control Toolbox – аналіз і синтез лінійних систем;
DSP Blockset – пристрої, що використовують *Discrete Signal Processing Toolbox* (обробка дискретних сигналів);
Motorola DSP Blockset – пристрої обробки дискретних сигналів фірми *Motorola*;

Fixed-Point Blockset	–	пристрої, що використовують операції з фіксованою точкою;
Fuzzy Logic Toolbox	–	*Fuzzy*-регулятори з демонстраціями;
System ID Blocks	–	блоки ідентифікації;
MPC Blocks	-	блоки, що використовують *Model Predictive Control Toolbox* (прогнозне керування);
NCD Blockset	–	пристрої, що використовують *Nonlinear Control Design Toolbox* (синтез нелінійних систем);
Neural Network Blockset	-	елементи нейрональних мереж;
Power System Blockset	–	елементи електричних схем і систем, силова електроніка, електричні машини;
Report Generator	–	генератор звітів;
Real-Time Workshop	–	майстерня реального часу;
Simulink extras	–	спеціальні блоки *Simulink*;
S-Function Demos	–	демонстрація блоків *Simulink*, побудованих з використанням *S*-функцій;
Comm Tbx library	–	комунікаційна бібліотека (з використанням *Communication Toolbox*);
Stateflow	–	система моделювання подій;
Virtual Reality Toolbox	–	система віртуальної реальності.

Ретельний опис кожної з цих бібліотек вимагає окремої книги.

Зі спеціальних бібліотек (*Simulink extras*), представлених на рис. 1.5, для спеціалістів в галузі електропривода найбільший інтерес уявляють:

Additional Linear	–	аналогові лінійні неперервні динамічні ланки з початковими умовами (п.у.);
Additional Discrete	–	дискретні динамічні ланки з п.у.;
Additional Sinks	–	пристрої, що реєструють, для спектрального аналізу;
Flip Flops	–	тригери.

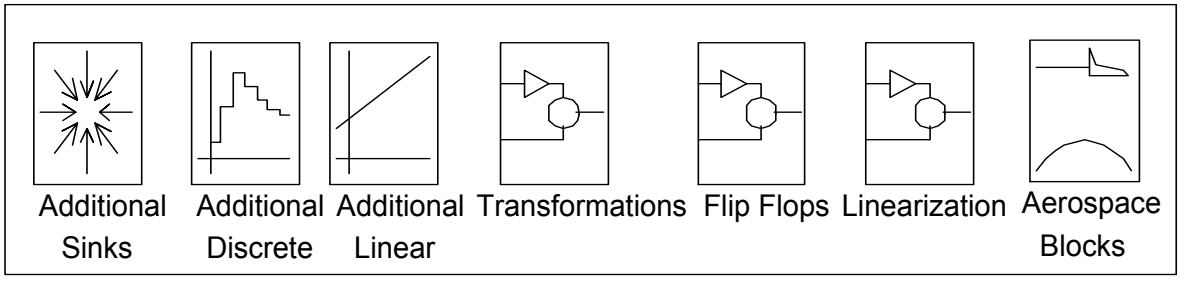


Рис. 1.5. Вікно бібліотек *Simulink extras*

При подвійному щиглику мишею по піктограмі блоку відкривається вікно введення його параметрів, у якому відображуються ім‘я блоку, його тип, коротенький опис і поля введення параметрів із супроводжуючими їх запитами. Більш докладну інформацію про блок можна побачити, звернувшись до функції *Help*. Іконка блоку найчастіше відображує його динамічні або статичні властивості і реагує на зміну параметрів.

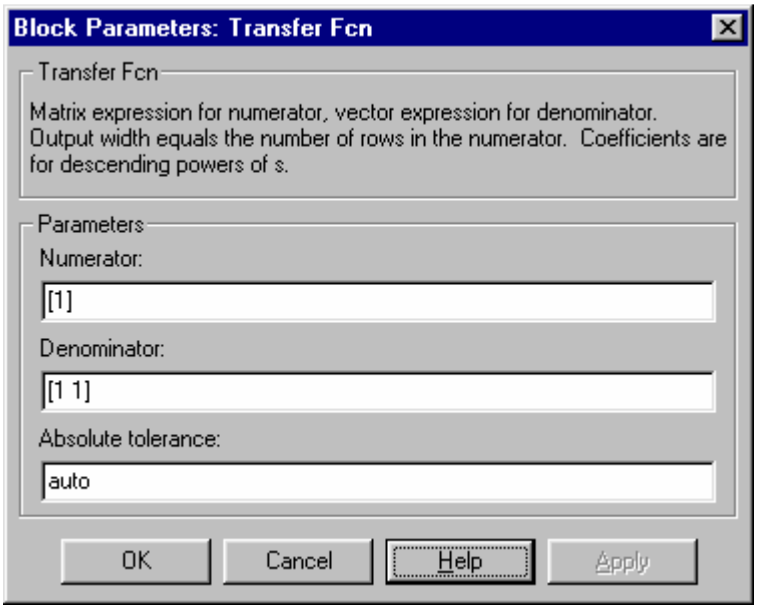


Рис. 1.6. Вікно введення параметрів блока *Transfer Fcn*

Як приклад, на рис. 1.6 наведено вікно введення параметрів блоку *Transfer Function.*

Параметри блоків можуть бути константами, змінними, функціями або виразами, припустимими в *MATLAB*. Будь-які змінні, від яких залежить параметр, повинні бути визначені в робочому середовищі (*Matlab Workspace*) до початку процесу моделювання, у противному випадку *Simulink* сигналізує про помилку в цьому блоці.

Клавіша *Apply* дозволяє оцінити результат зміни параметрів, не закриваючи вікна *Block Parameters*, клавіша *OK* закриває вікно введення параметрів із запам‘ятовуванням виконаних змін, клавіша *Cancel* закриває вікно введення параметрів зі скасуванням внесених змін.

Для знайомства з можливостями додатку *Simulink* та особливостями роботи деяких блоків можна скористуватися демонстраціями, які зручно запускати з вікна *MATLAB Demo Window*. Воно з’являється на екрані при виконанні команди

```
>>demo
```

або

```
>>demo simulink
```

та пропонує наступні демонстрації:

1) *General model demonstrations and samples* (основні демонстрації):

bounce – *Tracking a bouncing ball using reset integrator* (підстрибування м‘яча з використанням інтегратора зі скиданням вихідного сигналу),

simppend – *Simple pendulum simulation* (простий маятник),

onecart – *Spring-mass system* (візок з пружиною, до якого прикладається періодичне зусилля, з анімацією),

vdp – *Van der Pol equations simulation* (розв‘язання рівнянь Ван дер Пола),

simquat – *Quaternion rotation with animation* (обертання кватерніона),

toilet – *Toilet bowl flushing animation* (злив туалетного бачка),

countersdemo – *Counter demonstration* (демонстрація лічильників),

hardstop – *Friction with hard stops model* (модель тертя),

bangbang – *State events* (побудова фазового портрету),

dblcart1 – *Double cart demonstration* (двомасова система, що

складається з двох возиків, з‘єднаних між собою пружиною),

dblpend1 – *Pendulum with one joint* (маятник з одним суглобом),

dblpend2 – *Pendulum with two joints* (маятник з двома суглобами),

penddemo – *Inverted pendulum animation* (інверсний маятник, встановлений на візку, з дискретним спостерігачем стану),

thermo – *Thermodynamic model of a house* (термодинамічна модель будинку);

2) *Feature demonstrations and samples* (демонстрації сервісних блоків):

sl_subsys_semantics – *Subsystem examples* (приклади підсистем),

ifsub – *If block* (демонстрація умовного блоку, що керує виконанням підсистем),

triggeredsub – *Triggered subsystems* (підсистеми, які виконуються при перетині керуючим сигналом нуля та утримують вихідний сигнал до наступного перетину),

enablesub – *Enabled subsystems* (підсистеми, які виконуються при додатному значенні керуючого сигналу),

enabsubs – *Advanced enabled subsystems* (підсистеми з керуючим входом, що видає дозвіл на їх роботу, з додатковими можливостями),

logdemo – *Flip-flop blocks demonstration* (тригери),

prioritydemo – *Block priority demonstration* (блок пріоритету),

mergedemo – *Merge block demonstration* (блок поєднання),

busdemo – *Bus block demonstration* (блоки створення шини та її селектора),

matrixdemos – *Matrix signals demonstration* (матричні сигнали),

datatypedemo – *Data Typing demonstration* (демонстрація використання різних типів арифметичних даних),

combfilter – *Data Typing filter example* (фільтрація сигналів одиночної та подвійної точності),

zeroxing – *Accurate zero-crossing detection* (реєстрація перетину нуля),

sfundemos – *S-Functions (Simulink & Real-Time Workshop)* (демонстрація *S*-функцій);

3) *Automotive model demonstrations and samples* (автомобільні моделі);

4) *Aerospace model demonstrations and samples* (аерокосмічні моделі).

У підрозділі *Power System* розділу *Blockset* знаходяться демонстрації моделей з використанням електричних джерел, машини, трансформаторів, ліній електропередач, силової електроніки, тощо:

psbfilter – *Linear Filter* (лінійний електричний фільтра п‘ятої гармоніки),

psbtransient – *Transient Analysis* (аналіз перехідних процесів у лінійному електричному колі),

psbtransfo – *Linear Transformer* (лінійний трансформатор),

psbtransfosat – *Three-Phase Saturable Transformer* (трифазний трансформатор з насиченням),

psbctsat – *Current Transformer Saturation* (насичення трансформатора струму),

psbsurgnetwork – *AC Surge Arrester* (розрядник для захисту від атмосферних перенапруг мереж електропередач змінного струму),

psbmonophaseline – *Single-Phase Line* (однофазна мережа),

psbtriphaseline – *Three-Phase Line* (трифазна мережа),

psb2rectifiers – *Single-Phase Rectifiers* (однофазні випростувачі),

psbrectifier – *Three-Phase Diode Rectifier* (трифазний діодний випростувач),

psbconverter – *Thyristor Converter* (система трифазний тиристорний перетворювач - двигун постійного струму з пропорційно-інтегральним регулятором струму якоря),

psbswitching – *Ideal Switch* (переривання струму ідеальним ключем у колі з індуктивністю),

psbmosconv – *Mosfet Converter* (транзисторний *Mosfet*-перетворювач),

psbbuckconv – *GTO Buck Converter* (*GTO*-тиристор у складі перетворювача-компенсатора реактивної потужності),

psb1phPWM – *DC-DC and DC-AC PWM converters* (дискретні широтноімпульсні перетворювачі постійного та змінного струмів),

psb3phPWM – *Three-Phase PWM converters* (трифазні дискретні перетворювачі з широтноімпульсною модуляцією (ШІМ)),

psbbridges – *Universal Bridge in DC-AC PWM converter* (універсальний міст у ШІМ-перетворювачі),

psbloadshed – *Simplified Alternator* (синхронний генератор),

psbturbine – *Synchronous Machine* (синхронний генератор з гідравлічною турбіною),

psbpwm – *Asynchronous Machine* (асинхронна машина з живленням від ШІМ-інвертора),

psbpmmotor – *Permanent Magnet Synchronous Machine* (явнополюсна синхронна машина з живленням від ШІМ-інвертора),

psbmachines – *Machines and Load Flow* (асинхронна машина із дізель-генератором),

psbregulator – *Synchronous machine and regulator* (нелінійне керування гідравлічною турбіною та синхронним генератором),

psbdcmotor – *Starting of a DC Motor* (реостатний пуск двигуна постійного струму),

psbdcdrive – *Chopper Fed DC Motor Drive, Continuous* (двигун постійного струму з аналоговим імпульсним керуванням),

psbdcdrive_disc – *Chopper Fed DC Motor Drive, Discrete* (двигун постійного струму з дискретним імпульсним керуванням),

psbacdrive – *AC Motor Drive – Vector Control, Continuous* (дискретне векторне керування асинхронним двигуном),

psbcompensated – *Single-Phase Series Compensated Network* (однофазна

послідовна компенсована мережа),

psb3phseriescomp – Three-Phase Series Compensated Network, Discrete (дискретна трифазна послідовна компенсована мережа),

psbhvdc – Simple 6-pulse HVDC transmission system, Discrete (дискретна 6-пульсна високовольтна мережа електропередач постійного струму),

psbhvdc12pulse – Complete 12-pulse HVDC transmission system, discrete (дискретна 12-пульсна високовольтна мережа електропередач постійного струму),

psb3phsignaldq – Sequence and abc-dq0 transformation, discrete (реєстрація сигналів дискретного програмованого 3-фазного джерела з *abc-dq0*-перетворенням),

psb3phsignalsec – Three-Phase Programmable Source & Sequence Analysis, Discrete (частотний аналіз сигналів дискретного програмованого 3-фазного джерела

Імена демонстрацій співпадають з іменами *mdl*-файлів, що знаходяться відповідно у папках

Matlab \ *toolbox* \ *simulink* \ *simdemos* \ *simgeneral*

Matlab \ *toolbox* \ *simulink* \ *simdemos* \ *simnew*

Matlab \ *toolbox* \ *simulink* \ *simdemos* \ *simfeatures*

Matlab \ *toolbox* \ *simulink* \ *simdemos* \ *automotive*

Matlab \ *toolbox* \ *simulink* \ *simdemos* \ *aerospace*

Matlab \ *toolbox* \ *powersys* \ *powerdemo*

Так що кожну окрему демонстрацію можна виконати з командної строки введенням імені файлу моделі без поширення.

Для спілкування користувача з ЕОМ під час роботи у середовищі *MATLAB-Simulink* передбачені такі засоби:

- меню командного вікна *MATLAB* [4];
- основні меню вікон *Simulink*-моделей, *Simulink*-бібліотек та вікна *Simulink Library Browser* [2, 5, 6];
- контекстуальні меню цих вікон, що викликаються правою кнопкою миші при виділеному будь-якому елементові вікна;
- кнопки панелі інструментів;
- команди, що набираються та вводяться користувачем безпосередньо у вікні *Matlab Comand Window* або у програмному режимі при виконанні програми, що складається з відповідних команд.

1.2 Меню вікна *Simulink Library Browser*

Меню вікна *Simulink Library Browser*, відображеного на рис. 1.7, містить такі функції:

File – робота з файлами;
Edit – редагування;
View – вигляд вікна;

Help – допомога.

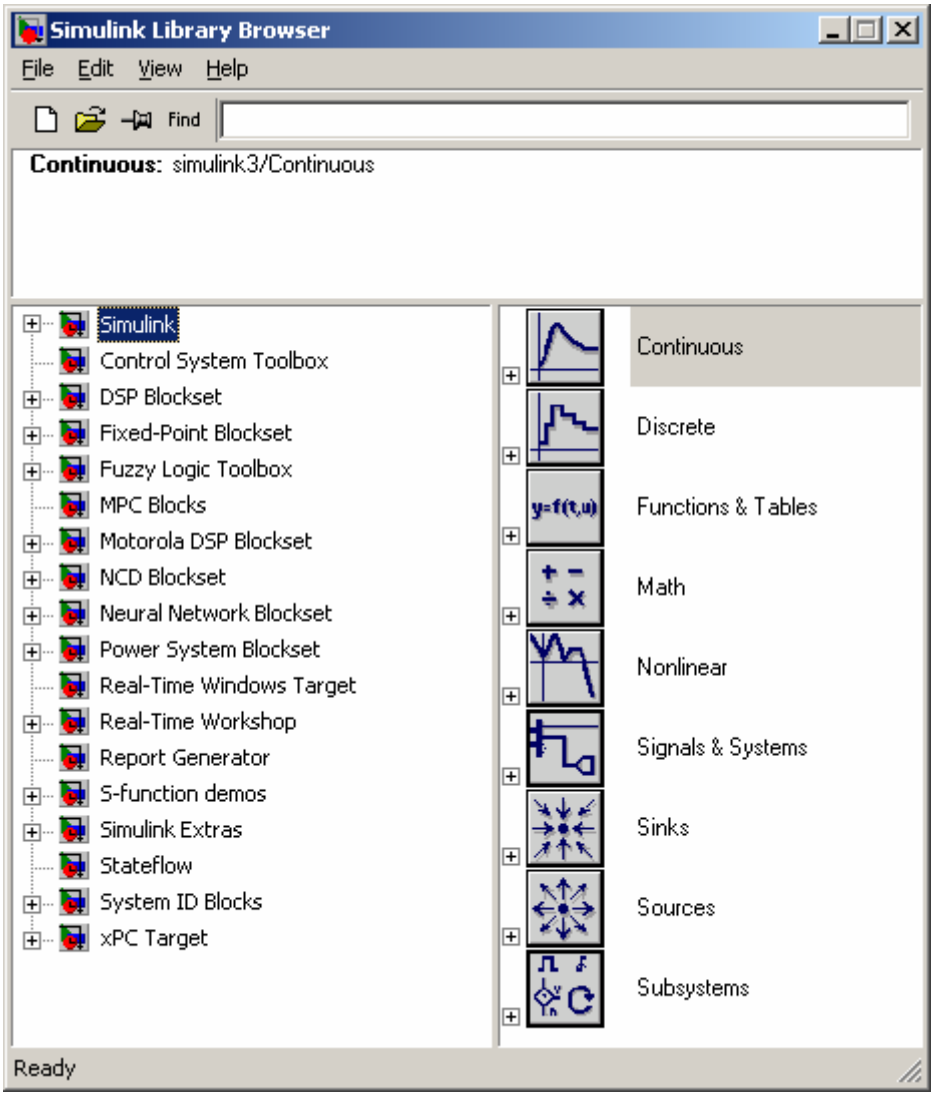


Рис. 1.7. Вікно навігатора *Simulink*-бібліотек

Кожну функцію цього меню можна вибрати не тільки мишею або клавішами <→>, <←> і <*Enter*>, але й комбінацією клавіш "<*Alt*>+<*s*>", де *s* – підкреслений символ імені обираної функції.

Функція *File* містить наступні операції:

New... – створити вікно для введення нової моделі (*Model (^N)*) або бібліотеки (*Library*);
Open... (^O) – відкрити модель;
Close – закрити вікно *Simulink Library Browser*;
Preferences… – настроювання середовища.

Операції підменю можна вибрати мишею, клавішами <↓>, <↑> та <*Enter*>, комбінацією клавіш, зазначеної в дужках, і клавішею <*s*>, де *s* – підкреслений символ імені обираної операції.

Для збереження моделей і інших файлів користувача в *MATLAB* передбачена папка *work*. У ній можна створити нову папку, наприклад, *C:\matlab\work\models*. Шлях до файлів-моделей необхідно додати у список

доступних директорій за допомогою операції *Set Path→ Add Folder… / Add with Subfolders…*функції *File* основного меню командного вікна *MATLAB*. Відповідне діалогове вікно, що з'являється при цьому, наведено на рис. 1.8. Нову папку можна поставити в початок списку (*Move to Top*), у кінець списку (*Move to Bottom*), перемістити на одну позицію вниз (*Move Down*) або догори (*Move Up*). Якщо не виконати запис нового списку у файл, то установка діє тільки протягом поточного сеансу роботи. Для того, щоб використовувати нову установку в майбутніх сеансах, потрібно записати її у файл (*MATLAB \ toolbox \ local \ pathdef.m*) за допомогою клавіші *Save.*

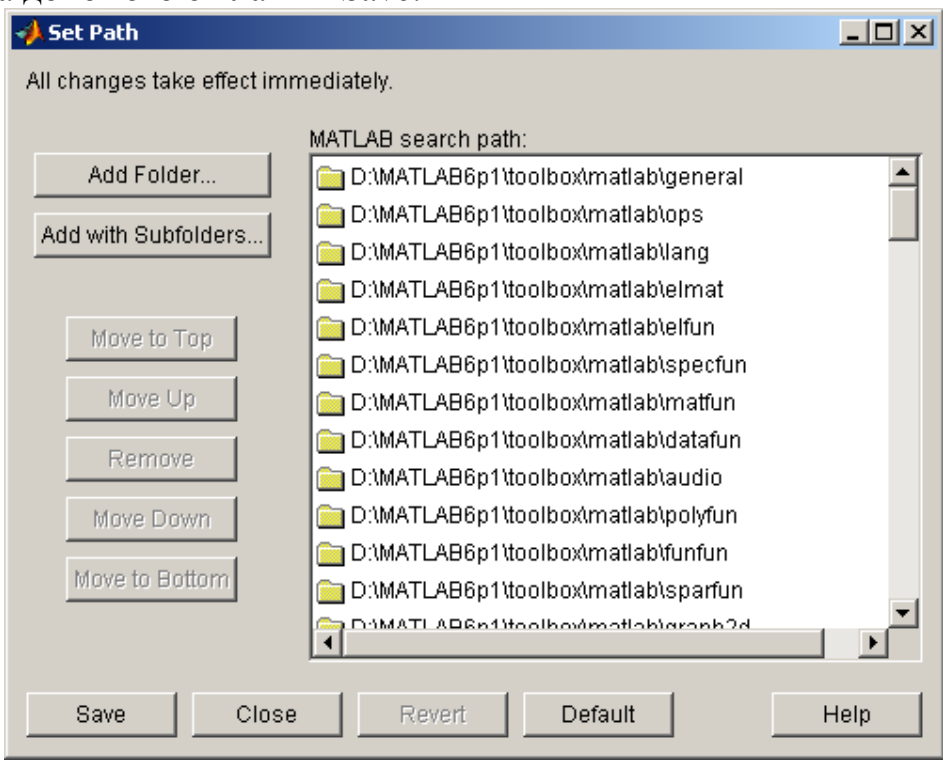


Рис. 1.8. Вікно встановлення/видалення доступу до каталогів

Файли моделей і бібліотек додатка *Simulink* повинні мати маску **.mdl*. Вони є текстовими файлами особливої структури і містять інформацію про структуру та параметри моделей, достатню для їхнього графічного відображення і моделювання. З вікон *Simulink* вони відкриваються у графічному вигляді, а з будь-якого текстового редактора (наприклад, вбудованого текстового редактора системи *MATLAB medit.exe*) – у текстовому.

Текстовий редактор системи *MATLAB* відкривається командою *File → New → M-file* меню командного вікна *MATLAB*.

Відкрити модель можна не тільки з меню *File* будь-якого *Simulink*-вікна, але й з меню *File* пакета *MATLAB* за допомогою функції *Open…*, а також з командного рядка *MATLAB* введенням у ній імені файлу-моделі без розширення. Оскільки в такий же спосіб (з командного рядка) у *MATLAB* викликаються на виконання *script*-файли (головні програми, написані алгоритмічною мовою системи *MATLAB*) і виводяться значення змінних [1-4], тож файлам потрібно давати оригінальні і не занадто короткі імена.

Тут слід зазначити, що при використанні імен змінних, функцій, файлів, програм, що збігаються з іменами, визначеними в робочому просторі або у файловій системі *MATLAB* раніше, вона не виводить попереджень про збіжність імен. При виконанні команди, що містить яке-небудь ім'я, *MATLAB* шукає це ім'я серед змінних, потім серед файлів з розширеннями *m*, *mex*, *mdl* і серед вбудованих функцій в указаному порядку. Перше з виявлених імен сприймається як об'єкт для виконання команди. Отже, використання неоригінальних імен файлів може привести до казусів.

Якщо при спробі відкрити модель або виконати яку-небудь програму з командного рядка на екран виводиться повідомлення «*Undefined function or variable 'name'*» (невизначена функція або змінна), то це означає, що користувач або помилився при наборі імені файлу, або цей файл знаходиться в недоступній для системи *MATLAB* директорії.

Функція *Preferences*, при виборі якої відкривається вікно, наведене на рис. 1.9, дозволяє виконувати настроювання середовища *MATLAB* і *Simulink*.

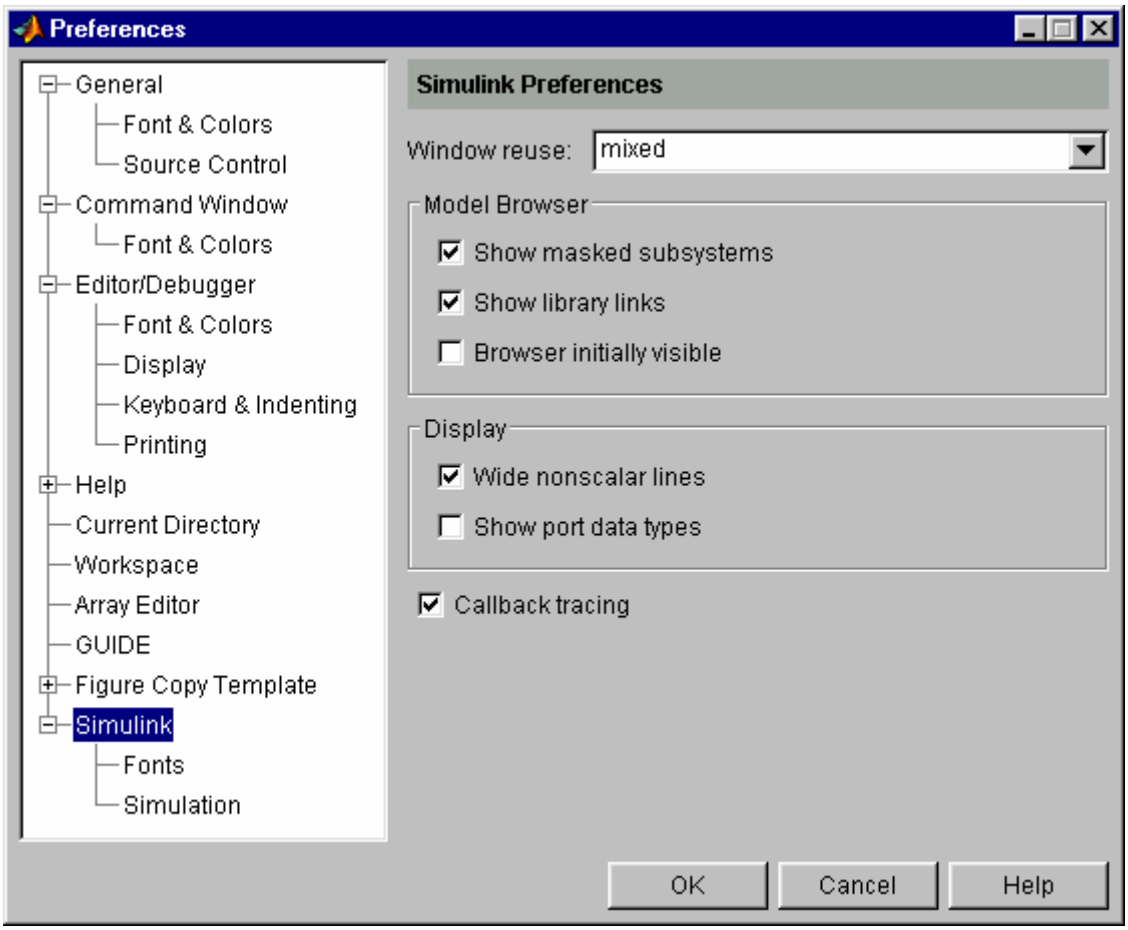


Рис. 1.9. Вікно настроювання середовища *MATLAB* і *Simulink*

На початковому етапі можна використовувати настроювання середовища, запропоноване у авторському варіанті. Вивчати функції настроювання слід поступово, при наявності необхідності. Докладний опис функцій цього вікна зайняв би забагато місця. Тому перелічимо лише деякі з тих параметрів, які можна корегувати за допомогою функції *Preferences*, не зупиняючись на

рекомендаціях щодо значень цих параметрів.

До них належать кольори, типи та розміри шрифтів (*Font & Colors*) різних вікон, формат та основні правила виведення чисельної інформації у командному вікні (*Command Window*), особливості інтерфейсу текстового редактора (*Editor/Debugger*), місце розташування файлів допомоги у файловій системі (*Help*), властивості вікон активних директорій (*Current directory*), робочого (оперативного) простору (*Workspace*) та редактора масивів (*Array editor*), особливості копіювання у буфер графічних фігур (*Figure Copy Template*), особливості зображення моделей зі складною структурою (*Simulink*) та параметри математичного моделювання (*Simulation*).

Для прикладу розглянемо детальніше параметр *Window reuse* (повторне використання вікна) панелі *Simulink Preferences*. Він визначає, чи повинен *Simulink* використовувати старе вікно, чи треба відкрити нове вікно для показу змісту підсистеми, і може приймати одне з 4-х значень: *none*, *reuse*, *replace* і *mixed*.

Дії, що виконуються при відкриванні й закриванні підсистеми з різними значеннями цього параметра, перераховані в табл. 1.1.

Таблиця 1.1

Значення параметра	Дії при відкриванні підсистеми	Дії при закриванні підсистеми
none	Зміст підсистеми відображається у новому вікні на передньому плані	Батьківське вікно переміщається на передній план, а вікно підсистеми – на задній
reuse	Зміст підсистеми заміщає батьківську модель у поточному вікні	Вихідна (батьківська) модель заміщає зміст підсистеми в поточному вікні
replace	Зміст підсистеми відображається у новому вікні, а батьківське вікно зникає	Батьківське вікно з'являється, а вікно підсистеми зникає
mixed	Зміст підсистеми відображається в її власному вікні	Батьківське вікно переміщається на передній план, а вікно підсистеми зникає

Установкою або скиданням прапорців у полях *Browser initially visible*, *Show library links* і *Look under masks*, що визначають властивості *Навігатора Моделей* (*Model Browser*), можна змінювати режими перегляду структури моделі.

При установці першого з перерахованих прапорців знов відкрите *Simulink*-вікно поділяється на дві частини. Ліва частина приділяється для навігатора, а права – для моделі. Це має сенс при наявності в моделі звичайних і замаскованих підсистем (*Subsystems*). Два інших прапорці дозволяють відповідно вмикати і вимикати зв'язок підсистеми з бібліотекою і переглядати зміст замаскованих блоків. Розрив зв'язку з бібліотекою передбачено для коректування структури підсистеми або вікна її маскування.

Для зображення векторних сигналів стовщеними єднальними лініями слід

установити прапорець у поле *Wide nonscalar lines*, а для виводу типів вихідних сигналів – у поле *Port data types* опції *Display*.

При установці прапорця в поле *Callback tracing* на екран виводяться команди, що виконуються в специфічних режимах роботи блоків чи блокових діаграм. Як приклад, можна навести команди, записані в поле параметра *Open function* вікна *Block Properties*, які виконуються при подвійному щиглику мишею на піктограмі цього блоку.

Функція *Edit* містить наступні команди:

Add to the current model (^I) – скопіювати обраний блок або бібліотеку в активне вікно моделі чи бібліотеки;

Find block…(^F) – знайти блок за його ім‘ям;

Find next block (<F3>) – знайти наступний блок.

Функція *View* містить команди керування включенням типових панелей (*Toolbar* – інструменти, *Status* – статус, *Description* – опис), команди керування розміром іконок у правій частині навігатора (*Large icons*, *Small icons*) та команду *Stay on top*, що у включеному стані забезпечує статус вікна *Simulink Library Browser* «поверх усіх вікон». Перелічені команди активізуються після відзначення їх прапорцями (галочками). Слід звернути увагу на можливість відкриття тільки основних бібліотек блоків *Simulink* (*Collapse entire Browser*) і повного їхнього комплекту (*Expand entire Browser*), а також на команду *Show parameters for selected block*, що забезпечує відкриття виділеної бібліотеки або блоку.

1.3 Меню вікон *Simulink*-моделей

Меню вікон *Simulink*-моделей відрізняється від меню вікна *Simulink Library Browser* і отримує функції та кнопки, які наведені на рис. 1.10.

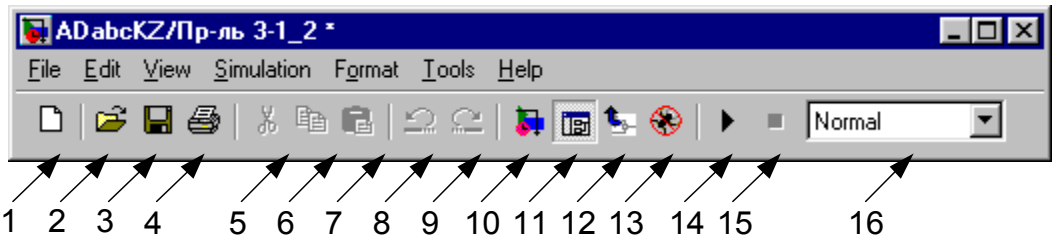


Рис. 1.10. Меню та кнопки вікна *Simulink*-моделі

Перші дев'ять кнопок виконують стандартні операції *Windows* (1 – відкрити нове *Simulink*-вікно, 2 – завантажити існуючу модель, 3 – зберегти модель у файлі, 4 – надрукувати модель, 5-7 – вирізати (5), скопіювати (6) виділений фрагмент моделі в буфер та причепити (7) зміст буфера до активного вікна, 8, 9 – скасувати / відновити результат останнього редагування), а інші – специфічні операції *Simulink*:

10 – *Library Browser* (активізувати *Навігатор Бібліотек*);

11 – *Toggle model browser* (увімкнути *Навігатор Моделі*);

12 – *Go to parent system* (активізувати батьківську модель);

13 – *Debug* (увімкнути засоби налагодження моделей);

14 – *Start simulation* (почати моделювання);

15 – *Stop simulation* (зупинити моделювання);

16 – вибір режиму моделювання.

Установку *popup*-меню 16 не слід змінювати починаючим користувачам.

У полі заголовка вікна рис. 1.10 відображено шлях до моделі у файловій системі. Символ “*” у кінці шляху означає, що після редагування моделі не виконано її запис у файл.

Функція *File* містить стандартні операції роботи з файлами та деякі специфічні функції:

New ▸ – створити вікно для введення нової моделі (*Model ^N*) або бібліотеки (*Library*);

Open... (^O) – відкрити модель;

Close (^W) – закрити модель;

Save (^S) – зберегти модель у колишньому файлі;

Save As... – зберегти модель у новому файлі;

Source Control… – використання зовнішніх систем керування вихідними текстами *MATLAB*-, *Simulink*- та *Stateflow*-програм;

Model Properties ▸ – властивості моделі;

Preferences… – настроювання середовища;

Print...(^P) – вивід на принтер;

Print Setup... – установки принтера;

Exit MATLAB – вихід із системи *MATLAB*.

Функцію *Source Control* не слід підключати без зайвої необхідності та при відсутності достатнього досвіду

Вікно *Model Properties*, яке відкривається при виборі відповідної операції, містить вкладки *Summary*, *Callbacks* і *History*.

Вкладка *Summary* дозволяє зафіксувати творця (*Creator*), дату створення (*Created*) і опис моделі (*Model description*).

Вкладка *Callbacks* дозволяє визначити команди або файли, що будуть автоматично виконані в таких ситуаціях:

Model pre-load function – перед завантаженням моделі,

Model initialization function – при її ініціалізації,

Simulation start function – перед стартом моделювання,

Simulation stop function – після його закінчення,

Model pre-save function – перед збереженням моделі у файлі.

Вкладка *History* містить дані про модифікацію моделі.

Вікно друку моделі *Print Model*, наведене на рис. 1.11, дає можливість, крім звичайних для *Windows* параметрів, визначати глибину роздруківки блоків моделі при наявності в ній підсистем: *Current system*, *Current system and above*, *Current system and bellow*, *All systems*. Зміст цих установок наочно продемонстровано у графічній формі.

Функція *Edit* містить як стандартні *Windows-операції* редагування, так і деякі специфічні операції:

Undo (^Z) – відмінити останню зміну;
Redo (^Y) – відновити останню зміну;
Cut (^X) – переміщення обраних об'єктів у буфер;
Copy (^C) – копіювання обраних об'єктів у буфер;
Paste (^V) – копіювання змісту буфера в обране місце графічного вікна;
Clear (Delete) – вилучення обраних об'єктів;
Select All (^A) – вибір всіх об'єктів в активному вікні
Copy model to clipboard – копіювання моделі в буфер;
Find… – пошук блоків, сигналів, коментарів та інших об'єктів моделі;
Open block – відкрити підсистему;
Mask parameters… – параметри замаскованої підсистеми;
Block parameters… – параметри блоку;
Block properties… – властивості блоку;
Mask subsystem (^M) – маскування підсистеми;
Create subsystem (^G) – об'єднання обраних об'єктів у підсистему (*subsystem*);
Mask subsystem (^M) – маскування підсистеми;
Look undo mask (^U) – заглянути під маску;
Link options ▸ – опції зв'язку підсистеми або замаскованого блоку з бібліотекою;
Update diagram…(^D) – обновити блок-діаграму.

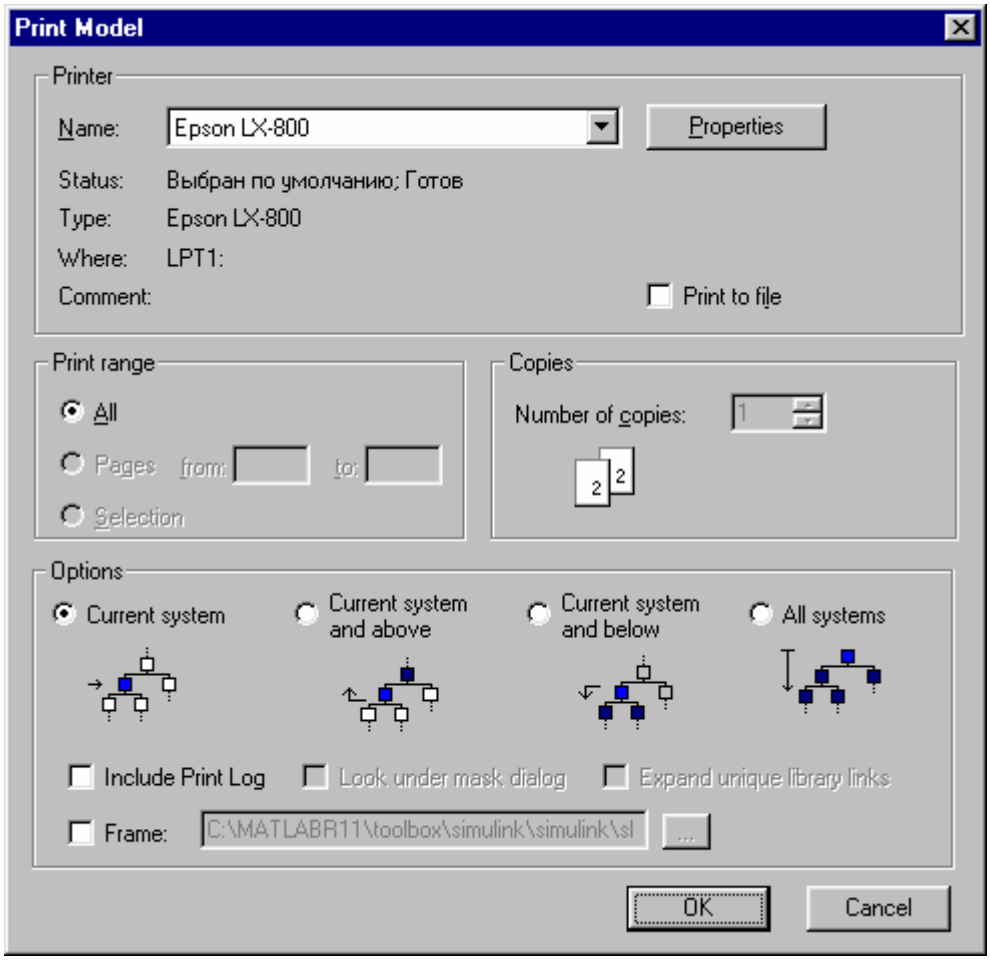


Рис. 1.11. Вікно друку *Simulink*-моделі

Команду *Update Diagram* необхідно використовувати в таких випадках:

- після зміни бібліотечних блоків, копії яких використовуються в моделі;
- після зміни кількості вхідних або вихідних аргументів у *S-функції*, що описує модель.

Функція *View*, крім звичайних для *Windows* прапорців *Toolbar* і *Statusbar* та команд керування масштабом зображення *Zoom in*, *Zoom out*, *Fit system to view* і *Normal* (100%), містить наступні операції:

Go to parent – перехід до батьківської моделі;
Model browser options ▸ – властивості *Навігатора Моделі*;
Block data tips options ▸ – властивості типів даних блоків;
Show Library Browser – перегляд *Навігатора Бібліотек*.

Навігатор моделі відкривається при установці прапорця *Model browser options* ▸ → ✔ *Model browser* або при натисканні кнопки 11 (рис. 1.10). При цьому модель займає праву (більшу) частину вікна, а *Model browser* – ліву. Навігатор моделі відображає її структуру (вкладеність підсистем).

Функція *Format* містить операції форматування моделі. Частина з них діє тільки після виділення фрагмента моделі. Тільки для помічених блоків діють такі операції:

Flip name – змінити положення імені блоку (для горизонтально розташованих блоків воно може розташовуватися вгорі або внизу, а для розташованих вертикально – праворуч або ліворуч),
Hide/Schow name – сховати/показати ім'я блоку,
Flip block (^I) – повернути блок із праворуч на ліворуч або зверху вниз,
Rotate block (^R) – повернути блок на 90° проти годинної стрілки,
Schow/Hide port labels – показати/сховати імена портів (для підсистем);

для виділених блоків і коментарів можна виконати наступні операції:

Text aligment – спосіб вирівнювання тексту в текстовому блоці,
Schow/Hide drop shadow – показати/сховати тінь від блоку
Foreground Color ▸ – установка кольору переднього плану (піктограма й ім'я блоку),
Background Color ▸ – установка кольору заднього плану (фон блоку);

для виділених блоків, ліній і коментарів дійсна операція

Font...– установка шрифту для текстових надписів (для зміни шрифту імені блоку повинний бути виділений блок, а не його ім'я).

Дія інших операцій поширюються на всю модель. До них відносяться

Screen Color ▸ – колір екрана,
Library link dysplay ▸ – відображення зв'язків блоків з бібліотеками,
Sample time colors – установка кольору блоку індикації часу,
Wide nonscalar lines – зображення векторних сигналів стовщеними лініями,
Signal dimensions – відображення розмірності сигналів,
Port data types – виведення інформації про типи сигналів,
Storage class – клас пам‘яті;
Execution order – виведення порядкового номера блоку в послідовності розрахунку їхніх вихідних сигналів при моделюванні.

Якщо фоновий колір блоку не збігається з кольором екрана, то вихідні лінії зв'язку цього блоку приймають колір його фону (*Background*), у противному випадку вони приймають колір піктограми й імені блоку (*Foreground*). Вищесказане відноситься і до контурних ліній блоку.

Функція *Simulation* керує наступними режимами:

Start (^T) – старт моделювання;
Stop – переривання (зупинка) процесу моделювання;
Simulation Parameters...(^E) – установка і редагування параметрів моделювання.

Вікно параметрів моделювання має декілька вкладок (панелей): *Solver*, *Workspace I/O*, *Diagnostics*, *Advanced*, та *Real-Time Workshop.*

Найбільш важливою є панель *Solver* (розв‘язання рівнянь, що описують математичну модель), яка передбачає введення таких параметрів:

Start time – час початку моделювання;

Stop time – час закінчення моделювання;
Type – тип розв‘язання: *Fixed/Variable-Step* (з фіксованим/змінним кроком).

Серед методів розв‘язання диференційних рівнянь (ДР) з фіксованим кроком (*Fixed Step*) в *Simulink* пропонуються методи Рунге-Кутта першого-п‘ятого порядків, а саме:

*ode*5 – метод Рунге-Кутта 5-го порядку (*Dormand-Prince formula*);
*ode*4 – метод Рунге-Кутта 4-го порядку (*fourth-order Runge-Kutta formula*);
*ode*3 – метод Рунге-Кутта 3-го порядку (*Bogacki-Shampine formula*);
*ode*2 – модифікований метод Ейлера (*improved Euler's formula / Heun's method*);
*ode*1 – метод Ейлера (*Euler's method*).

Серед методів розв‘язання ДР зі змінним кроком, що потребують заданої точності, в *Simulink* пропонуються комбіновані методи Рунге-Кутта (4-5)-го і (2-3)-го порядків та деякі методи, більш пристосовані для розв‘язання жорстких ДР:

*ode*45 – комбінований метод Рунге-Кутта (4-5)-го порядку з автоматичним вибором кроку, призначений для розв‘язання нежорстких ДР;
*ode*23 – комбінований метод Рунге-Кутта (2-3)-го порядку з автоматичним вибором кроку, призначений для розв‘язання нежорстких та слабко жорстких ДР;
*ode*113 – багатокроковий метод Адамса-Моултона-Бешфорта (метод прогнозу та корекцій змінного порядку), призначений для розв‘язання нежорстких ДР; може бути більш ефективним, ніж *ode*45, при високій точності;
*ode*15*s* – багатокроковий метод (1-5)-го порядку (за замовчанням 5-го), що базується на формулах чисельного диференціювання (*NDFs*), але може використовувати і формули зворотного диференціювання (*BDFs*) у відповідності з методом Гіра, спрямований на розв‘язання жорстких ДР;
*ode*23*s* – однокроковий метод, що базується на модифікованій формулі Розенброка 2-го порядку; для деяких жорстких ДР виявляється при невисокій точності виявляється більш ефективним, ніж *ode*15*s*;
*ode*23*t* – метод трапецій з використанням „вільного" інтерполянта для розв‘язання помірно жорстких ДР;
*ode*23*tb* – комбінований метод, що використовує на першому етапі формулу трапецій, а на другому –зворотну диференційну формулу другого порядку; як і метод *ode*23*s*, може бути при невисокій точності більш ефективним, ніж *ode*15*s*, для розв‘язання жорстких ДР.

Якщо модель має тільки дискретні змінні стану, або зовсім не має динамічних ланок, то для моделювання використовують метод *discrete* з фіксованим або змінним кроком. Змінний крок обирають, коли модель має у своєму складі дискретні динамічні ланки з різними періодами переривання, або коли треба фіксувати моменти часу перетину деякими сигналами нульового рівня.

При використанні методів *ode*15*s* та *ode*23*s* генерується чисельна матриця Якобі.

При використанні методу *ode*15*s* рекомендовано для підвищення стабільності розв‘язання ДР знизити максимальний порядок формули *NDF*, що задається параметром *Maximum order*, з 5 (за замовчанням) до 2.

Для методів з фіксованим кроком треба задати значення цього параметру в полі *Fixed step size*.

Для методів зі змінним кроком задаються такі параметри:

Max step size – максимальний крок; за замовченням він розраховується за формулою $h_{\max} = (t_{stop} - t_{start})/50$;

Min step size – мінімальний крок;

Initial step size – початковий крок;

Relative tolerance – відносна точність (за замовченням 10^{-3});

Absolute tolerance – абсолютна точність (за замовченням 10^{-6});

Refine factor – коефіцієнт збільшення кількості точок моделювання (ціле число); для методу *ode*45 рекомендовано значення 4, для інших методів – 1.

Підвищити якість візуалізації перехідних процесів за рахунок корекції вектора часу шляхом додавання до нього бажаних точок можна установкою параметра *Output options* у стан *Produce additional output* або *Produce specified output only* та визначивши параметр *Output times*. Більшу кількість точок вихідних сигналів моделі можна отримати і зменшенням максимального кроку чисельного інтегрування але таке рішення менш ефективне з точки зору затрат машинного часу.

Операції функції *Tools* призначені для взаємодії *Simulink* з іншими інструментами пакета *MATLAB*, наприклад,

Data explorer… – перегляд імен, розмірів і типів змінних,

Simulink debugger… – завантаження *Simulink*-відлагоджувача,

Model difference ▸ – відмінність моделей (має два режими: *Merge/Compare two models…* – об‘єднання / порівняння двох моделей і *Compare to last saved model …* – порівняння з останньою записаною моделлю),

Linear analysis… – лінійний аналіз,

Report generator… – завантаження генератора звітів.

Інші операції функції *Tools* не слід застосовувати без особливих на те підстав, тому що вони вимагають високої кваліфікації користувача і використовуються при розв‘язанні нетривіальних задач.

1.4 Меню вікон *Simulink*-бібліотек

Меню вікон *Simulink*-бібліотек має багато спільного з меню вікон *Simulink*-моделей.

Між ними існує така різниця:

- у меню бібліотек відсутні функції *Simulation* та *Tools*;

- у заблокованому стані у вікні бібліотек не можливо виконання операцій редагування, тобто усіх операцій функції *Format*, операцій *Clear*, *Cut*, *Paste*, *Create Subsystem*, *Mask Subsystem*, *Mask Parameters*, *Edit mask*, *Link options*, *Update diagram*, *Undo*, *Redo* функції *Edit* та операцій редагування, які здійснюються за допомогою миші (пересування блоків, зміна їх розміру, назви, тощо);
- для розблокування бібліотек функція *Edit* їхніх вікон має операцію *Unlock Library*.

Стан *Unlock Library* діє з моменту виконання відповідної операції до закриття вікна бібліотеки і поширюється на вкладені бібліотеки. При наступному відкритті бібліотеки вона буде знову заблокованою.

Користувач може редагувати існуючі бібліотеки (у розблокованому стані) та створювати власні.

Для створення нової бібліотеки треба відкрити її вікно через меню будь-якого вікна *Simulink*-моделі, *Simulink*-бібліотеки або *Simulink Library Browser: File* → *New* → *Library*, занести в нього необхідні блоки і закрити зі зберіганням у файлі.

Файли бібліотек, як і файли моделей, мають поширення *mdl.*

1.5 Типи *Simulink*-блоків

Simulink-блоки можуть створюватися різними способами.

Основу стандартних *Simulink*-бібліотек складають вбудовані блоки, які не доступні для перегляду користувачем у вигляді текстового файлу або структурної моделі.

Деякі стандартні блоки створені за допомогою вже існуючих блоків і являють собою або модифікацію деякого блоку з іншими параметрами і зміненою піктограмою, або комбінацію декількох блоків, об'єднаних в одну підсистему. Після імені такого блоку у вікні введення його параметрів стоять позначки (*mask*) і (*link*), що означають, що блок замаскований і зв'язаний з однією з бібліотек. У такий же спосіб може створювати нові блоки і користувач. Заглянувши під маску (*Edit* → *Look under Mask = ^U*), можна побачити в окремому вікні, з чого вони складаються (цю ж операцію можна виконати через контекстуальне меню, що відкривається щигликом правої кнопки миші по піктограмі блоку). Для перегляду і редагування параметрів маскування (*Edit* → *Edit mask…= ^M*) необхідно попередньо тимчасово порушити зв'язок з бібліотекою (*Edit* → *Link options* → *Disable link*). Після перегляду і внесення змін в установки цього вікна зв'язок з бібліотекою можна або відновити (*Restore link*) або перервати (*Break link*). При відновленні зв'язку можливі два варіанти: ігнорувати результати редагування (*Use library block*) чи поширити ці результати на бібліотечний блок (*Update library*). При опису імена цих блоків будуть позначені символом “*”.

S-функції можуть бути написані за особливими правилами на мовах *MATLAB (*.m)*, або *C++ (*.cpp)*, або *C (*.c)*, або *Ada (*.adb)*, або *Fortran (*.f)*, а потім конвертовані в блоки *Simulink*.

1.6 Створення та редагування моделей

Перед початком формування структурної схеми необхідно відкрити для неї нове вікно (*Simulink* → *File* → *New* → *Model (^N)* або *MATLAB* → *File* → *New* → *Model*). Нове вікно має заголовок "*Untitled*". При запису в файл (*File* → *Save / Save as…*) у якості заголовку вікна буде фігурувати призначене користувачем ім'я файла, яке за замовчанням отримає поширення *mdl*.

Вже існуючу модель можна завантажити з меню *Simulink* або *MATLAB* (*File* → *Open*) та із командного рядка *MATLAB* введенням в ньому імені файла, в якому збережена модель, без поширення.

В основу створення та редагування моделей покладено, як і в більшості графічних *Windows*-додатків, принцип *drag-and-drop* (перетягни та покинь).

Копіювання блоків із бібліотек або з будь-якої вже існуючої моделі у вікно створеного файла після їх відкриття виконується мишею за допомогою операції *drag* (тягнути при натиснутій лівій клавіші миші). Аналогічно здійснюється переміщення блоків всередині вікна. Копіювання блоків всередині вікна виконується операцією *drag right* (тягнути при натиснутій правій клавіші миші). При цьому до імені нового блока додається цифра, що відображає порядковий номер копіювання, чим забезпечується унікальність імені кожного блока однієї моделі.

Імена блоків можна змінювати безпосереднім редагуванням. При цьому не можна їх дублювати або залишати блок без імені (ім'я – пустий рядок), але можна сховати ім'я (*Format* → *Hide Name*). Зворотна операція здійснюється командою *Format* → *Show Name*. Для завершення редагування імені треба зробити щиглик мишею зовні поля введення тексту. Після цього ім'я аналізується системою та чи приймається, чи відкидається нею з виведенням повідомлення. Для зміни шрифту імені (*Format* → *Font…*) необхідно попередньо виділити сам блок, а не його ім'я.

В довільній точці активного *Simulink*-вікна, що виділяється щигликом лівої клавіші миші, можна вставити будь-які коментарі (*Annotations*). Введення та редагування коментарю виконуються точно так, як відповідні операції з ім'ям блока. В результаті утворюється новий специфічний блок, який відрізняється від інших блоків тим, що він не має ні піктограми, ні портів, ні вікна введення параметрів, а складається з одного імені, яке є текстом анотації. Рештою цей блок схожий на інші: його можна виділяти, копіювати, пересувати, знищувати і т.п.

Іменами можна відмічати і лінії зв'язку. Для цього необхідно клацнути двічі лівою кнопкою миші по обраній з'єднувальній лінії та ввести текст в утворене поле. Отриманий у такий спосіб надпис буде, на відміну від блока *Note*, прив'язаний до лінії зв'язку.

При пересуванні блока, до якого вже приєднані лінії зв'язку, останні витягуються або скорочуються для збереження зв'язків; кінці зв'язків, не приєднані до блоку, що пересовується не змінюють своє положення. Перемістити блок в межах одного вікна без ліній зв'язку (висунути блок з моделі) можна операцією <*Shift*>+*drag*.

Для виконання якої-небудь дії з будь-яким об‘єктом моделі (блоком, лінією зв‘язку, сукупністю блоків та/або зв‘язків, тобто фрагментом моделі, усією моделлю) його треба спочатку відмітити. Поодинокий об‘єкт виділяється щигликом лівої клавіші миші (*click*). Фрагмент моделі можна виділити у такі способи:

- помітити вибірково блоки або зв‘язки, не відпускаючи клавішу *Shift*;
- помітити підряд, тобто. заключити за допомогою мишки у прямокутник;
- помітити все (*Edit* $\rightarrow$ *Select All=^A*).

Про те, що виділення відбулося, свідчать маленькі нефарбовані квадратики, розташовані в кутах обраних блоків та поблизу кінців обраних зв‘язків.

З поміченим блоком можна виконувати такі операції:

- зміна розміру – *drag* за кут;
- поворот на 90^{O} – *Options* $\rightarrow$ *Rotate* або *^R*;
- поворот на 180^{O} – *Options* $\rightarrow$ *Flip Horisontal* або *^I*;
- знищення – *<Delete>*;
- всі операції функцій *Edit, Options, Style* меню *Simulink*.

Подвійний щиглик по піктограмі блока розкриває текстове вікно для введення його параметрів.

Параметри блока можуть бути подані як в чисельному вигляді, так і у вигляді імен змінних або математичних виразів. В останньому випадку до початку моделювання змінним повинні бути присвоєні значення. Це можна зробити з командного рядка *Matlab* або виконанням командного файла.

Зв‘язки встановлюються між вхідними та вихідними портами блоків. Стрілка на зв‘язку показує напрямок потоку даних. Утворюються зв‘язки у такі способи:

- довільно кусочно-неперервно (*drag* від порту з перериванням цієї операції в бажаних точках зламу);
- з автоматичним розташуванням точок зламу.

Автоматичне розташування точок зламу забезпечується при виконанні операції *drag* від порту до порту без переривання, а також при швидкому з‘єднанні блоків, яке виконується у такий спосіб: помічається початковий блок або блоки, натискається клавіша *Ctrl* і помічається кінцевий блок. Якщо кінцевих блоків декілька, то в першу чергу помічаються саме вони.

Для розгалуження лінії зв‘язку використовують операцію *drag right* або *^ drag left* або *drag* будь-якою клавішею але у зворотному напрямку, тобто від вхідного порту до точки розгалуження.

При проведенні ліній зв‘язку не варто намагатися влучити точно в порт; лінія приєднається до порту, якщо відпустити клавішу миші при находженні графічного курсора всередині блока або поблизу порту (на відстані менше 5 піксел).

З поміченими лініями зв‘язку можні виконувати такі операції:

- зміна напрямку – *Options* $\rightarrow$ *Rerout Lines,* або *^L*;
- паралельне пересування – *drag,* ухопившись за середину сегменту зв‘язку;
- зміна кута між сегментами зв‘язку – *drag*, ухопившись за вузол;

- додатковий злам сегмента зв‘язку – <*Shift*>+ *drag*, ухопившись за бажану точку зламу;
- ділення (розгалуження) зв‘язку – *drag*, ухопившись за першу ліву мітку зв‘язку або *drag right* від будь-якої точки зв‘язку;
- знищення – <*Delete*>.

З поміченими фрагментами моделі або з усією моделлю можна виконувати всі ті операції, що і з окремими блоками та/або зв‘язками.

Для того, щоб зробити розташування об‘єктів моделі більш зручним, вікна *Simulink* мають невидиму сітку 5х5 піксел, до вузлів або до ліній якої прив‘язуються всі об‘єкти. Дискретне пересування виділених об‘єктів можна виконати клавішами <→>, <←>, <↑>, <↓>.

При редагуванні моделі графічний курсор змінює свою форму, сигналізуючи користувачу про ту дію, до виконання якої підготовлена система:

– готовий для наступних дій;

– готовий для пересування сегмента лінії;

– готовий проведення лінії зв‘язку або для нанесення обмежувального прямокутника (виділення групи поруч розташованих об‘єктів);

– для перенесення або для створення нової точки зламу лінії;

– готовий до зміни розміру блока.

1.7 Вибір методу та параметрів моделювання

Для орієнтовного вибору методу та параметрів чисельного інтегрування (ЧІ) ДР можна користуватися такими відомостями та міркуваннями [9]:

1) Методи Рунге-Кутта можуть працювати з постійним кроком h_i та за алгоритмами, що організують автоматичний вибір кроку з умов забезпечення бажаної точності розрахунку ε.

2) Методи прогнозу та корекцій забезпечують бажану точність не за рахунок автоматичного вибору кроку, а за рахунок використання ітераційних формул.

3) Диференційні рівняння, що описують поведінку систем зі сталими часу, що значно відрізняються одна від одної, тобто $(T_{\max}/T_{\min})>>1$, називаються жорсткими (*stiff*). Для зменшення часу їх розв‘язання існують спеціальні методи ЧІ, наприклад, метод Гіра.

4) Постійний крок ЧІ слід обирати при наявності в моделі блоків, що отримують ділянки статичних характеристик з нескінченими коефіцієнтами підсилення у замкнених структурах, а також при моделюванні цифро-аналогових систем з постійними періодами дискретності.

5) Значення постійного кроку ЧІ та бажаної точності (максимально-припустимої похибки) повинно забезпечити компроміс між точністю та часом розрахунку перехідних процесів.

6) Вибір максимально припустимої похибки ЧІ Чим вище порядок p методу ЧІ, тим вище точність розв‘язання диференційних рівнянь при однаковому крокові ЧІ та тим менший крок забезпечує однакову точність:

$$\varepsilon = Ah^p \qquad (1.5)$$

(A – коефіцієнт, значення якого залежить від методу ЧІ, виду та параметрів системи ДР, що розв‘язується).

7) В добре спроектованих, тобто не дуже коливальних, неперервних системах крок ЧІ визначається величиною мінімальної сталої часу досліджуваної моделі. При моделюванні таких систем задовільна точність досягається орієнтовно для методів Рунге-Кутта (4-5)-го порядків при $h_i = T_{\min}/2$, для методів Рунге-Кутта (2-3)-го порядків при $h_i = T_{\min}/10$, і для методу Рунге-Кутта 1-го порядку (метод Ейлера) $h_i = T_{\min}/2$.

8) При моделюванні періодичних процесів на кожному періоді треба розраховувати близько 100 точок.

9) При моделюванні цифро-аналогових систем необхідно слідкувати за тим, щоб фіксовані у часу переключення та всі періоди переривання були кратними кроку ЧІ.

Для того, щоб упевнитись у правильності обраного кроку ЧІ, можна порівняти між собою результати трьох розрахунків перехідних процесів, один із яких виконано з кроком, обраним з урахуванням перелічених в попередньому пункті міркувань, другий – з удвічі меншим, а третій – з удвічі більшим кроком. Якщо всі результати збігаються із задовільною точністю, то обраний крок можна збільшити, інакше його треба зменшити.

При уточненні часу перехідних процесів та часів зміни вхідних сигналів домагаються, щоб у час закінчення розрахунку всі сигнали або їхні амплітуди (при наявності гармонічних складових) досягли своїх усталених значень та щоб ділянки з усталеними значеннями сигналів мали невелику тривалість. Це дасть змогу спостерігати в гарному масштабі саме перехідні процеси, а не усталені режими.

Якщо кількість розрахованих точок перехідних процесів виявляється недостатньою для гарної візуалізації („на око” помітні точки дискретної зміни похідних на графіках), то цю проблему розв‘язують одним з таких способів:

1) зменшують крок або максимально припустимі похибки ЧІ;
2) зменшують порядок метода ЧІ;
3) переходять від методу з постійним кроком ЧІ до методів, що забезпечують бажану точність розв‘язання ДР;
4) виконують інтерполювання вихідних сигналів поліномами другого-третього порядків.

1.8 Моделювання із командного рядка *MATLAB*

Існує три різних рівня використання *Simulink*.

Найбільш простий і наочний спосіб – це моделювання за допомогою меню, описаного у підрозділі 1.3 з фіксацією результатів блоком *Scope*. Робота таким методом доцільна на початковому етапі створення та налагодження моделі.

Другий спосіб складається у використанні функцій моделювання, аналізу і графічних функцій, які або вводяться з командного рядка пакета *Matlab*, або

записуються в командний файл (*script*-файл) та виконуються при запуску цього файла.

Третій спосіб полягає в безпосередньому доступі до *S*-функцій (функцій спеціального формату, що описують модель). При цьому від користувача вимагається достатньо висока кваліфікація.

В даному посібнику обмежимося викладанням першого і другого способів.

При моделюванні із командного рядка необхідно виконати команду:

```
[t, x, y] = sim (′model′, time, options, u)
```

де

model – ім‘я файла, в якому запам’ятовано модель;
time= tfinal / [tstart, tfinal] / [tstart, tout, tfinal]
tstart – початковий час моделювання (за замовчанням 0);
tfinal – конечний час моделювання;
tout – час фіксації результатів;
options – вектор-рядок параметрів моделювання, які установлюються функцією simset;
u – вхідний сигнал;
t – вектор-стовпець часу;
x – матриця змінних стану системи;
y – матриця вихідних змінних.

Функція sim дозволяє запам‘ятати значення не всіх, а тільки потрібних вихідних сигналів, якщо у переліку вихідних параметрів застосувати замість змінної y змінні y1, y2, ... , yn, де число, яким закінчується ім‘я змінної, означає номер вихідного порта.

Функція simset, що встановлює параметри моделювання, має формат:

```
options = simset (name1, value1, name1, value1, ...)
```

При зверненні до simset без аргументів на екран виводиться список імен параметрів, їх можливі значення та значення, що встановлюються за замовчанням:

```
» simset
Solver: [ 'VariableStepDiscrete' |
                'ode45' | 'ode23' | 'ode113' | 'ode15s' | 'ode23s' |
                'FixedStepDiscrete' |
                'ode5' | 'ode4' | 'ode3' | 'ode2' | 'ode1' ]
RelTol: [ positive scalar {1e-3} ]
AbsTol: [ positive scalar {1e-6} ]
Refine: [ positive integer {1} ]
MaxStep: [ positive scalar {auto} ]
InitialStep: [ positive scalar {auto} ]
MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
FixedStep: [ positive scalar ]
OutputPoints: [ {'specified'} | 'all' ]
OutputVariables: [ {'txy'} | 'tx' | 'ty' | 'xy' | 't' | 'x' | 'y' ]
```

```
MaxRows: [ non-negative integer {0} ]
Decimation: [ positive integer {1} ]
InitialState: [ vector {[]} ]
FinalStateName: [ string {''} ]
Trace: [ comma separated list of 'minstep', 'siminfo', 'compile' {''}]
SrcWorkspace: [ 'base' | {'current'} | 'parent' ]
DstWorkspace: [ 'base' | {'current'} | 'parent' ]
ZeroCross: [ {'on'} | 'off' ]
```

Встановлені параметри можна вивести на екран за допомогою функції simget.

1.9 Завдання

1. Вивчити інтерфейс середовища *MATLAB* та додатку *Simulink*, ознайомитися з основними та спеціальними бібліотеками *Simulink*. Випробувати засоби створення та редагування власної моделі.

2. Ознайомитися з основними демонстраціями *Simulink* та демонстраціями поширення *Power System Blockset*, не вдаючись до зайвих подробиць.

3. Ознайомитися з демонстраційною моделлю відповідно з номером варіанта (табл. 1.2) і створити власну бібліотеку, що складається з блоків, перелік яких наведено у табл. 1.2; вивчити способи збереження і редагування бібліотек. Спробуйте виконати моделювання демонстраційної моделі з командного рядка.

У звіті з лабораторної роботи представити:

- демонстраційну модель згідно з табл. 1.2 та моделі її підсистем;
- графіки перехідних процесів, що були отримані за допомогою демонстраційної моделі;
- створену власну бібліотеку блоків;
- власну модель.

Таблиця 1.2.

№ вар.	Демонстраційна модель для вивчення		Перелік стандартних блоків для створення бібліотеки
	Simulink	*Power System Blockset*	
1	*Automotive / Engine timing simulation*	*Power Electronics / Universal Bridge in DC/DC PWM Converter(Discrete)*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Dead Zone*
2	*Automotive / Engine timing eith closed loop control*	*Power Electronics / Three-phase PWM Converters (Discrete)*	*Integrator, Derivative, Memory, Fcn, Look-Up Table, Gain, Product, Sum*
3	*Automotive / Automotive suspension*	*Power Electronics / DC/DC and DC/AC PWM Converters (Discrete)*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Backlash*
4	*Automotive / Hydraulic system models*	*Drives / AC Motor Drive – Vector Control (Discrete)*	*Integrator, Derivative, State-Space, Fcn, Look-Up Table, Gain, Product, Sum*
5	*Aerospace / F-14 flight control simulation*	*Drives / Chopped-fed DC motor drive (Discrete)*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Quantizer*
6	*Aerospace / F-14 digital flight control simulation*	*Drives / Chopped-fed DC motor drive (Continuous)*	*Integrator, Derivative, Fcn, Look-Up Table, Zero-Pole, Gain, Product, Sum*
7	*Aerospace / Missile airframe trim and linearize demonstration*	*Drives / Starting a DC motor*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Rate Limiter*
8	*Aerospace / Radar tracking demonstration*	*Machines / Synchronous Machine and Regulator*	*Integrator, Derivative, Math Function, Look-Up Table, Gain, Product, Sum*
9	*Aerospace / Lunar Module digital autopilot*	*Machines / Machines and Load flow*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Relay*
10	*General / Friction with hard stops model*	*Machines / Stream turbine and Governor System*	*Integrator, Derivative, MinMax, Look-Up Table, Gain, Product, Sum*
11	*General / Counter demonstration*	*Machines / Permanent magnet Synchronous Machine*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Saturation*
12	*General / Inverted pendulum animation*	*Machines / Asynchronous Machine*	*Integrator, Derivative, Rounding Function, Look-Up Table, Gain, Sum*
13	*General / Double spring mass system*	*Machines / Synchronous Machine*	*Integrator, Transfer Fcn, Fcn, Gain, Product, Sum, Coulomb*

	simulation		*& Viscous Friction*

1.10 Методичні рекомендації

1.10.1 Рекомендована послідовність виконання 1-го пункту завдання

1) Завантажити систему за допомогою ярлика, через програмне меню, або через провідник (*Explorer*); ознайомитися з командами *MATLAB*-меню *File*, *Edit*, *View*, *Window*, *Help*, призначенням кнопок панелі інструментів, змістом та призначенням окремих вікон середовища *MATLAB*. Особливу увагу звернути на можливість зміни поточного каталогу (меню *Current Directory* на панелі інструментів) та встановлення доступу до каталогу (*File* → *Set Path*…).

2) Створити власну папку у робочій директорії *work* (розташування та назва узгоджується з викладачем) та зробити її доступною для пакета *MATLAB*.

3) Запустити *Simulink Library Browser* за допомогою кнопки або з командного рядку *MATLAB*; ознайомитися з командами його меню *File*, *Edit*, *View*, *Help* та призначенням кнопок панелі інструментів *Simulink*.

4) Відкрити бібліотеки *Simulink* у графічному вигляді, ознайомитися з ними та спромогтися усвідомити за назвою їхні зміст і призначення. Ознайомитися із блоками, що входять до кожної з бібліотек; особливу увагу приділити блоками бібліотек *Simulink* і *Power System Blockset*.

5) Відкрити вікно для створення нової *Simulink*-моделі, перетягти в нього декілька блоків з *Simulink*-бібліотек. Ознайомитися з командами меню *File*, *Edit*, *View*, *Simulation*, *Format*, *Help Simulink*-вікна та з основними прийомами створення і редагування моделей.

При виконанні останнього пункту спробуйте виконати такі дії: пересування блоку, його розмноження, поворот, зміна розміру, перейменування, приховування та візуалізація імені, зміна кольору фону та контурів блока, зміна розміру шрифту; з‘єднання блоків між собою, розгалуження лінії зв‘язку, вставка незалежних коментарів, та коментарів, прив‘язаних до ліній зв‘язку; запис моделі у створену вами папку.

1.10.2 Рекомендована послідовність виконання 2-го пункту завдання

1) Запустити демонстраційну програму *MATLAB*. Увагу зосередити на демонстраціях розділів *Simulink* і *Blocksets* → *Power System*. Усвідомити загальне призначення кожної з демонстраційних моделей, при цьому не вносити у моделі жодних змін! (Якщо вони випадково були внесені, тоді не слід зберігати модель при її закритті.)

2) Відкрити демонстраційну модель, назва якої наведена у табл. 1.2, та зберегти її копію у власній папці, надавши їй оригінальне ім’я, що відрізняється від імен демонстраційних моделей та стандартних файлів *MATLAB* (наприклад, *ivanov_demo_lab1*). Усі подальші зміни виконувати лише у моделі, що збережена у власній папці! Ознайомитися з призначенням моделі в цілому та її підсистем. Серед останніх виділити замасковані і незамасковані підсистеми і ознайомитися із їхньою структурою. Виконати демонстраційне моделювання процесів в системі, переглянути результати, звернувши увагу на засіб їх фіксації.

3) Ознайомитися з параметрами діалогового вікна керування процесом моделювання (меню *Simulation → Simulation parameters…*), уяснити їхнє значення. Вивчити способи встановлення параметрів окремих блоків моделі. Виконати декілька пробних сеансів моделювання при різних параметрах меню *Simulation → Simulation parameters…* та різних параметрах блоків моделі.

1.10.3 Рекомендована послідовність виконання 3-го пункту завдання

1) Відчинити вікно для побудування власної бібліотеки блоків (меню *File → New → Library*) та мишею перетягти до нього потрібні стандартні блоки *Simulink* (див. табл. 1.2). Розташувати блоки всередині вікна у зручній послідовності, після чого зберегти *mdl*-файл бібліотеки у власній папці з оригінальним ім'ям (наприклад, *ivanov_mylibrary_lab1*).

2) Закрити бібліотеку та відкрити її знову; упевнитися в тому, що тепер вона заблокована для редагування.

3) Розблокувати бібліотеку (*Edit → Unlock library*), залучити до неї новий блок (вибрати довільно), запам'ятати файл та закрити бібліотеку.

При використанні в моделях блоків з власних бібліотек треба знати, що при необхідності роботи з такою моделлю на іншій машині, необхідно копіювати в доступні для папки не тільки модель але й власні бібліотеки.

Другим виходом, що дозволяє обійтися без копіювання бібліотек, є розривання зв'язку блоків з нестандартними бібліотеками перед використанням моделей на інших машинах.

1.11 Контрольні завдання і запитання

1. Поясніть призначення команд меню середовища *MATLAB*, кнопок панелі інструментів та окремих вікон (*Command Window*, *Command History*, *Current Directory* та ін.).

2. Яким чином можна виконати встановлення за власним бажанням поточного каталогу та доступних каталогів *MATLAB*? Навіщо це робити?

3. Поясність призначення команд меню та кнопок панелей інструментів вікон *Simulink Library Browser* та *Simulink*.

4. Дайте загальну характеристику бібліотек та блоків *Simulink* і *Power System Blockset*.

5. Перелічить основні прийоми створення та редагування моделей. Як виконується встановлення потрібних параметрів моделювання, яка мета цього встановлення?

6. За якими правилами і навіщо створюють власні бібліотеки блоків?

7. З яких міркувань обирають метод і параметри чисельного інтегрування при моделюванні?

8. Які засоби моделювання у середовищі *Simulink* Ви знаєте?

9. Як виконати моделювання з командного рядка *MATLAB*?

10. Поясніть призначення параметрів чисельного інтегрування, що встановлюються функцією simset..

2 Лабораторна робота №2
ФОРМУВАННЯ ВХІДНИХ СИГНАЛІВ ТА РЕЄСТРАЦІЯ ВИХІДНИХ СИГНАЛІВ В СЕРЕДОВИЩІ *Simulink*

2.1 Основні засоби формування вхідних сигналів

Вхідні сигнали, що діють на системи автоматичного керування, здебільш є функціями часу. Їх можна розподілити на такі групи:

1) постійні;
2) ступеневі;
3) лінійні;
4) нелінійні неперіодичні, що описуються аналітичними виразами;
5) гармонічні;
6) лінійні періодичні
7) нелінійні періодичні;
8) випадкові.

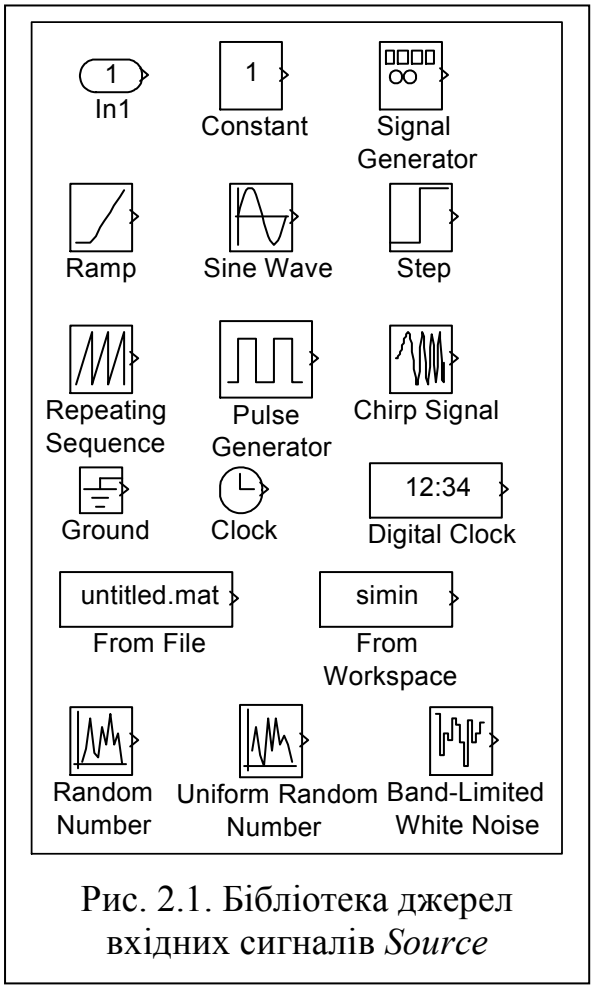


Рис. 2.1. Бібліотека джерел вхідних сигналів *Source*

Основу формування вхідних сигналів складають блоки бібліотеки джерел *Source*. У цю бібліотеку входить група блоків, що не мають вхідних портів, а мають тільки виходи. Їхні піктограми подані на рис. 2.1.

Блоки цієї бібліотеки можна розділити на 3 групи:

1) блоки, що не формують вхідних сигналів, а тільки помічають входи (блок *In*) або заземляють їх (блок *Ground*), щоб при старті моделювання в командному вікні *MATLAB* не виводилося попередження про наявність у моделі неприєднаного вхідного порту (Warning: Input port 1 of block … is not connected...);

2) блоки, що можуть формувати тільки один вихідний сигнал (*Ramp, Clock, Digital Clock* і *Repeating Sequence*);

3) блоки, що можуть формувати як один, так і декілька вихідних сигналів (*Const, Step, Signal Generator, Sine Wave, Pulse Generator, Chirp Signal, Random Number, Uniform Random Number* і *Band Limited White Noise*).

Для того, щоб блоки 3-ої групи формували один вихідний сигнал (скалярний) у вікні настроювання блоку повинен бути встановленим прапорець *Interpret vector parameters as 1-D.* Якщо цей прапорець не встановлено, то блок може формувати одночасно кілька вихідних сигналів, об‘єднаних в одномірний масив (вектор).

Векторні параметри джерел можуть бути задані у вікнах настроювання блоків як рядками, так і стовпцями за правилами алгоритмічної мови системи *MATLAB*.

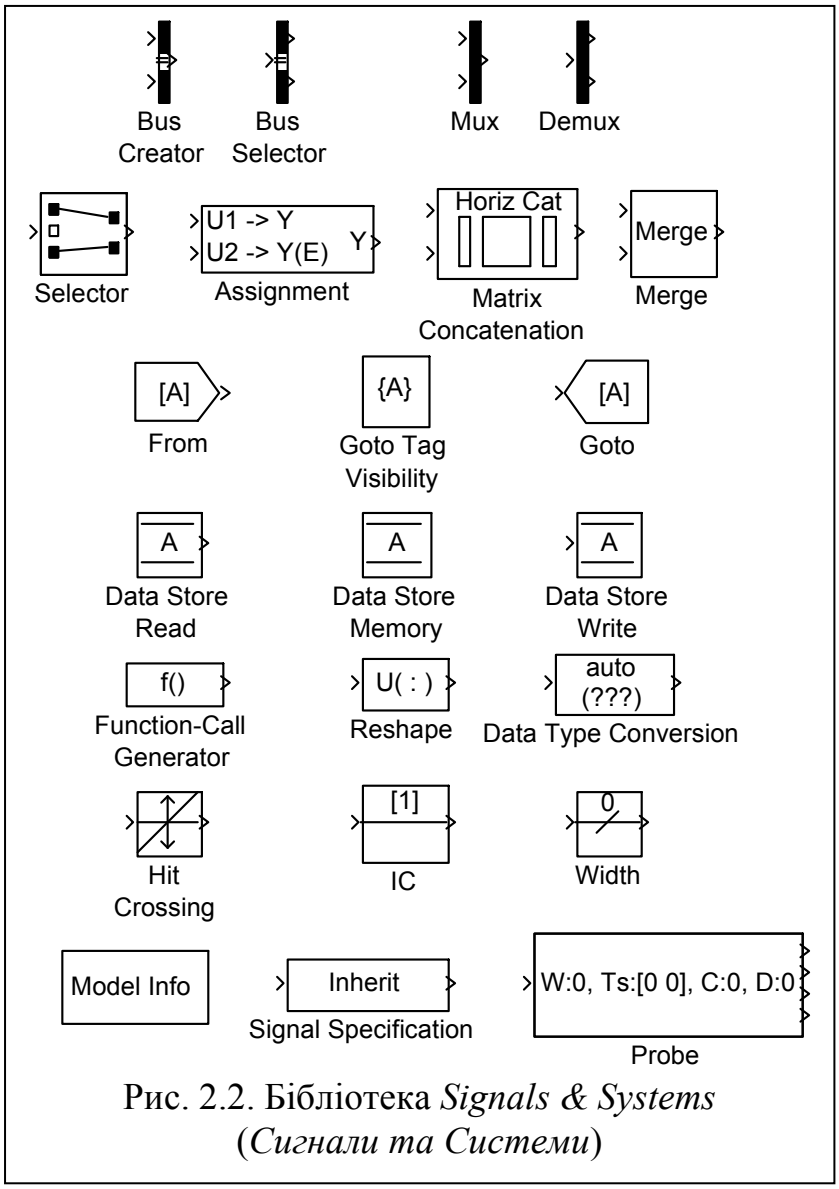


Рис. 2.2. Бібліотека *Signals & Systems* (*Сигнали та Системи*)

Векторний сигнал може бути розділений на скалярні сигнали або на векторні сигнали з меншою кількістю компонентів за допомогою блока *Demux* бібліотеки *Signals & Systems* (див. рис. 2.2). Зворотну операцію виконує блок *Mux*.

Такі джерела як *Step, Sine Wive, Pulse Generator, From File, From Workspace, Random Number* і *Uniform Random Number* можуть змінювати значення своїх вихідних сигналів як неперервно, тобто на кожнім кроці ЧІ, так і дискретно, тобто в моменти часу, кратні періоду дискретності, що задається параметром *Sample Time* (T_S).

Для того, щоб джерело працювало у неперервному режимі, значення цього параметра необхідно покласти рівним нулю (T_S =0).

Якщо джерело повинне формувати дискретний сигнал, то в поле параметра *Sample Time* може вводитися одне додатне значення, що інтерпретується як період дискретності *Ts*, або два додатних значення, перше з яких сприймається як період дискретності *Ts,* а друге – як зсув (запізнення) *offset.* У цьому випадку зміни вихідних сигналів ланок будуть здійснюватися в дискретні моменти часу

$$t_d = nT_s + offset\,; \qquad |offset| \le T_s\,. \tag{2.1}$$

У дискретному режимі кожен блок працює так, як працював би цей же блок у неперервному режимі з приєднаним до його виходу екстраполятора нульового порядку (блок *Zero-Order Hold* бібліотеки *Discrete*) з тим же значенням параметра *Sample Time*.

Якщо установити T_S =-1, то джерело працює в тім же режимі, що і блок, приєднаний до його виходу.

Виходом блоку ***Constant* (*Константа*)** є скалярне постійне значення c, визначене параметром *Constant value* і не залежне від часу $y = c = const$ або вектор констант у вигляді рядка $\mathbf{y} = [c_1 \quad c_2 \quad ... \quad c_n]$ або стовпця $\mathbf{y} = [c_1 \quad c_2 \quad ... \quad c_n]^{\mathrm{T}}$.

Джерело ***Step* (*Стрибок, Сходинка*)** забезпечує стрибкоподібну зміну вихідного сигналу між двома постійними рівнями c_1 (*Initial value*) і c_2 (*Final value*) у заданий момент часу t_3 (*Step time*):

$$y(t) = c_1 + (c_2 - c_1) \cdot 1(t) = \begin{cases} c_1 \text{ при } t \le t_3, \\ c_2 \text{ при } t > t_3. \end{cases} \tag{2.2}$$

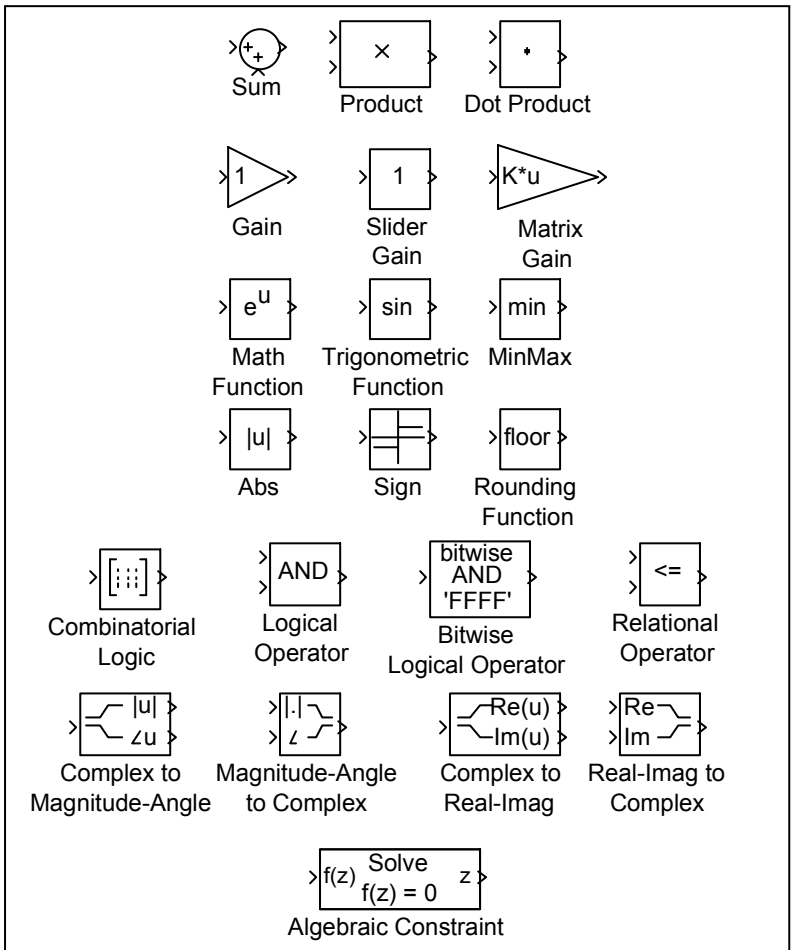


Рис. 2.3. Бібліотека математичних блоків *Math*

Декілька блоків *Step*, приєднаних до входів алгебраїчного суматора *Sum* з математичної бібліотеки *Math* (рис. 2.3) створюють багатосходинковий сигнал.

Блок ***Sum* (*Суматор*)** може мати круглу (*round*) або прямокутну (*rectangular*) форму, яка визначається параметром *Icon shape*.

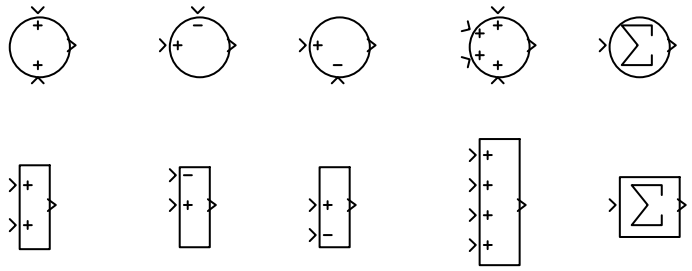
Він підсумовує вхідні сигнали з відповідними знаками, які встановлюються в полі параметра *List of signs* (входи нумеруються зверху вниз для прямокутного блоку та проти годинної стрілки для круглого блоку). Комбінація знаків “+”, ”–“ описує параметри кожного конкретного порту, де кількість портів відповідає кількості знаків, використаних у комбінації. Для пропуску позиції чергового порту у списку знаків використовують символ „|”.

Якщо замість списку знаків визначити кількість входів, то усі вони будуть підсумовуючими. При одному вході блок підсумовує з відповідним знаком елементи вхідного вектора:

$$y = sum(u) = \sum_{i=1}^{k} u_i \ . \tag{2.3}$$

При цьому в піктограмі блоку відображається не знак вхідного сигналу, а

символ Σ. Можливі різновиди блоку в залежності від значень його параметрів наведені на рис. 2.4.



а) *б)* *в)* *г)* *д)*

Рис. 2.4. Блоки *Sum* при таких значеннях параметру *List of signs*:
а) ++; *б)* –+|; *в)* |+–; *г)* 4; *д)* 1

Блок ***Clock* (*Годинник*)** забезпечує відлік і відображення часу моделювання: $y(t) = t$. Параметр *Decimation*, що може приймати цілочисельні додатні значення *d*, має сенс тільки при включеному прапорці *Display time* і при використанні для розв‘язання диференціальних рівнянь методів чисельного інтегрування з постійним кроком *h*. У цьому випадку в піктограмі блоку, що здобуває прямокутну форму, відображаються по черзі чисельні значення часу моделювання, кратні *d*h*.

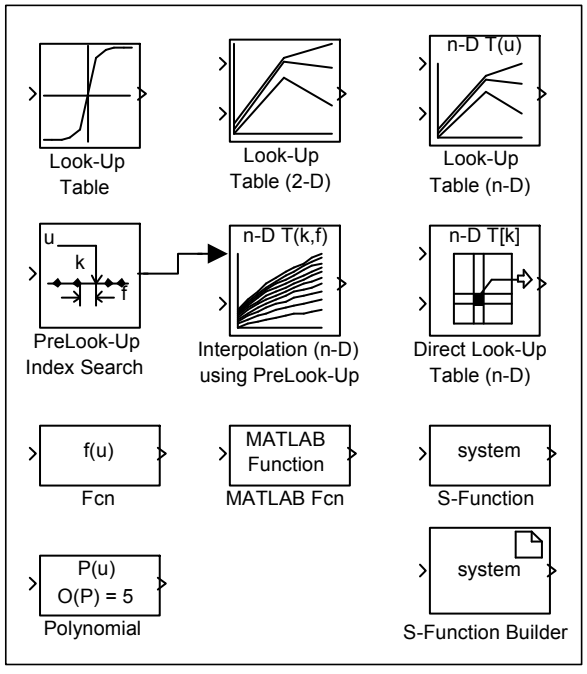


Рис. 2.4. Бібліотека *Functions & Tables*

Цей блок укупі з блоками *Math Function* і *Trigonometric Function* бібліотеки *Math* (див. рис. 2.3) , а також укупі з блоками *Fcn*, *MATLAB Fcn*, *Polynomial, Look-Up Table* бібліотеки *Functions & Tables*, наведеною на рис. 2.4, використовують для формування сигналів з відомим законом їхньої зміни у функції часу.

Приєднанням *Годинника* до блоку *To Workspace* (*У пам'ять*) бібліотеки *Sinks* можна запам'ятати вектор часу з метою використання його згодом для побудови графіків перехідних процесів.

Блок ***Trigonometric Function*** аналогічним способом відтворює елементарні тригонометричні (*sin*, *cos*, *tan*), зворотні тригонометричні (*asin*, *acos*, *atan*, *atan2*), гіперболічні (*sinh*, *cosh*, *tanh*), та зворотні гіперболічні (*asinh*, *acosh*, *atanh*) функції від вхідних сигналів.

Блок-функція ***Polynomial* (*Поліном*)** обчислює значення степеневого полінома, в якому незалежною змінною є вхідний сигнал, а вектор коефіцієнтів задається у полі параметра *Polynomial coefficients*.

Вихідний сигнал блоку ***Fcn*** **(*Функція*)** є аналітичною функцією, при будуванні якої можуть бути використані наступні компоненти:

- - *u*[*i*] – значення і-го вхідного сигналу (*u* без індексу еквівалентно *u*[1]);
- - числа;
- - операції +, -, *, /, ^;
- дужки ( );
- елементарні функції *sin*, *cos*, *tan*, *asin*, *acos*, *atan, sinh, cosh, tanh, exp, ln, log10, sqrt, floor, ceil, abs*, *atan2, rem*;
- операції порівняння ==, ~=, >, <, >=, <=;
- скалярні неіндексовані змінні.

Наприклад, *sin*(4**u*).**exp*(-*u*/*k*), *rem*(*u*[1], *u*[2]), *u*[1]**u*[2]>=*u*[3]+*u*[4].

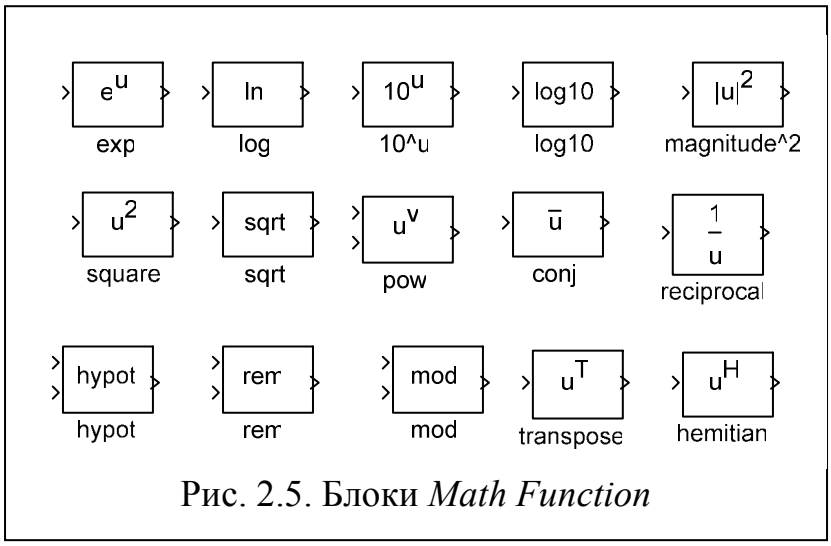


Рис. 2.5. Блоки *Math Function*

Блок ***Math Function*** може відтворювати елементарні одно- та двоаргументні функції від вхідних сигналів. Вибір функції здійснюється через меню. Він впливає на вигляд піктограми блоку. На рис. 2.5 наведені можливі різновидності цього блоку. Для наочності імена блоків замінені іменами відповідних функцій випадаючого меню.

Блок ***MATLAB Fcn*** **(*МАТЛАВ-Функція*)** обчислює задану *MATLAB*-функцію (стандартну або створену користувачем) від вхідного сигналу *u* (скалярного або векторного). Цей блок є пасивним двостороннім інтерфейсом між середовищами *Simulink* і *MATLAB*. Наприклад, якщо задана функція *sin*, то вихідний сигнал *y* буде обчислюватися по формулі *y*=*sin*(*u*). Якщо вхідних сигналів декілька, то вони позначаються як *u*(*1*), *u*(*2*), *u*(*3*) і т.д.

Блок *MATLAB Fcn* підтримує векторні входи і виходи, які перетворюються у скалярні блоком *Demux*, а навпаки – блоком *Mux*. Кількість входів повинна збігатися з кількістю вхідних аргументів *Matlab*-функції, а розмірність виходу (*Output width*) – з кількістю її вихідних аргументів. Якщо користувач задає параметр *Output width* рівним -1, то *Simulink* установлює розмірність вихідного сигналу рівною розмірності вхідного сигналу.

Блок ***Look-Up Table*** **(*Функціональний Перетворювач*)** забезпечує кусочно-лінійну апроксимацію табличної функції, заданої векторами аргументів (*Vector of input values*) і значень (*Vector of output values*) однакової довжини: $y = f(u)$:

$$y = \frac{y_i - y_{i-1}}{u_i - u_{i-1}} \cdot (u - u_{i-1}) + y_{i-1}, \qquad (2.4)$$

i=1...n – номера вузлових точок.

Вектор аргументів повинен бути монотонно зростаючим. Вихідні значення для вхідних сигналів, що попадають за межі інтервалу, заданого вектором аргументів, знаходять за допомогою екстраполювання по першим або останнім двом точкам.

Піктограма блоку *Look-Up Table* відображає графік залежності вхід-вихід. При зміні значень параметрів у діалоговому вікні блоку графік автоматично перемальовується. Якщо ж параметр заданий змінною, значення якої змінюється в робочому середовищі пакета *MATLAB*, то характеристика блоку змінюється відразу, але перемальовується тільки після відкриття його діалогового вікна, або після процесу моделювання.

Блок ***Digital clock* (*Цифровий Годинник*)** формує сигнал, який збігається з часом моделювання у моменти часу, кратні періоду дискретності *Sample Time*. Між цими моментами вихідний сигнал має постійне значення, що дорівнює останньому обмірюваному часу моделювання.

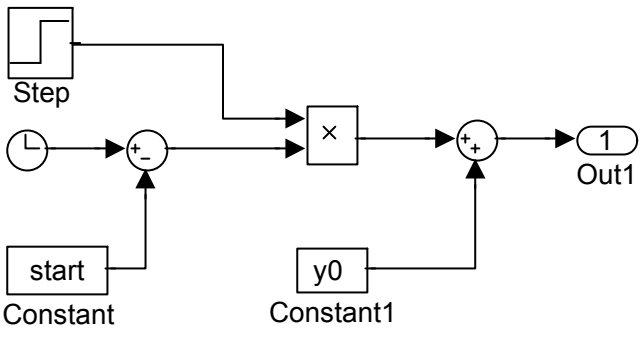


Рис. 2.6. Розгорнута модель блоку ***Ramp***

Джерело ***Ramp* (*Рампа, Скісна Площина*)** формує сигнал, що лінійно змінюється від значення y_0 (*Initial output*), починаючи з моменту часу t (*Start time*) з коефіцієнтом пропорційності k (*Slope*):

$$y(t) = y_0 + k \cdot 1(t - t_S)(t - t_S) \quad (2.5)$$

Розгорнута модель цього блоку приведена на рис. 2.6.

Блок ***Sine Wave* (*Синусоїда*)** може працювати в двох режимах (*Time-Based Mode* і *Sample-Based Mode*), котрі встановлюються у вікні налаштунку блоку за допомогою *pop-up*-меню (*Parameters*, *Sine type*). У режимі *Time-Based* можливе формування як неперервного, так і дискретного вихідного сигналу. У неперервному режимі (T_S=0) блок генерує синусоїду у функції часу відповідно до рівняння:

$$y(t) = A\sin(\omega t + \varphi_0) + y_0, \quad (2.6)$$

де

A – амплітуда (*Amplitude*);

ω – кругова частота, рад/с (*Frequency*);

φ_0 – фазовий зсув, рад. (*Phase*);

y_0 – вертикальний зсув (*Bias*).

У дискретному *Time-Based* режимі (T_S>0) при обчисленні вихідного сигналу використовуються формули:

$$\begin{cases} \sin(t + T_S) = \sin(t)\cos(T_S) + \sin(T_S)\cos(t), \\ \cos(t + T_S) = \cos(t)\cos(T_S) - \sin(t)\sin(T_S), \end{cases} \quad (2.7)$$

які дозволяють формувати вихідне значення набагато швидше, ніж у

неперервному режимі. Недоліком цього методу є нагромадження похибок округлення, тому що для обчислення вихідного сигналу в кожний наступний момент часу використовується значення вихідного сигналу в попередній момент часу. Для ліквідації цього недоліку блок *Sine Wave* передбачає можливість роботи в режимі *Sample-Based*. При цьому значення вихідного сигналу обчислюється за формулою

$$y = A\sin\left(2\pi\left(k + o\right)/\, p\right) + y_0, \qquad (2.8)$$

де

p – кількість розрахункових точок на періоді хвилі синусоїди (*Samples per period*);

o – зсув (відносне фазове зрушення) сигналу (*Number of offset Samples*);

k – цілочисельний параметр, що приймає на періоді синусоїди значення [0, 1, 2, …, p-1]...

Для узгодження методів *Time-Based* і *Sample-Based* необхідно забезпечити наступні співвідношення між їхніми параметрами:

$$p = \frac{2\pi}{\omega T_S}; \qquad o = \frac{p\varphi_0}{2\pi}. \qquad (2.9)$$

Ліквідуючи нагромадження похибок округлення, формула (2.8) має схований недолік, який виявляється при використанні цього блоку в підсистемі, що виконується за умовою (*Conditionally Executed Subsystem*). Цей недолік полягає в можливості неузгодженої роботи блоку *Sine Wave* з іншими блоками моделі після виконання підсистемою паузи з наступним поновленням її роботи. Отже, при виникненні потреби формування дискретної синусоїди в складі підсистеми, виконуваної за умовою, блок *Sine Wave* необхідно використовувати в режимі *Time-Based*.

Блок ***Pulse Generator* (*Генератор Прямокутних Імпульсів*)** також може працювати в неперервному (*Time-Based*) або дискретному (*Sample-Based*) режимах, вибір одного з яких здійснюється за допомогою параметра *Pulse Type*. В обох режимах блок формує періодичний сигнал, що складається з однополярних прямокутних імпульсів. В неперервному режимі він має наступні параметри:

ht – висота імпульсу (*Amplitude*);

T – період в одиницях виміру часу (*Period, sec)*);

du – ширина імпульсу у відсотках від періоду (*Puls width (in % of period)*);

stt – час початку пульсацій (*Phase delay, sec*).

У дискретному режимі у вікні настроювання блоку з'являється додатковий параметр *Sample Time* , а період і час початку пульсацій задаються у відносних одиницях (частках періоду дискретності – *Number of samples*).

Якщо при моделюванні обрано метод розв‘язання диференціальних рівнянь з фіксованим кроком, то *Simulink* автоматично використовує блок *Pulse Generator* у режимі *Sample-Based* з періодом дискретності, рівним кроку інтегрування. У цьому випадку крок інтегрування повинний бути обраний таким, щоб параметри *T, du*T/100* і *stt* були кратними кроку.

Блок ***Repeating Sequence* (*Повторювана Послідовність*)** виконує

повторення циклу, заданого таблицею двох параметрів: *Time values* (*tv*) – вектор часу і *Output values* (*yv*) – вектор вихідних сигналів. Вектор часу повинен бути монотонно зростаючим. Різниця між значеннями його останнього і першого елементів визначає період вихідного сигналу: $T = tv(end) - tv(1)$. Блок коректує свою піктограму відповідно до вигляду вихідного сигналу. Він реалізований за допомогою замаскованої підсистеми, зображеної на рис. 2.7.

Блок *Fcn* моделі рис. 2.7 визначає залишок від цілочисельного ділення часу моделювання на період, а блок *Look-Up Table* виконує кусочно-лінійну апроксимацію табличної функції, заданої параметрами *Time values* і *Output values*.

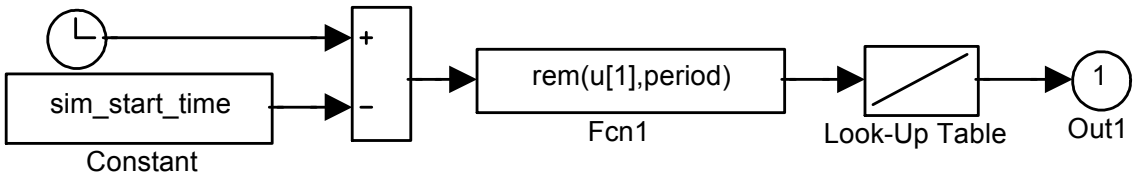


Рис. 2.7. Модель замаскованої підсистеми *Repeating Sequence*

Джерело ***Chirp Signal* (*Синусоїда зі Змінною Частотою*)**, реалізоване за допомогою замаскованої підсистеми рис. 2.8, формує синусоїду з одиничною амплітудою і частотою, що змінюється за лінійним законом:

$$y = \sin(At^2 + Bt), \qquad (2.10)$$

де

$B = 2\pi f_1$, $A = 2\pi(f_2 - f_1)/T$,

за вхідними параметрами:

f_1 (*Initial frequency*, *Hz*) – початкова частота [Гц];

T (*Target time*, *sec*) – заданий час [сек];

f_2 (*Frequency at target time* [*Hz*]) – значення частоти в заданий момент часу [Гц].

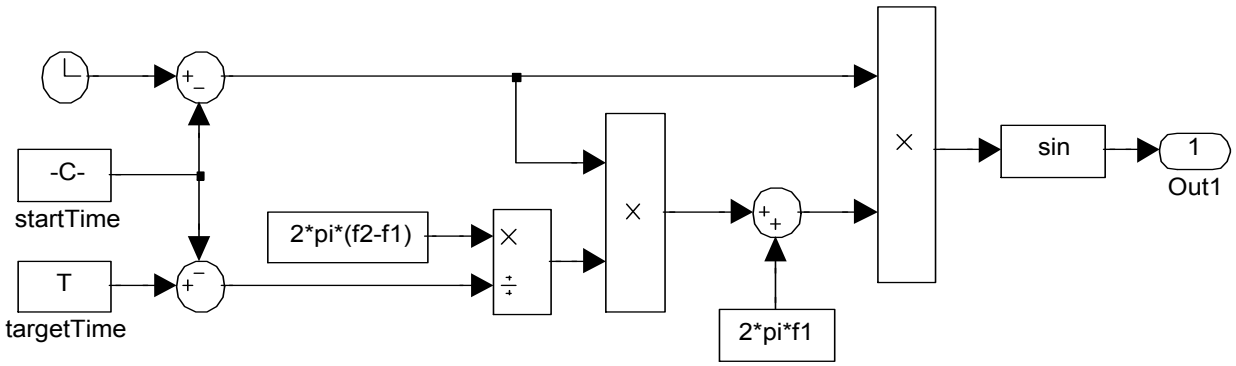


Рис. 2.8. Модель блоку *Chirp Signal*

Піктограма джерела зображує графік функції $y = \sin(t^2)$.

Комплексне джерело ***Signal Generator* (*Генератор Сигналів*)** генерує один з 4-х сигналів: синусоїду (*Sine*), прямокутний (*Square*) періодичний сигнал зі шпаруватістю 50%, пилкоподібний (*Sawtooth*) періодичний сигнал, а також випадковий сигнал (*Random*).

Тип сигналу вибирається за допомогою випадаючого меню *Wave form*. Чисельними параметрами є амплітуда *A* (*Amplitude*) і частота (*Frequency*). Частота

може задаватися як у Гц (*Hertz*), так і в рад/с (*rad/sec*) за допомогою параметра *Units*. Значення всіх сигналів змінюються від $+A$ до $-A$. Вихідні установки блоку можуть бути змінені протягом процесу моделювання, що особливо ефективно у сполученні з фіксацією результатів блоком *Scope* (*Осцилограф*), коли реакцію системи на різні типи вхідного сигналу потрібно одержати швидко.

Джерела випадкових чисел з нормальним **(*Random Number*)** та рівномірним **(*Uniform Random Number*)** розподілами генерують відповідно нормально розподілений (Гаусовський) і рівномірно розподілений випадкові сигнали для деяких заданих стартових чисел *Initial seed*, які ініціалізують генератори випадкових чисел.

Специфічними параметрами ланки *Random Number* є середнє значення сигналу *Mean* і середньоквадратичне відхилення *Variance*, а ланки *Uniform Random Number* – мінімальне *Minimum* і максимальне *Maximum* значення випадкового сигналу. Обидва блоки можуть працювати як в неперервному, так і у дискретному режимах.

Джерело ***Band-Limited White Noise* (*Білий Шум з Екстраполяцією*)** забезпечує формування "білого шуму" для неперервних систем за допомогою послідовно з'єднаних блоків *Random Number* з параметрами *Mean* = 0, *Variance* = 1 та *Gain* (*Пропорційна ланка*) на основі таких показників:

Cov (*Noise Power*) – потужність шуму (коваріація);
T_s (*Sample Time*) – період дискретності;
Seed – стартове число генератора випадкових чисел.

Коефіцієнт підсилення ланки *Gain* обчислюється за формулою

$$k = \sqrt{Cov} \big/ \sqrt{T_s} \,. \qquad (2.11)$$

Вектори *Noise Power* і *Seed* можуть бути однієї довжини. Для більш швидкого протікання процесу моделювання необхідно параметр *Sample Time* установлювати максимально великим, узгоджуючи однак його величину з мінімальної сталою часу системи.

При визначенні потужності шуму і періоду дискретності треба задовольнити умовам $Cov > 0$ і $T_s \neq -1$.

Блок ***In* (*Вхідний Порт*)** помічає входи моделі. Якщо у вкладці *Workspace I/O* вікна *Simulation Parameters* установити прапорець *Input* функції *Load from Workspace*, то зовнішній вхідний сигнал можна задавати в полі параметру *Input* у такий же спосіб, як за допомогою блоку *From Workspace*.

Основним параметром блоку *In* є номер порту (*Port number*), що відображається в його піктограмі. Усі порти в межах одного вікна повинні мати послідовну нумерацію. Номера портів формуються автоматично в порядку їхньої появи у вікні. При видаленні порту з вікна порти, що залишилися в ньому, перенумеровуються таким чином, щоб у послідовності номерів не було пропусків. Номера портів можна змінювати і вручну.

Параметром *Port dimensions* можна визначити розмірність вхідного сигналу. Значення за замовчуванням (-1) відповідає автоматичній установці цього параметра шляхом успадкування його від пов'язаного з ним блока.

Тип сигналу (*real, complex*), тип представлення даних (*double, single, int8, ..., boolean*), прапорець *Interpolate data*, а також параметр *Sample time* мають сенс тільки при використанні зовнішніх джерел у моделях верхнього рівня.

Блок ***From Workspace* (*З пам'яті*)** забезпечує читання даних з матриці, що повинна мати два або більш стовпців. Перший стовпець повинний містити монотонно зростаючі значення часу, а кожен додатковий – табличні значення функцій часу:

$$\begin{bmatrix} t_1 & y_1 & z_1 & \dots & w_1 \\ t_2 & y_2 & z_2 & \dots & w_2 \\ \dots & \dots & \dots & \dots & \dots \\ t_k & y_k & z_k & \dots & w_k \end{bmatrix}. \qquad (2.12)$$

Наприклад, для того, щоб блок *From Workspace* сформував на інтервалі $t \in [0, 2\pi]$ сигнал $y(t) = \sin(t) + \cos(t^2)$, у середовищі *MATLAB* до моделювання необхідно виконати послідовність операцій

```
Time = (0 : pi/50 : 2*pi)';
Out=sin(Time)+cos(Time.^2);
D=[Time  Out];
```

і установити в поле параметра *Data* у вікні настроювання блоку *From Workspace* ім'я матриці *D*.

Значення часу використовуються блоком для обчислення вихідного сигналу за допомогою лінійної інтерполяції або екстраполяції.

Піктограма блоку відображає ім'я матриці, з якої зчитуються його дані.

За допомогою випадаючого меню *Form output after final data value by* можна вибрати один з чотирьох способів зміни вихідного сигналу після перевищення часом моделювання значення останнього елемента першого стовпця матриці даних (максимального значення аргументу табличної функції):

Extrapolation – лінійна екстраполяція;
Setting To Zero – установка в нуль;
Holding Final Value – затримка останнього табличного значення;
Cyclic Repetition – циклічне повторення.

Для використання першого способу прапорець інтерполяції *Interpolate data* повинний бути обов'язково встановлений, а для останнього способу – скинутий. Для інших способів стан цього прапорця не має значення. При останньому способі (*Cyclic Repetition*) дані, що зчитуються, повинні мати формат не матриці, а структури. Наприклад, щоб розглянутий у попередньому прикладі сигнал циклічно повторювався при $t > 2\pi$, необхідно сформувати таку структуру даних:

```
D.Time = (0 : pi/50 : 2*pi)';
D.signals(1).Out=sin (Time)+cos (Time.^2);
D.signals(1).demensions = 1;
```

Останній параметр визначає розмірність вихідного сигналу.

Джерело ***From File* (*З Файлу*)** забезпечує читання даних з файлу, чиє ім'я

занесено в поле параметра *Filename* і відображається в піктограмі блоку.

Файл даних повинен мати розширенням *mat* (**.mat*) і містити матрицю з двома або більш рядками і мінімум двома стовпцями. Перший рядок інтерпретується як монотонно зростаючі значення часу, а інші рядки – як відповідні цим моментам часу значення вихідних сигналів джерела:

$$\begin{bmatrix} t_1 & t_2 & \dots & t_k \\ y_1 & y_2 & \dots & y_k \\ z_1 & z_2 & \dots & z_k \\ \dots & \dots & \dots & \dots \end{bmatrix}. \qquad (2.13)$$

Запис даних у *mat*-файл у пакеті *MATLAB* виконує команда

```
save FileName var1 var2 …
```

Наприклад,

```
save data1 D
```

збереже значення змінної *D* у файлі *data1.mat.* Для визначення вихідних сигналів у моменти часу, відсутні у файлі даних, використовується лінійна інтерполяція або екстраполяція. Файл даних для блоку *From File* може бути сформований блоком *To File* при моделюванні іншої системи.

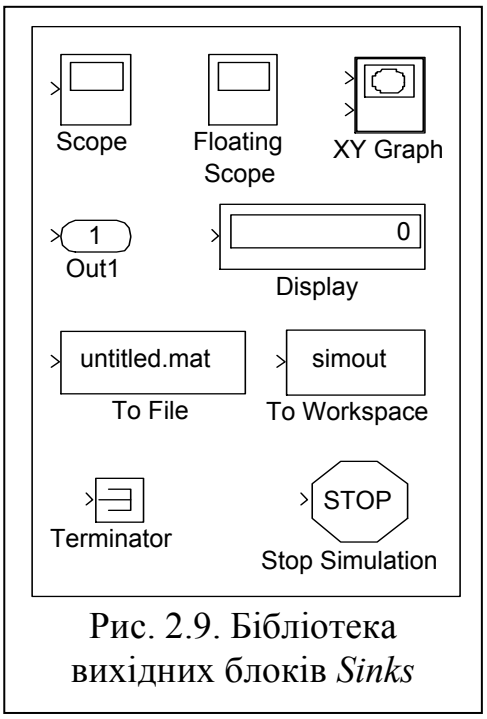


Рис. 2.9. Бібліотека вихідних блоків *Sinks*

2.2 Основні засоби реєстрації сигналів

Для реєстрації сигналів у *Simulink* існує бібліотека *Sinks*, блоки якої подані на рис. 2.9.

Більшість з них призначені для запам'ятання результатів моделювання (*To File, To Workspace*), а також для їх графічного (*Scope, Floating Scope, XY Graph*) або чисельного (*Display*) відображення. Усі вони мають тільки входи і не мають виходів.

Блок ***To Workspace*** **(*У Пам'ять*)** запам'ятовує дані, що надходять на його вхідний порт у змінній з ім'ям, завданим у поле параметра *Variable name.*

Параметр *Decimation* (*Проріджування*) визначає дискретність запам'ятовування або відображення інформації і може приймати тільки позитивні цілочисельні значення *d*, що вказують через скількох кроків чисельного інтегрування варто фіксувати результати моделювання. При *d*=1 (значення за замовчуванням) блоком запам'ятовується інформація, отримана на кожнім кроці розрахунку перехідних процесів, а при *d*=5 – тільки інформація, отримана на кожному п'ятому кроці. При інтегруванні з постійним кроком параметр *d* – це відношення кроку фіксації до кроку чисельного інтегрування: $h_{out} = dh_i$.

Якщо в процесі моделювання для розв‘язання диференціальних рівнянь обрано метод ЧІ зі змінним кроком, а крок фіксації повинний бути постійним, то

для дискретизації виведення замість параметра *Decimation* варто використовувати параметр *Sample Time*, обираючи його значення рівним бажаному кроку реєстрації: $T_s = h_{out}$. При $T_s = -1$ блок реєстрації успадковує період дискретності від блоку, вихідні сигнали якого він фіксує.

Параметри *Limit data point to last* і *Save format* визначають максимальну кількість точок для збереження реєстрованого сигналу (відлік ведеться від останньої розрахованої точки) і формат запам'ятовування даних. Для того, щоб не втратити інформацію, значення параметра *Limit data point to last* можна установити рівним ∞ (константа *Inf*).

Інформація може бути збережена в наступних форматах: *Structure with time* (*Структура з часом моделювання*), *Structure* (*Структура без часу моделювання*) і *Array* (*Масив*).

Найпростішим типом даних є *Array*. У цьому випадку кожен сигнал записується в окремий стовпець матриці. Один рядок матриці містить стан усіх сигналів у конкретний момент часу. Кількість рядків при *d*=1 дорівнює числу кроків моделювання. Якщо дані збережені в матриці *y*, то виділити з неї *j*-ий сигнал можна операцією *y*(*:,j*) , а всі сигнали в *i*-й зареєстрований момент часу – операцією *y*(*i,:*).

Якщо дані збережені у змінній *y*, що має формат *Structure with time*, то вектор-стовпець часу визначається звертанням до поля *time* змінної *y* (*y.time*), звертання *y.blockName* формує шлях до блоку (ім'я-моделі / ім'я підсистеми / ім'я-блоку), звертання *y.signals.dimensions* – кількість вхідних сигналів, звертання *y.signals.label* – мітку лінії зв'язку, а звертання *y.signals.values* – чисельні значення матриці вхідних сигналів, організованої так само, як при використанні формату *Array*. Структура без часу (*Structure*), відрізняється від структури з часом тим, що її поле *time* має значення порожньої матриці [].

Усякий раз при старті процесу моделювання дані, що зберігаються, обновляються, так що цей блок не може об'єднати результати кількох послідовних розрахунків. Результати не доступні в робочому середовищі доти, поки не закінчено або не зупинено розрахунок перехідних процесів.

Блок ***Scope*** **(*Осцилограф*)** у процесі моделювання відображає вихідні сигнали приєднаних до нього блоків. Для того, щоб побачити графіки перехідних процесів, необхідно попередньо відкрити його вікно (див. рис. 2.10), що поряд з полем виведення графіків (на відміну від попередніх версій їх може бути декілька) містить наступні «кнопки» (нумерація зліва направо):

1 – *Print* – друк;
2 – *Parameters* – настроювання параметрів;
3 – *Zoom* – рівномірна зміна масштабу по обох осях;
4 – *Zoom X-axis* – зміна масштабу по осі *X*;
5 – *Zoom Y-axis* – зміна масштабу по осі *Y*;
6 - *Autoscale* - автоматичне масштабування;
7 – *Save current axes settings* – запам'ятовування системи координат;
8 – *Restore saved axes settings* – відновлення системи координат (наприклад,

після зміни масштабу однієї з клавіш *Zoom*);

9 - *Floating Scope* - плаваючий осцилограф;

10 – *Lock/Unlock axes selection* – заблокувати/розблокувати вибір ліній зв'язку для плаваючого осцилографа;

11 - *Signal Selection* - вибір сигналів для плаваючого осцилографа.

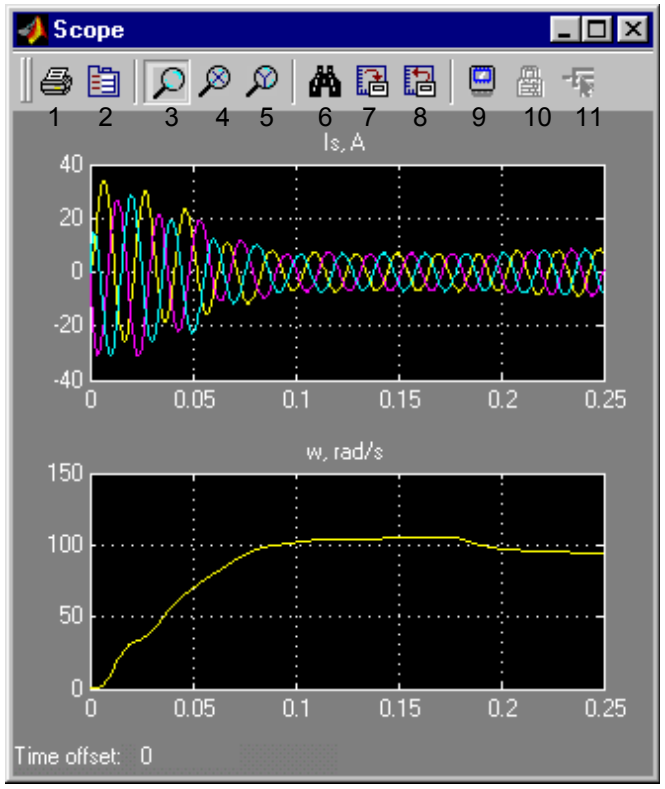


Рис. 2.9. Вікно блока *Scope*

Звичайно після закінчення процесу моделювання для того, щоб побачити графіки в нормальному масштабі, натискають кнопку *Autoscale*, а потім запам'ятовують діапазони систем координат кнопкою *Save current axes settings*.

Якщо результати автоматичного масштабування за якимись причинами не влаштовують користувача, то він може змінити межі координатних осей графіків вручну за допомогою кнопок 3-5 візуально або установкою границь системи координат у чисельному вигляді.

Масштабування зображення в обраному виміру кнопками можна виконувати двома способами. При першому з них після активізації відповідної кнопки курсор миші підводять до бажаної ділянки графіка і клацають на ньому лівою клавішею. При кожнім натисканні масштаб буде збільшуватися, що приведе до відображення у вікні всі меншого і меншого фрагмента графіка. При другому способі фрагмент графіка, який бажають збільшити, оточують прямокутником за допомогою курсору. Повернення до результатів попереднього масштабування виконується командою контекстуального меню *Zoom out*, а повернення до вихідного масштабу – кнопками *Autoscale* або *Restore saved axes settings*.

Для установки меж графіка по осі ординат у чисельному вигляді необхідно викликати контекстуальне меню щигликом правої кнопки миші в площині системи координат, вибрати з нього функцію *Axes Properties* і у вікні, що відкрилося, установити *параметри Y-min* і *Y-max*. У цьому ж вікні можна установити заголовок системи координат (параметр *Title*), замінивши коментар *%<SignalLabel>* бажаним текстом або вставивши цей текст перед коментарем.

Діапазон часу, що є спільним для всіх систем координат, можна змінити у вікні *Parameters*, що відкривається однойменною кнопкою й утримує 2 вкладки: *General* і *Data history.*

За допомогою вкладки *General* установлюються такі параметри:

- кількість систем координат (*Number of axes*), що визначає і кількість вхідних

портів;

- діапазон часу (*Time range*);
- режим візуалізації чисельних значень міток координатної сітки (*Tick labels*), що може приймати значення *all* (візуалізації усіх значень), *bottom axis only* (візуалізації значень часу тільки в самій нижній системі координат) і *none* (значення міток не виводяться, і графіки займають максимально можливу площу вікна);
- прапорець *floating scope*;
- параметри *Decimation* або *Sample Time*, що визначають дискретність відображення інформації (див. опис блока *To Workspace*).

За допомогою вкладки *Data history* назначаються параметри *Limit data point to last*, *Variable Name* і *Save format*. Два останніх параметри активізуються тільки при встановленому прапорці *Save data to workspace* (зберегти дані в оперативній пам'яті). При установці цього прапорця блок *Scope* починає виконувати, крім власних функцій, функції ланки *To Workspace*. За замовчанням для запису даних пропонується змінна у форматі *Structure with time*. Для збереження інформації багатоканальним *Осцилографом* не можна використовувати формат *Array*. Поле *signals* змінної *ScopeData* у цьому випадку є вектором структур, з кількістю елементів, рівним числу портів (каналів) *Осцилографа*. На додаток до полів *dimensions*, *values* і *label* кожен елемент змінної *signals* має ще поля *title* і *plotStyle*, які містять інформацію про заголовки графіків і стилі їхнього зображення.

Параметри *Осцилографа* можна редагувати в процесі моделювання.

Розмір і пропорції вікна *Scope* можна змінювати довільно.

Осцилограф, зображений на рис. 2.9, має 2 вхідних порти, на перший з який надходить векторний сигнал, а на другий – скалярний. Скалярний сигнал зображується завжди жовтим кольором, а для складових векторного сигналу використовуються повторювані циклічно 6 кольорів: *жовтий, малиновий, блакитний, червоний, зелений, синій*. Фон зображення завжди *чорний*.

Відсутність можливості змінювати колір фону, кольори і стиль графіків, що особливо незручно при необхідності їх друкування, можна віднести до недоліків блоку *Scope*.

Блок ***Floating Scope* (*Плаваючий Осцилограф*)** можна взяти з бібліотеки або переключити звичайний осцилограф *Scope* у режим, що плаває, відповідною кнопкою або установкою відповідного прапорця.

Floating Scope не має вхідних портів. Підключення до нього сигналів, що потребують реєстрації, можна виконати двома способами. Перший спосіб полягає в активізації однієї із систем координат *Плаваючого Осцилографа* з наступним вибором однієї або кількох ліній зв'язку моделі, як це описано в підрозділі 1.6. Активізація системи координат виконується щигликом лівої кнопки миші в полі цієї системи, що приводить до автоматичного вмикання кнопки *Lock/Unlock axes selection* у стан *Unlock* і до оточування активної системи координат синьою смугою. При другому способі вибір сигналів, що підключаються, здійснюється у *вікні Signal Selector*, що відкривається кнопкою 11. *Signal Selection*, завдяки

наявності в ньому засобів навігації, дозволяє вибрати будь-які сигнали системи, включаючи внутрішні сигнали закритих підсистем. На відміну від блоку *Scope*, плаваючий осцилограф не має буфера для збереження зареєстрованих даних. Тому для нього не можливі режими масштабування графіків (кнопки 3-6 не працюють).

Перевагою плаваючого осцилографа є економія оперативної пам'яті.

Блок ***XY Graph*** **(*XY-графіка*)** призначено для побудови фазових портретів, тобто для зображення одного із запам‘ятованих сигналів у функції іншого (а не у функції часу). Його вхідними параметрами є координати меж графіка : *x-min, x-max, y-min* і *y-max*, а також параметр дискретизації *Sample time*.

Обидва входи блоку є скалярними. При моделюванні *Simulink* автоматично відкриває графічне вікно *MATLAB* і динамічно відображає в ньому задану користувачем графічну залежність.

Графобудівник створений на основі *S*-функції *sfunxy*. Для демонстрації роботи цього блоку можна з командного рядка завантажити демонстраційну модель *lorenzs*.

Реєстратор ***To File*** **(*У Файл*)** записує часи моделювання і вхідні сигнали в *mat-файл* з ім'ям *File Name*, який має структуру матриці, що містить у першому рядку вектор часу, а в інших рядках – відповідні йому вектори вхідних сигналів. Ім'я матриці задається параметром *Variable name*. Цей формат ідентичний формату даних блоку *From File* (див. формулу (2.12)), що забезпечує сумісність файлів. Для дискретизації сигналів так само, як у блоці *To Workspace*, використовуються параметри *Decimation* і/або *Sample time*.

При кожнім старті процесу моделювання файл даних обновлюється.

Якщо якісь частини системи (наприклад, різні задавальні пристрої) працюють незалежно одна від одної, то результати моделювання цих підсистем можна записати у файли і зчитувати ці данні з файлів при моделюванні частини системи, що залишилася. При багаторазовому моделюванні це дозволить скоротити час розрахунку перехідних процесів.

Цей блок також рекомендується використовувати в тому випадку, коли для запам'ятовування результатів моделювання не вистачає оперативної пам'яті або коли моделювання займає занадто багато часу. Отриманий файл результатів моделювання можна потім використовувати для побудови графіків за допомогою послідовно з'єднаних блоків *From File* і одного з пристроїв, що реєструють, наприклад, *Scope*.

Піктограма блоку відображає ім'я файлу, у який записуються матричні дані.

Матриця, записана у файл, стає доступною в середовищі *Matlab* після завантаження файлу командою *load FileName* у вигляді значення змінної, ім'я якої визначено в полі параметру *Variable name*.

Блоками ***Outport*** **(*Вихідний Порт*)** помічають виходи моделей і підсистем.

У підсистемах вони необхідні для зв'язку підсистеми з моделлю більш високого рівня, а в моделях вищого рівня – для використання зовнішніх вихідних сигналів за допомогою опцій функції *sim* (при моделюванні з командного рядка)

або за допомогою функції *Save to workspace* вікна *Simulation Parameters* (вкладка *Workspace I/O*) при встановленому прапорці *Output*, а також для інформування додатків, призначених для аналізу і синтезу систем автоматичного керування (наприклад, *Control Toolbox*) про можливі точки знімання вихідних сигналів.

Нумерація вихідних портів і їхнє відображення в піктограмі згорнутої підсистеми відбувається точно так, як і для вхідних портів *In*.

Скалярні і векторні сигнали, що надходять на вихідні порти можуть мати як дійсні, так і комплексні значення.

Параметри *Output when disabled* і *Initial output* мають значення тільки в підсистемах, виконуваних за умовою.

Блок ***Display* (*Дисплей*)** застосовується для виведення чисельних значень сигналів у заданому форматі, що вибирається за допомогою параметра *Format* і може приймати наступні значення:

-*short* – 4 значущі цифри у форматі з фіксованою крапкою;

-*long* – 14 значущих цифр у форматі з фіксованою крапкою;

-*short-e* – 5 значущих цифр у мантисі чисел із плаваючою крапкою;

-*long-e* – 16 значущих цифр у мантисі чисел із плаваючою крапкою.

Скалярні і векторні сигнали, що надходять до блоку, можуть мати як дійсні, так і комплексні значення.

Якщо розмір блоку *Display* малий для відображення всієї інформації, то в його правому нижньому куті з'являється чорний трикутник, що вказує, у якому напрямку треба розтягти піктограму блоку.

Так само, як і *Осцилограф*, *Дисплей* може працювати в плаваючому режимі. Для цього у вікні параметрів блоку необхідно установити прапор *Floating display*, а у вкладці *Advanced* вікна *Simulation parameters* змінити стан *Optimisations*-опції *Signal storage reuse* (повторне використання запам‘ятованого сигналу) з *on* на *off*.

Дискретизація сигналів виконується так само, як і в блоку *To Workspace*.

Блок ***Terminator* (*Заглушка)*** приєднується до виходів блоків, не зв'язаних з іншими блоками для запобігання висновку попереджувального повідомлення про наявність у моделі не приєднаних виходів.

Блок ***Stop Simulation* (*Зупинка Моделювання*)** зупиняє моделювання при ненульовому скалярному вхідному сигналі. У випадку векторного вхідного сигналу досить, щоб хоча б один з його компонентів став ненульовим. Оскільки в *MATLAB* результатом логічних операцій і операцій порівняння є 1 (істина) або 0 (неправда), то вхідний сигнал блоку *Stop* зручно формувати за допомогою блоків *Relational Operator* і (або) *Logical Operator*. Наприклад, у приведеній на рис. 2.10 моделі процес моделювання перерветься, якщо вихідний сигнал інтегратора стане більш 10 або менше -5.

2.3 Завдання

1) створити модель, яка формує вхідні сигнали: $u_1(t) = \sin(t)$ та $u_2(t) = 0.5(t-6)$, і зафіксувати ці сигнали блоками бібліотеки *Sinks* у такі способи:

- двома блоками *Scope*,
- одним блоком *Scope* з двома системами координат,
- одним блоком *Scope* з одною системами координат,
- блоком *Floating Scope*,
- одним (спільним) та двома (окремими для кожного сигналу) блоками *Out*;
- одним та двома блоками *To Workspace*,
- одним та двома блоками *To File*,
- блоком *XY Graph*,
- блоками *Display*;

використати одержану інформацію для побудови графіків $u_1(t)$, $u_2(t)$ та $u_1(u_2)$;

2) створити модель, що складається з блоків бібліотеки *Sources*, приєднаних через ключовий елемент *Multiport Switch* до найбільш зручних блоків бібліотеки *Sinks*; ознайомитися з принципами роботи джерел вхідних сигналів, змінюючи їхні параметри;

3) сформувати вхідні сигнали, подані на рис. 2.11, за даними табл. 2.1.

4) сформувати вхідні сигнали $U = f(t)$ за формулами, поданими в останньому стовпці табл. 2.1.

У звіті представити моделі та отримані графіки.

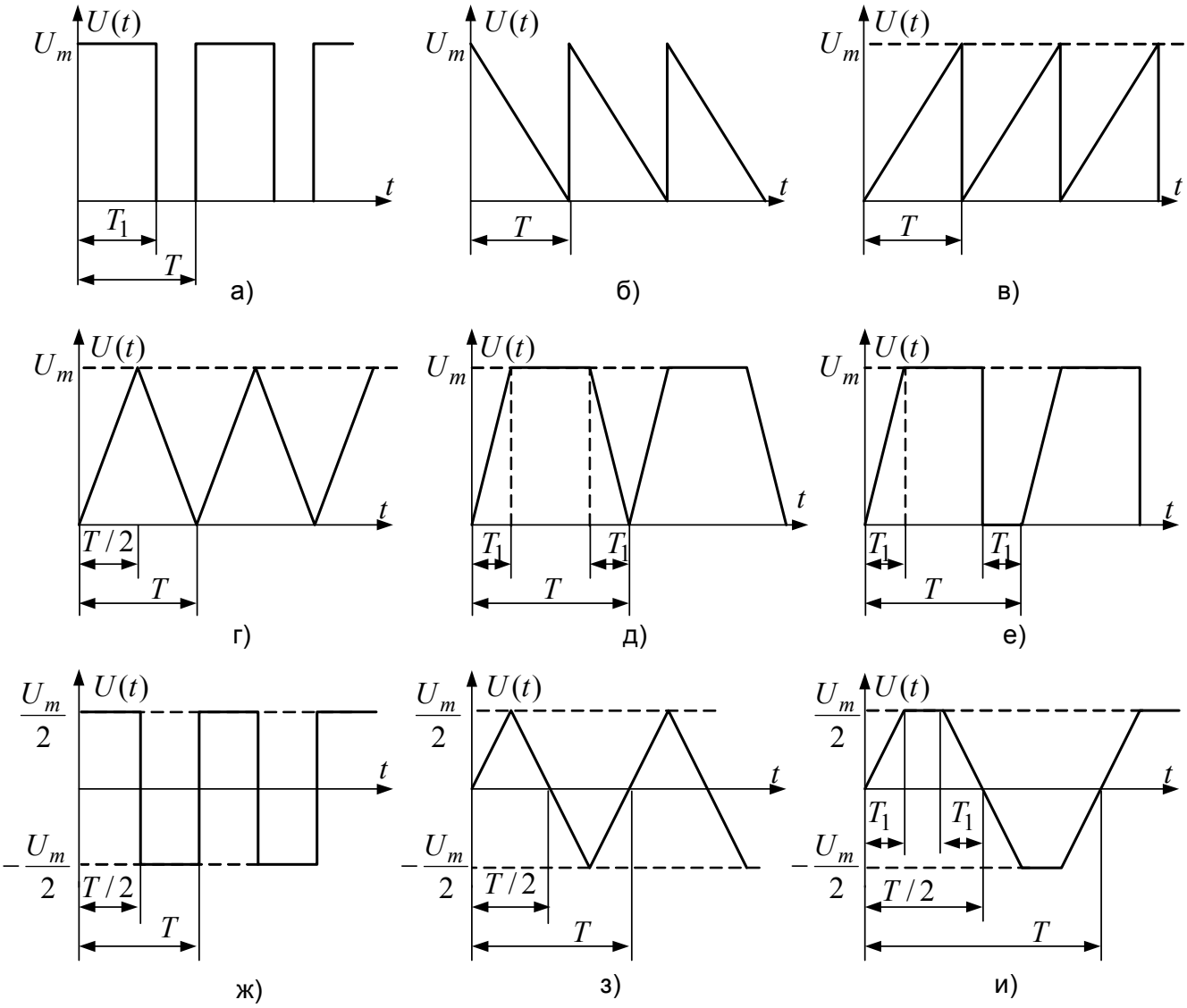


Рис. 2.11. Графіки вхідних сигналів

Таблиця 2.1

№ вар.	U_m	T	T_1	№ графіка	$U = f(t),\ t \in [0, 10]$
1	100	1	0,2	а, б, д	$U = 150\sin(60t + \pi/10)$
2	100	1	0,4	а, в, е	$U = 125\cos(50t - \pi/12)$
3	100	1	0,6	а, г, и	$U = 100\sin(10t)e^{-t/4}$
4	150	2	1	а, д, з	$U = 100e^{-t/4}$
5	150	2	0,8	ж, б, д	$U = 200\sqrt{t}$
6	150	2	1,2	ж, б, е	$U = 100\sqrt[3]{t}$
7	200	0,5	0,1	ж, в, и	$U = \dfrac{100}{t + 0.1}$
8	200	0,5	0,2	ж, г, д	$U = \dfrac{150t}{t + 1}$
9	200	0,5	0,15	ж, г, е	$U = t^2 + 2t + 10$
10	220	0,1	0,01	ж, г, и	$U = 100\sin(\lg(t + 1))$
11	220	0,1	0,02	а, б, е	$U = 200\cos(\ln(t + 1))$
12	220	0,1	0,05	а, б, и	$U = 150\dfrac{1}{1 - 0.5\sin t}$
13	250	10	1	а, г, д	$U = 100\dfrac{t^2}{1 + t^3}$
14	250	10	2	а, г, е	$U = 200\cos^2(t)\cos(2t)$
15	250	10	4	а, з, и	$U = 200\sin^2(t)\sin(2t)$
16	75	5	1	а, з, е	$U = e^{\sin 2t}\sin t$
17	75	5	0,5	а, д, и	$U = 100\sqrt[3]{t} \cdot \sin(3t)$
18	75	5	1,5	ж, г, е	$U = 200\sqrt[2]{t} \cdot \cos(2t)$

2.4 Методичні рекомендації

1) Без зайвої потреби не використовуйте вікна моделей, бібліотек, осцилографів у повноекранному режимі.

2) Для швидкого з‘єднання двох блоків виділіть щигликом миші вхідний блок, натисніть на клавішу <*Ctrl*> та виділіть мишею вихідний блок.

3) Для розмноження блоків та розгалуження ліній зв‘язку використовуйте операцію *drag* (*тягти*) при натиснутій правій клавіші миші.

4) Деякі операції з блоками зручно робити через меню, що випадає при щиглику по блоку правою клавішею миші.

5) Слідкуйте за наочністю моделі, для чого притримуйтесь таких правил:

- намагайтеся використовувати мінімальну кількість зломів та перетинів ліній

зв‘язку;

- не робіть блоки занадто великими (модель повинна бути компактною);
- приховуйте імена блоків, призначення яких зрозуміло з їхніх піктограм;
- якщо це сприяє кращому порозумінню моделі, замінюйте імена стандартних блоків іншими іменами, іменуйте лінії зв‘язку та доповнюйте моделі текстовими коментарями;
- при використанні різних кольорів ретельно обміркуйте, що саме ви бажаєте виділити тим чи іншим кольором.

6) Для реєстрації часу моделювання застосовуйте один з таких засобів:

- блок *Clock* приєднайте до блоку *To Workspace*, визначивши у полі *Variable name* цього блоку будь-яку змінну, наприклад, *t*.;
- у вкладці *Workspace I/O* вікна *Simulation Parameters* установіть прапорець *Time* функції *Save to workspace*.

7) При реєстрації даних блоками *To Workspace* і *Scope*, а також за допомогою функції *Save to workspace* вікна *Simulation Parameters* використовуйте формат *Array*. При великій кількості точок у векторі часу моделювання не забувайте збільшити або зробити безкінечним (*Inf*) параметр *Limit data point to last*.

8) Для об‘єднання декількох скалярних сигналів в один векторний використовуйте блок *Mux* бібліотеки *Signals & Systems*. Це дасть змогу скоротити кількість блоків реєстрації.

2.5 Контрольні запитання

1) Охарактеризуйте бібліотечні блоки реєстрації вихідних сигналів, перелічте їхні недоліки та переваги.

2) Опишіть структуру даних блоків *To Workspace* і *From Workspace*, *To File* і *From File*.

3) Поясніть призначення параметру *Decimation* блоків .

4) Поясніть призначення параметру *Interpolate data* блоків і сенс можливих значень цього параметра.

5) Поясніть призначення параметру *Limit data points to last* блоків

6) Поясніть роботу моделей рис. 2.7 та рис. 2.8.

3 Лабораторна робота №3
ЗНАЙОМСТВО З БІБЛІОТЕЧНИМИ НЕПЕРЕРВНИМИ ДИНАМІЧНИМИ БЛОКАМИ ПРОГРАМИ *Simulink*

3.1 Математичний опис лінійних неперервних систем

У більшості випадків на першому етапі досліджень роблять такі припущення, які дозволяють вважати САУ лінійними та стаціонарними (*LTI-systems*). Методи аналізу та синтезу таких системи є найпростішими та добре розвинутими.

Нехай довільна неперервна стаціонарна динамічна система n-го порядку має k входів та r виходів. Тоді вона може бути описана у просторі стану системою лінійних диференційних рівнянь (ДР)

$$\dot{\mathbf{x}}(t)=\mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{u}(t) \tag{3.1}$$

з початковими умовами

$$\mathbf{x}(t_0)=\mathbf{x}_0 \tag{3.2}$$

та лінійним матричним рівнянням виходу

$$\mathbf{y}(t)=\mathbf{C}\mathbf{x}(t)+\mathbf{D}\mathbf{u}(t), \tag{3.3}$$

де

$\mathbf{x}=\begin{bmatrix}x_1 & x_2 & \dots & x_n\end{bmatrix}^T$ – вектор змінних стану;

$\mathbf{y}=\begin{bmatrix}y_1 & y_2 & \dots & y_r\end{bmatrix}^T$ – вектор вихідних сигналів;

$\mathbf{u}=\begin{bmatrix}u_1 & u_2 & \dots & u_k\end{bmatrix}^T$ – вектор вхідних сигналів;

$\mathbf{x}_0=\begin{bmatrix}x_{10} & x_{20} & \dots & x_{n0}\end{bmatrix}^T$ – вектор початкових умов (*Initial conditions*);

A – матриця стану розміром $n\times n$;

B – матриця входу розміром $n\times k$;

C – матриця виходу розміром $r\times n$;

D – матриця обходу розміром $r\times k$, яка характеризує прямий (пропорційний) зв'язок між входом і виходом.

Для систем з одним входом (k = 1) та кількома виходами (*SIMO*-систем) матриці **B** і **D** вироджуються у вектори-стовпці:

$$\mathbf{B}=\mathbf{b}=\begin{bmatrix}b_1 & b_2 & \dots & b_n\end{bmatrix}^T, \qquad \mathbf{D}=\mathbf{d}=\begin{bmatrix}d_1 & d_2 & \dots & d_n\end{bmatrix}^T.$$

Для так званих *SISO*-систем, тобто систем з одним входом та одним виходом (k = 1, r = 1), матриця **B** залишається вектором-стовпцем, матриця **C** стає вектором-рядком розміром $1\times n$, а матриця **D** вироджується у скаляр:

$$\mathbf{C}=\mathbf{c}=\begin{bmatrix}c_1 & c_2 & \dots & c_n\end{bmatrix}, \qquad \mathbf{D}=d.$$

ДР (3.1) неоднорідні. Вони описують поведінку системи під дією зовнішніх вхідних сигналів. Якщо такі сигнали відсутні, то ДР стають однорідними:

$$\dot{\mathbf{x}}(t)=\mathbf{A}\mathbf{x}(t). \tag{3.4}$$

У сукупності з рівнянням виходу

$$\mathbf{y}(t)=\mathbf{C}\mathbf{x}(t) \tag{3.5}$$

вони описують власні рухи системи під дією відхилення початкових умов від значень, що відповідають усталеному режиму.

Для переходу до операторної форми запису рівнянь (3.1), (3.2) в них треба замінити операцію диференціювання операцією множення на оператор Лапласа s:

$$s\mathbf{x}(s)=\mathbf{A}\mathbf{x}(s)+\mathbf{B}\mathbf{u}(s), \tag{3.6}$$

$$\mathbf{y}(s)=\mathbf{C}\mathbf{x}(s)+\mathbf{D}\mathbf{u}(s). \tag{3.7}$$

Математичному опису (МО) у формі (3.7), (3.8) відповідає структурна схема, що зображена на рисунку 3.1 [10-12].

З рівнянь (3.6), (3.7) можна знайти матричні передавальні функції (ПФ) від вектора входу до вектору змінних стану розміром $n\times k$ та до вектора виходу

розміром $r \times k$:

$$\mathbf{W}_x(s) = \frac{\mathbf{x}(s)}{\mathbf{u}(s)} = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \frac{Adj(s\mathbf{I} - \mathbf{A})}{det(s\mathbf{I} - \mathbf{A})}\mathbf{B}, \tag{3.8}$$

$$\mathbf{W}_y(s) = \frac{\mathbf{y}(s)}{\mathbf{u}(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} = \mathbf{C}\frac{Adj(s\mathbf{I} - \mathbf{A})}{det(s\mathbf{I} - \mathbf{A})}\mathbf{B} + \mathbf{D}, \tag{3.9}$$

де $\mathbf{I}$ – одинична діагональна матриця розміром $n \times n$.

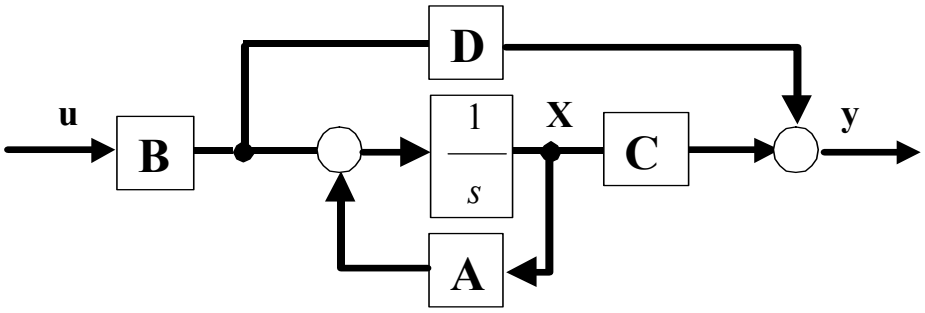


Рис. 3.1 – Структурна схема *LTI-MIMO*-системи у просторі стану

Знаменник передавальних функцій звуть характеристичним поліномом (ХП) системи:

$$G(s) = det(s\mathbf{I} - \mathbf{A}) = s^n + \alpha_{n-1}s^{n-1} + ... + \alpha_1 s + \alpha_0, \tag{3.10}$$

а рівняння

$$G(s) = 0$$

– характеристичним рівнянням.

Корені характеристичного рівняння

$$\mathbf{s} = [s_1 \quad s_2 \quad ... \quad s_n] \tag{3.11}$$

називають полюсами системи або власними числами матриці **A**.

Чисельних кожної з скалярних ПФ, що є компонентами матричних ПФ (3.9) та (3.10), також уявляє собою ступеневий поліном відносно змінної Лапласа, який називають поліномом впливу:

$$H(s) = \beta_m s^m + \beta_{m-1}s^{m-1} + ... + \beta_1 s + \beta_0. \tag{3.12}$$

Його корені

$$\mathbf{s}_z = [s_{z1} \quad s_{z2} \quad ... \quad s_{zm}] \tag{3.13}$$

звуть нулями системи.

Порядок поліному впливу пов‘язаний з порядком ХП умовою можливості фізичної реалізації

$$m \le n. \tag{3.14}$$

Слід відзначити, що умова $m = n$ може бути виконана тільки для передавальних функцій від вхідного сигналу до вихідного, якщо вони мають між собою прямий зв‘язок (без ланок затримки). Математичним виразом такого зв‘язку є ненульове значення відповідного елементу матриці обходу **D**.

Отже, скалярні ПФ можуть бути записані у поліноміальній формі, наприклад:

$$W_{ij}(s)=\frac{y_i(s)}{u_j(s)}=\frac{H_{ij}(s)}{G(s)}=\frac{\beta_m s^m+\beta_{m-1}s^{m-1}+...+\beta_1 s+\beta_0}{s^n+\alpha_{n-1}s^{n-1}+...+\alpha_1 s+\alpha_0}. \quad (3.15)$$

Від ПФ, записаної у формі (3.15), дуже легко перейти до МО *SISO*-системи у просторі стану в одній із канонічних форм, наприклад,

$$\mathbf{A}=\begin{bmatrix} -\alpha_{n-1} & 1 & 0 & ... & 0 \\ -\alpha_{n-2} & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... \\ -\alpha_1 & 0 & 0 & ... & 1 \\ -\alpha_0 & 0 & 0 & ... & 0 \end{bmatrix}. \quad (3.16)$$

$$\mathbf{B}=\mathbf{b}=\begin{bmatrix}\beta_{n-1} & \beta_{n-2} & ... & \beta_o\end{bmatrix}^T, \quad (3.17)$$

$$\mathbf{C}=\mathbf{c}=\begin{bmatrix}1 & 0 & ... & 0\end{bmatrix}, \quad (3.18)$$

$$\mathbf{D}=d=\beta_n. \quad (3.19)$$

Як бачимо, ХП, визначений за формулою (3.10), і застосований у ПФ (3.15) виявляється нормованим за коефіцієнтом при старшому степені оператора Лапласа.

У класичній теорії автоматичного керування більш прийнято нормування поліномів у чисельнику та знаменнику передавальної функції за коефіцієнтами при вільному члені або, при його відсутності, за коефіцієнтами при найменших степенях оператора Лапласа, наприклад,

$$W_{ij}(s)=\frac{y_i(s)}{u_j(s)}=k\frac{b_m s^m+b_{m-1}s^{m-1}+...+b_1 s+1}{a_n s^n+a_{n-1}s^{n-1}+...+a_1 s+1}=k\frac{H_{0ij}(s)}{G_0(s)}. \quad (3.20)$$

Коефіцієнти ПФ (3.16) і (3.17) пов‘язані між собою співвідношеннями:

$$a_n=\frac{1}{\alpha_0},\qquad a_i=\frac{\alpha_i}{\alpha_0},\qquad b_i=\frac{\beta_i}{\beta_0},\qquad k=\frac{\beta_0}{\alpha_0}. \quad (3.21)$$

Скалярній ПФ (3.16) відповідає ДР

$$\frac{d^n y_i(t)}{dt^n}+\alpha_{n-1}\frac{d^{n-1}y_i(t)}{dt^{n-1}}+...+\alpha_1\frac{dy_i(t)}{dt}+\alpha_0 y_i(t)=$$

$$=\beta_m\frac{d^m u_j(t)}{dt^m}+\beta_{m-1}\frac{d^{m-1}u_j(t)}{dt^{m-1}}+...+\beta_1\frac{du_j(t)}{dt}+\beta_0 u_j(t).$$

Використовуючи розкладення поліномів у чисельнику та знаменнику ПФ (3.16) або (3.17) на множники, отримуємо

$$W_{ij}(s)=K\frac{(s-s_{z1})(s-s_{z2})\ldots(s-s_{zm})}{(s-s_1)(s-s_2)\ldots(s-s_n)}, \quad (3.22)$$

де

$$K=\beta_m=k\frac{b_m}{a_n}. \quad (3.23)$$

В *Simulink* передбачено застосування усіх описаних вище форм математичного опису лінійних динамічних систем.

3.2 Відомості про блоки бібліотеки *Continuous* програми *Simulink*

Бібліотека *Continuous*, що представлена на рис. 3.2, містить у собі блоки аналогового інтегрування (*Integrator*) і диференціювання (*Derivative*), лінійні неперервні динамічні ланки довільного порядку в різних формах їхнього математичного опису (*Transfer Fcn*, *Zero-Pole* і *State-Space*) і ланки запізнювання (*Memory*, *Transport Delay* і *Variable Transport Delay*).

3.2.1 *Integrator* (*Інтегратор*)

У *Simulink 2.x* було 3 типи аналогових інтеграторів *(Integrator, Limited Integrator* і *Reset Integrator).* Починаючи з версії 2.0 усі три інтегратори об'єднані в одному, котрий може працювати в різних режимах за рахунок вибору відповідних установок у вікні параметрів, яке показано на рис. 3.3. Зміни режиму роботи інтегратора відображаються на вигляді його піктограми.

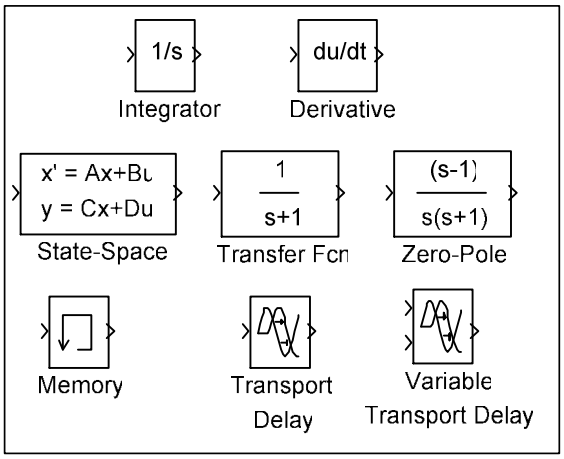


Рис. 3.2. Бібліотека неперервних динамічних блоків (*Continuous*)

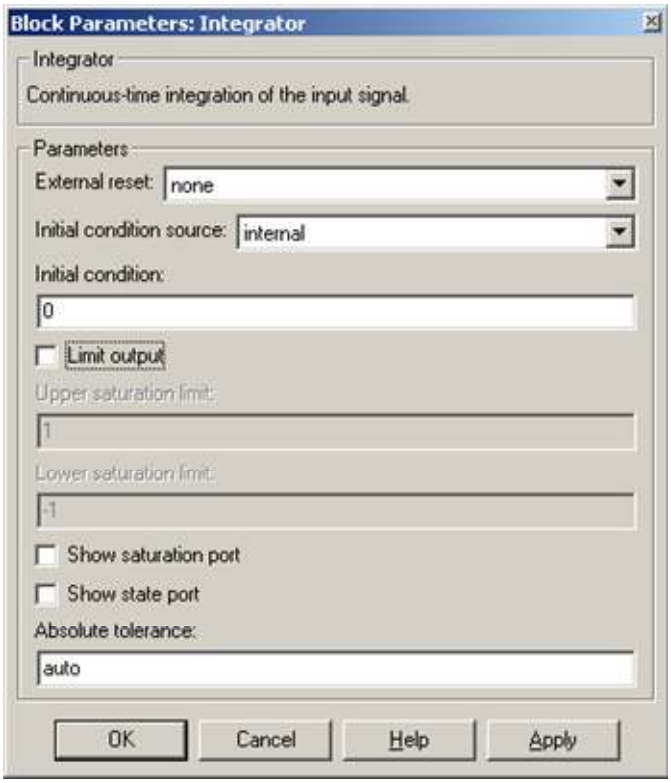


Рис. 3.3. Вікно визначення параметрів блоку *Integrator*

На рис. 3.4 зображені піктограми блоку *Integrator* з різними установками режимів.

Найпростіший інтегратор (див. ланку *Integrator* на рис. 3.4) інтегрує вхідний сигнал *u* із заданою початковою умовою x_0 (*Initial condition*).

Для того, щоб обмежити вихідний сигнал аналогового інтегратора (див. блок *Limited Integrator*), необхідно установити прапорець *Limit output* і ввести значення параметрів y_{max} (*Upper saturation limit*) і y_{min} (*Lower saturation limit*). Припустимим значенням цих параметрів є константа *Inf* (∞), що дозволяє обмежувати блок тільки знизу або тільки зверху.

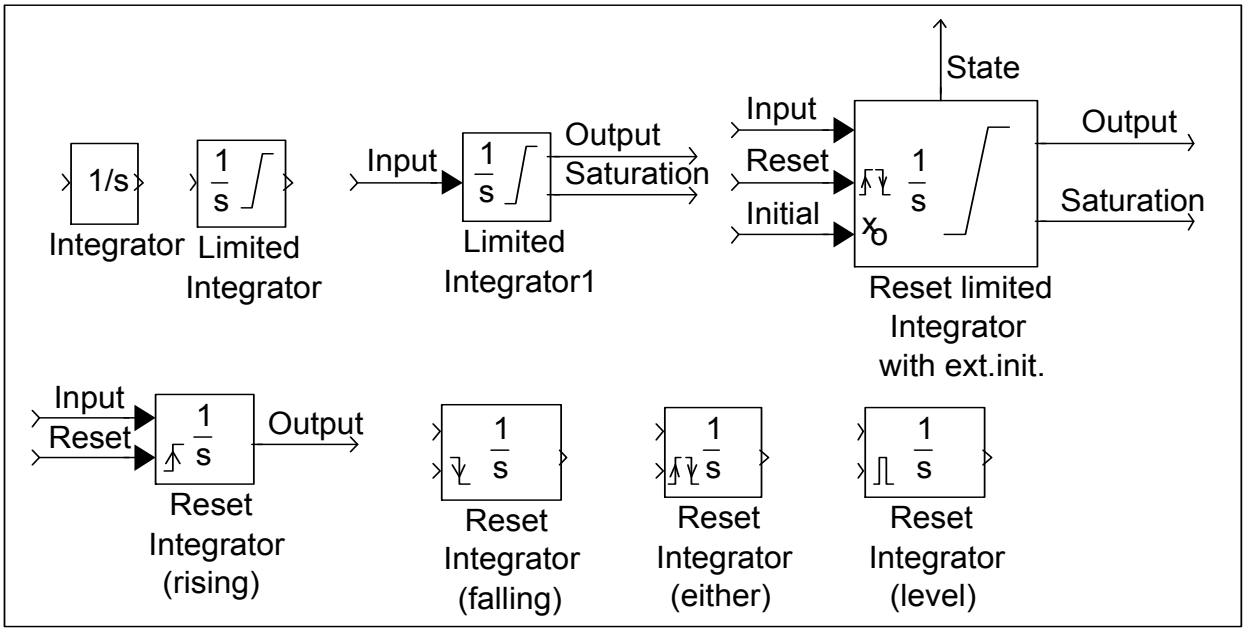


Рис. 3.4. Піктограми інтеграторів, які працюють у різних режимах

Слід зазначити, що інтегратор з обмеженням не можна замінити послідовним з'єднанням інтегратора без обмеження і типової нелінійності «обмеження координат» (блок *Saturation* бібліотеки *Nonlinear*). Для правильного обмеження вихідного сигналу інтегратора при перетинанні ним однієї з меж необхідно перевести блок з режиму інтегрування в режим збереження інформації підключенням до його входу замість сигналу нульової константи, який інтегрується, як це показано на рис. 3.5. Повернення в режим інтегрування відбувається при зміні знака вхідного сигналу.

Можна візуалізувати порт обмеження, установкою прапорця *Show saturation port* (див. блок *Limited Integrator1*).

Інтегратор може працювати зі скиданням вихідного сигналу в початкове значення (див. ланки *Reset Integrator*). Цим режимом керує меню, що випадає, опції *External reset*, яке пропонує наступні варіанти:

- *none* – скидання не виконується;
- *rising* – скидання виконується при перетинанні керуючим сигналом нульової лінії знизу нагору;
- *falling* – скидання виконується при перетинанні керуючим сигналом нульової лінії зверху вниз;
- *either* – скидання виконується при перетинанні керуючим сигналом “нуля” в обох напрямках;
- *level* – при нульовому значенні керуючого сигналу виконується інтегрування, а при ненульовому – скидання та утримання інтегратора в початковому стані.

При виборі будь-якого варіанта, крім *none*, до блоку приєднується керуючий вхідний порт (*Reset port*).

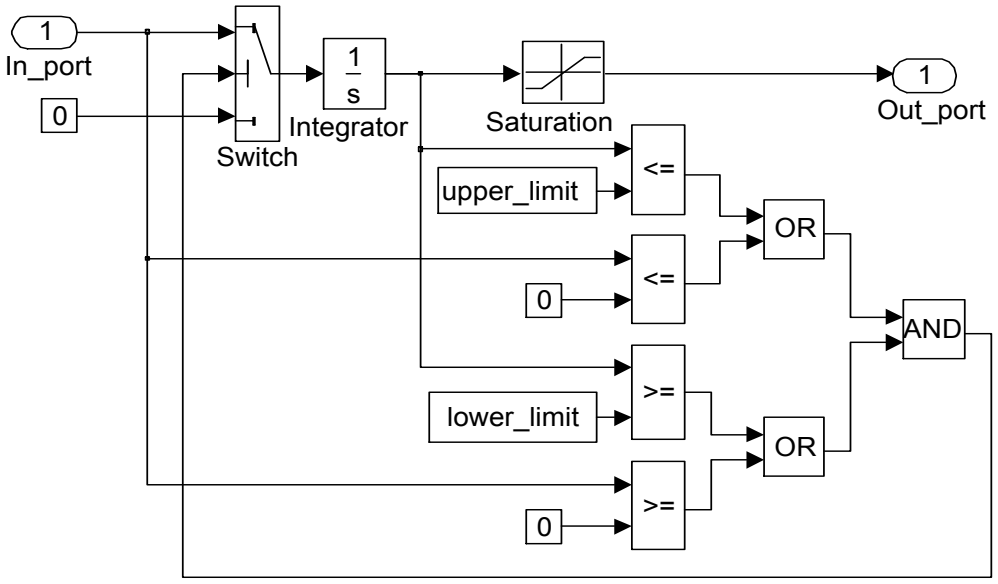


Рис. 3.5. Розгорнута модель інтегратора з обмеженням вихідного сигналу

Інтегратор може використовувати як внутрішні (*internal*), так і зовнішні (*external*) початкові умови. Вибір здійснюється за допомогою меню опції *Initial condition source*. При виборі режиму *external* блок одержує додатковий вхідний порт, до якого можна приєднати блок *IC* (*Початкова умова*) бібліотеки *Connections*. При виборі режиму *internal* діє початкова умова, що встановлюється в полі параметра *Initial condition*. Задавати зовнішні початкові умови має сенс тільки для інтегратора зі скиданням (див. ланку *Reset limited integrator with ext. init.*).

Для організації скидання або зміни початкових умов у функції вихідного сигналу інтегратора необхідно, щоб уникнути утворення алгебраїчної петлі, замість вихідного порту використовувати порт стану, для чого потрібно візуалізувати його установкою прапорця *Show state port*. Сигнал стану інтегратора формується раніш, ніж вихідний сигнал, хоча і має з ним однакові значення. Приклад використання ланки *Integrator* у режимах *Reset* і *Limited* можна подивитися в демонстраційному прикладі *Bouncing ball* (модель *bounce*).

Друга область застосування порту стану – це передача сигналів з однієї підсистеми, виконуваної за умовою, в іншу. Для демонстрації цього приклада можна скористуватися моделлю *clutch*.

3.2.2 *Derivative* (*Похідна*)

Виконує чисельне диференціювання вхідного сигналу:

$$y(t_i) = \frac{\Delta u_i}{\Delta t_i} = \frac{u_i - u_{i-1}}{t_i - t_{i-1}} \approx \frac{du}{dt}. \qquad (3.24)$$

Точність результату залежить від розміру кроку інтегрування. Менший розмір кроку дозволяє одержати більш точний результат.

Перед початком моделювання значення вхідного сигналу дорівнює нулю.

При диференціюванні дискретного сигналу з періодом переривання T_S вихідний сигнал являє собою послідовність імпульсів у моменти часу, які

збігаються з моментами зміни вхідного сигналу:

$$y(t_i)=\begin{cases}\dfrac{u(kT_S)-u(kT_S-T_S)}{t_i-t_{i-1}} & \text{при } t_i=kT_S,\\ 0 & \text{при } t_i\neq kT_S.\end{cases} \tag{3.25}$$

У цьому випадку блок може бути описаний дискретною передавальною функцією

$$\frac{y(z)}{u(z)}=\frac{1-z^{-1}}{\Delta t}=\frac{z-1}{\Delta t\cdot z}. \tag{3.26}$$

3.2.3 *Transfer Fcn* (*Передавальна Функція*)

Блок *Transfer Fcn* реалізує передавальну функцію загального вигляду в поліноміальній формі:

$$W(s)=\frac{B_m(s)}{A_n(s)}=\frac{\sum_{i=0}^{m}b_{i+1}s^{m-i}}{\sum_{i=0}^{n}a_{i+1}s^{n-i}}. \tag{3.27}$$

Вхідними параметрами є вектори коефіцієнтів степеневих поліномів чисельника (*Numerator*) і знаменника (*Denominator*), упорядковані за зменшенням степенів оператора Лапласа *s*. Вектор, що складається з (*n*+1) елементів, задає поліном степеня *n*. Порядок знаменника повинний бути більшим від порядку чисельника або дорівнювати йому ($m \le n$). Блок має нульові початкові умови.

Коефіцієнти можуть бути введені трьома способами:

1) константами і скалярними змінними, об'єднаними квадратними дужками у вектор;
2) змінною без дужок;
3) змінною або виразом у круглих дужках.

При першому способі в піктограмі блоку відображаються поліноми з коефіцієнтами у вигляді введених констант і змінних при відповідних степенях оператора Лапласа *s*. Наприклад, введення вектора [1 -2 *T* 4] відобразиться в блоці як

$$s^3-2s^2+Ts+4\ ,$$

а введення вектора [*b*0, *b*1, *b*2] – як

$$b0s^2+b1s+b2\,.$$

Змінні, що використані в цих векторах, повинні одержати значення в робочому середовищі *MATLAB* до початку процесу моделювання.

При другому способі поліном відображається у вигляді *змінна*(*s*). Наприклад, введення *num* відображається як *num*(*s*), введення *An* – як *An*(*s*) та т.і.

При третьому способі *Similink* шукає значення змінних, які входять до виразу, у робочому просторі *MATLAB,* обчислює значення виразу і відображає його так само, як і при першому способі введення. Наприклад, якщо поліном визначено у діалоговому вікні як (*G*+*B*), а змінні *G* і *B* визначено в *MATLAB* як *G*=[4 1 1], *B*=[0 1 0], то відображення полінома в блоці має вигляд:

$4s^2+2s+1$.

Значення змінних повинне існувати в робочому середовищі в будь-який момент зображення блоку, інакше в піктограмі блоку відобразяться три знаки питання: *???*.

Simulink дозволяє визначити передавальну функцію не тільки для систем з один входом та одним виходом, але і для систем з одним входом та декількома виходами. Це досягається введенням чисельника передавальної функції як матриці коефіцієнтів, кількість рядків якої відповідає кількості виходів. Відображення коефіцієнтів чисельника в піктограмі блоку в цьому випадку не підтримується, так що чисельник представляється тільки в загальному вигляді, наприклад, *num*(*s*), тобто матриця коефіцієнтів чисельника повинна задаватися тільки ім'ям перемінної без дужок. Якщо блоки, приєднані до вихідного порту такої ланки, не сприймають векторних вхідних сигналів, то отриманий вихідний векторний сигнал необхідно пропустити через блок *Demux* (*Демультіплексор*).

3.2.4 *Zero-Pole* (*Нулі-Полюси*)

Являє собою ланку загального виду з передавальною функцією (3.22) та такими параметрами:

K – посилення (*Gain*);

s_{zi} – нулі (*Zeros*);

s_i – полюси (*Poles*).

Нулі і полюси можуть мати не тільки дійсні, але і комплексно-спряжені значення. Вони можуть уводитися тими ж трьома способами, що і вектори степеневих багаточленів блоку *Transfer Fcn*. При відсутності нулів у лінійному блоку параметр *Zeros* задають пустою матрицею „[]”.

Розглянутий блок не підтримує режим *SIMO*, тобто не може бути використано, на відміну від ланки *Transfer Fcn*, для опису лінійної динамічної системи з декількома виходами.

3.2.5 *State-Space* (*Простір Стану*)

Забезпечує моделювання неперервної динамічної ланки загального виду за її математичному описом у просторі стану (3.1)-(3.3). Параметрами цього блоку є матриці коефіцієнтів **A**, **B**, **C**, **D** та вектор початкових умов (*Initial conditions*).

3.2.6 *Memory* (*Пам'ять*)

Здійснює затримку на один крок чисельного інтегрування. На виході утримується попереднє значення входу. Цей блок можна використовувати для розв'язки алгебраїчних контурів. Разом із блоком *Clock* його можна використовувати для визначення кроків інтегрування. У цьому блоці задається тільки один параметр: *Initial condition* – початкова умова.

3.2.7 *Transport Delay* (*Чисте Запізнювання*) і *Variable Transport Delay* (*Змінне Запізнювання*)

Передавальна функція блоків має вигляд:

$$\frac{y(p)}{u(p)} = e^{-s\tau}, \qquad (3.28)$$

тобто вихідний сигнал відтворює вхідний сигнал із запізнюванням τ (*Time Delay*):

$$y(t)=\begin{cases} y_0 & \text{при} \quad t \le \tau \\ u(t-\tau) & \text{при} \quad t > \tau \end{cases}, \tag{3.29}$$

де y_0 – початкова умова (*Initial condition*).

Ланки функціонують за рахунок збереження необхідної кількості послідовних значень вхідного сигналу в круговому буфері, початковий розмір якого задається параметром *Initial Buffer Size*. Отже, при великих часах запізнювання і порівняно малому кроці інтегрування блоки можуть використовувати велику кількість оперативної пам'яті.

Блок *Variable Transport Delay* відрізняється від блоку *Transport Delay* тим, що величина запізнювання в ньому є не внутрішнім, а зовнішнім параметром, який подається на другий вхідний порт блоку, і може змінюватись за будь-яким законом. Для обмеження запізнювання на заданому рівні і для розрахунку кількості точок у круговому буфері в ньому передбачено параметр *Maximum Delay*.

Ще одним параметром ланок запізнення є порядок ряду Пада (*Pade order*), який використовується при лінеаризації цих блоків і не впливає на результати моделювання.

3.3 Відомості про лінійні арифметичні блоки бібліотеки *Math*

Описані в попередньому підрозділі блоки використовуються при моделюванні лінійних неперервних систем. Для зв‘язку динамічних елементів між собою у складі таких систем застосовуються лінійні арифметичні блоки *Sum*, *Gain*, *Slider Gain* і *Matrix Gain* бібліотеки *Math* (рис. 2.3).

Блок *Sum* описано в підрозділі 2.1 (див. рис. 2.4).

Блоки ***Gain* (*Підсилення*, *Пропорційна Ланка*)** та ***Matrix Gain* (*Матричне Підсилення*)**, починаючи з версії *MATLAB*-6.0 реалізують однакові функції. Вони можуть виконувати за вибором користувача поелементне (*Element-wise K.*u*) або матричне (*Matrix*(*K*u*), *Matrix*(*u*K*)) множення вхідного сигналу на коефіцієнт передачі. Тип множення задається параметром *Multiplication*. Обидва із множників можуть бути скалярами, векторами або матрицями відповідного розміру, тобто можливі такі варіанти виконання цієї операції:

$$y = K \cdot u, \quad \underset{(m,n)}{\mathbf{y}} = \underset{(m,n)}{\mathbf{K}} .* \underset{(m,n)}{\mathbf{u}}, \quad \underset{(m,n)}{\mathbf{y}} = \underset{(m,k)}{\mathbf{K}} * \underset{(k,n)}{\mathbf{u}}, \quad \underset{(m,n)}{\mathbf{y}} = \underset{(m,k)}{\mathbf{u}} * \underset{(k,n)}{\mathbf{K}}. \tag{3.30}$$

При поелементному множенні, яке виконується над масивами однакового розміру,

$$y_{ij} = K_{ij} u_{ij}, \tag{3.31}$$

а при матричному –

$$y_{ij} = \sum_{l=1}^{k} K_{il} u_{lj}. \tag{3.32}$$

Ураховуючи те, що сигнали моделі досить рідко бувають двомірними, останні три операції, відображені формулами (3.27), здебільш мають такий формат:

$$\underset{(n,1)}{\mathbf{y}} = \underset{(n,1)}{\mathbf{K}} .* \underset{(n,1)}{\mathbf{u}}, \quad \underset{(m,1)}{\mathbf{y}} = \underset{(m,n)}{\mathbf{K}} * \underset{(n,1)}{\mathbf{u}}, \quad \underset{(1,n)}{\mathbf{y}} = \underset{(1,m)}{\mathbf{u}} * \underset{(m,n)}{\mathbf{K}}, \quad y = \underset{(1,n)}{\mathbf{K}} * \underset{(n,1)}{\mathbf{u}}, \quad \underset{(n,1)}{\mathbf{y}} = \underset{(n,1)}{\mathbf{K}} * u. \quad (3.33)$$

Піктограма блоку відображує скалярний коефіцієнт підсилення в тій же формі, у якій він визначений при введенні (змінна або константа). Якщо коефіцієнт заданий у вигляді змінної в круглих дужках, то усередині блоку відображується її значення. Якщо значення змінної або її ім'я, або вираз занадто великі для того, щоб розташувати їх у межах блоку, то в піктограма зображуються символи "-*K*-". Щоб побачити вираз або значення коефіцієнта підсилення, необхідно збільшити розміри блоку.

Блок ***Slider Gain*** **(*Повзункове Підсилення*)** виконує множення вхідного сигналу на коефіцієнт, що обирається в заданих межах *Low-High* повзунком віртуального реостату, розташованого у вікні вибору параметрів блоку.

3.4 Завдання

1) Для вивчення властивостей інтегратора при роботі його у різних режимах, набрати модель, зображену на рис. 3.5, зафіксувати результати моделювання (приклад див. на рис. 3.6) та пояснити їх.

2) За допомогою блоків *Transfer Function*, *Zero-Pole* та *State Space* реалізувати моделі, задані передавальними функціями у табл. 3.1. Впевнитися у збігу результатів моделювання.

3) Отримати вектор кроків ЧІ $h_i = t_i - t_{i-1}$ $(i = 1, 2, ..., lentgh(t))$ для останньої моделі з використанням блоку *Memory*. Порівняти одержаний результат з результатом, отриманим за допомогою функції diff (t). Побудувати графік $h(i)$ для методів *ode*45, *ode*23, *ode*23*s*, *ode*15*s* та *ode*23*t*.

4) Продемонструвати роботу блоків запізнення.

3.5 Методичні рекомендації

1) Для визначення нулів zer та полюсів pol за коефіцієнтами поліномів у чисельнику та знаменнику передавальної функції скористуйтеся *MATLAB*-функцією roots з аргументами num та den відповідно, а для вирішення зворотної задачі – функцією poly:

```
pol = roots (den),    zer = roots (num),    gain = num(1) / den(1)
den = real (poly (pol)),    num =gain * real (poly (zer))
```

2) Для математичного опису лінійної системи у просторі стану скористайтеся рівняннями (3.16)-(3.19):

```
n = length (den) - 1,    m = length (num) - 1
den1=den / den(1),    num1 = num / den(1),    num1 = [zeros(1,n-m), num1)]
A = [ -den1(2:n+1)', [eye(n-1); zeros(1,n-1)] ]
B = num1(2:n+1)'
```

C = zeros(1,n); C(1)=1

D = num(1)

3) Для дослідження блоків запізнення подавайте їм на входи синусоїдальні сигнали. Змінне запізнення організуйте за допомогою послідовного з‘єднання блоків *Clock*, *Math Fn* та *Gain*.

3.6 Контрольні запитання

1) Які форми математичного опису лінійних неперервних систем Ви знаєте?

2) Що таке характеристичний поліном, характеристичне рівняння, нулі і полюси системи?

3) Як скласти математичний опис системи у просторі стану?

4) Як отримати передавальну функцію з диференційного рівняння?

5) У яких режимах може працювати блок *Integrator*?

6) Для чого потрібний блок *Memory*?

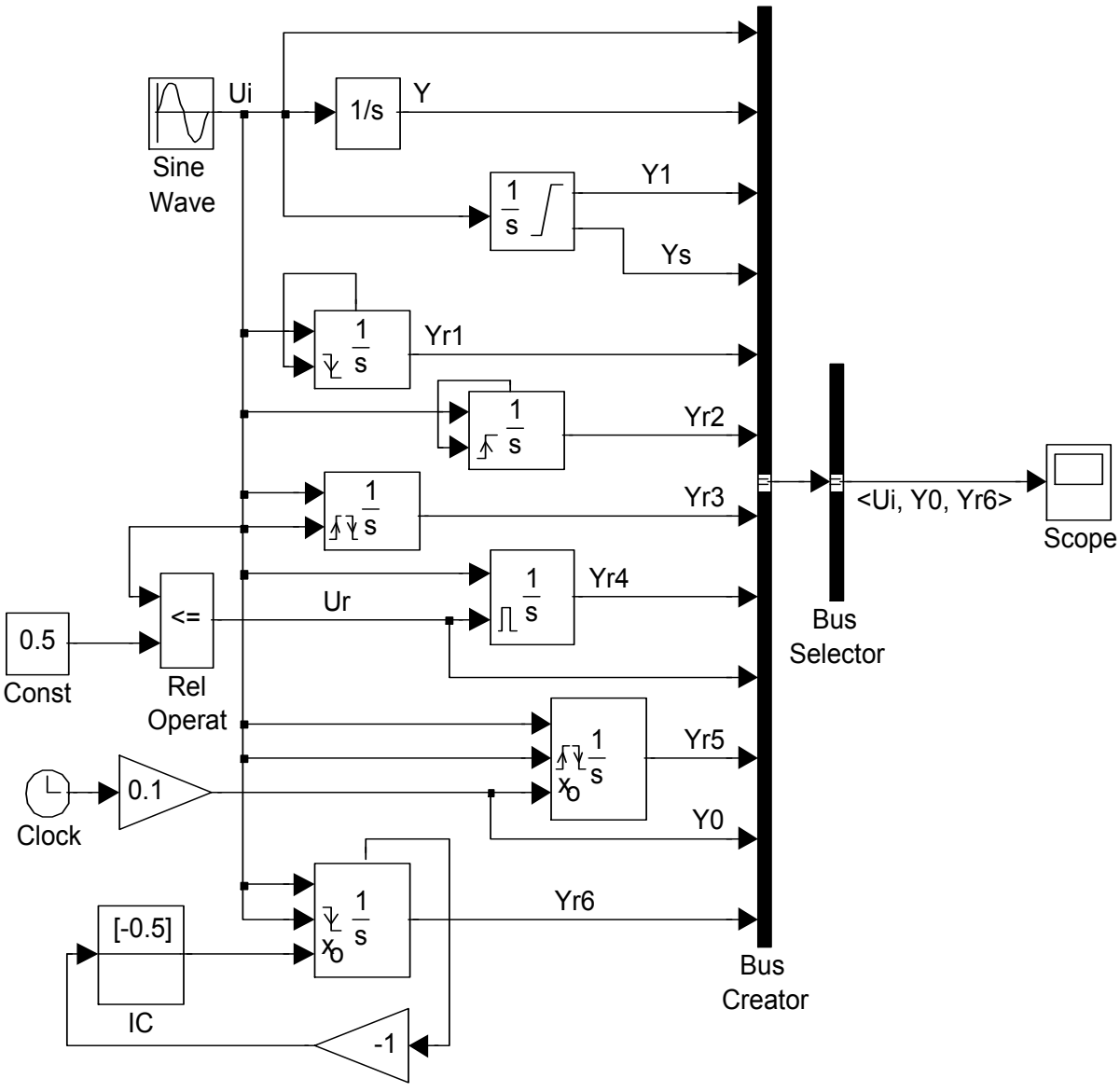
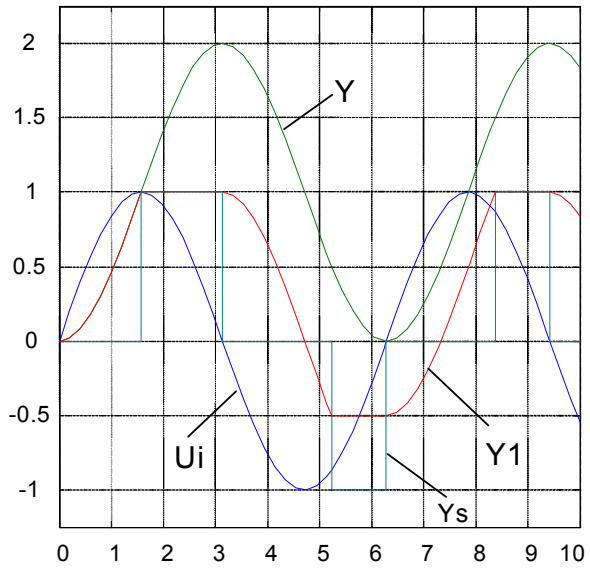
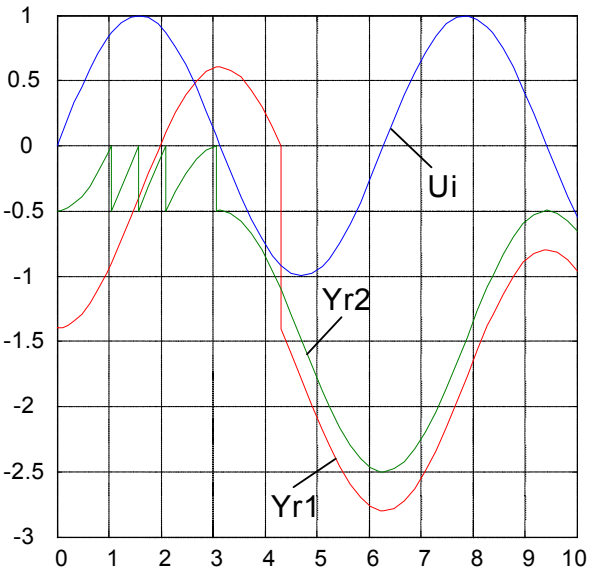


Рис. 3.6. Модель для вивчення принципу роботи інтегратора

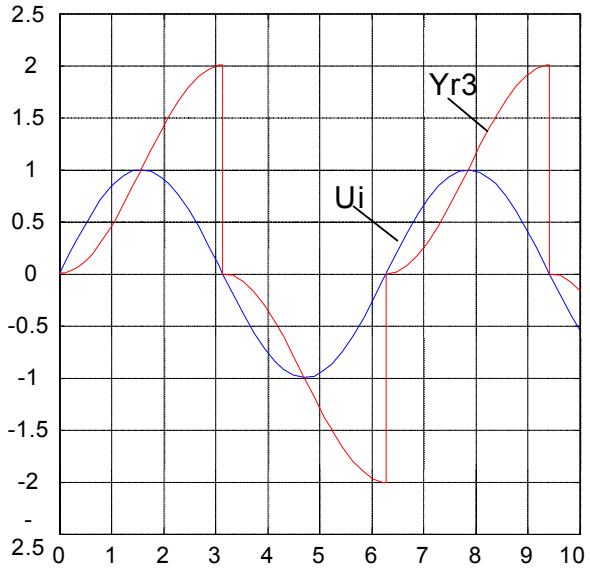
в різних режимах



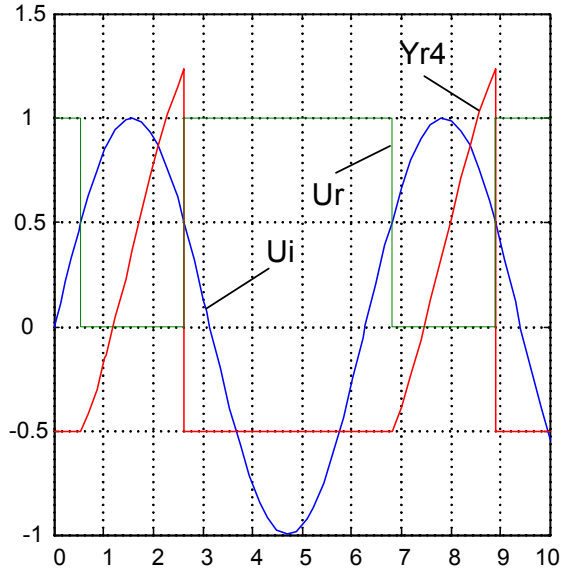
а)



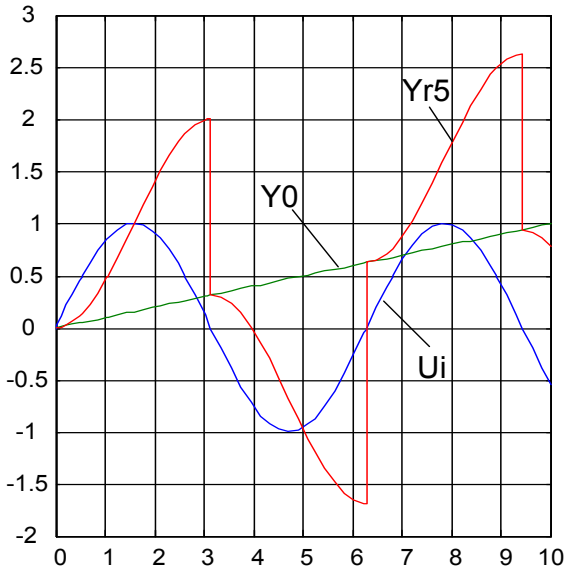
б)



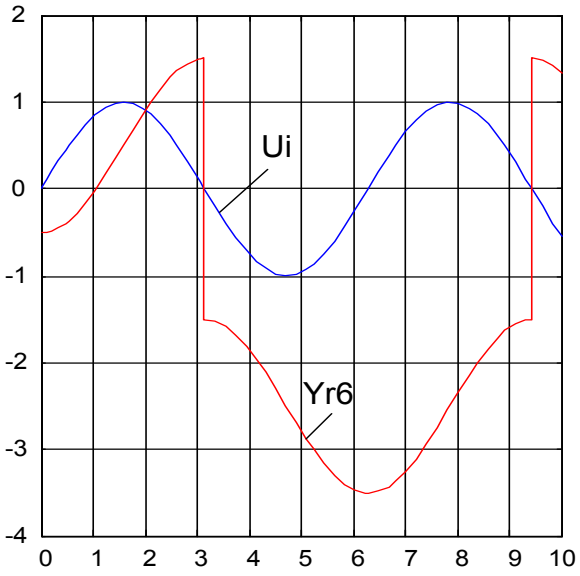
в)



г)



д)



е)

Рис. 3.6. Перехідні процеси, одержані при роботі моделі рис. 3.5

Таблиця 3.1

№ вар.	Передавальна функція	Параметри
1	$W(s)=\dfrac{k}{T^2s^2+2\xi Ts+1}$	$k=1,\ T=2,\ \xi=0.25$
2		$k=1.5,\ T=3,\ \xi=\sqrt{2}/2$
3	$W(s)=\dfrac{K}{(s-s_1)(s-s_2)}$	$K=2,\ s_{1,2}=-2.3\pm 3j$
4		$K=2,\ s_{1,2}=-3\pm 2.3j$
5	$W(s)=\dfrac{T_1s}{T^2s^2+2\xi Ts+1}$	$T=2,\ \xi=0.25,\ T_1=1.5$
6		$T=3,\ \xi=\sqrt{2}/2,\ T_1=4$
7	$W(s)=K\dfrac{s}{(s-s_1)(s-s_2)}$	$K=2,\ s_{1,2}=-2.3\pm 3j$
8		$K=2,\ s_{1,2}=-3\pm 2.3j$
9	$W(s)=\dfrac{k}{T^2s^2+1}$	$k=1,\ T=2$
10		$k=1.5,\ T=3$
11	$W(s)=K\dfrac{s}{(s-s_1)^2}$	$K=2,\ s_{1,2}=-2.3\pm 3j$
12		$K=2,\ s_{1,2}=-3\pm 2.3j$
13	$W(s)=\dfrac{T_1s}{T^2s^2+1}$	$T_1=2,\ T=3$
14		$T_1=1.5,\ T=4$
15	$W(s)=k\dfrac{T_1s+1}{T^2s^2+2\xi Ts+1}$	$k=1,\ T_1=2,\ \xi=0.25,\ T=3$
16		$k=1.5,\ T_1=8,\ \xi=\sqrt{2}/2,\ T=4$
17	$W(s)=K\dfrac{(s-s_3)}{(s-s_1)(s-s_2)}$	$K=2,\ s_{1,2}=-2.3\pm 3j,\ s_3=-1$
18		$K=2,\ s_{1,2}=-3\pm 2.3j,\ s_3=-2$
19	$W(s)=k\dfrac{T_1s+1}{T^2s^2+1}$	$k=1,\ T_1=2,\ T=3$
20		$k=3,\ T_1=1.5,\ T_3=4$
21	$W(s)=K\dfrac{(s-s_3)}{(s-s_1)^2}$	$K=2,\ s_{1,2}=-2.3\pm 3j,\ s_3=-1$
22		$K=2,\ s_{1,2}=-3\pm 2.3j,\ s_3=-2$
23	$W(s)=\dfrac{(T_1s+1)(T_2s+1)}{T^2s^2+2\xi Ts+1}$	$T_1=2,\ T_2=3,\ T=5,\ \xi=0.4$
24		$T_1=2,\ T_2=3,\ T=5,\ \xi=0.12$
25	$W(s)=\dfrac{(T_1s+1)(T_2s+1)}{T^2s^2+1}$	$T_1=2,\ T_2=3,\ T=5$
26		$T_1=5,\ T_2=3,\ T=2$

4 Лабораторна робота №4
ЗНАЙОМСТВО З БІБЛІОТЕЧНИМИ ДИСКРЕТНИМИ ДИНАМІЧНИМИ ЛАНКАМИ ПРОГРАМИ *Simulink*

4.1 Математичний опис лінійних дискретних систем

Лінійні стаціонарні дискретні динамічні системи з періодом переривання T описуються у просторі стану різницевими матричними рівняннями

$$\mathbf{x}(kT)=\mathbf{A}\mathbf{x}(kT-T)+\mathbf{B}\mathbf{u}(kT-T) \tag{4.1}$$

з початковими умовами

$$\mathbf{x}(t_0)=\mathbf{x}_0 \tag{4.2}$$

та лінійним матричним рівнянням виходу

$$\mathbf{y}(kT)=\mathbf{C}\mathbf{x}(kT)+\mathbf{D}\mathbf{u}(kT). \tag{4.3}$$

Для переходу до операторної форми сигнали, що запізнюються у часі на період T, помножують на z^{-1}, де

$$z=\exp(Ts) \tag{4.4}$$

– дискретний оператор. В результаті такого перетворення рівняння (4.1) та (4.2) набувають вигляду:

$$z\mathbf{x}(z)=\mathbf{A}\mathbf{x}(z)+\mathbf{B}\mathbf{u}(z), \tag{4.5}$$

$$\mathbf{y}(z)=\mathbf{C}\mathbf{x}(z)+\mathbf{D}\mathbf{u}(z). \tag{4.6}$$

Формальна аналогія математичного опису неперервних та дискретних систем в операторній формі простору часу дає можливість використовувати для їх перетворення, аналізу та синтезу однаковий математичний апарат.

Зокрема, перехід від рівнянь у просторі стану до дискретних матричних передавальних функцій здійснюється за формулами, аналогічними формулам (3.9) і (3.10) для неперервних систем:

$$\mathbf{W}_x(z)=\frac{\mathbf{x}(z)}{\mathbf{u}(z)}=(z\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}=\frac{Adj(z\mathbf{I}-\mathbf{A})}{det(z\mathbf{I}-\mathbf{A})}\mathbf{B}, \tag{4.7}$$

$$\mathbf{W}_y(z)=\frac{\mathbf{y}(z)}{\mathbf{u}(z)}=\mathbf{C}(z\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}+\mathbf{D}=\mathbf{C}\frac{Adj(z\mathbf{I}-\mathbf{A})}{det(z\mathbf{I}-\mathbf{A})}\mathbf{B}+\mathbf{D}. \tag{4.8}$$

Складові дискретних матричних ПФ записують у поліноміальній формі відносно оператора z

$$W_{ij}(z)=\frac{y_i(z)}{u_j(z)}=\frac{H_{ij}(z)}{G(z)}=\frac{b_m z^m+b_{m-1}z^{m-1}+...+b_1 z+b_0}{a_n z^n+a_{n-1}z^{n-1}+...+a_1 z+a_0}=,\quad m\le n \tag{4.9}$$

або відносно зворотної до нього величини z^{-1}

$$W_{ij}(z^{-1})=\frac{y_i(z)}{u_j(z)}=\frac{b_0 z^{-m}+b_1 z^{-(m-1)}+...+b_{m-1}z^{-1}+b_m z^{-(n-m)}}{a_0 z^{-n}+a_1 z^{-(n-1)}+...+a_{n-1}z^{-1}+a_n}. \tag{4.10}$$

Знаменник дискретної ПФ також зветься характеристичним поліномом а чисельник – поліномом впливу

Їх корені утворюють вектор полюсів $\mathbf{z}=[z_1 \quad z_2 \quad ... \quad z_n]$ та вектор нулів $\mathbf{z}_z=[z_{z1} \quad z_{z2} \quad ... \quad z_{zm}]$.

Скалярній передавальній функції (4.10) відповідає різницеве рівняння

$$a_n y_i(kT) + a_{n-1} y_i(kT - T) + ... + a_1 y_i(kT - (n-1)T) + a_0 y_i(kT - nT) =$$
$$= b_m u_j(kT) + b_{m-1} u_j(kT - T) + ... + b_1 u_j(kT - (m-1)T) + b_0 u_j(kT - mT).$$

Використовуючи розкладення поліномів у чисельнику та знаменнику передавальної функції (4.9) на множники, отримуємо

$$W_{ij}(s) = K \frac{(z - z_{z1})(z - z_{z2})\ldots(z - z_{zm})}{(z - z_1)(z - z_2)\ldots(z - z_n)}, \qquad K = \frac{b_m}{a_n}. \tag{4.11}$$

4.2 Відомості про блоки бібліотеки *Discrete* програми *Simulink*

У цю бібліотеку, представлену на рис. 4.1, входять дискретний інтегратор, екстраполятори, ланка запізнювання на період дискретності і дискретні динамічні ланки загального виду з різними способами їхнього математичного опису.

Усі ці блоки мають у діалоговому вікні введення параметрів поле з ім'ям *Sample Time*. Правила визначення цього параметру, який керує моментами зміни вихідних сигналів дискретних блоків t_d згідно з формулою (2.1), викладені в підрозділі 2.1.

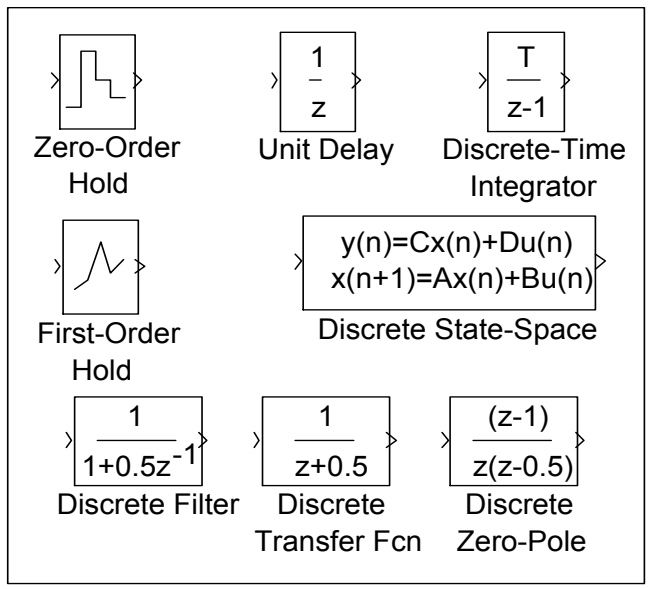


Рис. 4.1. Бібліотека *Discrete*

Кожен дискретний блок має вбудований переривач (ідеальний імпульсний елемент) на вході і екстраполятор нульового порядку на виході, так що вхідні сигнали дискретних блоків змінюються тільки в моменти часу t_d, а вихідні сигнали між двома сусідніми перериваннями утримуються на постійному рівні.

4.2.1 *Unit Delay* (*Одиничне Дискретне Запізнювання*)

Робить затримку сигналу й утримання його на цьому рівні на один період квантування.

$$W(z) = z^{-1}. \tag{4.12}$$

Початкова умова *Initial condition* задає значення виходу блоку на першому кроці моделювання системи, на якому вихід блоку *Unit Delay* не визначений.

Крім періоду квантування, може бути заданий зсув *offset*. Блок підтримує тільки скалярні вхід і вихід.

4.2.2 *Discrete-Time Integrator* (*Дискретний Інтегратор*)

Виконує чисельне інтегрування вхідного сигналу з періодом дискретності T_S (*Sample Time*) і початковою умовою x_0 (*Initial condition*). Дискретний інтегратор може використовувати один із трьох методів чисельного інтегрування (*Integrator method*):

Forward Euler – метод нижніх (лівобічних) прямокутників:

$$y(nT_S) = y(nT_S - T_S) + T_S u(nT_S - T_S), \tag{4.13}$$

$$W(z) = \frac{y(z)}{u(z)} = T_S \frac{z^{-1}}{1 - z^{-1}} = T_S \frac{1}{z - 1}; \tag{4.14}$$

Backward Euler – метод верхніх (правобічних) прямокутників:

$$y(nT_S) = y(nT_S - T_S) + T_S u(nT_S), \tag{4.15}$$

$$W(z) = \frac{y(z)}{u(z)} = T_S \frac{1}{1 - z^{-1}} = T_S \frac{z}{z - 1}; \tag{4.16}$$

Trapeziodal – метод трапецій (середніх прямокутників):

$$y(nT_S) = y(nT_S - T_S) + T_S \frac{u(nT_S - T_S) + u(nT_S)}{2}, \tag{4.17}$$

$$W(z) = \frac{y(z)}{u(z)} = T_S \frac{1 + z^{-1}}{2(1 - z^{-1})} = T_S \frac{z + 1}{2(z - 1)}. \tag{4.18}$$

Застосування різних методів чисельного інтегрування (ЧІ) можна продемонструвати за допомогою розгорнутої моделі дискретного інтегратора, зображеної на рис. 4.2, і перехідних функцій, приведених на рис. 4.3.

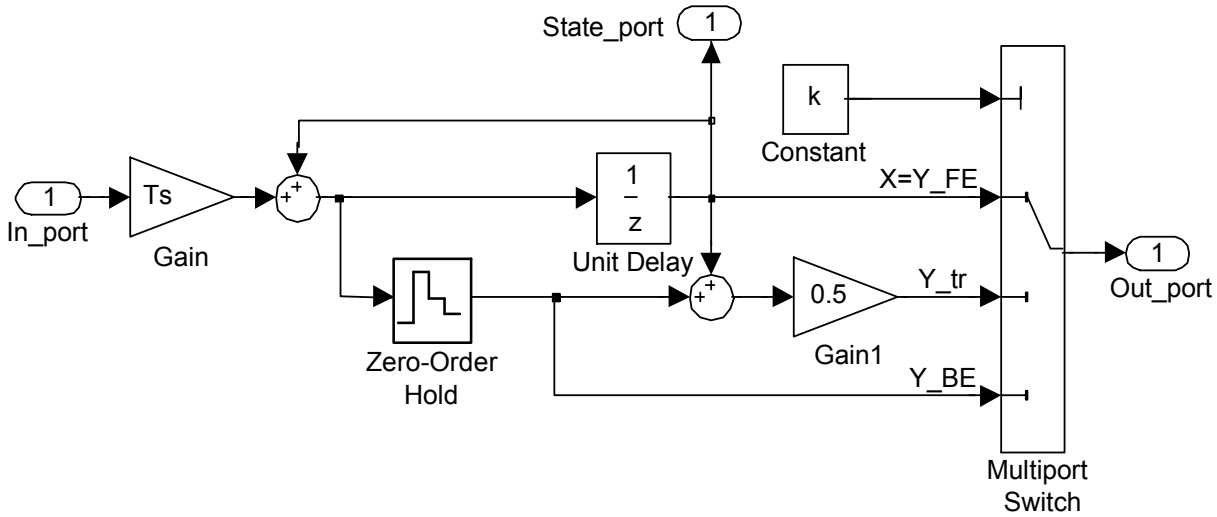
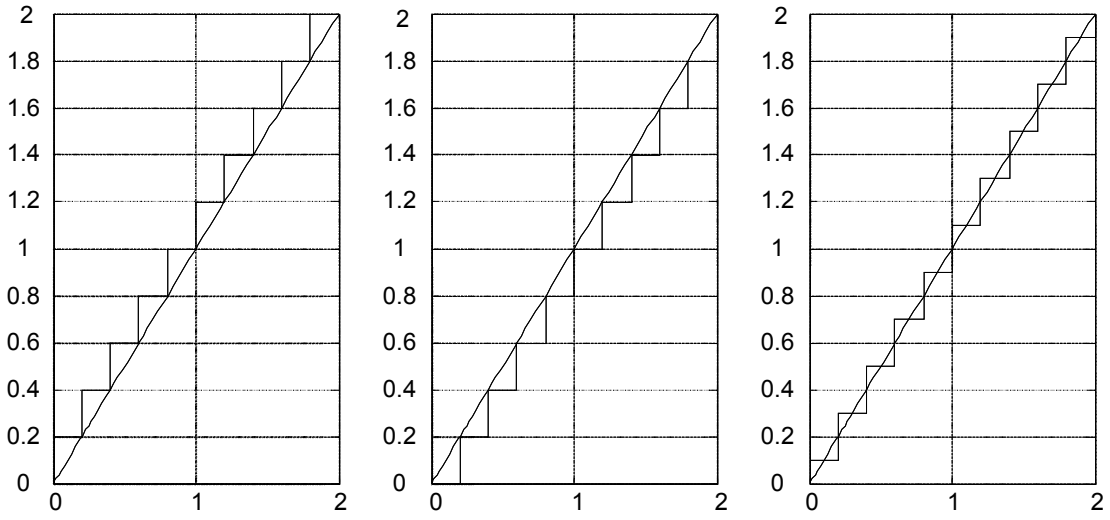


Рис. 4.2. Розгорнута модель дискретного інтегратора
з різними алгоритмами чисельного інтегрування

На останньому рисунку для порівняння приведена також перехідна функція неперервного інтегратора.

Як видно з рис. 4.2, основу цифрового інтегратора складає ланка запізнювання на період дискретності *Unit Delay*, замкнена додатним зворотним зв'язком. Один від одного інтегратори з різними алгоритмами ЧІ відрізняються точкою знімання вихідного сигналу. Варто звернути увагу на те, що інтегратори, які використовують методи *Backward Euler* і *Trapezoidal*, мають прямий зв'язок входу з виходом. Тому при замиканні їхнього виходу на вхід утворюється алгебраїчний контур. Уникнути цього можна заміною вихідного сигналу

сигналом стану *State port*, попередньо зробивши його видимим установкою прапорця *Show State port*, або включити в замкнений контур блок *Unit Delay*.



а) *б*) *в*)

Рис. 4.3 – Перехідні функції дискретних інтеграторів
з різними методами чисельного інтегрування:
а) *Backward Euler*, *б*) *Forward Euler*, *в*) *Trapeziodal*

Слід відзначити, що алгоритм, закладений в роботу блоку *Discrete-Time Integrator*, дещо складніший, ніж це показано на рис. 4.2, і не зовсім ретельно продуманий, що приводить до деяких похибок в його роботі. Зокрема, при встановленні на вході цього блоку ланки *Const* або ланки *Step* з нульовим значенням параметру *Step time* перехідні функції інтегратора не змінюються при зміні методу чисельного інтегрування і мають вигляд рис. 4.3, *а*. Так само веде себе дискретний інтегратор і при використанні його порту стану для розв‘язання алгебраїчного контуру. Отже, в таких випадках краще використовувати не бібліотечний блок , а створити його власноруч за схемою рис. 4.2.

Так само, як і аналоговий інтегратор (див. підрозділ 3.2), дискретний інтегратор може працювати в режимах обмеження (*Limit Output*) і скидання (*External reset*) вихідного сигналу в початковий стан, який може встановлюватися усередині блоку (*internal*) і поза нього (*external*). Зміни установок відображаються на піктограмі блоку.

Для обмеження вихідних сигналів дискретних інтеграторів у моделі рис. 4.2 треба блок *Unit Delay* замінити блоком запізнювання з обмеженням вихідного сигналу, який можна створити за допомогою моделі рис. 4.4, а на виходах блоків *Zero-Order Hold* та *Gain1* установити ланки *Saturation* з тими ж самими рівнями обмеження, що задані в блоках *Constant* і *Constant1* моделі рис. 4.4.

У перемикачах *Switch* і *Switch1* моделі рис. 4.4 значення u_T параметру *Threshold* повинно задовольняти умові $0 < u_T < 1$. Зазвичай приймають середнє значення указаного діапазону, тобто $u_T = 0.5$.

4.2.3 *Discrete Filter* (*Дискретний Фільтр*) і *Discrete-Time Fcn* (*Дискретна Передавальна Функція*)

Обидва блоки являють собою дискретну динамічну ланку загального виду з передавальною функцією поліноміального типу з тією лише різницею, що поліноми блоку *Discrete-Time Fcn* мають незалежною змінною оператор z (див. формулу (4.9)), а поліноми блоку *Discrete Filter* – z^{-1} (див. формулу (4.10)):

Способи введення векторів коефіцієнтів степеневих поліномів чисельника (*Numerator*) і знаменника (*Denominator)* і їхня інтерпретація аналогічні таким для блоку *Transfer Fcn* (див. п. 3.2.3).

Блок *Discrete-Time Fcn* базується на різновидності математичного опису дискретних ланок, що використовується у поширенні *Control System Toolbox*, а блок *Discrete Filter* – на опису, що використовується у поширенні *Signal Processing Toolbox*.

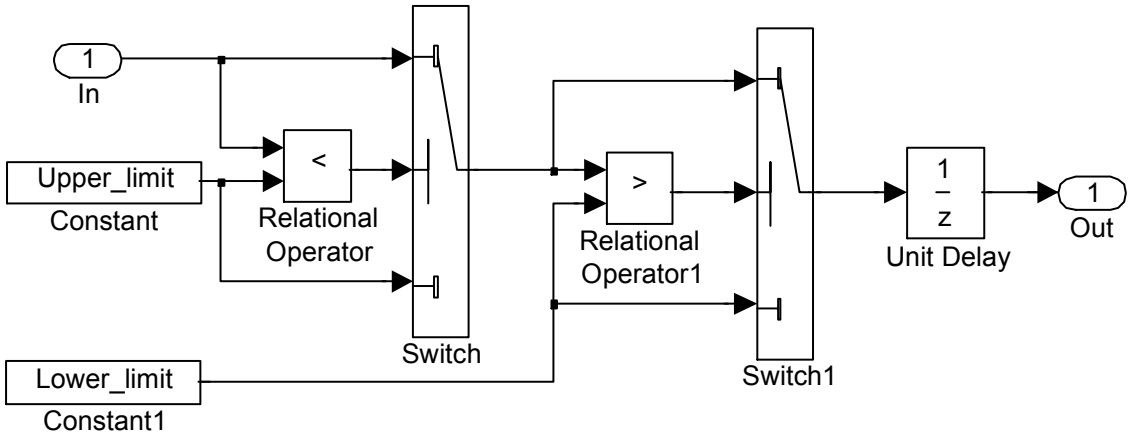


Рис. 4. 4. Модель блоку запізнення на період дискретності з обмеженням вихідного сигналу

4.2.4 *Discrete Zero-Pole* (*Дискретні Нулі-Полюси*)

Являє собою дискретну динамічну ланку загального виду з ПФ (4.11), де

K – коефіцієнт підсилення (*Gain*);

z_{zi} – нулі (*Zeros*);

z_j – полюси (*Poles*).

Вектори нулів і полюсів можуть уводитися тими ж трьома способами, що і вектори степеневих поліномів блоку *Transfer Fcn*.

4.2.5 *Discrete State-Spase* (*Дискретний Простір Станів*)

Забезпечує моделювання дискретної динамічної ланки загального вигляду за її описом у просторі станів (див. рівняння (4.1)-(4.6)).

Вхідними параметрами блоку є матриці *A, B, C, D,* вектор початкових умов (*Initial conditions*) і період переривання (*Sample time*).

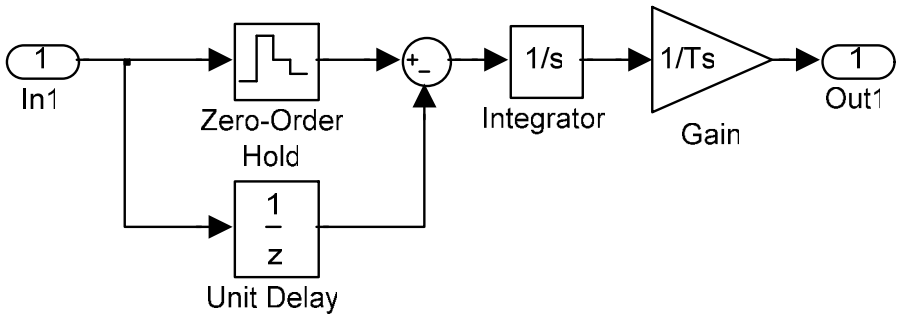
4.2.6 *Zero-Order Hold* (*Екстраполятор Нульового Порядку*)

Вимірює вхідний сигнал у дискретні моменти часу (2.1) і утримує значення цього сигналу на виході протягом періоду дискретності.

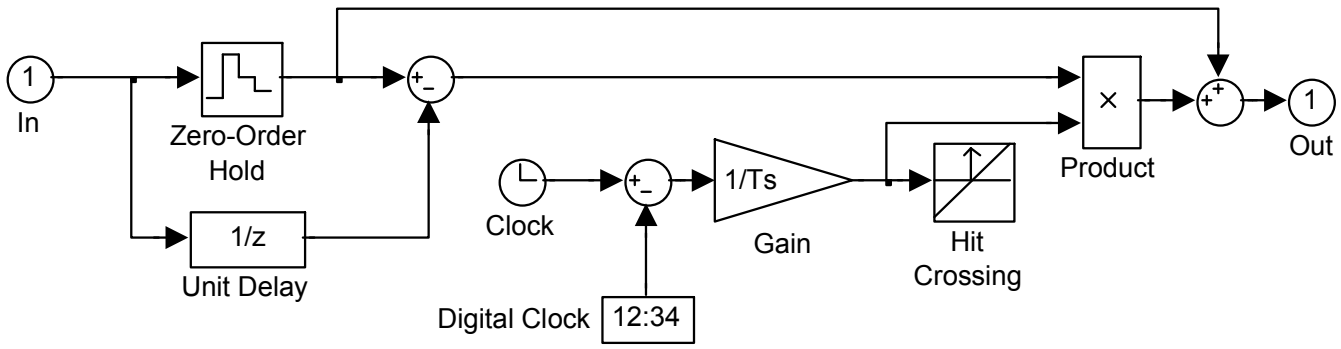
$$W(s) = \frac{e^{-T_S s} - 1}{e^{-T_S s} s} = \frac{z - 1}{zs}. \qquad (4.19)$$

4.2.7 *First-Order Hold* (*Екстраполятор Першого Порядку*)

В пакеті *MATLAB* 4.*x* блок *First-Order Hold* (*FOH*) мав структурну схему, зображену на рис. 4.5, *а*. Він вимірював вхідний сигнал у дискретні моменти часу (2.1) і формував вихідний сигнал у вигляді ламаної лінії, яка складалась з відрізків прямих, що з'єднують між собою 2 сусідні обмірювані точки входу з запізнюванням на період дискретності.



а)



б)

Рис. 4.5. Моделі екстраполяторів першого порядку в пакеті *MATLAB*:
а) версій до 4.x; *б*) версій, починаючи з 5.x

Починаючи з *MATLAB* 5.x, структуру блоку *FOH* було змінено у відповідності з рис. 4.5, *б*, що, на наш погляд, виявилось не дуже вдалим рішенням. Про це свідчать подані на рис. 2.6 графіки перетворення вхідної синусоїди екстраполяторами нульового і першого порядків з наведеними на рис. 4.5 структурними схемами.

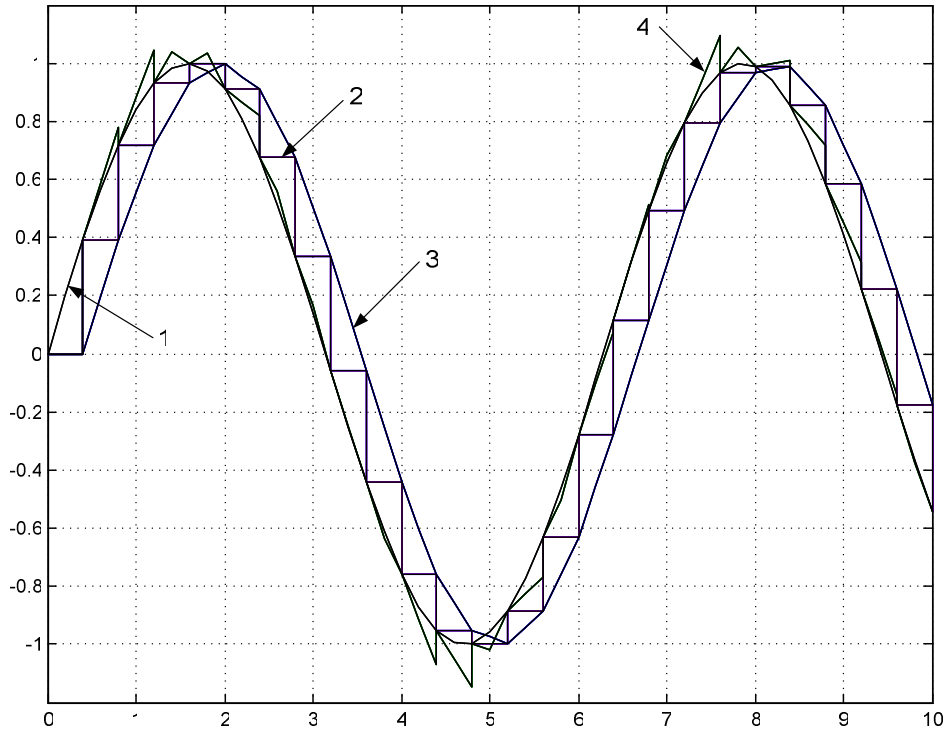


Рис. 4.6. Вхідний синусоїдальний сигнал (1) та результати перетворення його екстраполяторами нульового порядку (2) та екстраполяторами першого порядку, побудованими за схемами рис. 4.5, а (3) та рис. 4.5, б (4)

4.3 Завдання

1) Дослідити роботу блоку *Discrete-Time Integrator* з різними методами чисельного інтегрування при відпрацьовуванні ступінчатого, лінійного, та синусоїдального сигналів. Пояснити характер перехідних процесів.

2) Розробити дискретний інтегратор, здійснюючий чисельне інтегрування методом Сімпсона за рівнянням

$$y_S(nT) = y_S(nT - 2T) \; + \frac{T}{3}\big(u(nT-2T) + 4u(nT-2T) + u(nT)\big). \qquad (4.20)$$

Приєднати його до моделі, отриманої при виконанні попереднього пункту завдання.

3) Створити екстраполятор першого порядку за схемою рис. 4.5, *а* та порівняти його роботу з роботою бібліотечних блоків *First-Order Hold* та *Zero-Order Hold*.

4) Зібрати модель, що складається із двох послідовно з‘єднаних дискретних блоків з різними періодами дискретності за даними табл. 4.1.

5) Порівняти перехідні функції цифро-аналогових моделей, що складаються з послідовно з‘єднаних дискретного інтегратора та неперервної аперіодичної ланки, замкнених одиничним зворотним зв‘язком.

(В одной схеме интегратор с ограничением, а в другой – последовательное соединение интегратора и ограничения)

6) Розробити дискретний інтегратор, здійснюючий чисельне інтегрування методом Сімпсона, з обмеженням вихідного сигналу. Приєднати його до моделі,

отриманої при виконанні попереднього пункту завдання. Довести адекватність розробленого блоку

4.3 Методичні вказівки та рекомендації

При моделюванні цифро-аналогових систем можна використовувати методи ЧІ з автоматичним вибором кроку або методи з постійним кроком при $h = T_s$.

При моделюванні цифрових та цифро-аналогових систем, які отримують у своєму складі дискретні блоки з різними періодами дискретності, можна використовувати тільки методи ЧІ з автоматичним вибором кроку.

Для узгодження двох послідовно з'єднаних дискретних блоків, які працюють з різними періодами дискретності T_{s1} і T_{s2}, необхідно дотримуватись таких правил:

- якщо $T_{s1} > T_{s2}$, розташуйте між ними ти блок *Unit Delay* з $T_s = T_{s1}$;
- якщо $T_{s1} < T_{s2}$, розташуйте між ними ти блок *Zero-Order Hold* з $T_s = T_{s2}$.

При виконанні пункту 6 деталізуйте передавальну функцію дискретного інтегратора, поміняйте блоки *Unit Delay* на підсистеми рис. 4.4 (назвіть їх *Limited Unit Delay*), на виході інтегратора поставте блок *Saturation* та продумайте, як узгодити між собою рівні обмеження блоків *Saturation* і *Limited Unit Delay*.

4.4 Контрольні запитання

1) Поясніть різницю між різними методами чисельного інтегрування.

2) Поясніть, як можна знайти дискретну передавальну функцію деякого об'єкту за його різницевим рівнянням.

3) Які недоліки має бібліотечний блок *Discrete-Time Integrator*?

4) Чим відрізняються один від одного блоки *DiscreteTransfer Function* і *Discrete Filter*?

5) Які недоліки має бібліотечний блок *First-Order Hold*? Як їх позбутися?

6) Як треба узгоджувати між собою дискретні блоки, що мають різні періоди дискретності?

7) Які методи ЧІ можна використовувати при моделювання цифро-аналогових та цифрових об'єктів?

5 Лабораторна робота №5
ЗНАЙОМСТВО З БІБЛІОТЕЧНИМИ НЕЛІНІЙНИМИ БЛОКАМИ ПРОГРАМИ *Simulink*

5.1 Теоретичні відомості

Нелінійні ланки, що найчастіше використовуються при моделюванні електромеханічних систем можна розподілити на такі групи:

1) блоки множення-ділення декількох сигналів;

2) блоки, що виконують логічні операції та операції порівняння;

3) функціональні перетворювачі, описувані нелінійними аналітичними

виразами;

4) функціональні перетворювачі, задані таблицями вхідних та вихідних сигналів у деяких вузлових точках.

Серед блоків третьої та четвертої груп виділяють так звані типові нелінійності. Так називають блоки з досить простими кусочно-лінійними характеристиками вхід-вихід, що часто зустрічаються на практиці.

До них належать такі ідеалізовані блоки як *Обмеження координат*, *Сухе тертя*, *В'язке тертя*, *Зона нечутливості*, *Модуль*, *Люфт* (*Зазор*), *Петля гістерезису*, *Реле*, *Компаратор* та деякі інші.

З іншого боку всі нелінійності можна поділити на однозначні та багатозначні (найчастіше двозначні). У багатозначних нелінійностей зв'язок між вхідним та вихідним сигналами визначається не тільки формою статичної характеристики, але й передісторією вхідного сигналу, наприклад, від того, зменшується вхідний сигнал чи наростає. Типовими двозначними нелінійностями є блоки *Петля гістерезису* та *Зазор в кінематичній передачі*.

При визначенні вихідних сигналів блоків четвертої групи в точках між вузловими, застосовують або апроксимацію, або інтерполювання [ ].

При апроксимації найчастіше використовують метод найменших квадратів (МНК) або кусочно-лінійну апроксимацію. У якості апроксимуючих аналітичних функцій здебільш застосовують степеневі поліноми, комбінації експонент та розкладення періодичних характеристик в ряд Фур'є.

Серед методів інтерполяції для більшості технічних розрахунків достатню точність можна забезпечити локальним інтерполюванням степеневими поліномами першого-третього порядків. Для підвищення точності можна використовувати глобальну інтерполяцію або інтерполяцію кубічними сплайнами. Але застосування останніх методів може значно збільшити час розрахунку перехідних процесів.

5.2 Нелінійні блоки програми *Simulink*

Нелінійні ланки програми зосереджені в декількох бібліотеках.

Блоки першої, другої та частково третьої груп, описаних у попередньому підрозділі, знаходяться в бібліотеці *Math* (див. рис. 2.3). Крім описаних у підрозділі 2.1 блоків *Math Function* та *Trigonometric Function*, до них належать блоки *Product*, *Dot Product*, *Abs*, *Sign*, *MinMax*, *Rounding Function*, *Relational Operator*, *Logical Operator*, *Combinatorial Logic* та *Bitwise Logical Operator*.

Блок ***Product*** **(*Множення-Ділення*)** перемножує вхідні сигнали, кількість яких *n* визначається параметром *Number of inputs*:

$$y = u_1 \cdot u_2 \cdot \ldots u_n . \tag{5.1}$$

При n=1 він накопичує добуток значень єдиного вхідного векторного сигналу:

$$y = prod(\mathbf{u}) = \prod_{i=1}^{k} u_i . \tag{5.2}$$

При цьому в піктограмі блоку символ “*”замінюється символом “П”.

Якщо замість кількості входів задати список, зіставлений з послідовності знаків операцій множення „*” та ділення „/”, то блок буде виконувати над вхідними сигналами саме операції, задані списком.

Якщо вхідні сигнали є векторами однакового розміру, то операції множення / ділення можуть здійснюватись як матрично (*Matrix*(*)), так і поелементно (*Element-wise*(.*)), що задається через меню параметру *Multiplication*.

Блок ***Dot Product*** **(*Скалярний добуток*)** обчислює скалярний добуток векторних вхідних сигналів **u** та **v** однакового розміру:

$$y = \mathrm{dot}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^{\mathrm{T}}\mathbf{v} = \sum_{i=1}^{n} u_i v_i \ . \qquad (5.3)$$

Складається з послідовно з'єднаних блоку *Product* із двома входами і блоку *Sum* з одним входом. Якщо вхідні сигнали – скаляри, то обчислюється їхній добуток.

Виходом ланки ***Abs*** **(*Модуль*)** є абсолютне значення входу: $y = |u|$.

Блок ***Rounding Function*** **(*Округлення*)** округляє значення вхідного сигналу одним з обраних способів: округлення до цілого з недостачею (*floor*), з надлишком(*ceil*), та за правилами арифметики (*round*), скорочення до цілого, тобто відкидання дробової частини (*fixed*).

Блок ***MinMax*** **(*Мінімум / Максимум*)** визначає мінімальний або максимальний (за вибором користувача) із вхідних сигналів, кількість яких задається параметром *Number of input ports*.

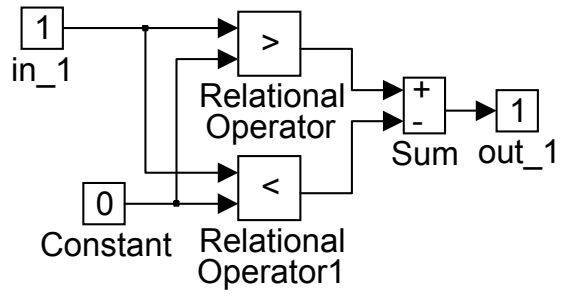


Рис. 5.1 – Структура блоку *Sign*

Блок ***Sign*** **(*Знакова Функція, Ідеальне Реле*)** формує вихідний сигнал, величина якого визначається знаком вхідного сигналу:

$$y = \begin{cases} 1 & \text{при} \quad u > 0, \\ 0 & \text{при} \quad u = 0, \\ -1 & \text{при} \quad u < 0. \end{cases} \qquad (5.4)$$

Цей замаскований блок має структурну схему рис. 5.1 і належить до типових нелінійностей. Якщо вхідним сигналом блоку *Sign* є швидкість (лінійна або кутова), а на виході встановлено ланку з коефіцієнтом, рівним амплітуді сили тертя (при поступальному руху) або моменту тертя (при обертальному руху), то одержують модель типової нелінійності *Сухе тертя*.

Блок ***Relational Operator*** **(*Операції Порівняння*)** виконує зазначену в полі параметра *Operator* операцію порівняння:

< – менше;
<= – менше або дорівнює;
>= – більше або дорівнює;
> – більше;

== – дорівнює;

~= – нерівно.

Результатом є одиничний (істина) або нульовий (неправда) сигнал.

Блок ***Logical Operator*** **(*Логічна операція*)** виконує одну з логічних операцій *and, or, xor, nor, nand* з елементами векторів, які надходять на вхідні порти, кількість яких задається в полі параметра *Number of Input Ports.* Операція *not* допускає тільки один вхідний порт.

Блок ***Combinatorial Logik*** **(*Комбінаторна логіка*)** забезпечує перетворення вхідного сигналу відповідно до заданої таблиці істинності *Truth Table,* котра являє собою список можливих вихідних сигналів ланки. При завданні цієї таблиці необхідно дотримуватись таких правил:

- кількість стовпців дорівнює кількості виходів блоку;
- кількість рядків дорівнює 2^n, де n – розмірність вхідного сигналу;
- нульове значення сигналу в таблиці трактується як “неправда”, будь-яке ненульове – як “істина”;
- входи таблиці вважаються заданими; зокрема при двоелементному вхідному сигналі варіація входів як [0 0; 0 1; 1 0; 1 1] для .

Наприклад, при введенні як параметр блоку матриці [0 1; 1 0; 1 1; 0 0] повна таблиця істинності матиме вигляд табл. 5.1.

Таблиця 5.1

u_1	u_2	y_1	y_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

За допомогою цього блоку можна описати на рівні логіку роботи будь-якого пристрою або системи, якщо їх вхідні і вихідні дані можуть бути представлені у формі булевих величин (0 - “неправда”, 1 - “істина”). Отже, розглянуту ланку можна назвати моделлю кінцевого детермінованого автомата.

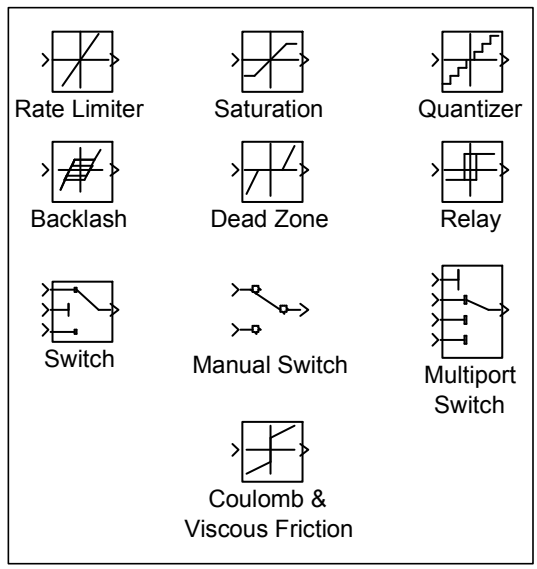


Рис. 5.2 – Бібліотека *Nonlinear*

В бібліотеці ***Nonlinear*** (див. рис. 5.2) знаходяться найбільш поширені типові нелінійності: *Saturation* (*Обмеження координат*), *Dead Zone* (*Зона нечутливості*), *Coulomb & Viscous Friction* (*Сухе та в‘язке тертя*), *Quantizer* (*Квантизатор*), *Rate Limiter* (*Обмеження темпу*), *Backlash* (*Люфт*), *Relay* (*Реле з гістерезисом*).

Піктограми перелічених блоків зображують статичні характеристики відповідних типових нелінійностей.

Блок ***Saturation*** **(*Обмеження Координат*)** являє собою пропорційну ланку з одиничним коефіцієнтом підсилення, вихідний сигнал якої обмежений зверху на рівні *U* (*Upper output limit*) і знизу – на рівні *L*

(*Lower output limit*):

$$y = \begin{cases} u & \text{при} \quad L \le u \le U, \\ U & \text{при} \quad u > U, \\ L & \text{при} \quad u < L. \end{cases} \tag{5.5}$$

Його властивості продемонстровані на рисунку 5.3.

Блок ***Dead Zone*** **(*Зона Нечутливості*)** має статичну характеристику, що описується формулою

$$y = \begin{cases} 0 & \text{при} \quad L \le u \le R, \\ u + L & \text{при} \quad u < L, \\ u - R & \text{при} \quad u > R, \end{cases} \tag{5.6}$$

де L, R – ліва (*Start of dead zone*) і права (*End of dead zone*) границі зони нечутливості відповідно. Процес перетворення цією ланкою синусоїдального сигналу відображено на рис. 5.4.

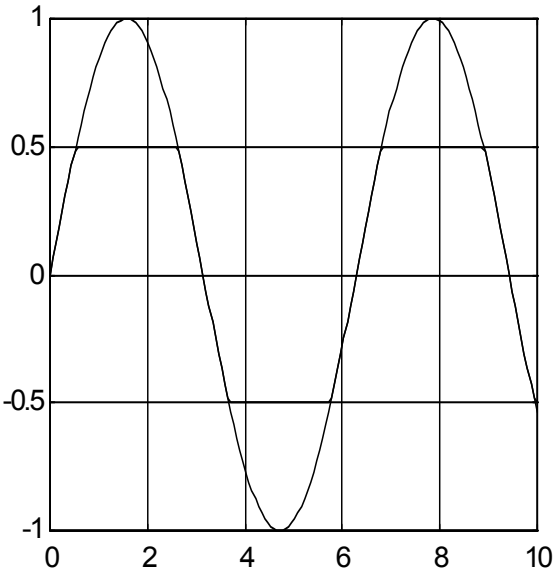


Рис. 5.3 – Перетворення синусоїди ланкою *Saturation*

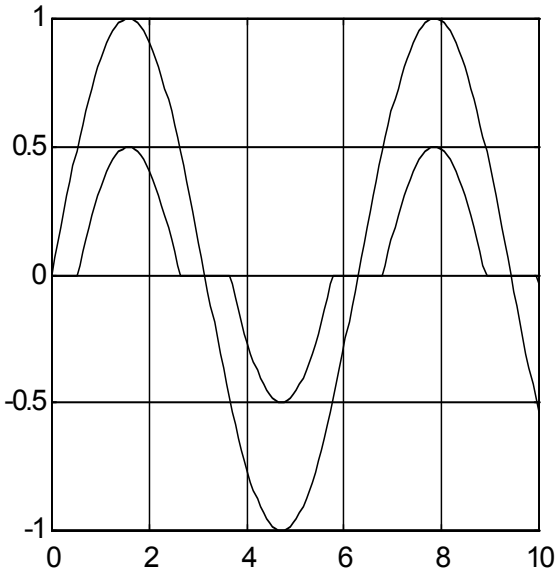


Рис. 5.4 – Перетворення синусоїди ланкою *Dead Zone*

Блок ***Coulumbic & Viscous Friction*** **(*Сухе та В‘язке Тертя*)** реалізує статичну характеристику, що описується рівнянням лінійну модель тертя:

$$y = (K \cdot |u| + y_0) \cdot \text{sign}(u), \tag{5.7}$$

де

K *(Gain)* – коефіцієнт передачі лінійної ділянки статичної характеристики;

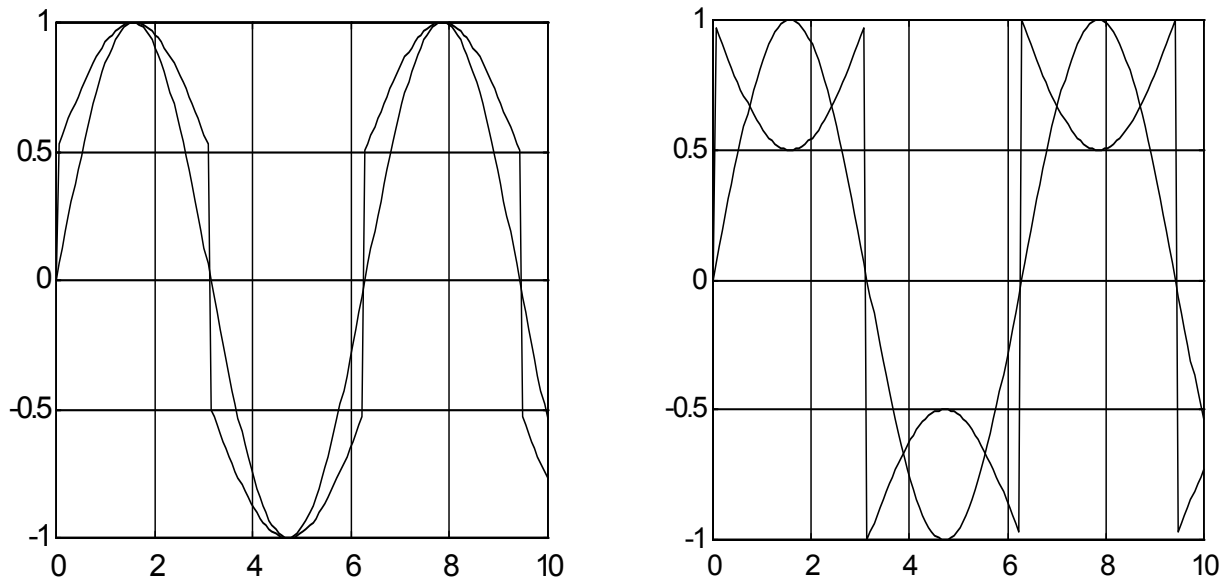
y_0 (*Offset*) – абсолютне значення вихідного сигналу в нульовій точці.

Якщо вхідним сигналом є швидкість руху деякого механізму, то вихідний сигнал можна інтерпретувати як силу або момент тертя ідеалізованої лінійної моделі цього процесу.

При $K = 0$ тертя буде сухим, тобто незалежним від величини швидкості. При $K > 0$ тертя буде в‘язким, тобто воно збільшуватимеся при підвищенні швидкості,

що має місце, наприклад, при руху в рідинному або газовому середовищах. При $K < 0$ тертя зменшується при підвищенні швидкості за рахунок виникнення підіймальної сили, що знижує зчеплення між поверхнями, які спіткаються між собою в процесі руху.

На рисунку 5.5 наведені діаграми перетворення синусоїдального вхідного сигналу ланкою *Coulumbic Friction* при додатному (*а*) і від'ємному (*б*) значеннях коефіцієнта *Gain*.



а) *б*)

Рис. 5.5 – Перетворення синусоїди ланкою *Coulumbic & Viscous Friction*:
а) *Offset* = 0.5, *Gain* = 0.5; *б*) *Offset* = 1, *Gain* = -0.5

Блок ***Quantizer* (*Квантизатор*)** квантизує вхідний сигнал за рівнем з заданою дискретою Δy (*Quantization interval*):

$$y = \text{round}(u / \Delta y). \tag{5.8}$$

Блок ***Rate Limiter* (*Обмеження Темпу*)** відтворює вхідний сигнал *u(t)* з обмеженням його першої похідної *du/dt* на рівні *R (Rising slew rate*) при збільшенні сигналу і *F (Falling slew rate*) – при його зменшенні відповідно до формули

$$y_i = \begin{cases} \Delta t \cdot \text{abs}(R) + y_{i-1} & \text{при} \quad \left|\dfrac{du}{dt}\right| > R, \\ -\Delta t \cdot \text{abs}(F) + y_{i-1} & \text{при} \quad \left|\dfrac{du}{dt}\right| < F, \\ u_i & \text{при} \quad F \le \left|\dfrac{du}{dt}\right| \le R, \end{cases} \tag{5.9}$$

як це показано на рисунку 5.6.

Значення похідної обчислюється за формулою:

$$\frac{du}{dt} \approx \frac{\Delta u}{\Delta t} = \frac{u_i - u_{i-1}}{t_i - t_{i-1}} \tag{5.10}$$

Ланка ***Backlash*** **(*Зазор, Люфт*)** моделює зазор (люфт) у кінематичній передачі:

$$y_i = \begin{cases} u - \dfrac{\delta}{2} & \text{при} \quad u - \dfrac{\delta}{2} > y_{i-1}, \\ u + \dfrac{\delta}{2} & \text{при} \quad u + \dfrac{\delta}{2} < y_{i-1}, \\ y_{i-1} & \text{при} \quad u - \dfrac{\delta}{2} \le y_{i-1} \le u + \dfrac{\delta}{2}, \end{cases} \tag{5.11}$$

де δ – величина зазору (*Deadband width)*.

Вхідним сигналом тут є положення активної маси, а вихідним – пасивної.

Установка початкових значень вхідного (*Initial input value*) і вихідного (*Initial output value*) параметрів дозволяють цілком визначити початковий стан системи. Початкове значення виходу повинне бути в межах 1/2 величини зазору; у противному випадку *Simulink* сигналізує про помилку.

Діаграма перетворення синусоїдального вхідного сигналу ланкою *Backlash* наведена на рисунку 5.7.

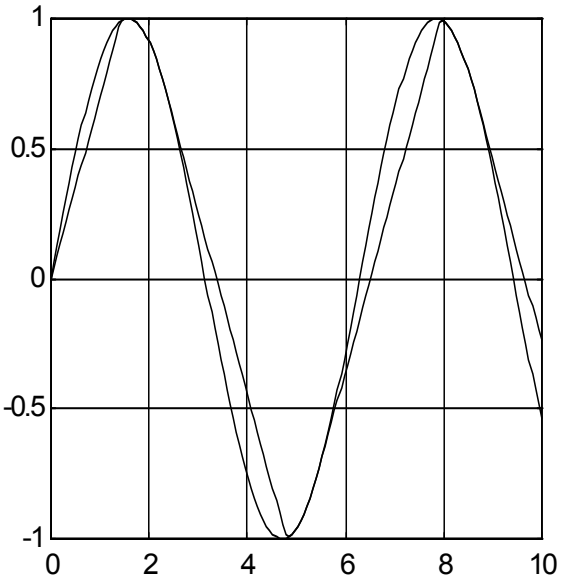


Рис. 5.6 – Перетворення синусоїди ланкою *Rate Limiter*

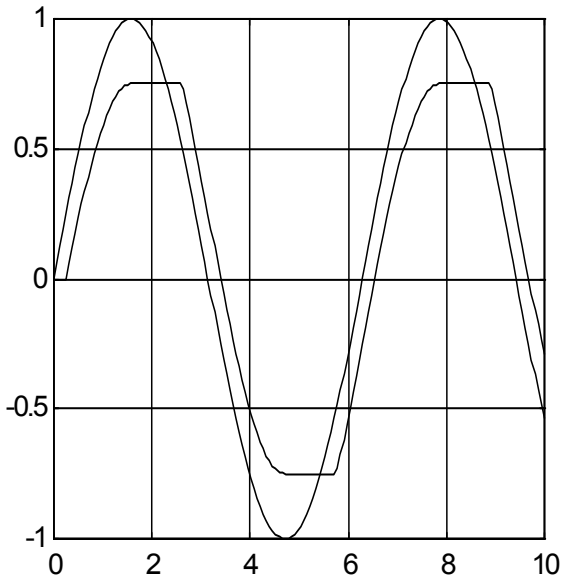
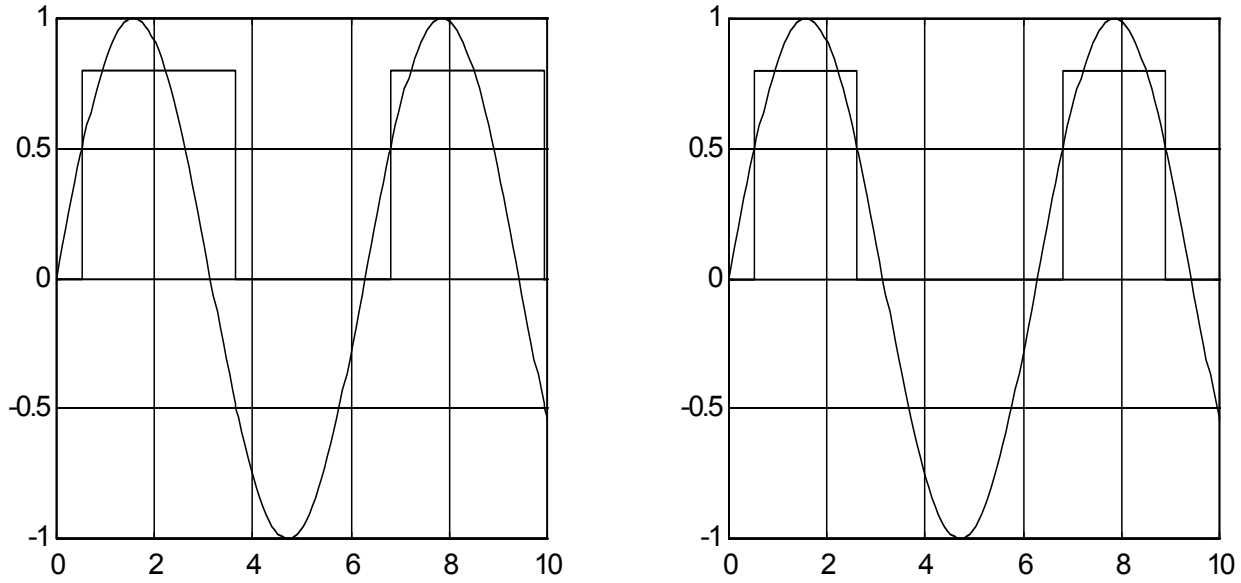


Рис. 5.7 – Перетворення синусоїди ланкою *Backlash* (δ=1)

Блок ***Relay*** **(*Реле з Гістерезисом*)** дозволяє перемикати вихід між двома заданими значеннями *Output when on* (y_{on}) і *Output when off* (y_{off}). Коли реле включено, воно залишається в цьому стану доти, поки значення вхідного сигналу не упаде до заданого значення для вимикання реле *Input for off* (x_{off}). Коли реле вимкнено, воно залишається в цьому стану доти, поки значення входу не перевищить задане значення для включення реле *Input for on* (x_{on}).

При $x_{on} > x_{off}$ моделюється реле з петлею гістерезіса, а при $x_{on} = x_{off}$ – ідеальне реле (див. рис. 5.8).



а) *б*)

Рис. 5.8 – Перетворення синусоїди ланкою *Relay*:
а) x_{on}= 0.5, x_{off}= -0.5; *б*) x_{on}= x_{off} = 0.5

Бібліотека *Nonlinear* отримує також 3 ключа: *Switch*, *Manual Switch* і *Multiport Switch*.

Блок ***Switch* (*Керований Ключ*)** пропускає на вихід один із двох вхідних сигналів, що подаються на перший і третій вхідні порти блоку в залежності від стану керуючого сигналу, що подається на другий порт, відповідно до алгоритму

$$y=\begin{cases} u_1 & при \quad u_2 \geq c \\ u_3 & при \quad u_2 < c \end{cases} \tag{5.12}$$

де c – граничне значення (*Threshold*) керуючого вхідного сигналу u_2.

Блок ***Manual Switch* (*Ручний Ключ*)** пропускає на вихід один із двох вхідних сигналів згідно з положенням перемикача, яке змінюється щигликом миші (вручну) як при підготовці до моделювання, так і в період розрахунку перехідних процесів.

Блок ***Multiport Switch* (*Багатоканальний Ключ*)** пропускає на вихід один із декількох вхідних сигналів, кількість яких задається параметром *Number of inputs*. Номер входу, що пропускається, визначається округленим до цілого значенням керуючого сигналу, який подається на перший вхід ключа.

Кусочно-лінійну апроксимацію заданої табличної нелінійності виконує блок ***Look-Up Table*** (див. підрозділ 2.1). Для того, щоб статична характеристика такої нелінійної ланки була більш плавною можна замість блоку *Look-Up Table* використати блок *MATLAB Fn*, звертаючись через нього до функції, що здійснює кусочно-кубічну інтерполяцію interp1(X,Y,u,'cubic') або інтерполяцію сплайнами interp1(X,Y,u,'spline'), де X, Y – вектори аргументів та значень табличної функції однакового розміру.

Для поліноміальної апроксимації табличної нелінійності методом найменших квадратів спочатку знаходять вектор коефіцієнтів степеневого полінома n -го порядку виконанням операції.

```
A = polyfit (X, Y, n)
```

а потім використовують визначені коефіцієнти у блоку *Polynomial*.

Досить часто при моделюванні нелінійних систем з перемиканнями використовують блок ***Hit crossing*** **(*Перетин*)** бібліотеки *Signals & Systems*, призначений для фіксації моменту перетину вхідним сигналом *и* заданого рівня *о* (*Offset Value*) в заданому напрямку: *rising* (*при наростанні*), *falling* (*при спаданні*) або *either* (*в обох напрямках*), що задається параметром *Hit crossing direction* (*Напрямок перетину*). При включеному стані вихідного порту він формує в момент, коли різниця сигналів *и-о* змінює знак на протилежний, у відповідності з обраним напрямком перетину, одиничний імпульс. При *и=о* вихідний сигнал утримує одиничне значення. У всіх інших випадках вихідний сигнал блоку дорівнює нулю.

Демонстраційна модель *hardstop* ілюструє використання блоку *Hit crossing* для моделювання тертя з урахуванням різниці між тертям покою та тертям руху і симуляції процесу миттєвого стопоріння рухомого органу.

5.3 Завдання

1) Одержати перехідні процеси перетворення синусоїдального сигналу типовими нелінійностями та статичні характеристики цих блоків.

2) Реалізувати кусочно-лінійну апроксимацію, апроксимацію методом найменших квадратів кубічну інтерполяцію та інтерполяцію кубічними сплайнами табличних нелінійностей, заданих в табл. 5.1. Порівняти час, що витрачається на розрахунок перехідних процесів при використанні цих методів.

3) Проілюструвати роботу блоку *Hit crossing*, встановленням на його вході джерела синусоїдального сигналу, а на виході – лічильника кількості перетинів, що утворюється замиканням блоку *Memory* одиничним додатним зв‘язком.

4) Пояснити роботу демонстраційної моделі *hardstop*.

5.4 Методичні вказівки та рекомендації

1) Організуйте виконання перших трьох пунктів завдання в програмному режимі.

2) Для реалізації кусочно-лінійної апроксимації табличної нелінійності скористайтесь блоком *Look-Up Table*, для апроксимації методом найменших квадратів – блоком *Polinomial Fn*, а для кубічної інтерполяції та інтерполяції кубічними сплайнами – блоком *Matlab Fn*.

3) Вектор коефіцієнтів A степеневого полінома k -го порядку для апроксимації табличної нелінійності Y(X) методом найменших квадратів знайдіть за допомогою функції polyfit:

```
A = polyfit (X, Y, k)
```

Таблиця 5.1

№ вар.	Табличні функції									
1,2	x_t	-1	1	3	5	7	9	11	13	15
	y_t	8.71	109,8	124.4	122.5	112.1	96.6	80.2	63	57.9
3,4	x_t	2	3.2	4.4	6.2	7.8	9.5	10.9	11.5	12.7
	y_t	19.9	22	30	42.1	65	99.5	120	126.8	133.4
5,6	x_t	-3.5	-1.5	0.5	2.5	4.5	6.5	8.5	10.5	12.5
	y_t	0.45	-3.09	-4.01	-3.9	-3	-1.62	-0.18	0.99	1.72
7,8	x_t	1.25	2.59	4.4	6.54	8.5	11.5	13.5	14.5	15
	y_t	3.0	5.0	7.0	8.5	9.3	9.9	10.6	11.2	11.64
9,10	x_t	-2	0	2	4	6	8	10	12	14
	y_t	7.84	7.13	6.31	5.29	4.03	2.5	0.87	-0.68	-0.79
11,12	x_t	-1.5	1	2.7	5.5	6.5	8.3	9.6	11.2	12.75
	y_t	2.45	1.12	-1	-2.1	-2.3	-1.9	-1	2	3.5
13,14	x_t	0,67	1,5	2,5	3,5	5	6,5	10	12,4	14
	y_t	110	118,7	124,5	125,2	122,5	115,1	88,3	70	61,2
15,16	x_t	0,5	2,5	4,5	6,5	8,5	10,5	12,5	14,5	16,5
	y_t	23,7	20,1	27,8	45,3	79,2	115,4	132,9	141,1	147
17,18	x_t	-2,77	-0,5	1	2	3,5	7	10	11,5	12,5
	y_t	-1,5	-3,65	-4,03	-4,0	-3,54	-1,58	0,73	1,4	1,83
19,20	x_t	0,5	2,0	3,5	5,0	6,5	8,5	9,5	11,0	12,5
	y_t	1,23	0,92	0,78	0,68	0,6	0,53	0,49	0,47	0,45
21,22	x_t	-1	1	3	5	7	9	11	13	15
	y_t	1,02	2,57	5,51	7,52	8,69	9,38	9,79	10,35	11,64
23,24	x_t	-3	0,5	1,5	2,5	4,3	6,2	7,7	9,0	11
	y_t	9,4	7,52	6,75	5,8	3,6	0,53	-1,5	-2,94	-4,4
25,26	x_t	-4	-2	0	2	4	6	8	10	12
	y_t	3,1	2,66	1,74	0,35	-1,26	-2,28	-2,07	-0,54	2,53
27,28	x_t	0	0,4	1,5	3,0	4,6	7	9,2	11,5	13
	y_t	1,47	1,26	0,99	0,82	0,7	0,57	0,5	0,46	0,44

4) Для оцінювання часу, що витрачається на розрахунок перехідних процесів, скористайтесь функціями tic і toc.

5) При виконанні п. 3 проаналізуйте три різновидності перетинів: знизу догори, згори вниз та в обох напрямках.

5.5 Контрольні запитання

1) На які групи можна розподілити нелінійні блоки?

2) Що таке типові нелінійності?

3) В яких бібліотеках знаходяться блоки, що роблять систему нелінійною?

4) Які засоби моделювання табличних нелінійностей Ви знаєте?

5) Як реалізувати кусочно-лінійну апроксимацію табличної нелінійності?

6) Як здійснити апроксимацію табличної нелінійності степеневим поліномом методом найменших квадратів?

7) Як скористуватись функціями інтерполювання при моделюванні нелінійних функцій, заданих у вигляді таблиць?

6 Лабораторна робота №6
СТВОРЕННЯ ПІДСИСТЕМ ТА ЇХ МАСКУВАННЯ

Підсистеми використовують для зменшення кількості блоків в моделі, що сприяє її компактності, та для об‘єднання фрагментів моделі, що виконують деяку спільну функцію, в єдине ціле.

6.1 Створення підсистем

Створити підсистему можна двома способами.

Перший з них полягає у тому, що в вікно моделі вставляється блок *Subsystem*. При подвійному щиглику мишею по його піктограмі відчиняється вікно, в якому збирається модель підсистеми, в якій до входів приєднують вхідні (*In*), а до виходів – вихідні (*Out*) порти. Після закривання цього вікна блок *Subsystem* має кількість входів та виходів відповідну до кількості встановлених в моделі підсистеми вхідних і вихідних портів. Порядок, в якому вони розташовані, також збігається з нумерацією портів.

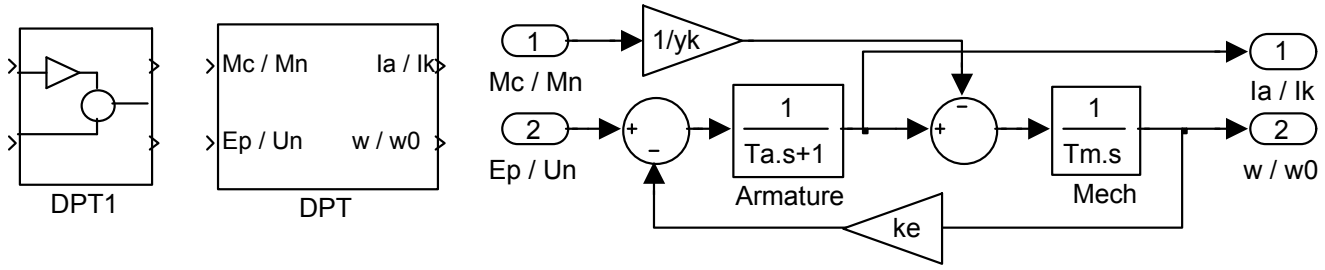
Згідно з другим способом вже існуючий фрагмент моделі виділяється мишею оточенням його у прямокутник та об‘єднується в підсистему командою *Edit → Create Subsystem* (*^G*). В цьому разі вхідні та вихідні порти додаються до фрагменту, що утворює підсистему, автоматично. Якщо автоматична установка портів, їх нумерація або їхні імена не влаштовують користувача, він може виконати редагування підсистеми.

Блок *Subsystem* може бути відображеним на екрані в режимах *Hide Port Labels* (імена портів сховані) та *Show Port Labels* (з зображенням імен портів), які обираються користувачем через меню *Format*. Якщо підсистема має декілька входів та виходів, то стиль *Show Port Labels* є кращим, бо він дозволяє безпомилково з‘єднувати блок *Subsystem* з іншими блоками, не звіряючись для

перевірки з моделлю підсистеми. Для наочності імена портів бажано називати іменами вхідних та вихідних змінних моделі. Створену підсистему також бажано перейменувати у відповідності з назвою об‘єкту, структуру якого вона відображує.

Для приклада на рис. 6.1 показана у згорнутій (*а, б*) та розгорнутій (*в*) формах підсистема, яка отримує у собі модель двигуна постійного струму з регулюванням у колі якоря, подану у в.о.

Щоб розгорнути незамасковану підсистему, достатньо зробити подвійний щиглик на піктограмі згорнутої підсистему. В розгорнутому вигляді підсистему можна коригувати, якщо вона не занесена до якої-небудь бібліотеки. Для коригування бібліотечної підсистеми спочатку треба розірвати її зв‘язок з бібліотекою (*Edit → Break Library Link*). Після редагування стару підсистему в бібліотеці можна замінити новою, для чого бібліотеку треба перевести у стан *Unlock Library.*



а) *б*)

Рис. 6.1 – Приклад згорнутих (*а*) та розгорнутої (*б*) підсистем

6.2 Маскування підсистем

Будь-яку підсистему можна замаскувати.

Існують 2 рівня маскування.

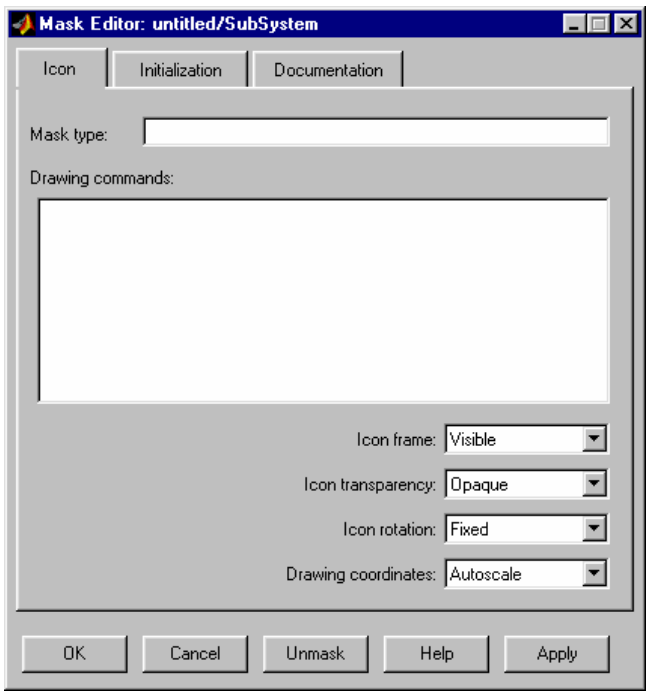
Перший рівень полягає тільки у заміні іконки блоку. При такому способі подвійний щиглик по іконці як і для незамаскованої підсистеми відчиняє вікно з розгорнутою моделлю підсистеми.

При використанні другого рівня маскування підсистема стає схожою на звичайний блок, тобто при подвійному щиглику на його піктограмі відкривається не розгорнута модель підсистеми, а вікно введення параметрів *Block Parameters*. Для того, щоб побачити розгорнуту модель треба „зазирнути під маску” виконанням команди *Edit → Look Under Mask* (*^U*).

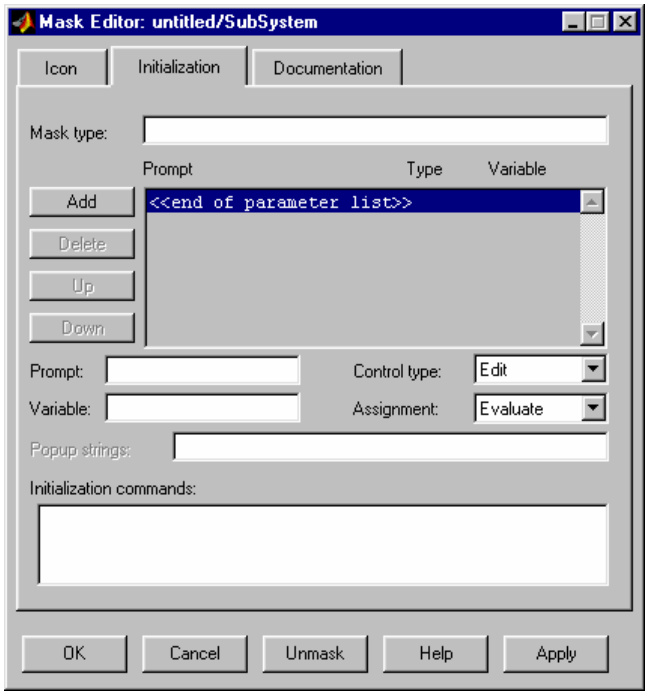
Маскування підсистеми починають командою *Edit → Mask Subsystem* (*^M*), на яку система реагує відкриттям вікна маскування *Mask Editor*, що має 3 вкладки (панелі): *Icon*, *Initialization* та *Documentation*, які показані на рис. 6.2.

В поле *Mask type* необхідно ввести тип блока. Пізніше цей тип з поміткою (*Mask*) будет відображуватися у вікні введення параметрів замаскованого блока.
В поле *Block description* панелі *Documentation* можна ввести краткий коментар про призначення блока та про його параметри, який буде відображений у вікні *Block Parameters*, а в поле *Block Help* – більш докладний опис блока, який буде

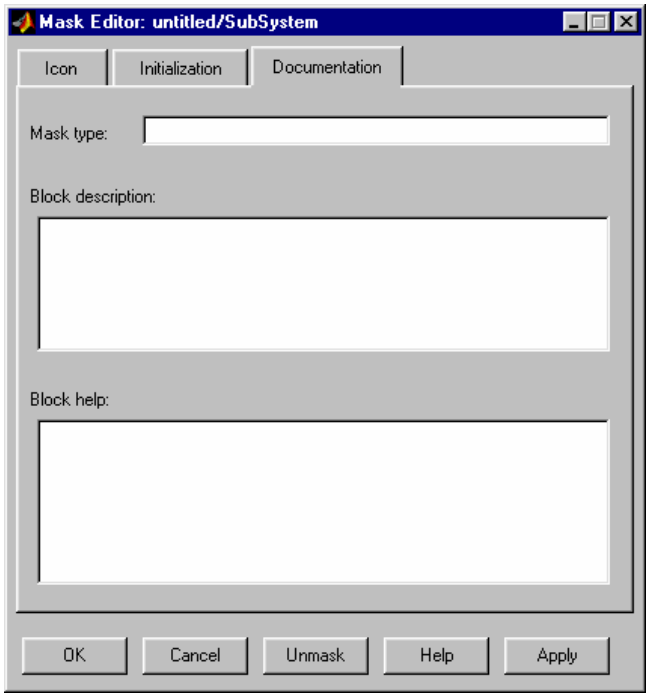
виводитися на екран в *html*-форматі через при натисканні клавіші *Help* в одноименному вікні.



а)



б)



в)

Рис. 6.2 – Вкладки вікна маскування підсистем

Для того, щоб параметри підсистеми можна було б визначати в одному вікні, не користуючись розширеною моделлю, у вкладці *Initialization* треба після натискання клавіші *Add* визначати в полі *Prompt* запрошення на введення параметра, а в полі *Variable* – ім‘я змінної, яка отримає значення після введення його у відповідь на запрошення. У вікні *Block Parameters* запрошення будуть розташовані у порядку, зворотному до порядку їх визначення у вікні *Mask Editor*. Цей порядок можна змінити клавішами *Up* і *Down*. Зайві параметри можна знищити клавішею *Delete*.

Для кожного параметра спосіб, яким буде обиратися або вводитися значення параметру, визначається вибором у полі *Control type*. У ньому пропонуються наступні засоби:

Edit – введення константи, змінної або виразу;

Checkbox – вибір одного з двох альтернативних варіантів *on* або *off*, перший з яких діє при встановленому „прапорці”, тобто при поміченому „галочкою” полі параметру, а другий – при невстановленому„прапорці”, тобто при відсутності помітки в полі параметру;

Popup – вибір варіанта з випадаючого меню.

При використанні способу *Popup* треба у полі *Popup strings* ввести варіанти строк (без апострофів) для вибору, відокремлюючи їх одну від одної символом „|”.

Одним з бібліотечних блоків, який має усі три перелічені вище типи визначення параметрів є джерело *From Workspace*, вікно якого показано на рис. 6.3.

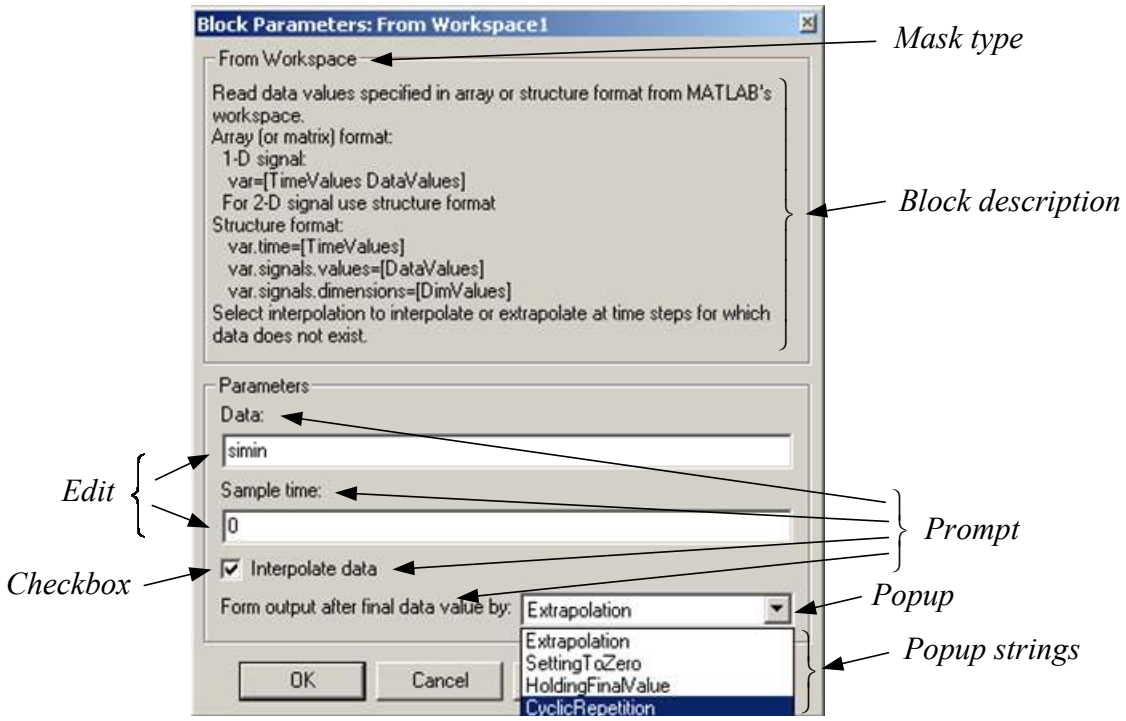


Рис. 6.3 – Вікно введення параметрів блока *From Workspace*

На прикладі цього блоку показано, які поля вікна *Mask editor* та у який спосіб формують вікно введення параметрів. Зауважимо тільки, що блок *From Workspace* є вбудованим, а не замаскованим, так що подивитися його вікно маскування не представляється можливим.

Значення обраного або введеного параметра може використовуватись як у чисельній формі, так і у форматі рядка символів. Тип операції присвоювання визначається станом поля *Assigned*: *Evaluate* (обчислити значення) або *Literal* (інтерпретувати значення як рядок символів).

Спосіб обчислення значення параметра в режимі *Evaluate* пояснюється таблицею 6.1.

Таблиця 6.1

Control type	***Numerical value***
Edit	Введена константа, обчислене значення введеної змінної або введеного виразу
Checkbox	1 – при встановленому „прапорці” (стан *on*) 0 – при відсутності „прапорця” (стан *off*)
Popup	Ціле число, що відповідає порядковому номеру (зверху вниз) обраного варіанту

Як видно з рис. 6.3, вигляд вікна *Block Parameters* замаскованої підсистеми визначається змістом комірок *Mask type*, *Block description*, *Prompt*, *Variable* та *Control type*. Для прикладу в табл. 6.2 показані параметри маскування підсистеми з чотирма параметрами., зображеної на рис. 6.1, а на рис. 6.4 – одержане в результаті проініціалізоване вікно введення параметрів замаскованої підсистеми.

Таблиця 6.2

Mask type	Двигун постійного струму
Block description	Shunt-connected DC machine, p.u. (wb=w0, Ib=Ikz, Eb=Un)
***Prompt*1**	Taking into account feedback of EMF:
***Variable*1**	ke
***Control type*1**	*Checkbox*
***Prompt*2**	Mkz/Mn:
***Variable*2**	yk
***Control type*2**	*Edit*
***Prompt*3**	Electromagnetic time constant:
***Variable*3**	Ta
***Control type*3**	*Edit*
***Prompt*4**	Electromechanical time constant:
***Variable*4**	Tm
***Control type*4**	*Edit*

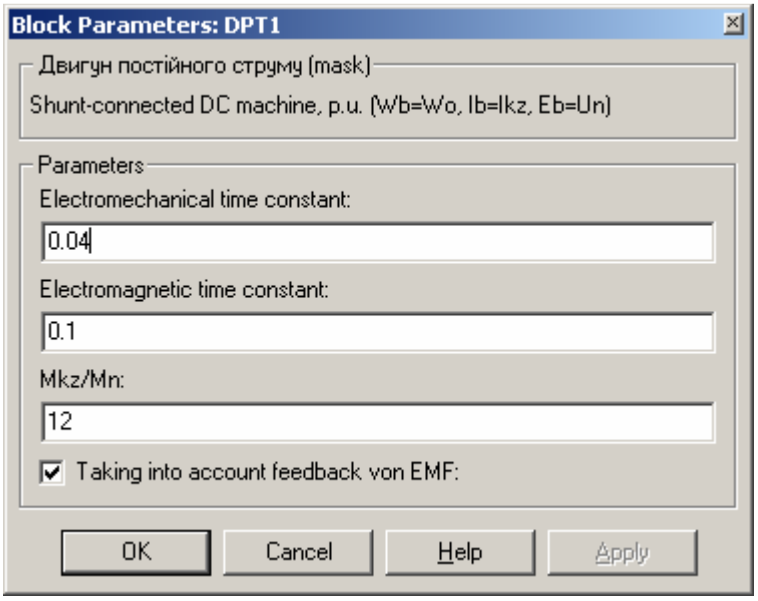


Рис. 6.4 – Вікно введення параметрів замаскованої підсистеми рис. 6.1

При визначенні запрошень (*Prompt*) кириличні шрифти краще не використовувати, бо це може привести до непередбачених результатів. В інших текстових полях для більшої надійності також краще користуватися латиницею.

Панель *Icon* призначена для зміни піктограми (іконки) підсистеми та її властивостей.

Зовнішнім видом іконки керують команди, занесені в поле *Drawing commands*. Принципово у піктограмі блоку може бути виведеним будь-який текст, передавальна функція, графік статичної або дина-мічної характеристики та не дуже складне графічне зображення.

Для виведення текстів можна використовувати один з наступних операторів:

disp (’text’) або disp (StringVariableName)
text (x, y, ’text’) або text (x, y, StringVariableName)
text (x, y, ’text’, ’HorizontalAligment’, HorAl, ’VerticalAligment’, VertAl)
fprintf (’text’) або fprintf (’format’, VariableName)
port_label (port_type, port_number, label)

Для розподілу тексту на рядки в операторах disp і text можна використовувати комбінацію символів *n*, наприклад, disp (’Mask\nSubsystem’).

В операторах text параметри x,y уявляють собою координати точки в піктограмі блоку, до якої прив‘язується текст. Для явного визначення системи координат, що використовується при зображенні іконки можна скористатися оператором

plot (xlb, ylb, xrt, yrt)

де xlb, ylb і xrt, yrt – координати лівого нижнього та правого верхнього кутів графічного вікна. Використання операторів xlim, ylim та axes [] з цією ж метою у полі *Drawing commands* не припустимо.

Стиль прив‘язки визначається параметрами HorAl (вирівнювання по горизонталі) та VertAl (вирівнювання по вертикалі), які можуть приймати такі значення: ’center’, ’left’, ’right’, і ’middle’, ’base’, ’bottom’, ’cap’, ’top’ відповідно. За замовчанням діють опції ’left’ та ’middle’. Результат використання перелічених опцій пояснюється на рис. 6.5.

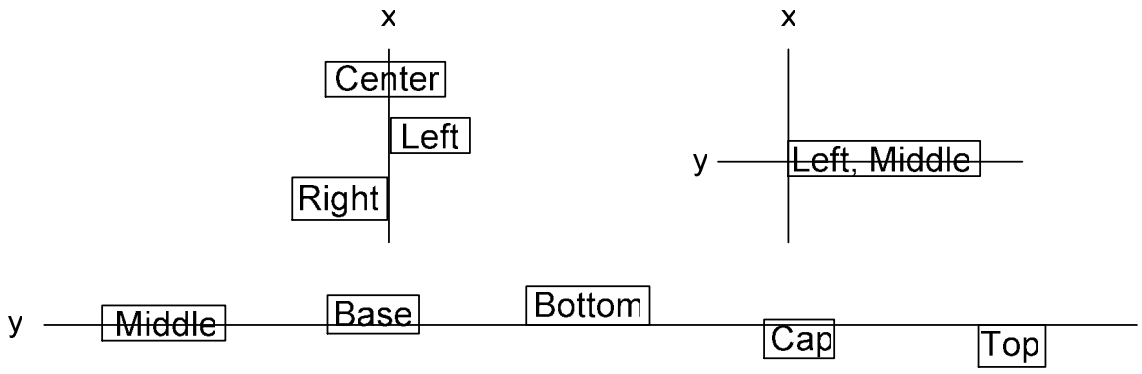


Рис. 6.5. Демонстрація дії опцій вирівнювання

При застосуванні команди port_label параметр port_type може приймати значення 'input' або 'output', а параметр label – будь-якого рядка символів, наприклад, в результаті виконання команди

port_label ('output', 2, 'a')

другий вихідний порт згорнутої підсистеми буде помічено символом a, незалежно від імені цього порта у розгорнутій підсистемі.

Для зображення на піктограмі блока його передавальної функції можна застосовувати одну із наступних команд:

dpoly (num, den) або dpoly (num, den, char)

droots (z, p, k) або droots (z, p, k, char)

Команда dpoly виводить передавальну функцію в поліноміальній формі, а droots – у вигляді розкладення на нулі-полюси (див. опис блоків *Transfer Function* і *Zero-Pole-Gain* в підрозділі 3.2). Необов'язковий символьний параметр char має за замовчанням значення 's'. Для виведення дискретних ПФ його можна змінити на 'z' або 'z-'. (див. опис блоків *Discrete Transfer Function* і *Discrete Filter* в підрозділі 4.2).

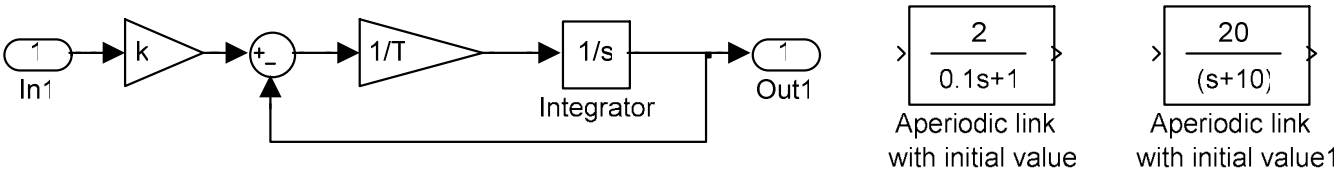
Параметри команд dpoly і droots звичайно визначаються через параметри підсистеми у полі *Initialization commands* вкладки *Initialization*. Для прикладу на рис. 6.6 показана у розгорнутому та згорнутому виглядах підсистема *Aperiodic link with initial value* (*Аперіодична ланка з початковими умовами*). На рис. 6.6, *б* іконка замаскованої підсистеми сформована командою dpoly, а на рис 6.6, *в* – командою droots, для чого у полі *Initialization commands* записані оператори присвоювання

num = [k]; den = [T 1];

і

z = [ ]; p = -1/T; K = k/T;

відповідно.



а) *б*) *в*)

Рис. 6.6. Розгорнута (*а*) та згорнуті (*б*, *в*) підсистеми *Aperiodic link with initial value*

Як бачимо параметри піктограм, сформованих у такий спосіб виводяться тільки у вигляді констант. Для того, щоб результати операцій присвоювання не відображувались у командному вікні, пасля кожного з операторів треба поставити точку з крапкою (;).

Команди ініціалізації виконуються в таких випадках:

- при завантаженні моделі;
- при старті моделювання;
- при повороті блоку;
- в усіх випадках, коли треба заново зобразити іконку блоку, а команди зображення використовують результати команд ініціалізації.

Для виведення на піктограмі графічних зображень, заданих координатами вузлових точок, можна скористуватись командами

```
plot (y)    plot (x, y)    plot (x1, y1, x2, y2, …, xN, yN)
```

де x, y – вектори абсцис та ординат табличної функції, або

```
plot (xlb, ylb, xrt, yrt, y)    plot (xlb, ylb, xrt, yrt, x, y)
plot (xlb, ylb, xrt, yrt, x1, y1, x2, y2, …, xN, yN)
```

Якщо аргументи x табличної функції не задані, то по осі абсцис відкладаються індекси вектора y. Остання група операторів plot явно визначає систему координат.

Як приклад, на рис. 6.7 подана у розгорнутому та згорнутому виглядах підсистема *Comparator* (*Компаратор*), у якої іконку сформовано оператором

```
plot ([-1 1], [0 0], [0 0], [0 1.2], [x1 x1 1], [0 1 1])
```

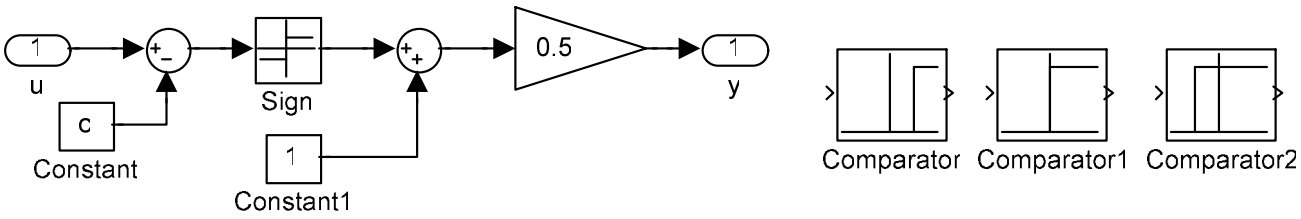
Параметр x1 визначено командою ініціалазації

```
if c<0, x1=-0.5; elseif  c>0, x1=0.5; else x1=0; end
```

внаслідок виконання якої іконка в залежності від знаку аргумента c має вигляд *б*, *в* або *г*.

Іконки, які уявляють собою досить прості графічні зображення, що складаються з обмеженої кількості відрізків прямих, можна створити за допомогою утіліти iconedit. При виклику її з командного рядка *MATLAB* на екрані по черзі з'являються 2 запити: *Name of block diagram* (*Ім'я моделі*) і *Name of block* (*Ім'я підсистеми*). Після відповіді на ці запроси на екрані відчиняється графічне вікно, у якому можна намалювати бажане зображення, фіксуючи за допомогою миші вузлові точки рисунку та використовуючи наступні клавішні команди:

d=delete – знищити останню з зафіксованих точок, *n=new pt* – почати нову лінію і *q=quit* – завершити створення іконки.



а) *б*) *в*) *г*)

Рис. 6.7. Розгорнута (*а*) та згорнуті (*б*, *в*, *г*) підсистеми *Comparator*

Для зображення зафарбованих плоских замкнених фігур використовують команду

```
patch (x, y, [r g b])
```

де x, y – вектори абсцис та ординат точок замкненої фігури, [r g b]) – її колір, визначений інтенсивністю трьох кольорів: червоного, зеленого та синього у вигляді констант зі значеннями в діапазоні від 0 до 1. За замовчанням (при відсутньості останнього параметру) фігура зафарбовується кольором „заднього фону” (*Background color*), тобто тим кольором, яким зображені контури іконки. Наприклад, оператор

```
patch ([ -1 0 1], [0 1 0], [0 1 1])
```

створює піктограму у вигляді трикутника, зафарбованого бірюзовим кольором, зображену (у чорно-білому варіанті) на рис. 6.8.

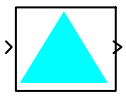


Рис. 6.8. Замаскована підсистема, іконка якої створена командою patch

Для зображення на іконці малюнків, створених іншими програмними продуктами, треба скористуватися командою

```
image (imread ('FileName.Extension'))
```

де FileName – ім‘я графічного файла, а Extension – його розширення.

MATLAB підтримує такі типи графічних файлів: **.bmp* (*Bitmap*), **.hdf* (*Hierarchical Data Format*), **.jpg* (*Joint Photographic Experts Group*), **.pcx* (*Paintbrush*), **.png* (*Portable Network Graphics*), **.tif* (*Tagged Image File Format*), **.xwd* (*X Window Dump*).

Властивостями піктограми керують опції, імена яких розташовані нижче поля *Drawing commands* вкладки *Icon*, а значення обираються за допомогою *popup-menu*. Інформація про ці опції наведена в табл. 6.3.

Таблиця 6.3

Опція	Значення	Сенс
Icon Frame (*Рамка Іконки*)	*Visible*	Рамку видно
	Invisible	Рамку не видно
Icon Transparency (*Прозорість Іконки*)	*Opaque*	Іконка непрозора
	Transparent	Іконка прозора
Icon Rotation (*Обертання Іконки*)	*Fixed*	При повороті блоку іконка не обертається
	Rotates	При повороті блоку іконка обертається
Drawing Coordinates (*Система Координат*)	*Autoscale*	$x_{min} = \min(\mathbf{X}),\quad x_{max} = \max(\mathbf{X}),$ $y_{min} = \min(\mathbf{Y}),\quad y_{max} = \max(\mathbf{Y})$
	Normalized	$x_{min} = 0,\quad x_{max} = 1,\quad y_{min} = 0,\quad y_{max} = 1$
	Pixel	$x_{min} = 0,\quad x_{max} = block\ width,$ $y_{min} = 0,\quad y_{max} = block\ hight$

Прикладом блоку з невидимою рамкою є блок *Manual Switch*.

Якщо зробити іконку прозорою, то крізь неї можуть бути видні, наприклад, імена портів.

Ширину та висоту блока в пікселах при використанні стилю *Pixel* при визначенні системи координат можна розрахувати у полі *Drawing commands* за допомогою операторів

```
pos = get_param (gcb, 'Position'));
width = pos(3) – pos(1);
hight = pos(4) – pos(2);
```

Відмінити маскування можна клавішею *Unmask*. При виконанні команди *Edit → Mask Subsystem* (*^M*) вікно маскування відновлює свої попередні установки.

6.3 Створення „кнопок”, що керують процесом виконання модельного експерименту

За допомогою блоку *Subsystem* можна створити „кнопки”, що керують процесом виконання модельного експерименту, тобто такі відокремлені блоки (без вхідних та вихідних портів), подвійний щиглик мишею на іконці яких дозволяє виконувати бажані оператори, функції або команди пакета *MATLAB*.

Звичайно користувачі-початківці створюють моделі, параметри яких задають константами. В такому разі для досліджень впливу зміни якого-небудь параметру на властивості досліджуваного об‘єкта доводиться відкривати вікна визначення параметрів усіх блоків, до яких входить варійований параметр, вручну змінювати значення цього параметру, виконувати моделювання через *Simulink*-меню і запам‘ятовувати його результати. Таку послідовність дій багатократно повторюють, а потім обробляють результати модельного експерименту. Викладену методику не можна назвати раціональною при наявності розвинутої алгоритмічної мови і можливості виконувати моделювання в програмному

режимі.

Більш ефективно дотримуватися такої методики:

- параметри моделі задавати іменами змінних;
- для ініціалізації моделі створити файл даних, в якому задаються або обчислюються значення цих змінних;
- для виконання файла даних, не виходячи з вікна *Simulink*-моделі, створити „кнопку” з умовною назвою *Init*;
- при налагодженні моделі виконувати моделювання досліджуваної системи після процесу ініціалізації через *Simulink*-меню, фіксуючи результати блоками *Scope*;
- після того, як ви впевнитеся у працездатності моделі, треба розробити план модельного експерименту і *MATLAB*-програму або декілька програм, що його реалізують;
- для виконання розроблених програм, не виходячи з вікна *Simulink*-моделі, створити „кнопки” з умовною назвою *Run-Plot*;
- наприкінці можна створити кнопку, активізація якої виводить в окремому вікні інформацію про модель та рекомендації щодо її застосування.

Модель, параметри якої задані іменами змінних, а не числами є більш наочною. При програмному визначенні параметрів зменшується час коригування параметрів та можливість помилок, особливо тоді, коли один і той же параметр, як, наприклад, період дискретності, треба визначити в декількох блоках.

Програма модельного експерименту може отримувати у своєму складі цикли, в яких змінюються варійовані параметри або параметри, що керують ключовими елементами моделі. В цих циклах можна здійснювати розрахунок перехідних процесів, частотних характеристик, тощо, та обробляти отримані результати, зокрема, виводити їх у вигляді графіків.

Створення „кнопок” допомагає пов‘язати модель з програмами, які її ініціюють та досліджують.

Цей процес можна виконати у такому порядку:

1) відкрити бібліотеку *Subsystems* і перетягти з неї у вікно досліджуваної моделі блок *Subsystem*;

2) відкрити блок *Subsystem*, знищити його вхідний та вихідний порти і лінію зв‘язку поміж ними;

3) активізувати контекстуальне меню щигликом правої кнопки миші по піктограмі блоку та обрати в ньому функцію *Block Properties*, що приведе до відчинення відповідного вікна;

4) у рядку *Open function* відчиненого вікна набрати ім‘я файла або послідовність *MATLAB*-команд, які ви бажаєте виконати при натисканні створеної кнопки, після чого натиснути клавішу *OK*;

5) виділити блок і виконати з ним послідовно операції *Hide Name* і *Show Drop Shadow* функції *Format Simulink*-меню;

6) замаскувати підсистему (^*M*) виведенням в її іконці назви „кнопки”, наприклад, disp (‘Init’), остання команда набирається в полі *Drawing commands*

вікна маскування;

7) Зафарбувати „кнопку” яким-небудь (бажано світлим) кольором, наприклад, за допомогою операцій *Format → Background Color → Yellow.*

Власне кажучи, необхідними є тільки перша і третя операції. Усі останні дії спрямовані на те, щоб придати „кнопці” презентабельний вигляд (див. рис. 6.9).

Рис. 6.9. Вигляд „кнопки” *Init*, створеної за наведеною методикою

Для створення інформаційної „кнопки” після виконання пункту 2, тобто після знищення портів, треба у звільненому вікні набрати текст, який пояснює призначення моделі, або якусь іншу інформацію. Пункти 3 і 4 виконувати не треба, а в пункті 6, тобто при маскуванні підсистеми, треба вивести в іконці „кнопки” відповідний текст, наприклад, ‘Info’ або ‘?’.

6.4 Завдання

1) Створити незамасковану підсистему задатчика інтенсивності (ЗІ) або задатчика положення (ЗП) у форматі *Show Port Label*.

2) Створити замасковану підсистему регулятора з обмеженням вихідного сигналу. В іконці відобразити передавальну функцію регулятора та графічне зображення характеристики вхід-вихід блоку *Saturation*.

3) Створити замасковану підсистему фільтра на основі блоків *Transfer Function* або *Zero-Pole-Gain*. В іконці відобразити перехідну функцію або амплітудно-частотну характеристику (АЧХ).

4) Створити замасковані підсистеми „Обмеження координат” та „Зона нечутливості”, у яких рівні обмеження та межі зони задаються не параметрами, а вхідними сигналами.

5) Створити інформаційні „кнопки” та „кнопки” для керування моделями, розробленими при виконанні попередніх лабораторних робіт за вказівкою викладача.

Варіанти завдань до пунктів 1-4 відображені у табл. 6.4.

Таблиця 6.4

№ варі-антів	Задаваль-ний пристрій	Регулятор з обмеженням	Фільтр			Нелінійні блоки
		Тип	Тип	Маск.блок	Іконка	Спосіб
1, 5, 9, 13	І-ЗІ	Дискретний ПІ	Бартер-ворта	*Transfer Function*	Перехідна функція	*MATLAB Function*
2, 6, 10, 14	$І^2$-ЗІ	Аналоговий ІФ	Бесселя	*Zero-Pole-Gain*	АЧХ	Розгорнута структура
3, 7 11, 15	ІПД-ЗІ	Дискретний ПІ	Бартер-ворта	*Zero-Pole-Gain*	Перехідна функція	*MATLAB Function*
4, 8,	ЗП	Аналоговий	Бесселя	*Transfer*	АЧХ	Розгорнута

12, 16		ІФ		*Function*		структура

Перевірте працездатність створених підсистем та кнопок. Підсистеми занесіть у власні бібліотеки.

6.5 Методичні вказівки та рекомендації

1) Для створення задавальних пристроїв скористуйтеся моделями, поданими на рис. 6.10.

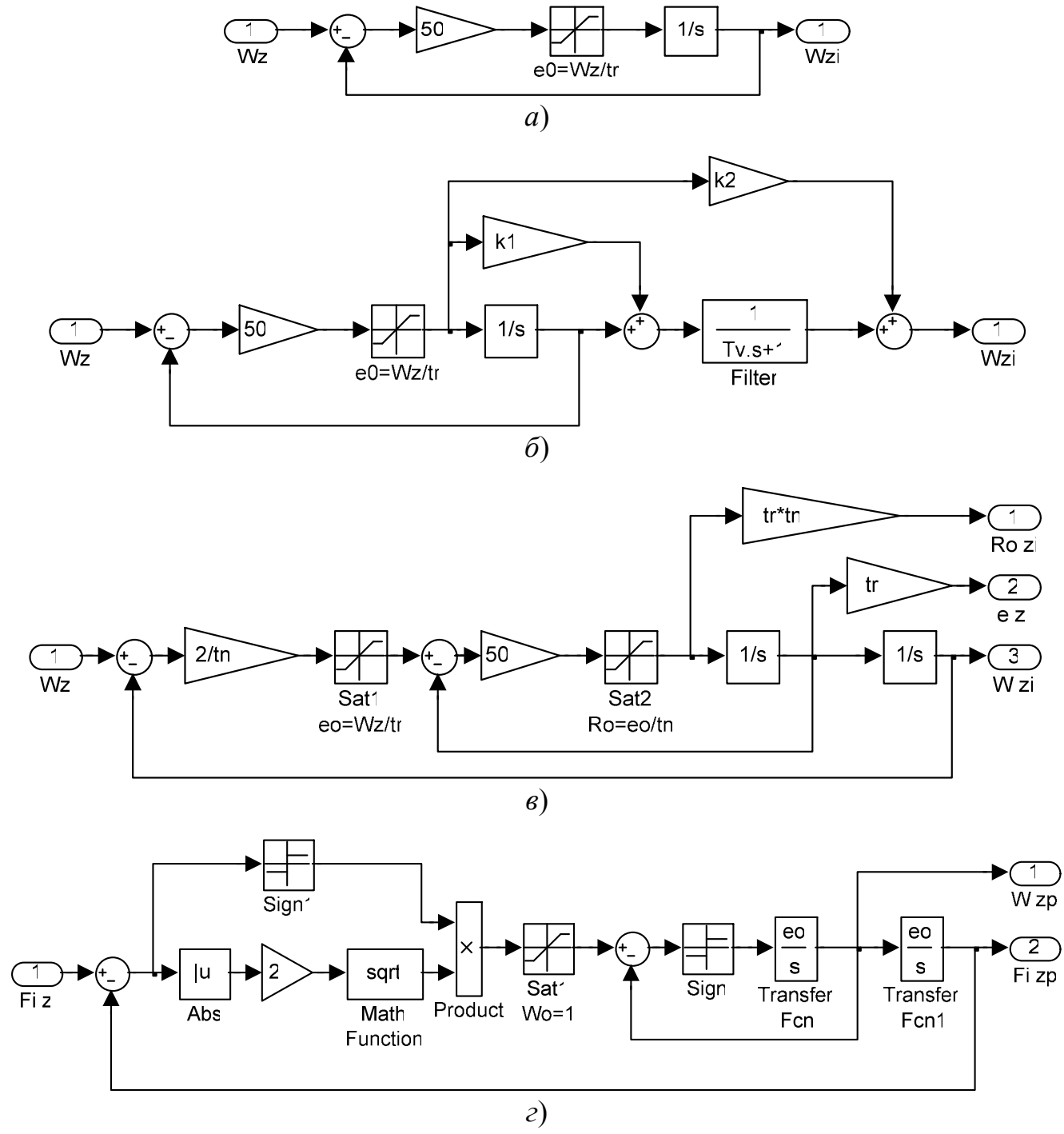


Рис. 6.10. Моделі задавальних пристроїв:
а) І-ЗІ, *б*) ІПД-ЗІ, *в*) $І^2$-ЗІ, *г*) ЗП

На моделях позначені такі параметри:

Wz – завдання на усталену швидкість;

Fiz – завдання на усталене положення;

Ro, eo, Wo – обмеження на ривок, прискорення та швидкість відповідно;

tr – час наростання завдання на швидкість до заданого значення або до рівня обмеження;

tn – час наростання прискорення до рівня обмеження.

Для перевірки працездатності моделей скористайтесь такими параметрами: Wz=1; Fiz=1.2; tr=100; tn=20; Wo=1; eo=Wo/tr; Ro=eo/tn; Tv=4; k2=b2/Tv; k1=b1-k2; b2=50; b1=10.

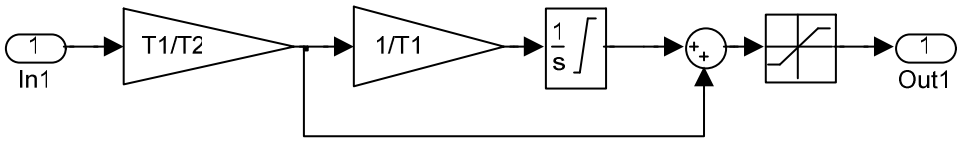
Для моделювання ЗП використайте метод *ode5* з кроком 0.1, для моделювання ЗІ – метод *ode23s*.

2) Для створення аналогових регуляторів з передавальними функціями

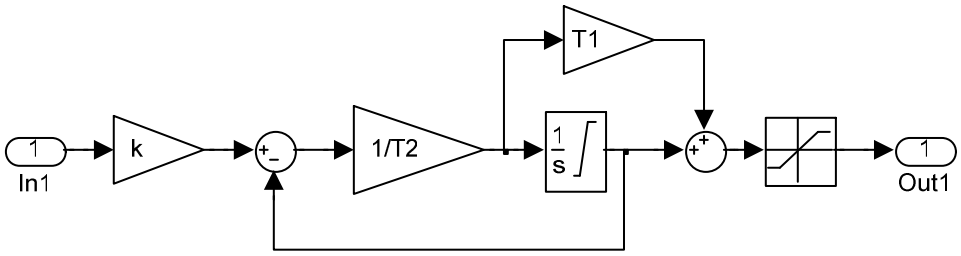
$$W_{\text{ПІ}}(s)=\frac{T_1 s+1}{T_2 s}=\frac{T_1}{T_2}+\frac{1}{T_2 s}=\frac{T_1}{T_2}\left(1+\frac{1}{T_1 s}\right),$$

$$W_{\text{ПІ}}(s)=k\frac{T_1 s+1}{T_2 s+1}.$$

та обмеженням вихідного сигналу скористайтесь деталізованими моделями, наведеними на рис. 6.11.



а)



б)

Рис. 6.11. Моделі регуляторів з обмеженням вихідного сигналу:
а) ПІ; *б*)ІФ

Рівні обмеження інтеграторів та блоків *Saturation* повинні співпадати з рівнем обмеження регулятора.

Для створення відповідних дискретних регуляторів треба в моделях рис. 6.11 замінити аналоговий інтегратор дискретним.

При перевірці працездатності та адекватності розроблених регуляторів скористайтесь такими параметрами: $T_1=2$; $T_2=4$; $k=1$; $y_{\max}=0.75$;

$y_{\min} = -0.75$. На входах моделей за допомогою блока *Step* сформуйте сигнал, який при $t = 5$ змінює своє значення з 1 на -1.

3) У перелічених в табл. 6.1 фільтрів параметрами є порядок фільтра n та частота пропускання w0. Створити фільтри можна за допомогою маскування блоків *Transfer Function* або *Zero-Pole-Gain*, сформувавши параметри цих блоків і масиви даних для зображення іконки у полі *Initialization commands*.

Фільтр Баттерворта можна створити послідовністю команд

```
[num, den] = butter (n, w0, 's');
[z, p, K] = tf2zp (num, den);
```

а фільтр Бесселя –

```
[z, pn, K] = besselap (n);
p = pn * w0;
[num, den] = zp2tf ([z, p, K]);
```

Для побудови перехідних функцій треба задати вектор часу

```
t = linspace (0, 20/w0, 21);
```

та розрахувати реакцію блоку на одиничний стрибок оператором

```
y = step (num, den, t);
```

Для зображення АЧХ треба задати вектор частот

```
w = linspace (0, w0*5);
```

та розрахувати вектор амплітуд:

```
A = bode (num, den, w);
```

Вивід графіків у вікно іконки блоку здійснюється однією з команд

```
plot (t, y)
plot (w, A),
```

що набираються у полі *Drawing commands* вкладки *Icon*.

4) При структурному способі створення підсистеми „Змінне обмеження координат" скористуйтеся блоком *MaxMin* з двома входами.

При структурному способі створення підсистеми „Змінна зона нечутливості" обміркуйте, у який спосіб можна отримати цю типову нелінійність із нелінійності „Обмеження координат".

Для перевірки адекватності створених підсистем скористайтеся моделлю, зображеною на рис. 6.12.

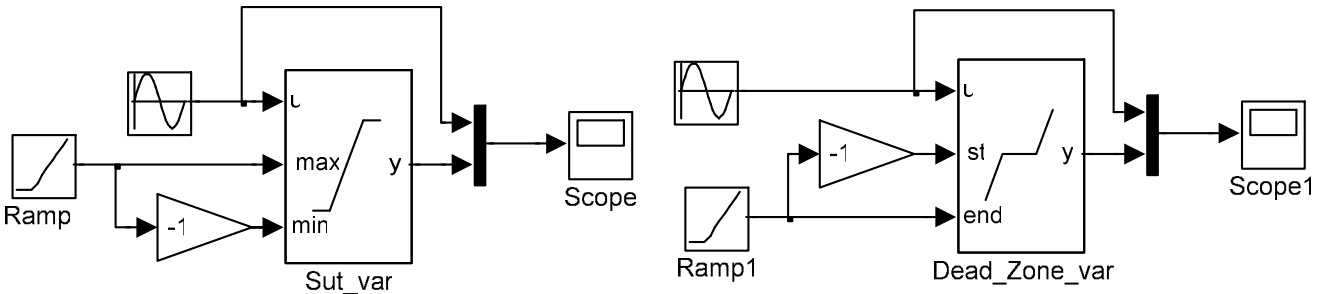


Рис. 6.12. Модель для перевірки адекватності підсистем
„Змінне обмеження координат” та „Змінна зона нечутливості”

Спробуйте створити для підсистем, розроблених при виконанні четвертого пункту завдання, іконки, подібні до показаних на рис. 6.12.

6.6 Контрольні запитання

1) Які типи підсистем Ви знаєте?
2) Чим відрізняється замаскована підсистема від незамаскованої?
3) Як замаскувати підсистему?
4) Як змінити піктограму блоку?
5) Що можна відображати в іконках?
6) Як організувати введення параметрів замаскованої підсистеми?
7) Як визначити „допомогу” для користування замаскованою підсистемою?
8) Як створити „кнопку”, пов‘язану з моделлю?
9) Для чого доцільно використовувати при моделюванні віртуальні „кнопки”?

ЛІТЕРАТУРА

1. Потемкин В.Г., Рудаков П.И. Система МАТЛАБ 5 для студентов. – 2-е изд., испр. и дополн. – М.: ДИАЛОГ-МИФИ, 1999. – 448 с.
2. Лазарев Ю.Ф. МАТЛАБ 5.х. – К.: Издательская группа BHV, 2000.– 384с. (Серия “Библиотека студента”)
3. Мартынов Н.Н., Иванов А.П. МАТЛАБ 5.х. Вычисления, визуализация, программирование. – М.: КУДИЦ-ОБРАЗ, 2000.– 336с.
4. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения / Дьяконов В.П. – М.: СОЛОН-Пресс, 2004. –768 с.
5. MATLAB und SIMULINK. Beispielorientierte Einführung in die Simulation dynamischer Systeme / Josef Hoffman – Bonn Addison-Wesley-Longman. – 1998.
6. Дьяконов В.П. Simulink 4. Специальный справочник. – СПб.: Питер, 2001. – 528с.
7. Герман-Галкин С.Г. Компьютерное моделирование полупроводниковых систем в МАТЛАБ 6.0: Учебное пособие. – СПб.: КОРОНА принт, 2001. – 320 с.
8. Толочко О.И. Использование пакета Matlab и его расширения Simulink при исследовании систем электропривода / Методическое пособие (для студентов специальности 7.0922.08). – Донецк: ДонГТУ, 1999. – 87 с.
9. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. – М.: Мир, 1980. – 279 с.
10. Деруссо П., Рой Р., Клоуз Ч. Пространство состояний в теории управления: Пер. с англ. - М.: Наука, 1970. - 477 с.
11. Голубь А.П., Кузнецов Б.И., Опрышко И.А., Соляник В.П. Системы управления электроприводами / Под ред. В.П. Соляника / Учебное пособие. - К.: УМК ВО, 1992. - 374 с.
12. Толочко О.І. Аналіз та синтез електромеханічних систем зі спостерігачами стану. – Донецьк: НОРД-ПРЕС, 2004. – 298 с.
13. Бессекерский В.А., Попов Е. П. Теория систем автоматического регулирования. - М.: Наука, 1972. - 768 с.
14. Филипс Ч., Харбор Р. Системы управления с обратной связью. – М.: Лаборатория базовых знаний, 2001. – 616 с.
15. Lutz H., Wendt W. Taschenbuch der Regelungstechnik. – Tun und Frankfurt am Main: Verlag Harry Deutsch, 2000. – 1148 S.

ЗМІСТ