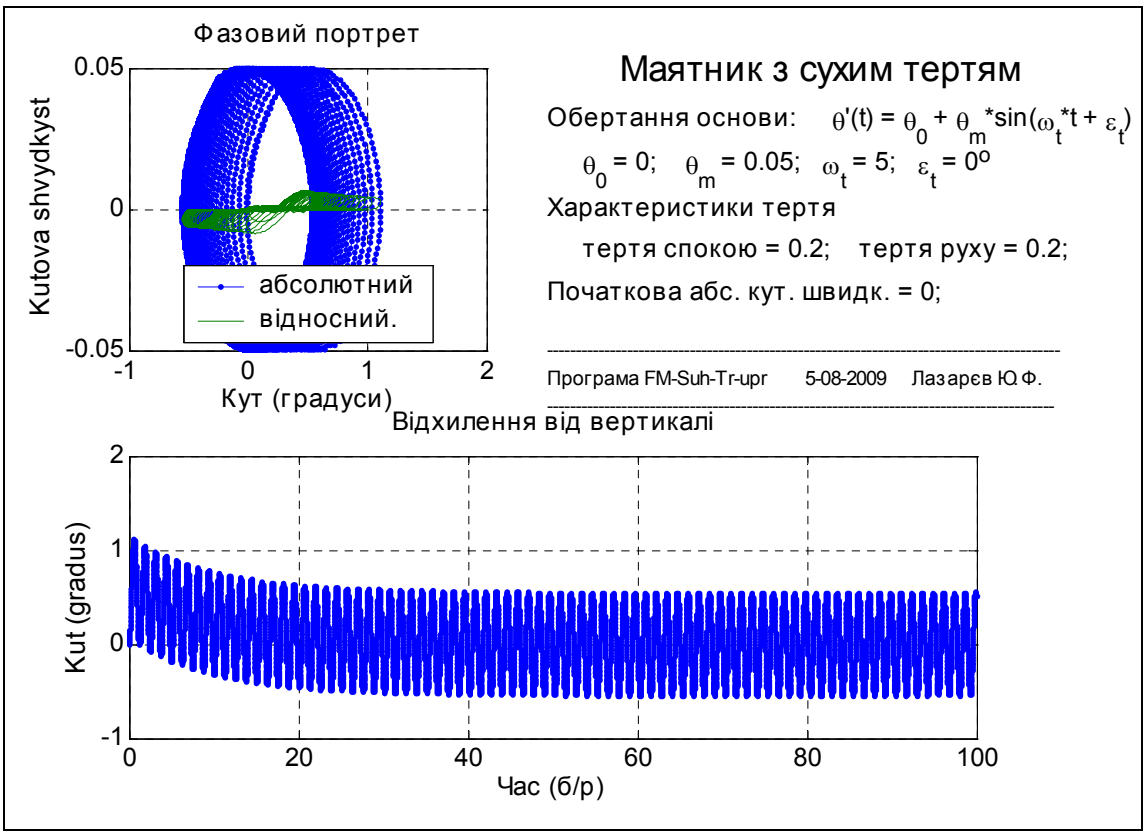
Ю. Ф. Лазарєв

MATLAB і моделювання динамічних систем

Навчальний посібник

Глава 4
Засоби взаємодії Matlab з Simulink



Київ – 2009

УДК 681.3(0.75)
Л17

Лазарєв Ю. Ф.
Л17 MATLAB і моделювання динамічних систем. Навчальний посібник. Глава 4. Засоби взаємодії Matlab з Simulink. – Київ: НТУУ "КПІ", 2009. – 63 с.

Зміст

4. Засоби взаємодії Matlab з SimuLink

Моделювання процесов за допомогою S-моделей поряд зі значними перевагами має й деякі суттєві недоліки.

До переваг використання S-моделей можна віднести:

- ефективність створення програм моделювання складних динамічних систем шляхом складання блок-схеми системи з стандартних готових блоків, які є візуальними відображеннями відповідних математичних програм (візуальне програмування);
- зручні і наочні засоби, що дозволяють перетворити готову блок-схему, або одержати додаткову інформацію про змінювання проміжних процесів;
- широкий набір ефективних програм розв'язувачів (Solvers), які реалізують методи чисельного інтегрування диференційних рівнянь з фіксованим кроком інтегрування, з автоматично змінюваним змінним кроком інтегрування, а також розв'язувачів для так званих жорстких систем диференційних рівнянь;
- відсутність необхідності у спеціальній організації процесу чисельного інтегрування диференційних рівнянь;
- унікальні можливості щодо інтегрування рівнянь нелінійних систем з суттєвими нелінійностями (когда нелінійна залежність має стрибкоподібний характер);
- можливість вельми швидкого і зручного отримування графічної інформації про змінювання модельованих величин з часом.

Недоліками використання S-моделей є:

- жорстка і незручна форма графічного подання сигналів у блока – оглядових вікнах – ***Scope*** і ***XYGraph*** (на відміну від засобів, що використовуються у середовищі Matlab);
- відсутність можливості автоматично (програмно) обробити одержані результати багаторазовог моделювання однієї чи кількох S-моделей;
- відсутність можливості раціонально (наприклад, у діалоговій формі) організувати процес змінювання первісних даних S-моделі і параметрів її блоків.

Важливо відзначити, що для окремих видів диференційних рівнянь набагато прстіше, зручніше і швидше скласти процедури для обчислення їхніх правих частин, чим сформувати відповідну блок-схему.

З викладеного випливає, що програмна реалізація процесу моделювання і моделювання шляхом створення S-моделей маю взаємодоповнювальними можливостями. Тому бажано скористатися перевагами цих двох засобів моделювання, поєднавши програмну реалізацію з застосуванням S-моделей.

4.1. Об'єднання S-моделей з програмами Matlab

Щоб здійснити поєднання програми Matlab з S-моделлю, необхідно мати у наявності засоби, які дозволяють забезпечити:

- передавання даних з середовища Matlab у S-модель і зворотно;
- запуск процеса моделювання S-моделі з середовища Matlab, а також можливість змінювання параметрів моделювання і S-блоків з цього середовища;
- виклик програм Matlab з S-моделі;
- створення S-блоків не тільки із інших готових блоків, а і шляхом використання програм, записаних на M-мові.

4.1.1. Керування процесом моделювання у Simulink

Кожний блок S-моделі має такі внутрішні характеристики (рис. 4.1):

- вектор вхідних величин ***u***;
- вектор вихідних величин ***y***;
- вектор змінних стану ***x***.

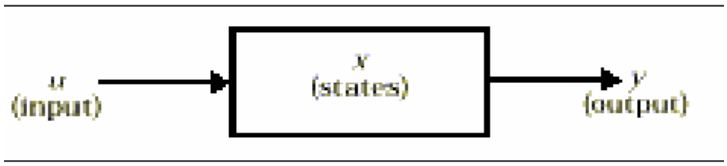


Рис. 4.1. Схема взаємодії величин, що визначають поточний стан блоку

Вектор змінних стану може складатися зі змінних x_c неперервних станів, змінних x_d дискретних станів, або їх комбінації.

Математичні зв'язки між цими величинами можуть бути подані у виді наступних рівнянь:

- формування виходу:

$$y = f_o(t, x, u);$$

- оновлення (формування нового значення) змінних дискретних станів:

$$x_d(k+1) = f_u(t, x, u),$$

- формування значень похідної від вектору змінних стану:

$$\frac{dx}{dt} = f_d(t, x, u),$$

де

$$x = \begin{cases} x_c \\ x_d(k) \end{cases}.$$

Моделювання складається з двох фаз – ініціалізації і власне моделювання

У фазі ініціалізації виконуються наступні дії:

1) параметри блоків пердаються у Matlab задля оцінювання (обчислення); результати числових операцій використовуються як фактичні параметри блоків;

2) ієрархія моделі згладжується: кожна не умовно виконувана підсистема замінюється блоками, з яких вона складається;

3) блоки сортуються у тому порядку, в якому їх потрібно змінювати; алгоритм сортування забезпечує такий порядок, що будь-який блок з прямим підключенням не змінюється, поки змінюються блоки, які визначають вхідні величини; на цьому кроку виявляються алгебричні цикли;

4) перевіряються зв'язки між блоками (перш за все збіжність довжини вектора вихідних величин кожного блоку з очікуваною довжиною векторів вхідних величин керованих ними блоків).

Власне моделювання здійснюється шляхом чисельного інтегрування. Кожний з наявних методів інтегрування (ODE) залежить від здатності моделі визначати похідні її неперервних станів. Розрахунок цих похідних здійснюється у два етапи. Спочатку кожна вихідна величина блоку обчислюється у порядку, визначеному у процесі сортування. На другому етапі обчислюються похідні кожного блоку для поточного моменту часу, вхідні змінні і змінні стану. Отриманий вектор похідних використовується для обчислення нового вектора змінних стану у наступний момент часу. Як тільки завершується обчислення нового вектору змінних стану, блоки даних і блоки – оглядові вікна оновлюються.

З переліком програм розв'язувачів (інтеграторів), що прикладаються до пакету Simulink, можна ознайомитися у вікні Configuration Parameters (рис. 4.2), яке виникає на екрані після виклику команди Simulation > Configuration Parameters з меню блок-схеми.

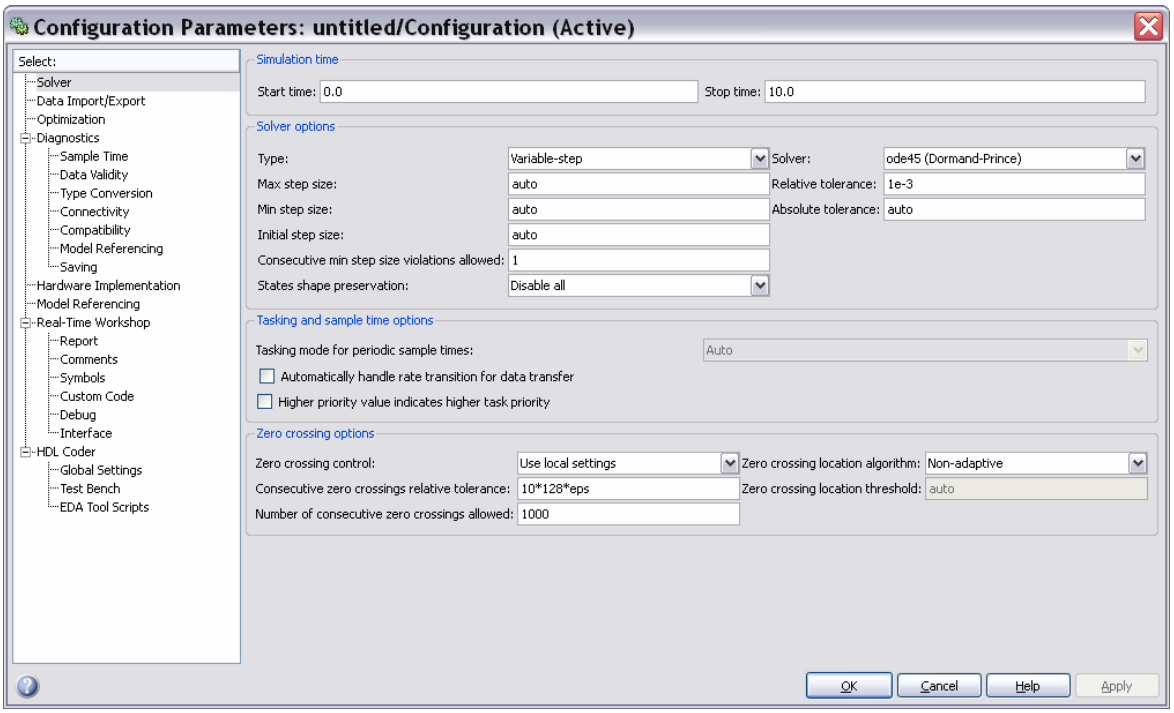


Рис. 4.2. Вкладення Solver вікна Configuration Parameters

У верхній частині вкладення *Solver* вікна *Configuration Parameters* містяться поля введення *Start time* (Початковий час) і *Stop time* (Кінцевий час), в яких встановлюється відповідно початкове і кінцеве значення аргументу (часу). В області Solver options (Параметри розв'язувача) у списку *Type* (Тип) обира-

ється тип розв'язувачів, а у спадному списку *Solver* (Розв'язувач) праворуч від нього - конкретний розв'язувач.

Якщо обраний тип розв'язувачів *Fixed-step* (з фіксованим кроком), у списку праворуч з'явиться такий набір розв'язувачів:

- *discrete (no continuous states)* – дискретний (не неперервні стани);
- *ode5 (Dormand-Prince)* – метод Дормана-Принса (пятого порядку);
- *ode4 (Runge-Kutta)* – метод Рунге-Кутта (четвертого порядку);
- *ode3 (Bogacki-Shampine)* – метод Богацького-Шампена (третього порядку);
- *ode2 (Heun)* – метод Хойна (другого порядку);
- *ode1(Euler)* – метод Ейлера (першого порядку);
- *ode14x (extrapolation)* - екстраполяція .

При цьому у нижній частині вікна (рис. 4.3) виникає поле *Fixed step size* (Розмір фіксованого кроку), у яке потрібно ввести значення кроку інтегрування. У списку *Tasking mode for periodic sample times*, що виникає нижче, слід обрати один з трьох можливих режимів роботи: *Auto* (автоматический), *SingleTasking* (однозадачний) або *MultiTasking* (багатозадачний).

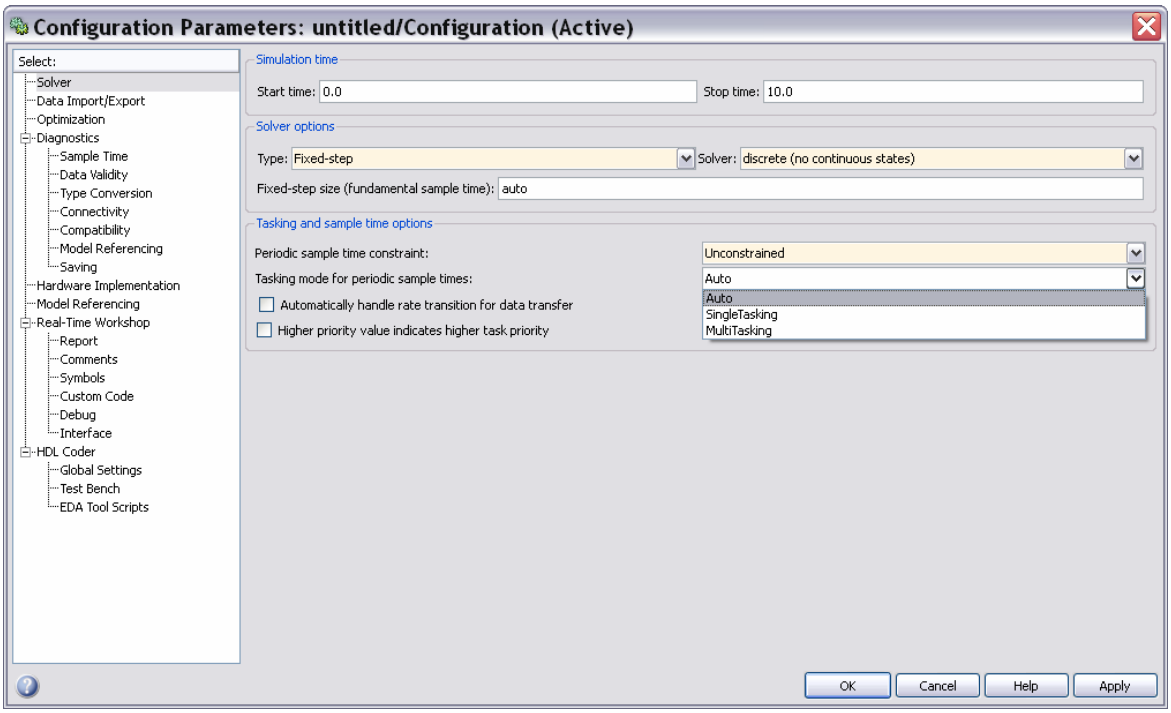


Рис. 4.3. Вид вікна Configuration Parameters при встановлені Fixed-step

При обранні у списку *Type* (Тип) елемента *Variable-step* (зі змінним кроком), у спадному списку праворуч виникне інший список інтеграторів (методів чисельного інтегрування) (див. рис. 4.2):

- *discrete (no continuous states)* – дискретний (не неперервні стани);
- *ode45 (Dormand-Prince)* – метод Дормана-Принса;
- *ode23 (Bogacki-Shampine)* – метод Богацького-Шампена;
- *ode113 (Adams)*– метод Адамса;
- *ode15s (stiff\NDF)* – метод NDF для жорстких систем;

- *ode23s (stiff\Mod. Rozenbrock)* – модифікація Розенброка для жорстких систем;

- *ode23t (Mod. stiff\Trapezoidal)* – метод трапецій, модифікація для жорстких систем;

- *ode23tb(stiff\TR-BDF2*) – метод TR-BDF2 для жорстких систем.

У цьому випадку у нижній частині поля *Solver options* виникають наступні поля введення:

- *Max step size* (Максимальний розмір кроку);
- *Min step size* (Мінімальний розмір кроку);
- *Initial step siz*e (Початковий розмір кроку);

а праворуч від них – поля

- *Relative tolerance* (Відносна точність);
- *Absolute tolerance* (Абсолютна точність).

В усіх полях, окрім *Relative tolerance*, встановлено значення *auto*, тобто ці параметри задаються автоматично і змінюються користувачем лише у випадку, коли йому потрібно встановити їх конкретні значення, відмінні від прийнятих за замовчуванням. Відносна точність (точніше, відносна похибка) за замовчуванням дорівнює $1 \cdot 10^{-3}$.

4.1.2. Виявлення перетинання нуля

При моделюванні систем, що містять елементи з розривними характеристиками, такі як реле, люфт, сухе тертя, необхідно максимально точно відтворити особливості поводження ситеми у моменти часу, коли відбувається стрибкоподібне змінення значення розривної характеристики. Зазвичай у цей момент стрибкоподібно змінюються й властивості самої системи, а також початкові умови для продовження процесу. У зв'язку з цим вельми важливо максимально точно (з машинною точністю) визначити момент часу, у який здійснюється подібне змінення розривної характеристики, і зафіксувати поточні значення параметрів руху системи. Це потрібно для того, щоб на наступному кроці почати процес інтегрування саме з цього моменту часу з новими початковими умовами, що відповідають стану системи, набутому саме у цей момент часу.

Зазвичай стрибкоподібне змінення розривної характеристики відбувається у момент, коли одна зі змінних стану системи при своєму змінюванні у часі переходить через деякий рівень. При цьому інколи має значення напрямок, у якому здійснюється перетинання змінною цього рівня – при її збільшенні чи зменшенні – від цього може залежати, як саме зміняться (стрибкоподібно) характеристики системи і значення її змінних стану.

Задача виявлення перетинання сигналом деякого сталого рівня легко зводиться до задачі виявлення перетинання нульового рівня. Тому у подальшому процес точного визначення параметрів стану системи у момент, коли здійснюється стрибкоподібне змінення деякої характеристики системи, називатимемо *виявленням перетинання нуля*. У пакеті Simulink виявлення перетинання нуля використовується для того, щоб зафіксувати різкі зміни у неперервних сигна-

лах. Ця процедура відіграє важливу роль при керуванні стрибками стану і при точному інтегруванні переривчастих сигналів.

Система зазнає стрибка стану, коли змінювання значень змінних стану системи викликає значні миттєві змінення у системі. Простой приклад стрибков стану – відскакування м'яча від підлоги. При моделюванні такої ситуації використовується метод інтегрування зі змінним кроком. Метод чисельного інтегрування зазвичай не передбачає заходів, які дозволили би точно визначити момент контакту м'яча з підлогою. Внаслідок цього при моделюванні м'яч, переходячи через контактну точку, нібито проникає крізь підлогу. Використання виявлення перетинання нуля у Simulink гарантує, що момент стрибка стсану ситеми визначений точно (з машинною точністю). Тому в результаті чисельного моделювання не відбувається проникнення м'яча крізь пол, і перехід від від'ємної швидкості м'яча до додатної у момент контакту відбувається надзвичайно різко.

У пакеті Simulink передбачені наступні блоки, які використовують виявлення перетинання нуля:

Integrator (Інтегратор) – коли представлений порт насичення *Show saturation port* (див. п. 3.2.3), виявляється момент, коли насичення відбувається; якщо вихід обмежений, то тричі виявляється перетинання нуля: коли досягається верхня межа насичення, коли досягається нижня межа насичення і коли зона насичення покинута;

Hit Crossing (Уловлювання перетинання) – виявляє момент, коли сигнал на вході перетинає заданий рівень;

Abs (Абсолютне значення) – визначає момент, коли сигнал на вході перетинає нуль у будь-якому напрямку – зменшуючись чи збільшуючись;

BackLash (Люфт) – використовується двічі: коли вхідний сигнал сягає верхнього і нижнього порогів;

Dead Zone (Мертва зона) – використовується двічі: коли сигнал входить у зону нечутливості (до цього вихідний сигнал був більше вхідного на величину нижньої границі зони) і коли залишає цю зону (вихідний сигна стає менше вхідного на величину верхньої границі;

MinMax (Мінімум-максимум) – для кожного елемента вихідного вектора виявляє момент часу, коли вхідний сигнал стає мінммальним або максимальним;

Relay (Реле) – виявляє момент часу, коли реле потрібно включити (якщо воно виключено) або виключити (якщо воно включено);

Relational Operanor (Оператори відношення) – виявляє момент часу, коли відношення змінюється;

Saturation (Насичення) – використовується двічі: коли вхідний сигнал досягає верхнього порогу або залишає його і коли сигнал досягає нижнього порогу або залишає його;

Sign (Сігнум-функція) – виявляє момент проходження вхідного сигналу через нуль;

Step (Сходинка) – виявляє момент часу, коли відбуватиметься стрибкоподібне змінення рівня вихідного сигнала.

Як показовий приклад розглянемо застосування функції перетинання нуля у програмі демонстрування поводження підстрибуючого м'яча. Для цього введить у командному вікні Matlab команду ***bounce***. В результаті її виконання виникне вікно з зображенням блок-схеми S-моделі поводження м'яча (рис. 4.4).

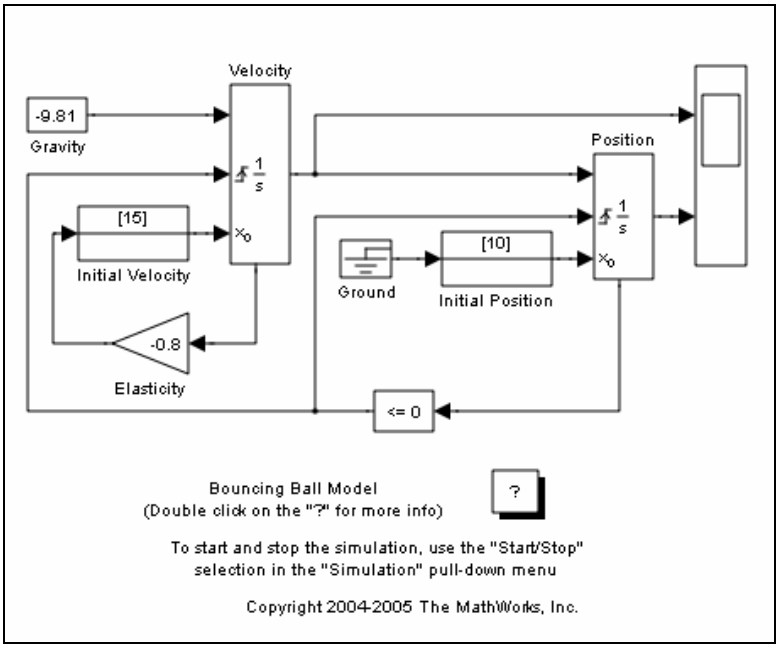


Рис. 4.4. Блок-схема S-моделі Bounce

Ця модель здійснює чисельне інтегрування методом *ode23* (з автоматичним змінюванням кроку інтегрування) диференційне рівняння

$$\frac{d^2x}{dt^2} = -g\,.$$

(де g - прискорення вільного падіння; x - поточна висота м'яча над підлогою) у проміжках між моментами часу, в яких відбувається зіткнення м'яча з підлогою.

Інтегрування здійснюється за допомогою двох блоків- інтеграторів - ***Velocity*** і ***Position***. На виході першого з них одержують значення поточної швидкості руху м'яча, а на виході другого – висоти м'яча над підлогою.

Більш детальне знайомство з інтеграторами (блок ***Position*** і блок ***Velocity***) (рис. 4.5 і 4.6) дає можливість впевнитися, що в обох випадках введено зовнішнє керування (встановлений елемент *rising* (при збільшуванні) у списку *External reset*), що приводить до появи другого зверху вхідного порту. В обох інтеграторах введено зовнішнє введення початкових умов (встановлений елемент *external* (зовнішнє) у списку *Inutial condition source* (джерело початкової умови)). При цьому на лівому боці кожного блоку виникають треті зверху вхідні порти.

У блоці ***Position*** встановлена ніжня границя (нуль) змінювань висоти м'яча (див. рис. 4.5), а у блоці ***Velocity*** встановлений прапорець *Enable zero-crossing detection* (Дозволити визначати перетинання нуля) (див. рис. 4.6).

Окрім того, в обох блоках встановлені прапорці *Show state port* (Показати порт стану), у зв'язку з чим на зображенні цих блоків виникли (знизу) додатково допоміжні вихідні порти (див. рис. 4.4). На ці порти виводяться поточні значення вихідних величин інтеграторів: швидкості на порт блоку ***Velocity*** і висоти м'яча – на порт блоку ***Position***.

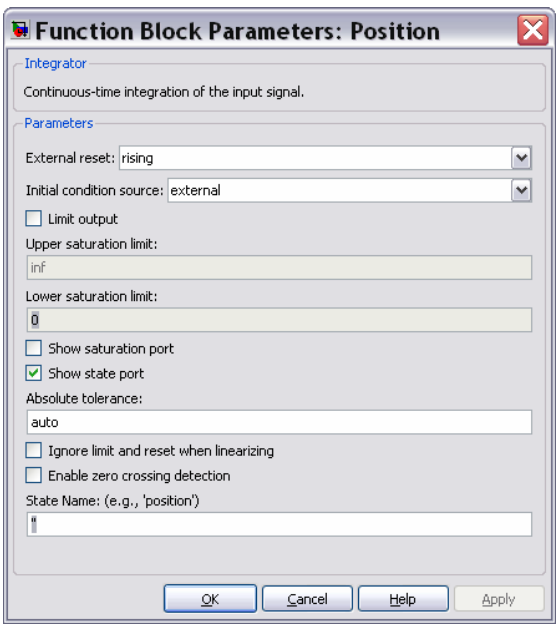


Рис. 4.5. Вікно блоку Position

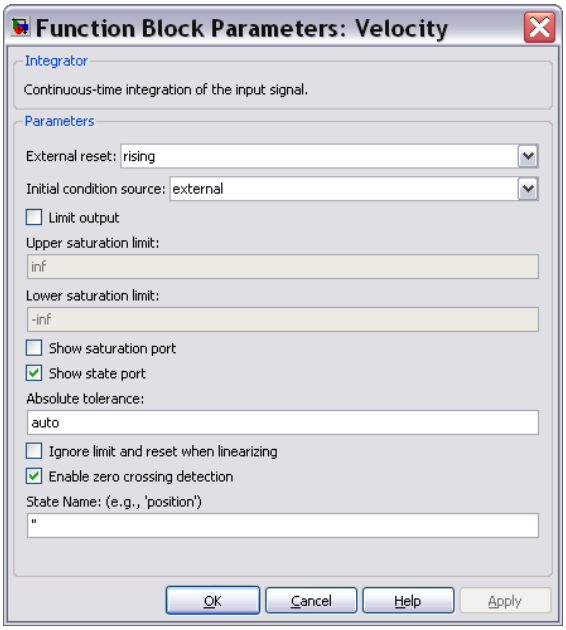


Рис. 4.6. Вікно блоку Velocity

Як бачимо з рис. 4.4, початкові умови встановлені такими: зі швидкості 15 м/с, з положення – 10 м.

До нижніх входів блоків-інтеграторів під'єднані блоки *Initial Position* (Початкове положення) і *Initial Velocity* (Початкова швидкість), побудовані на основі стандарних блоків *IC (Initial Condition – початкова умова)* з поділу *Signal Attributes* бібліотеки Simulink. Призначення їх – встановлювати початкові умови для інтегратора, показані на зображенні цих блоків.

Моделювання здійснюється у такий спосіб. Інтегрування починається при початкових умовах, вказаних на зображенні блоків *Initial Position* і *Initial Velocity*. У момент, коли на другому інтеграторі (***Position***) фіксується перетинання м'ячем нуля висоти, здійснюється точне (з машинною точністю) обчислення момента часу, в який м'яч контактує з підлогою, перераховуються значення швидкості і висоти на момент перетинання на першому і другому інтеграторах і цей момент встановлюється як новий початковий момент часу.

Найдене значення швидкості через додатковий вихідний порт інтегратора ***Velocity*** поступає на вхід блоку ***Elasticity*** (Еластичність), зміню свій знак на протилежний і зменшується на 20% (цим враховується зменшення швидкості при відскоку за рахунок втрати енергії внаслідок неідеальній пружності м'яча) і використовується як нова початкова швидкість. Початкове значення висоти пі-

сля контакту м'яча з підлогою встановлюється таким, яким є вихідний сигнал блоку Ground, який з'є днаний з блоком ***Initial Position***. А цей сигнал завжди дорівнює нулеві. Тому початкова висота після контакту з підлогою завжди буде встановлена рівною нулю. Потім інтегрування продовжується при нових початкових умовах. Результат роботи моделі відображується в оглядовому вікні блоку ***Scope*** (рис. 4.7).

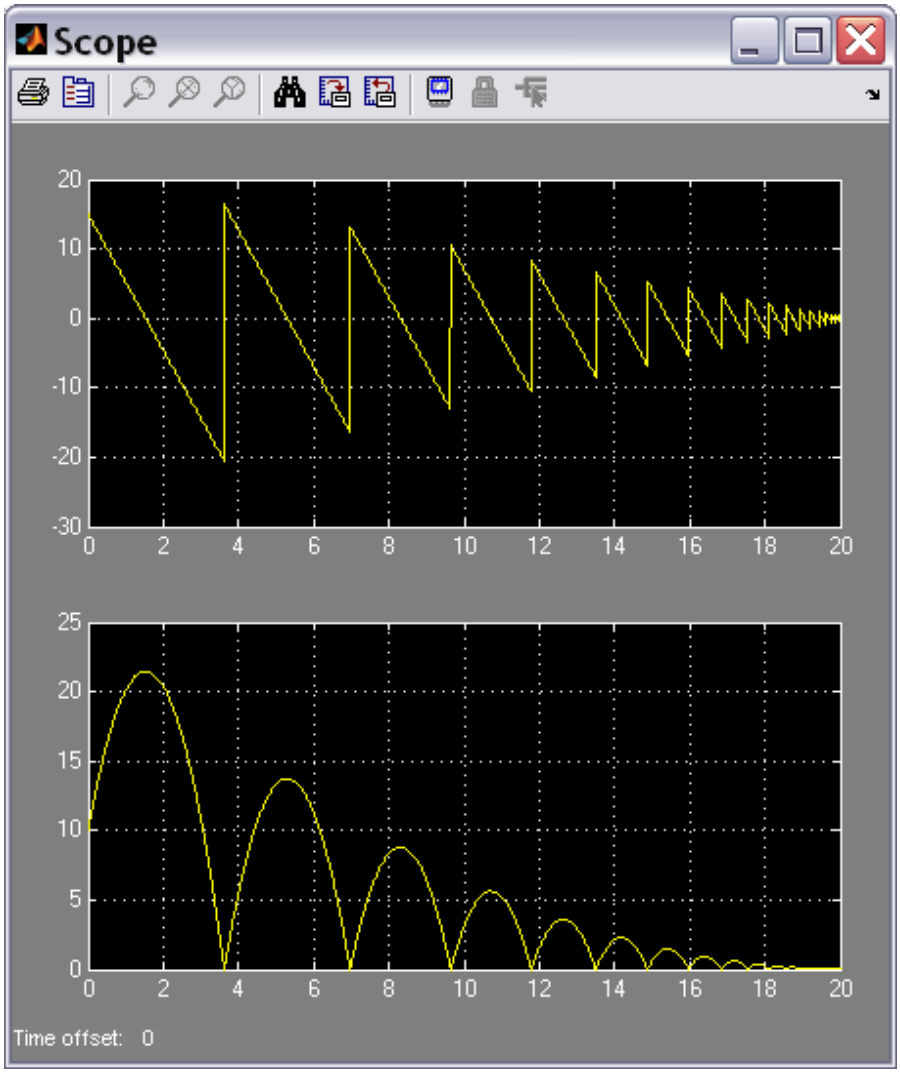


Рис. 4.7. Результат роботи S- моделі Bounce

Варто відмітити, що керування процесом переривання інтегрування і його продовження при нових початкових умовах здійснюється другим інтегратором ***Position*** при перетинанні величини на його виході встановленої нижньої границі (нуля). При цьому значення величини висоти й швидкості на виході обох інтеграторів не можуть бути використані для розрахунку нового їхнього значення. для цієї мети необхідно застосовувати додаткові виходні порти інтеграторів, а тому потрібно встановити прапорці *Show state port* (Показати порт стану), і подати це розраховане значення не безпосередньо на вхідний порт початкової умови інтегратора, а обов'язково через блок початкової умови ***IC***.

Описані вище засоби пакета Simulink дозволяють моделювати вельми важливі і важкопрограмовані особливості поводження суттєво неінійних систем.,

таких як "зчеплення" і "розчеплення" рухомих частин механізмів під дією сил сухого тертя, удароподібні процеси і пов'язані з ними режими, "ковзні" режими, автоколивання, різкі переходи від одного режиму до іншого.

4.1.3. Обмін даними між середовищем Matlab і S-моделлю

Робочий простір середовища Matlab є досяжним для використовуваної S-моделі. Це означає, що якщо у якості значень параметрів у вікнах настроювання блоків S-моделі використані ймення змінних, а значення цих змінних були попередньо встановлені у робочому просторі, то ці значення одразу передадуться відповідним блокам S-моделі. Тому, щоб організувати зручне змінювання параметрів блоків S-моделі (наприклад, у діалоговому режимі), достатньо зробити наступне:

1) у вікнах настроювання блоків S-моделі у якості параметрів вказати ідентифікатори (ймення) замість чисел;

2) організувати засобами середовища Matlab (наприклад, програмно) присвоювання числових значень цим ідентифікаторам, а також (у випадку необхідності) – їхнє змінювання у діалоговому режимі;

3) після присвоювання числових значень усім ідентифікаторам (наприклад, через запуск відповідної M-програми) провести запуск S-моделі на моделювання.

Деякі засоби обміну даними були вже розглянуті раніше. Це блок ***From Workspace*** поділу ***Sources*** і блок ***To Workspace*** поділу ***Sinks*** стандартної бібліотеки ***Simulink***. Першій прислуговується для включення сигналів, попередньо одержаних (обчислених) і записаних у робочий простір Matlab (наприклад, в результаті обчислень в середовищі Matlab), у процес моделюваня S-моделі. Другий забезпечує можливість запису результатів, одержаних при моделюванні з використанням S-моделі, у робочий простір середовища Matlab.

Для записи одержаного в результаті моделювання процесу у робочий простір (див. рис. 3.19, п 3.2.1) слід вставити у блок схему S-моделі блок ***To Workspace***, подати на його вхід потрібний для запису сигнал і вказати у полі *Variable name* (Імя змінної) вікна настроювання блоку ймення, під яким цей процес потрібно зберегти у робочому просторі системи Matlab. Відповідні моменти модельного часу при цьому не записуватимуться.

Для визначення процеса, який буде використаний у S-моделі, користуючись даними, записаними у робочий простір, слід вставити у блок-схему S-моделі блок ***From Workspace***, з'єднати його вихід з одним з входів інших блоків, розкрити вікно настроювання блоку (рис. 4.8), і у полі введення *Data* вказати вектор, складений з двох імен – ймення масиву значень аргументів (моментів часу, у які визначений цей процес) і ймення масиву значень процеса при вказаних значеннях аргументу, наприклад: [T, D] (рис. 4.8). У цьому випадку з масиву T робочого простору будуть зчитані усі значення, які будуть відігравати в S-моделі роль значень модельного часу, а вихідна величина блоку при моделюванні у моменти часу, що відповідають записаним у масиві T, прийматиме значення, записані у масиві D. Якщо при цьому реальні значення моментів часу

при моделюванні не збігатимуться з записаними у масиві T, вдбудеться лінійна інтерполяція значень масиву D, що відповідають попередньому і наступному значенням моментів часу у масиві T.

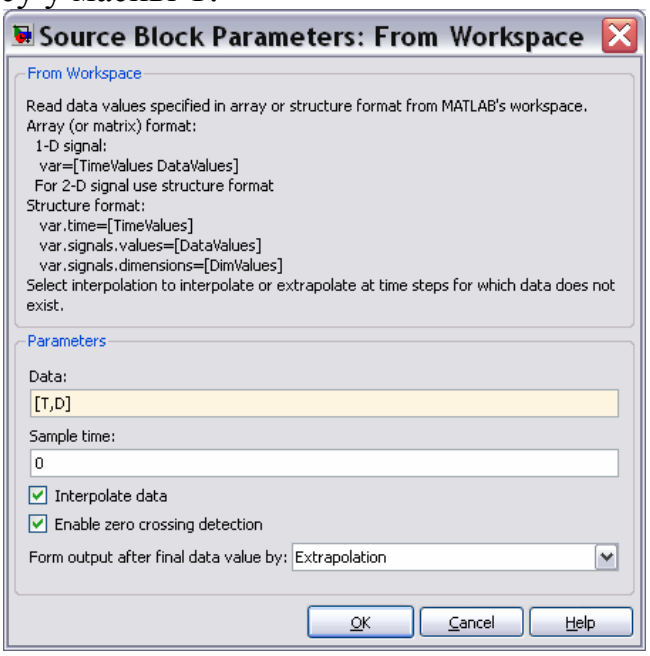


Рис. 4.8. Вікно настроювання блоку From Workspace

Але існує й простішій спосіб виконання вищеописаних операцій – без використання зазначених блоків.

Щоб підключити визначеий у програмі Matlab процес у S-модель як вхідний, передбачений механізм включення портів входу і виходу. для цього потрібно зробити наступне.

1. Вставити блок вхідного порту ***In*** у блок-схему S-моделі і з'єднати його з одним з блоків S-моделі.
2. У вікні S-моделі викликати команду Simulation > Configuration Parameters, щоб відчинити вікно ***Configuration Parameters***, в якому обрати команду *Data Import/ Export* (Імпорт-експорт даних) (рис. 4.9).
3. В області *Load from workspace* (Завантажити з робочого простору) встановити прапорець Input (Вхід) і у полі праворуч ввести ім'я, що складається з ймення вектор значень аргумента і ймення вектора значень вхідного сигнала при цих значеннх аргумента, наприклад: [t, u].
4. Встановити значення цих векторів у Matlab, наприклад, так:
$$t = (0:0.1:1)'; \qquad u = [\sin(t), \cos(t), 4*\exp(t)]$$
5. Запустити S-модель на моделювання.

Щоб вивести деякі сигнали, що формуються в S-моделі, у робочий простір Matlab, потрібно виконати наступні дії.

1. У блок схему моделі вставити блоки портів виходу ***Out*** і під'єднати до них необхідні вихідні величини інших блоків.

2. У вікні S-моделі викликати команду Simulation > Configuration Parameters, щоб відчинити вікно ***Configuration Parameters***, в якому обрати команду *Data Import/ Export* (Імпорт-експорт даних) (рис. 4.9).
3. В області *Save to workspace* (зберегти у робочому просторі) відчиненого вікна встановити прапорці Time і Output, і у полі праворуч ввести ймення, під якими будуть записані значення часу і величин, що подаються на вихідні порти, у робочий простір. За замовчуванням ці імена є **tout** (для модельного часу) і **yout** (для даних з вихідних портів).

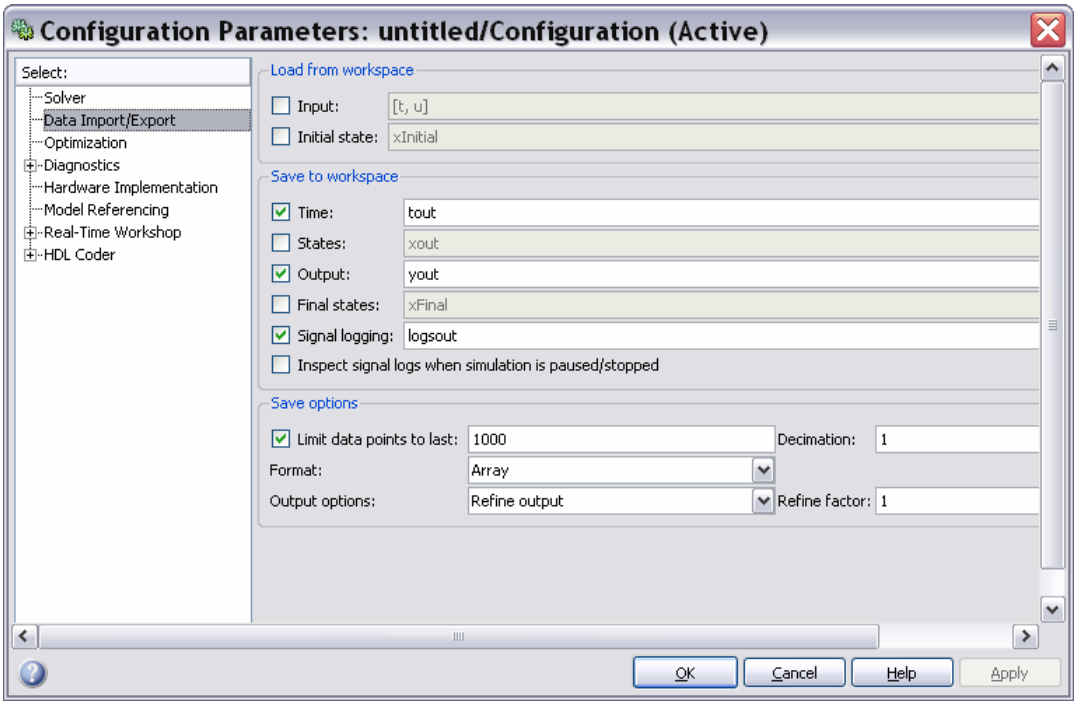


Рис. 4.9. Вкладення Data Import/ Export вікна Configuration Parameters

У цьому випадку значення модельного часу будуть записуватися у робочий простір у масив під ім'ям **tout**, а відповідні значення вихідних сигналів при цих значеннях часу – у стовпці матриці **yout** (у першій стовпець – процес, який поданий на першій вихідний порт ***Out1***, у другий стовпець – процес, поданий на другий порт ***Out2*** і т. д.).

Після встановлення прапорця *Initial state* (Початковий стан) в області *Load from workspace* (Завантажити з робочого простору) можна ввести в S-модель початкові значення змінних стану системи. Встановивши прапорець *States* (Змінні стану) в області *Save to workspace* (Зберігти у робочому просторі), можна записати поточні значення змінних стану системи уробочий простір під ім'ям **xout** (або під іншим ім'ям, якщо його записати у поле праворуч від прапорця *State*). Нарешті, можна записати й кінцєві значення змінних стану у вектор **xFinal**, якщо встановити прапорець *Final state* (Кінцевий стан).

4.1.4. Запуск процеса моделювання S-моделі з середовища Matlab

Розглянемо засоби, які дозволяють запускати процес моделювання утворених S-моделей із програми Matlab.

S-модель запускається на виконання, якщо у програмі Matlab (або у командному вікні Matlab) викликати процедуру sim:

[t,x,y1,y2,...,yn] = sim(model, timespan,options,ut).

Тут **model** – симольний рядок, що містить ім'я MDL-файлу, в якому записана відповідна S-модель; **timespan** – вектор, що складається з двох елементів – значень початкового і кінцевого моментів часу моделювання; **options** – вектор значень параметрів інтегрування, який встановлюється процедурою **simset**

options = simset('Властивість1**',**Значення1**,'**Властивість2**',**Значення2**...);**

Процедура **sim** повертає наступні значення: t – вектор вихідних значень моментів модельного часу; x – масив (вектор) змінних стану системи; y1 – першій стовпець матриці вихідних змінних системи (які надаються до вихідних портів) і т. д.

Змінювати параметри розв'язувача і процесу інтегрування у Matlab можна за допомогою функції **simset**, як це показано вище. У такий спосіб можна задати значення наступних властивостей розв'язувача:

- 'Solver' – назва розв'язувача; значення (вказується між двома апострофами) може бути однією з наступних: ode45, ode23, ode1b, ode15s, ode23s – для інтегрування з автоматично змінюваним кроком; ode5, ode4, ode3, ode2, ode1 – для інтегрування з фіксованим кроком;
- 'RelTol' – відносна припустима похибка; значення може бути додатним скаляром; за замовчуванням встановлюється 1e-3;
- 'AbsTol' – абсолютна припустима похибка; значення може бути додатним скаляром; за замовчуванням встановлюється 1e-6;
- 'FixedStep' – фіксований крок (додатний скаляр);
- 'MaxOrder' – максимальний порядок методу (застосовується лише для методу ode15); може бути одним з цілих чисел 1, 2, 3, 4; за замовчуванням дорівнює 5;
- 'MaxRows' – максимальна кількість рядків у вихідному векторі; невід'ємне ціле;
- 'InitialState' – вектор початкових значень змінних стану;
- 'FinalStateName' – ім'я вектора, в який буде записуватися кінцеве значення вектора змінних стану моделі;
- 'OutputVariables' – вихідні змінні системи; за замовчуванням має значення {txy}; можливі варіанти tx, ty, xy, t, x, y; усі вони неявно вказують, які саме вихідні змінні не будуть виводитися.

4.1.5. Створення S-блоків з використанням програм Matlab

У системі Matlab передбачений механізм перетворення деяких процедур, написаних мовами високого рівня, у блок S-моделі. Він реалізується за допомогою S-функцій.

S-функція – це відносно самостійна програма, яка написана користувачем мовою Matlab або C і має візуальне подання у виді блока Simulink. Застосування S-функцій дозволяє вирішити наступні задачі:

- утворювання нових (користувацьких) блоків, які доповнюють бібліотеку пакета Simulink;
- використовування опису модельованої системи у виді системи математичних рівнянь;
- включення раніше створених програм, написаних M-мовою або мовою C, у S-модель.

Програмний код S-функції має чітку структуру. У випадку, коли S-функція утворюється на основі M-мови, ця структура наведена у файлі SfinTMPL.m, який міститься у папці TOOLBOX\ SIMULINK\BLOCKS. заголовок S-функції у загальному випадку має вид:

function [sys,x0,str,ts] = <Ім'я S-функції>**(t, x, u, flag**{,<Параметри>**})**

Стандартними аргументами S-функції є:

t – поточне значення аргументу (модельного часу);

x – поточне значення вектора змінних стану;

u – поточне значення вектора вхідних величин;

flag – цілочислова змінна, яка відбиває етапи дії S-функції;

<Параметри> - перелік додаткових ідентифікаторів, які характеризують модельовану систему і значення яких використовуються у S-функції (їхня наявність не є обов'язковою).

В результаті обчислень, що виконуються при роботі S-функції, присвоюються значення таким змінним:

sys – системна змінна, вміст якої залежить від значення, що набуває змінна flag;

x0 – вектор початкових значень змінних стану;

str – символьна змінна стану (зазвичай вона є пустою []);

ts – матриця, що містить інформацію продискрети часу.

Текст S-функції складається з стандартного тексту самої S-функції і текстів наступних внутрішніх процедур, якими вона використовує:

- ***mdlInitializeSizes*** **–** встановлює розміри змінних S-функції і початкові значення змінних стану;
- ***mdlDerivatives*** **–** процедура обчислення поточних значень правих частин диференційних рівнянь системи, записаних у формі Коші у випадку, коли змінні стану об'явлені як неперервні;
- ***mdlUpdate*** **–** процедура оновлення на наступному інтервалі дискрету часу значень змінних стану, об'явлених як дискретні;

- ***mdlOutputs*** – процедура обчислення значення вектора вихідної змінної блоку S-функції;
- ***mdlGetTime0NextVarHit*** – допоміжна функція для визначення моменту часу, коли конкретна змінна перетинає заданий рівень;
- ***mdlTerminate*** – функція переривання роботи S-функції.

Деякі з вказаних процедур можуть не використовуватися. Це залежить від типу рівнянь (алгебричні, диференційні або різницєві), якими описуєтьсямодельвана S-функцією система. Так, якщо поводження ситеми описується лише алгебричними рівняннями, то не використовуються майже усі вказані внутрішні процедури, за виключенням процедур ***mdlInitializeSizes*** і ***mdlOutputs***. В останній й обчислюються відповідні алгебричні співвідношення, що визначають зв'язок між вхідними змінними **u** і вихідними змінними **y**. У тому випадку, коли поводження системи визначено системою диференційних рівнянь, не використовується функція ***mdlUpdate***, якщо рівняння системи є різницєвими - процедура ***mdlDerivatives***.

Головна процедура S-функції містить, у головному, звернення до тієї чи іншої внутрішньої процедури у відповідності до значення змінної *flag*. Наприклад:

```
switch flag,
 case 0,
        [sys,x0,str,ts]=mdlInitializeSizes;
 case 1,
        sys=mdlDerivatives(t,x,u);
 case 2,
        sys=mdlUpdate(t,x,u);
 case 3,
        sys=mdlOutputs(t,x,u);
 case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
 case 9,
        sys=mdlTerminate(t,x,u);
 otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end
```

У залежності від значення змінної *flag*, виконуються наступні дії (через звернення до відповідної внутрішньої процедури):

0 – ініціалізація блока;

1 – звернення до процедури правих частин диференційних рівнянь у формі Коши;

2 – обчислення нових значень змінних стану на наступному кроці дискретизації (для різницєвих рівнянь);

3 – формування значення вектора вихідних величин;

4 – формування нового значення модельного часу, яке відліковується від моменту перетинання заданого рівня певноє змінною стану;

9 – припинення роботи блока.

Встановлювання і змінювання значень змінної *flag* відбувається автоматично, без втручання користувача, у відповідності до логіки функціонування блоків Simulink при моделюванні.

Отже, використовування S-функції дозволяє моделювати роботу як звичайних алгебричних, так і динамічних (неперервних або дискретних) ланок.

Щоб утворити S-блок на основі використання S-функції, виконайте наступні дії.

1. Напишіть текст S-функції, наприклад у виді M-файлу, користуючись файлом-шаблоном SfunTMPL.m.
2. Перетягніть стандартний блок S-функції (рис. 4.10) з поділу ***User-Defined Functions*** бібліотеки ***SIMULINK*** у вікно блок-схеми, в який буде створюватися новий S-блок.

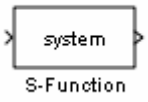


Рис. 4.10. Заготівка S- блоку S-функції

3. Двічі клацніть на зображенні блоку ***S-function***. Це призведе до виникнення на екрані вікна його настроювання (рис. 4.11). Вікно містить поля введення *S-function name* (Ім'я S-функції), в яке вводиться ім'я файлу з написаним текстом S-функції, і *S-function parameters* (Параметри S-функції), в яке вводяться ймення або значення параметрів блоку, вказаних у розділі <Параметри> M-файла, що містить написаний текст S-функції.

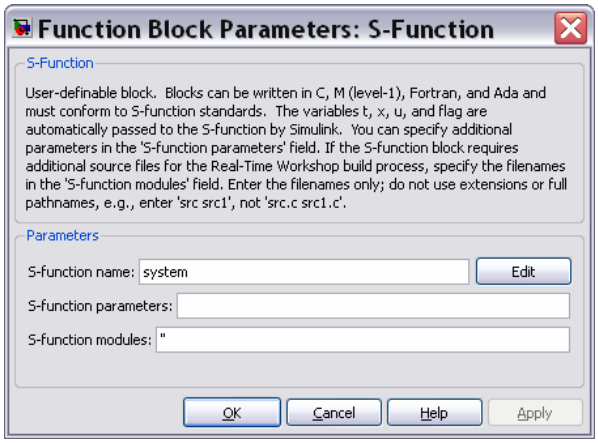


Рис. 4.11. Вікно настроювання блоку S-function

4. Введіть у вказані поля ім'я M-файла, у якому записаний текст S-функції, і список значень параметрів. Якщо, наприклад, у перше поле ввести ім'я S_KA, а у друге – рядок J, Ug0, UgSk0, вікно набуде виду, поданому на рис. 4.12.

5. Клацніть мішкою на кнопці OK. Якщо система виявить M-файл з вказаним ім'ям у папках які досяжні Matlab, вікно, що подане на рис. 4.12, зникне, а на зображенні блока у вікні блок-схеми виникне введене ім'я S-функції (точніше, написаного M-файла) (рис. 4.13).

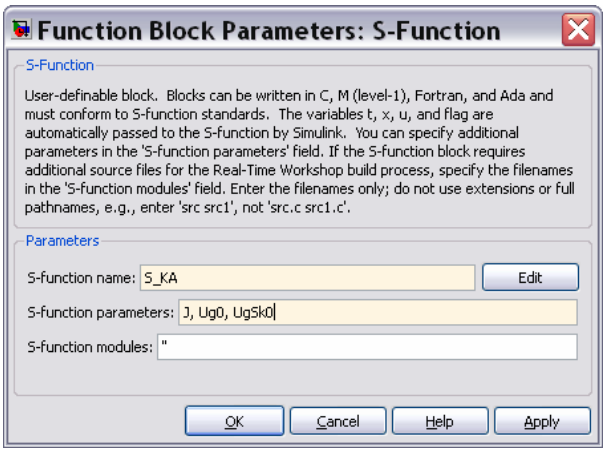


Рис. 4.12. Вікно S-function після введення даних

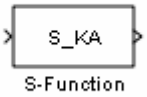


Рис. 4.13. Блок S-function після введення даних

S-блок на основі S-функції, що міститься у M-файлі S_KA.m буде створений. Тепер його можна використовувати як звичайний S-блок у блок-схемі S-моделі. До входу цього блоку має надходити векторний сигнал **u**. Виходом блоку стане векторний сигнал **y**, який сформований S-функцією у внутрішній процедурі ***mdlOutputs***.

4.1.6. Приклад утворення і роботи з S-функцією

Створимо S-функцію, яка релізує динамічні властивості твердого тіла при його обертальному русі. Для опису динаміки тіла скористаємося динамічними рівняннями Ейлера у матричній формі:

$$\mathbf{J}\frac{d\boldsymbol{\omega}}{dt}+(\boldsymbol{\omega}\times)\mathbf{J}\boldsymbol{\omega}=\mathbf{M}. \qquad (4.1)$$

Тут **J** - матриця моментів тіла відносно декартових осей, зв'язаних з тілом; **ω** - матриця-стовпець з проекцій абсолютної кутової швидкості тіла на ті самі осі; $(\boldsymbol{\omega}\times)$ - кососиметрична матриця виду

$$(\boldsymbol{\omega}\times)=\begin{bmatrix}0 & -\omega_z & \omega_y\\ \omega_z & 0 & -\omega_x\\ -\omega_y & \omega_x & 0\end{bmatrix}, \qquad (4.2)$$

складена з тих самих проекцій; **M** - матриця-стовпець з проекцій моменту зовнішніх сил на пов'язані осі.

Такими рівняннями описується, наприклад, обертальний рух космічного апарату. Тому у подальшому тіло іноді будемо ототожнювати з космічним апаратом.

Утворимо М-файл відповідної S-функції. Назвемо його ***S_DUE***.m.

```
function [sys,x0,str,ts] = S_DUE(t,x,M,flag,J,UgSk0)
 % S-функция S_DUE Динамических Уравнений Ейлера
 % Реализует динамику вращательного движения твердого тела,
 % отыскивая вектор абсолютной угловой скорости тела
 % по заданному вектору моментов внешних сил,
 % действующих на тело
 % ВХОД блока:
 %       M - вектор проекций момента внешних сил на оси
 %            X, Y i Z связанной с телом системы координат
 % ВЫХОД блока:
 %       y - вектор из шести элементов: первые три - проекции
 %            абсолютной угловой скорости om тела на указанные оси,
 %            последние три - проекции на те же оси
 %            углового ускорения тела
 % Входные ПАРАМЕТРЫ S-функции:
 %       J - матрица моментов инерции тела в указанных осях;
 %       UgSk0 - вектор начальных значений проекций
 %                  угловой скорости тела

 % Лазарев Ю.Ф., Украина, 18-12-2001

IJ=inv(J);        % вычисление обратной матрицы моментов инерции
switch flag,
  case 0
    [sys,x0,str,ts] = mdlInitializeSizes(UgSk0);
  case 1,
    sys = mdlDerivatives(t,x,M,J,IJ);
  case 3,
    sys = mdlOutputs(x,M,J,IJ);
  case 9
    sys = [];
end
%          Конец процедуры
%=======================================================
function [sys,x0,str,ts] = mdlInitializeSizes(UgSk0,Ug0)
sizes = simsizes;
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 6;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = UgSk0;
str = [];
ts  = [0 0];
 % Конец процедуры mdlInitializeSizes
 %=======================================
function z = mdlDerivatives(t,x,M,J,IJ)
 % ВХОДНОЙ вектор M является вектором проекций моментов внешних сил,
 % действующих на космический аппарат соответственно по осям X Y Z
 % x(1)=om(1); x(2)=om(2); x(3)=om(3);
```

```
 % z(1)=d(om(1))/dt; z(2)=d(om(2))/dt; z(3)=d(om(3))/dt;
 %    |Jx   Jxy  Jxz|
 % J = |Jxy   Jy    Jyz| - матрица моментов инерции КА
 %    |Jxz   Jyz Jz |
om=x(1:3); omx=vect2ksm(om);
z=lJ*(M-cross(om,J*om));       % ДИНАМИЧЕСКИЕ уравнения Ейлера
 % Конец процедуры mdlDerivatives
 %==================================
function y = mdlOutputs(x,M,J,lJ)
y(1:3)=x;
om=x(1:3);
zom=lJ*(M-cross(om,J*om))   ;    % Определения УСКОРЕНИЙ
y(4:6)=zom;
 % Конец процедуры mdlOutputs
```

В якості вхідного вектора утворюваного S-блоку прийнятий вектор M, який складається з трьох значень поточних проекцій вектора моменту зовнішніх сил, що діють на тіло, на осі системи декартових координат, пов'язаної з тілом. Утворимо вихідний вектор **y** із шести елементів: перші три – поточні значення проекцій абсолютної кутової швидкості тіла, другі три – проекції на ті самі осі вектора абсолютного кутового прискорення тіла:

```
y = [omx, omy, omz, epsx, epsy, epsz]
```

Утворюваний S-блок розглядається як неперервна система (з трьома неперервними змінними стану **x = [omx, omy, omz]**. Тому з тексту головної програми S-функції вилучена процедура ***mdlUpdate*** і залишена процедура ***mdlDerivatives***, яка є фактично підпрограмою правих частин динамічних рівнянь Ейлера.

Утворимо нове (пусте) вікно блок-схеми. Перетягнемо в нього стандартний блок S-функції з поділу ***User-Defined Functions*** бібліотеки ***SIMULINK***.

Подвійним клацанням мішкою на зображенні блоку викличем його вікно настроювання і запишемо в нього назву M-файла створеної S-функції і його параметри **J, UgSk0** (рис. 4.14). Клацнемо на кнопці ОК. Внаслідок цього вікно настроювання зникне, а на зображенні блока виникне ім'я **S_DUE**.

Тепер у цьому ж вікні з S-блоком створимо блок-схему для перевірки правильності роботи цього блоку. Але для цього попередньо потрібно продумати умови тестового прикладу.

Розглянемо такий випадок:

- на тіло не діють моменти зовнішніх сил, тобто тіло вільно обертається у просторі;
- осі декартової системи координат, яка жорстко пов'язана з тілом, спрямовані вздовж головних осей інерції тіла; за цих умов матриця моментів інерції буде діагональною;
- тіло є динамічно симетричним, а його вісь фігури спрямована вздовж другої осі тіло є динамічно симетричним, а його вісь фігури спрямована вздовж другої осі (Y) зв'язаноъ системи координат; це означає, що матриця моментів інерції матиме такий вид:

$$\mathbf{J} = \begin{bmatrix} J_e & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & J_e \end{bmatrix},$$

де J_e - екваторіальний момент інерції; J - момент інерції тіла відносно його осі фігури (осьовий момент інерції тіла);

- тіло попередньо приведено у обертання з кутовою швидкістю Ω навколо своєї осі фігури і має незначну (у порівнянні з Ω) початкову кутову швидкість ω_0 навколо осі X.

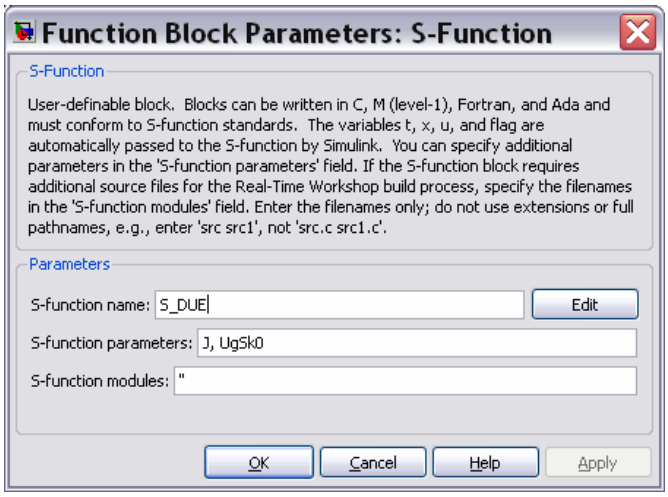


Рис. 4.14. Вікно настроювання блоку S_DUE

За цих умов рівняння (4.1) набудуть такого вигляду:

$$\begin{cases} J_e \dfrac{d\omega_X}{dt} = (J - J_e)\omega_Y \omega_Z \\ J \dfrac{d\omega_Y}{dt} = 0 \\ J_e \dfrac{d\omega_Z}{dt} = -(J - J_e)\omega_Y \omega_X \end{cases}$$

і при заданих початкових умовах матимуть такий розв'язок:

$$\omega_X = \omega_0 \cos(k\Omega t); \qquad \omega_Y = \Omega; \qquad \omega_Z = \omega_0 \sin(k\Omega t), \tag{4.3}$$

де позначено

$$k = \frac{J - J_e}{J_e}. \tag{4.4}$$

Отже, якщо утворен модель є правильною (адекватною процесові, що описується рівнянням (4.1)) і для неї забезпечені задані умови, при моделюванні ми маємо отримати результати, що відповідають формулам (4.3).

Додамо у блок-схему блок констант, який формує нульовий вектор моментів зовнішніх сил, а також блоки Scope, які дозволять проконтролювати результати моделювання у виді графіків залежностей проекцій кутової швидкості і кутового прискорення тіла від часу. В результаті одержимо блок схему, подану на рис. 4.15.

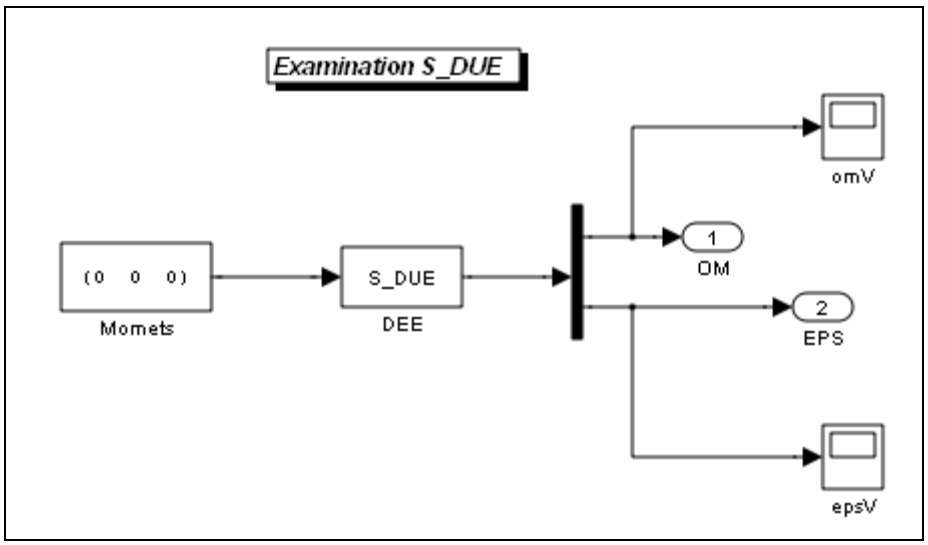


Рис. 4.15. Блок-схема S-моделі Prov

Перед запуском цієї S-моделі, якій дамо ім'я Prov, встановимо кінцевий час моделювання у 5000 с через команду Simulation > Configuration Parameters > Solver > Stop time, а також введемо у командному вікні Matlab оператори

```
>> J=diag([400 600 400])
J =
  400    0    0
    0  600    0
    0    0  400
>> UgSk0=[0.001 0.01 0]
UgSk0 =
   0.0010    0.0100         0
```

які задають матрицю моментів інерції і початкові умови.

Тепер слід перейти у вікно блок-схеми і запустит блок-схему на моделювання.

Звернувшися до оглядових блокві по завершенні процесу моделювання, можна впевнитися, що утворена модель дає результати, що повністю збігаються з одержаними з формул (4.3).

Показати у графічній формі результати роботи створеної моделі досить важко з огляду на наступні обставини. Блоки Scope виводять гріфіки на чорному полі. Тому при копіюванні відповідного графічного вікна на папір виходить неякісне зображення. Можна скопіювати його на папір за допомогою команди друку графічного вікна блоку Scope, але тоді відповідне зображення займе цілий аркуш – його неможливо зменшити засобами текстового редактору. Лінії на графіку після друку но чорно-білому прінтері не відрізнятимуться одна від одної. На них неможливо нанести написи, щоб вказати особливості кривих, до того ж складно змінити стиль лінії.

Виходячи з зазначеного можна висновувати, що найраціональнішим рішенням буде передати результати у робочий простір шляхом введення у блок схему виходних портів (див. рис. 4.15) і відправки на них тих сигналів, які потрібно подати графічно. Потім слід побудувати необхідні графіки, використовуючи графічні засоби Matlab. Останнє можна зробити бкзпосередньо, за допо-

могою команд Matlab, у командному вікні, але доцільніше виконати цю процедуру програмно, причому бажано об'єднати в ній усі дії:

1) введення значень параметрів, початкових умов тощо;

2) встановлення параметрів інтегрування;

3) звернення до S-моделі і запуск її на моделювання;

4) обробку одержаних результатів, побудову і оформлення графіків.

Приклад такої програми за ім'ям **Prov_upr** наведений нижче.

```
% Prov_upr
% Керуюча програма для запуска модели Prov.mdl

% Лазарев Ю.Ф.  23-07-2009
J=[400 0 0; 0 600 0; 0 0 400]; % Введення значень матриці інерції
UgSk0=[0.001 0.01 0];   % Введення початкових значень проекцій кутової швидкості
      % Встановлення параметрів моделювання
options=simset('Solver','ode4','FixedStep',5e1);
      %  МОДЕЛЮВАННЯ на S-моделі
sim('Prov',5000,options);
      % Формування даних і виведення графіків
tt=tout;    omx=yout(:,1);  omy=yout(:,2);  omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);

subplot(2,1,1), h=plot(tt,omx,'.-',tt,omy,'o-',tt,omz,'--');grid
set(h,'LineWidth',2);   set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','Fontsize',16)
title(' Проекції кутової швидкості'),   ylabel('рад / с'),
legend('\omega_X','\omega_Y','\omega_Z',0)

subplot(2,1,2), h=plot(tt,epsx,'.-',tt,epsy,'o-',tt,epsz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','Fontsize',16)
title('Проекції кутового прискорення'),   ylabel('1/c^2)'), xlabel('Час (с)')
set(gcf,'color','white'),   legend('\epsilon_X','\epsilon_Y','\epsilon_Z',0)
```

Запуск цієї програми дозволяє одержати графіки, показані на рис. 4.16. Тепер читач може наочно впевнитися у адекватності моделі.

Зауважимо, що смодельований рух відповідає вільному рухові симетричного гіроскопа – його нутаційним коливанням. У гіроскоп тіло перетворює надання йому порівняно швидкого обертання навколо однієї з його осей ($\omega_Y = \Omega = 0{,}01\ \ \text{с}^{-1}$). Матриця моментів інерції прийнята діагональною, тобто припускається,що тіло динамічно збалансовано відносно осей X, Y і Z. Нарешті, моменти інерції відносно осей X і Z прийняті равними. Це означає, що тіло є динамічно симетричним з віссю фігури Y.

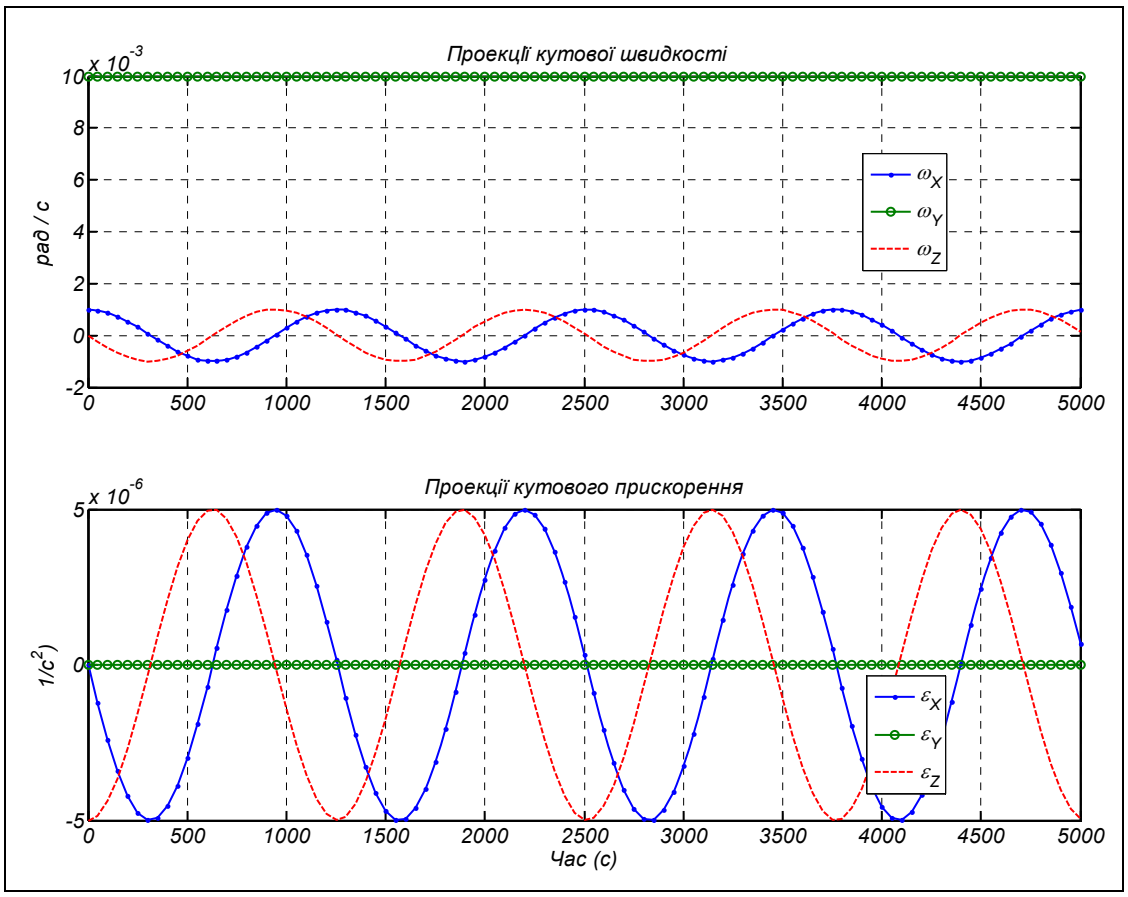


Рис. 4.16. Результати роботи програми Prov_upr

4.1.7. Запуск M-програм із S-моделі

Слід вказати ще один, більш зручний спосіб поєднання S-моделі з програмами Matlab. Він полягає у виклику M-файлів безпосередньо з S-моделі за допомою спеціально передбачених для цього засобів.

Припустимо, що перед початком завантажування S-моделі MODEL.mdl потрібно викликати M-файл, наприклад за ім'ям PERVdan, який містить операції присвоювання первісних значень усім даним. Це модна здійснити, якщо при створенні S-моделі з вказаним ім'ям у командному вікні Matlab ввести наступну команду:

set_param('MODEL', 'PreLoadFcn', 'PERVdan')

Вона пов'яже файл PERVdan.m з S-моделлю MODEL.mdl у такий спосіб, що він буде автоматично викликатися при виклику цієї S-моделі. Якщо після виконання цієї команди записати на диск дану S-модель, то при подальших її викликах спочатку автоматично буде викликаний файл PERVdan.m і лише потім на екрані виникне блок-схема S-моделі, готова для моделювання.

Перевірити, який саме M-файл використовується в даній S-моделі у якості попередньо виконуваного, можна шляхом виклику команди

get_param('ім'я S-моделі', **'PreLoadFcn')**

За допомогою функції **set_param** можна встановити в S-моделі значення багатьох її параметрів, у тому числі і параметрів окремих блоків моделі. У загальному виді звернення до функції може мати такий вид:

set_param('ім'я S-моделі / Ім'я блоку', 'Параметр1', Значення1, 'Параметр2', Значення2 ...)

Якщо вказаний параметр Ім'я блоку, то наступні значення присвоюються параметрам цього блоку. Наприклад, звернення

set_param('MODEL', 'Solver', 'ode15s, 'StopTime', 3000)

приведе до встановлення у S-моделі розв'язувача *ode15s* і часу завершення процесу моделювання – 3000.

При використанні звернення до цієї функції виду

set_param('MODEL / Rivnyannya', 'Gain', '1000')

у блоці Rivnyannya S-моделі MODEL параметру Gain буде присвоєно значення 1000.

Команда

set_param('MODEL / Fcn', 'Position', [50 100 110 120])

встановить зображення блоку Fcn у S-моделі MODEL у прямокутник з координатами **[50 100 110 120]** у вікні блок-схеми. При зверненні

set_param('MODEL / Compute', 'OpenFcn', 'my_open_fcn')

блок **Compute** S-моделі MODEL буде пов'язаний з M-програмой Matlab, яка записана у файлі **my_open_fcn.m**. Після цього файл **my_open_fcn.m** буде викликатися кожного разу після подвійного клацання на зображенні блоку **Compute**.

Якщо потрібно викликати деякий M-файл перед проведенням власне моделювання на S-моделі або після нього (наприклад, потрібна викликати програму, яка дозволяє змінити значення параметрів моделі у діалоговому режимі, або використати програму виведення результатів моделювання у графічній формі), можна встановити на вільному місці блк-схеми пусті блоки ***Subsystem*** (з поділу ***Ports & Subsystems***). Кожен з них здійснюватиме виклик відповідного M-файла.

Пусті блоки ***Subsystem*** блок-схеми можа зв'язати з певними M-програмами, набравши у командному вікні команду, аналогічну приведеній раніше. Спробуємо організувати таку форму керування процесом моделювання моделі ***Prov***.

Для цього створимо на основі раніше створеного файла ***Prov_upr*** три окремих файли:

- ***Prov_DUE_Pred.m***, який виконує присвоювання значень первісним величинам;
- ***Menu_DUE.m***, який здійснює змінювання первісних даних у діалоговому режимі;
- ***Graf_DUE.m***, який забезпечує виведення результатів моделювання у графічне вікно.

Тексти цих програм наведені нижче.

Програма Prov_DUE_Pred

```
% Prov_DUE_Pred
 % Програма встановлення первісних (вшитих) значень параметров моделі Prov.mdl

 % Лазарєв Ю.Ф.  23-07-2009
clc, clear all
 J=[400 0 0; 0 600 0; 0 0 400]; % Введення значень матриці інерції
UgSk0=[0.001 0.01 0];       % Введення початкових значень проекцій кутової швидкості
% Встановлення параметрів чисельного інтегрування
hi=10;  TK=10000;
```

Програма Menu_DUE

```
% MENU_DUE
% Програма змінювання первісних дазначень параметров модели Prov.mdl

% Лазарев Ю.Ф.  23-07-2009
k=1;
while k<12
   k=menu('Дані для моделі Prov. Що змінити?',...
        sprintf('Jx = %g',J(1,1)),       sprintf('Jy = %g',J(2,2)),       sprintf('Jz = %g',J(3,3)),...
       sprintf('Jxy = %g',J(1,2)),       sprintf('Jxz = %g',J(1,3)),       sprintf('Jyz = %g',J(2,3)),...
        sprintf('OMx(0) = %g',UgSk0(1)),   sprintf('OMy(0) = %g',UgSk0(2)),...
     sprintf('OMz(0) = %g',UgSk0(3)),  sprintf('hi = %g',hi),sprintf('TK = %g',TK),...
     'Нічого не змінювати');
   if k==1
      J(1,1)=input([sprintf('Поточне значення Jx=%g     ',...
                        J(1,1)),'Встановіть нове значення Jx=']);
   end
   if k==2
      J(2,2)=input([sprintf('Поточне значення Jy=%g     ',...
                        J(2,2)),'Встановіть нове значення Jy=']);
   end
   if k==3
      J(3,3)=input([sprintf('Поточне значення Jz=%g     ',...
                        J(3,3)),'Встановіть нове значення Jz=']);
   end
   if k==4
      J(1,2)=input([sprintf('Поточне значення Jxy=%g    ',...
                        J(1,2)),'Встановіть нове значення Jxy=']);
   end
   if k==5
      J(1,3)=input([sprintf('Поточне значення Jxz=%g    ',...
                        J(1,3)),'Встановіть нове значення Jxz=']);
   end
   if k==6
      J(2,3)=input([sprintf('Поточне значення Jyz=%g    ',...
                        J(2,3)),'Встановіть нове значення Jyz=']);
   end
   if k==7
      UgSk0(1)=input([sprintf('Поточне значення OMx(0)=%g   ',...
                        UgSk0(1)),'Встановіть нове значення OMx(0)=']);
   end
   if k==8
      UgSk0(2)=input([sprintf('Поточне значення OMy(0)=%g   ',...
                        UgSk0(2)),'Встановіть нове значення OMy(0)=']);
   end
   if k==9
      UgSk0(3)=input([sprintf('Поточне значення OMz(0)=%g   ',...
                        UgSk0(3)),'Встановіть нове значення OMz(0)=']);
   end
   if k==10
      hi=input([sprintf('Поточне значення hi=%g   ',hi), ...
                        'Встановіть нове значення hi=']);
   end
   if k==11
      TK=input([sprintf('Поточне значення TK=%g   ',TK),...
                         'Встановіть нове значення TK=']);
   end
end
J(2,1)=J(1,2); J(3,1)=J(1,3); J(3,2)=J(2,3);
```

Програма Graf_DUE

```
% Graf_DUE
% Програма побудови в графічному вікні графіків результатів роботи моделі
Prov.mdl

% Лазарєв Ю.Ф.  23-07-2009
        % Формування даних
tt=tout;
omx=yout(:,1);  omy=yout(:,2);  omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);
StrJ=[sprintf('Jx= %g; ',J(1,1)),sprintf('Jy= %g; ',J(2,2)),sprintf('Jz= %g; ',J(3,3))];
StrU=[sprintf('OMx= %g; ',UgSk0(1)),sprintf('OMy= %g; ',UgSk0(2)),sprintf('OMz= %g;
',UgSk0(3))];
StrJ1=[sprintf('Jxy= %g; ',J(1,2)),sprintf('Jxz= %g; ',J(1,3)),sprintf('Jyz= %g; ',J(2,3))];
tt=tout;   omx=yout(:,1);  omy=yout(:,2);  omz=yout(:,3);
epsx=yout(:,4); epsy=yout(:,5); epsz=yout(:,6);
        % Виведення графіків
    figure
subplot(2,1,1), h=plot(tt,omx,'.-',tt,omy,'o-',tt,omz,'--');grid
set(h,'LineWidth',2);   set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','Fontsize',16)
title(' Проекції кутової швидкості'),   ylabel('рад / с'),
xlabel([StrJ,'          ',StrJ1,'          ',StrU])
legend('\omega_X','\omega_Y','\omega_Z',0)

subplot(2,1,2), h=plot(tt,epsx,'.-',tt,epsy,'o-',tt,epsz,'--');grid
set(h,'LineWidth',2); set(gca,'LineWidth',2)
set(gca,'FontAngle','italic','Fontsize',16)
title('Проекції кутового прискорення'),   ylabel('1/c^2)'), xlabel('Час (c)')
set(gcf,'color','white'),   legend('\epsilon_X','\epsilon_Y','\epsilon_Z',0)
```

Тепер введемо у блок-схему Prov.mdl два блоки Subsystem. Здійснимо у цих блоках наступні перетворення.

1. Змінимо імена блоків (підписи під їхніми зображеннями): першій блок назвемо ***MENU***. а другий – ***GRAFIKI*** (рис. 4.17).

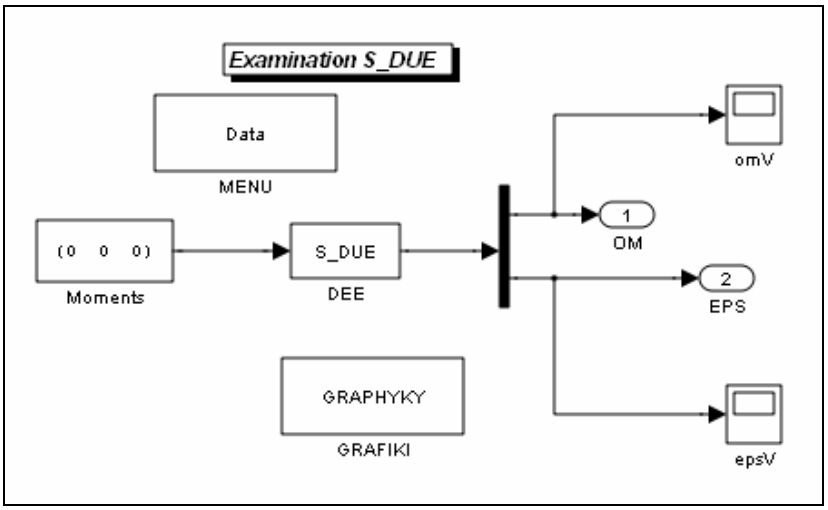


Рис. 4.17. Блок-схема S-моделі Prov1

2. Подвійно клацнувши на кожному з цих двох блоків, відчинемо їхні блок-схеми і видалимо увесь вміст – зробимо блоки порожніми.

3. У вікні блок схеми клацнемо на зображенні цих блоків правою кнопкою миші; з контекстного меню, що виникне після цього, оберемо команду *Mask Subsystem* (Створити маску підсистеми); в результаті відчиниться вікно *Mask Editor* (Редактор маски) (рис. 4.18).

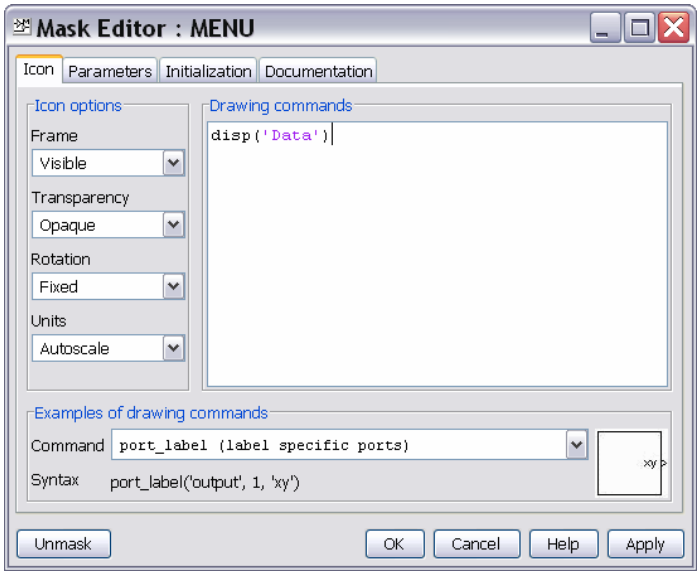


Рис. 4.18. Вікно Mask Editor

4. У полі *Drawing commands* (Команди рисування) вкладинки Icon (Значок) цього вікна (див. рис. 4.18) введемо команду **disp** з вказанням у якості аргумента тексту, якого потрібно розмістити на зображенні блоку; для першого блоку це буде текст *Data*, для другого – *GRAPHYKY*.
5. Зачинемо вікно *Mask Editor* клацанням по кнопці ОК.

Примітка. *В усіх написах усередині блок-схем слід використовувати тільки латиницю. Інакше відповідні моделі можуть не сприйнятися пакетом Simulink.*

Введемо у командному вікні команди, що пов'язують складені М-програми з S-моделлю:

```
set_param('Prov1', 'PreLoadFcn', 'Prov_DUE_Pred')
set_param('Prov1 / MENU', 'OpenFcn', 'Menu_DUE')
set_param('Prov1 / GRAFIKI', 'OpenFcn', 'Graf_DUE')
```

Після виконання цих команд керування усіма діями з моделювання буде здійснюватися з вікна самої S-моделі. Назвемо цю модифікацію S-моделі ***Prov1***.

Тепер моделювання можна робити у наступному порядку.

1. Спочатку викликаємо S-модель ***Prov1*** шляхом введення у командному вікні Matlab команди

```
>> Prov1
```

після її виконання первісні (вшиті) значення даних вже будуть записані у робочий простір, оскільки перед появою вікна блок-схеми на екрані буде запущена на виконання програма **Prov_DUE_Pred.**

2. Двічі клацнемо у вікні блок-схеми на блоці ***MENU*** (Data). При цьому відчиниться вікно меню (рис. 4.19).

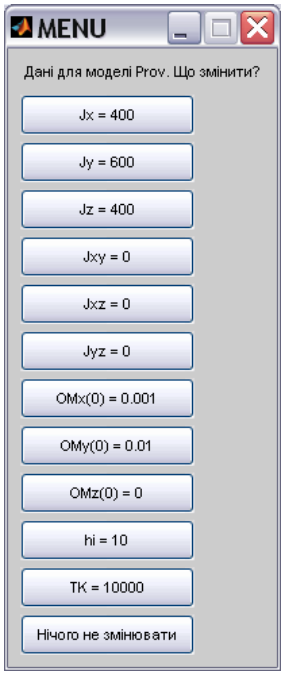


Рис. 4.19. Вікно меню змінювання даних

3. Встановивши потрібні значення параметрів, завершимо роботу з меню, клацнувши на кнопці *Нічого не змінювати*.
4. Встановимо параметри інтегрування через меню блок-схеми Simulation > Configuration Parameters > Solver, а саме (рис. 4.20):
   а) у полі Type встановимо Fixed-step;
   б) у полі Solver встановимо ode4;
   в) у полі Fixed-step size введемо: hi;
   г) у полі Stop time введемо: TK.
5. Запустимо S-модель на моделювання, клацнувши на кнопці з трикутною стрілкою на панелі інструментів вікна блок-схеми.
6. По закінченні процесу моделювання для виведення графіків у графічне вікно двічі клацнемо на блоці ***GRAFIKI***

Такий спосіб зв'язування S-моделі з існуючими M-файлами є, мабуть, найбільш зручним, бо, по-перше, дозволяє викликати M-файли лише у випадку потреби і у довільному порядку, а по-друге, керування моделюванням і викликом програм здійснюється лише з блок-схеми S-моделі.

На рис. 4.21-4.23 наведені результати моделювання вільного обертання тіла при різних сполученнях параметрів. В усіх випадкаху тілу попередньо надано обертання навколо осі Y з кутовою швидкістю $\omega_Y = 0{,}01$ рад/с.

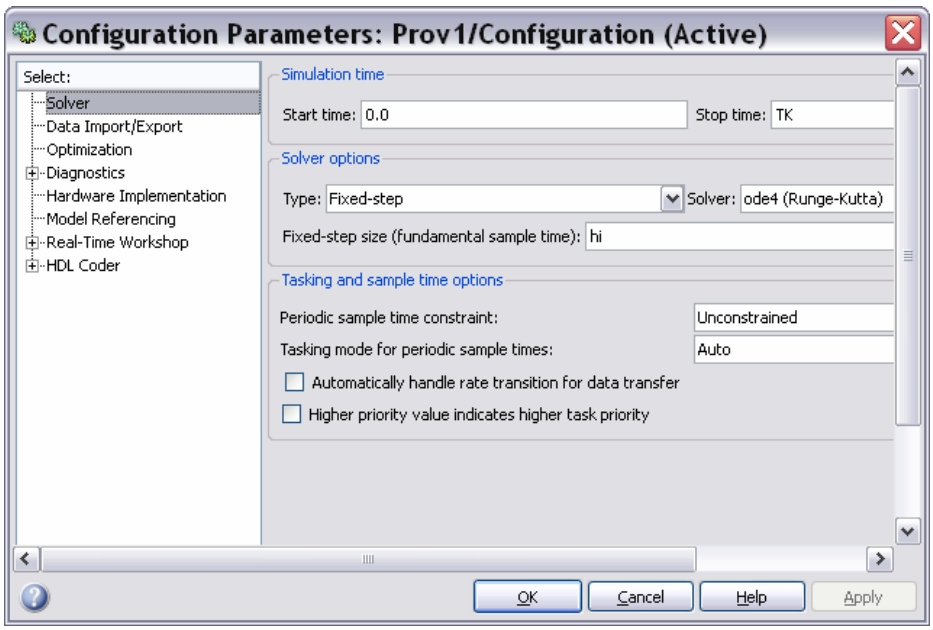


Рис. 4.20. Вікно Configuration Parameters

Графіки на рис. 4.21 відповідають динамічно несиметричному, але динамічно збалансованому тілу ($J_{XY} = J_{XZ} = J_{YZ} = 0$; $J_X = 400$; $J_Y = 600$; $J_Z = 200$ Н м с^2). Тіло має початкову швидкість $\omega_X = 0{,}002$ рад/с навколо осі X.

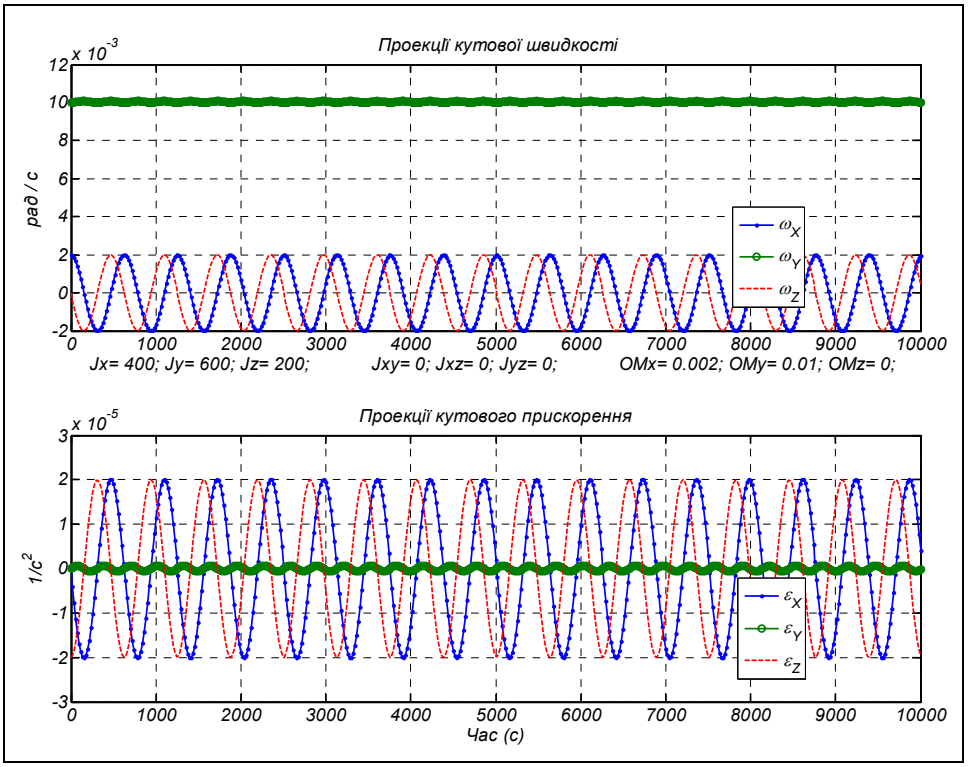


Рис. 4.21. Нутаційні коливання несиметричного гіроскопа

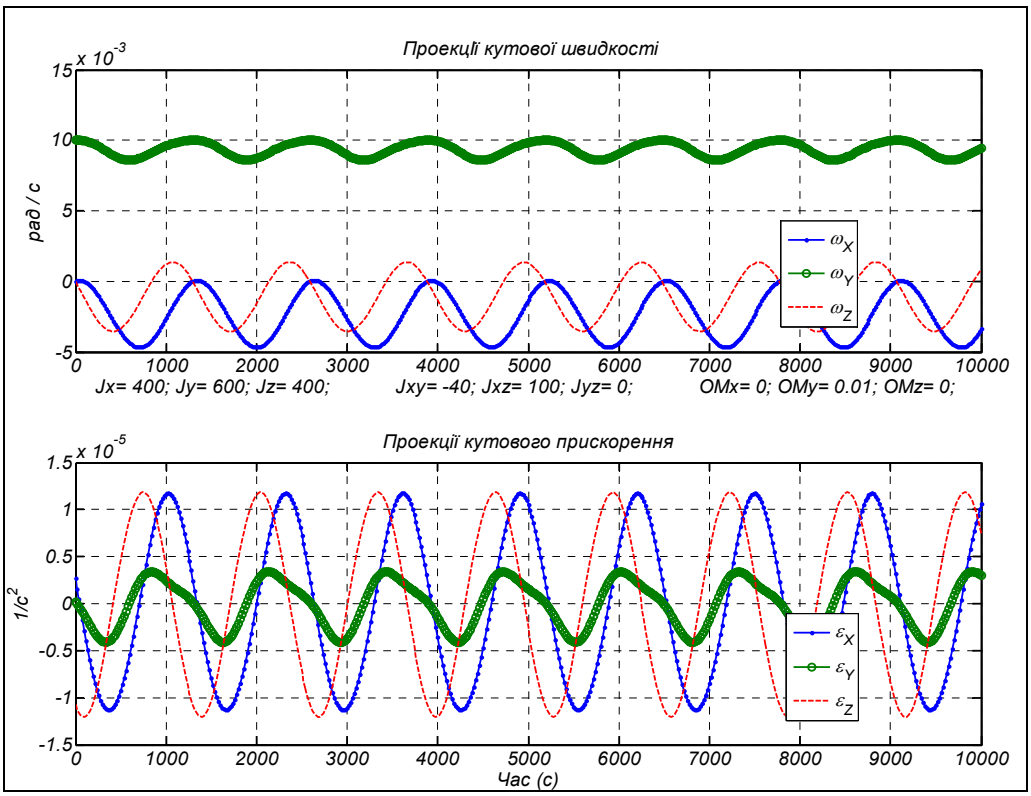


Рис. 4.22. Вільний рух динамічно незбалансованого гіроскопа

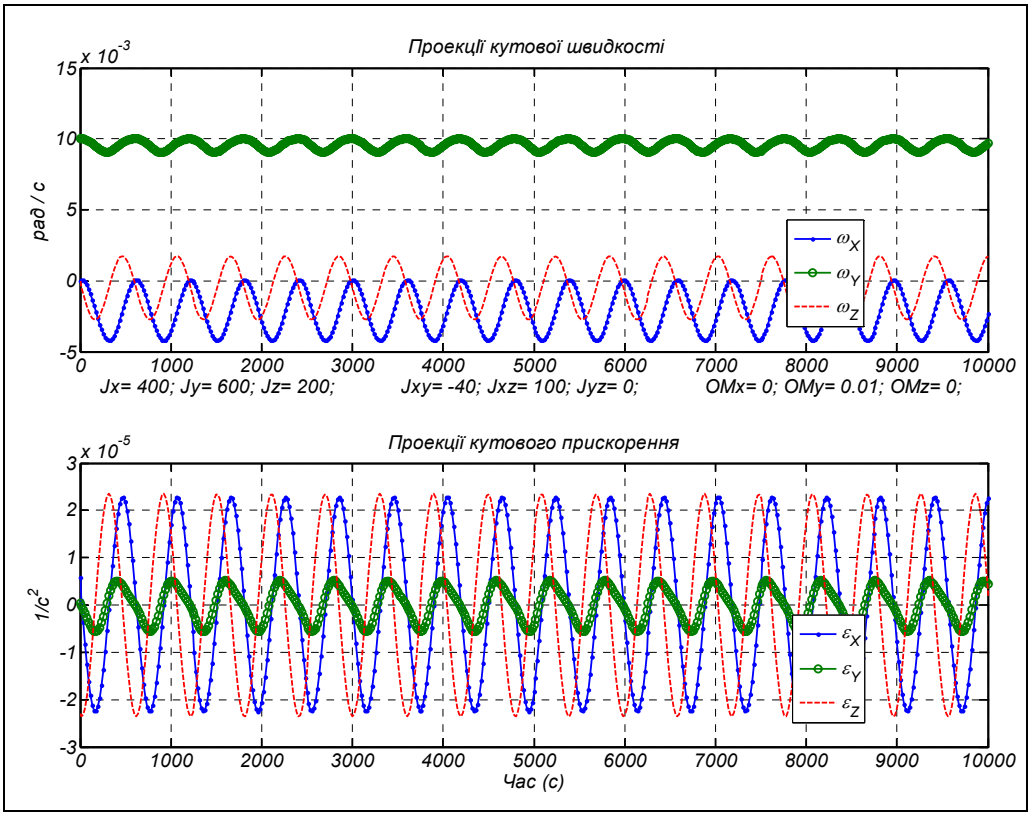


Рис. 4.23. Вільний рух несиметричного і незбалансованого гіроскопа

У наступних двох випадках $\omega_X = 0$, тобто тіло у початковий момент часу обертається лише навколо осі Y.

У другому випадку тіло є динамічно незбалансованим ($J_{XY} = -40;\quad J_{XZ} = 100;\quad J_{YZ} = 0;\ J_X = 400;\ J_Y = 600;\ J_Z = 400$ Н м с^2).

Третій варіант відповідає випадку, коли тіло є динамічно і несиметричним, і незбалансованим ($J_{XY} = -40;\quad J_{XZ} = 100;\quad J_{YZ} = 0;\ J_X = 400;\ J_Y = 600;\ J_Z = 200$ Н м с^2).

Як можна впевнитися по результатах цих "есперименів", вільний обертальний рух тіла суттєво залежить від його інерційних характеристик.

4.2. Користувацькі бібліотеки S-блоків

Ті, хто займається моделюванням систем, рано чи піздно наштовхуються на необхідність підготовки власних блоков, що мають властивості стандартних бібліотечних блоків пакету Simulink. Потреба у цьому виникає, коли користувач при виконанні різних задач моделювання в власній предметній області змушений неодноразово застосовувати створені їм елементарні блоки, яки є оригінальними і не входять у склад стандартних бібліотек Simulink, або багаторазово використовувати ті самі блоки у певних стійких їхніх сполученнях. У таких випадках, розмістивши нові блоки у бібліотеці, можна значно скоротити час утворення нових моделей того самого типу і запобігти появі похибок.

Перевага розміщення власних блоків у бібліотеці користувача полягає у тому, що їх можна застосовувати неодноразово, перетягуючи зображення блоку з бібліотеки у вікно блок-схеми. Користуватися такими блоками зручніше за все через спеціальні вікна настроювання, аналогічних тим, які розглядалися при опису стандартних блоков Simulink.

Створення вікон настроювання блоків здійснюється через формування так званої маски блоку, яка й відіграє роль вікна настроювання.

4.2.1. Утворення бібліотеки

Розглянемо процес створення бібліотеки S-блоків на конкретних прикладах.

Формування нової бібліотеки починається з відкриття вікна нової блок-схеми моделі. У цьому вікні слід викликати команду File > New > Library (Файл > Новий > Бібліотека). В результаті на екрані виникне порожнє вікно бібліотеки (рис. 4.24) з ім'ям Library: untitled1. У цьому вікні можна утворювати S-блоки, можна також перетягувати в нього блоки, яки вже створені.

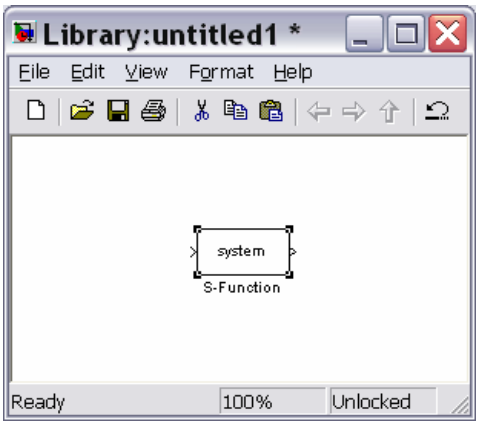


Рис. 4.24. Вікно новоутворюваної бібліотеки

У загальному випадку сформувати S-блок можна на основі стандартних блоків двох видів: блоку ***S-function*** з поділу *User-defined Function* бібліотеки Simulink і блоку ***Subsystem*** з поділу *Ports & Subsystems* той самої бібліотеки.

При утворенні S-блока на грунті блоку ***S-function*** використовуються файли S-функцій, написані мовою Matlab; такий блок має лише один вхід (можливо, векторний) і один вихід (векторний). S-блок, утворюваний на основі блоку ***Subsystem***, являє собою блок-схему, яка вміщує вже існуючі блоки, і може мати довільну кількість входів і виходів різного виду.

Утворимо у відчиненій нами бібліотеці S-блок, який назвемо ***S_DUE***, на основі створеної раніше однойменної S-функції.

1. Перетягнемо у вікно утворюваної бібліотеки блок ***S-function*** з поділу *User-defined Function* бібліотеки Simulink. Вікно бібліотеки набуде виду, наведеному на рис. 4.24.

2. Двічі клацнувши на зображенні цього блоку, викличемо вікно його настроювання (рис. 4.25).

3. У полі *S-function name* (Ім'я S-функції) вікна настроювання введемо ім'я S_DUE, а у полі *S-function parameters* (Параметри S-функції) – параметри *J*, *UgSk0*, потім клацнемо на кнопці OK. В результаті (за умови, що відповідний файл міститься у папках, досяжних для Matlab, а список введених параметрів відповідає списку параметрів, вказаних у S-функції), вікно настроювання зникне, і зображення блоку у вікні бібліотеки зміниться (рис. 4. 26).

4. Щоб точніше відобразити сутність перетворень, які здійснює блок, надамо йому назву ***Euler dynamic equations***.

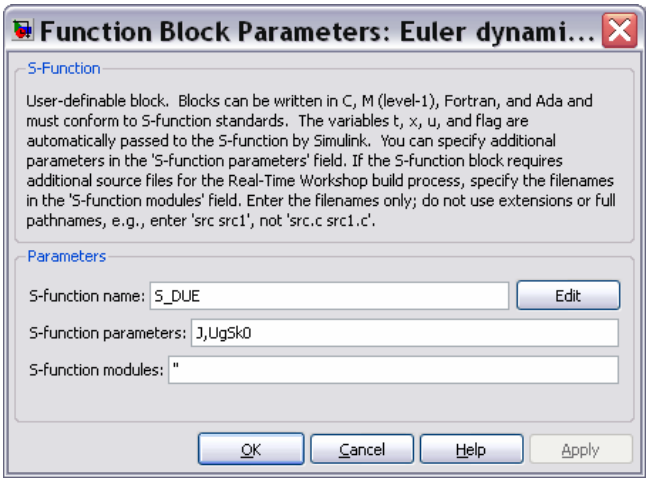


Рис. 4.25. Вікно настроювання блоку S_DUE

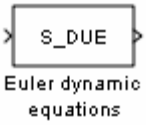


Рис. 4.26. Зображення блоку S_DUE

У подальшому для моделювання процесу керування орієнтацією, наприклад, космічного апарату (КА), що рухається навколо планети за певною замкненою орбітою, знадобиться ще один блок, який здійснює інтегрування кінематичних рівнянь орієнтації. В результаті обчислюються значення параметрів,

які визначають поточне кутове положення корпусу КА відносно орбітальної системи координат. Якщо у якості такого параметру обрати кватерніон повороту $\mathbf{Q}$, який описує перехід від орбітальної системи координат до зв'язаною з корпусом КА, відповідні кінематичні рівняння набудуть наступнох кватерніонної форми:

$$\frac{d\mathbf{Q}}{dt} = \frac{1}{2}(\mathbf{Q} \circ \boldsymbol{\omega} - \boldsymbol{\Omega} \circ \mathbf{Q}),$$

де $\boldsymbol{\omega}$ -вектор-кватерніон абсолютної кутової швидкості КА; $\boldsymbol{\Omega}$ - вектор-кватерніон абсолютної кутової швидкості орбітальної системи координат (жорстко зв'язаної з положенням КА на орбіті); $\circ$ - знак кватерніонного добутку.

Кватерніонне кінематичне рівняння не вельми зручно використовувати для проведення обчислень, з огляду на те, що дії над кватерніонами суттєво відрізняються від дій над матрицями і не передбачені у Matlab. Доцільніше перетворити це рівняння у систему матричних рівнянь:

$$\begin{cases} \dfrac{dq_0}{dt} = -\dfrac{1}{2}\mathbf{q}^t(\boldsymbol{\omega} - \boldsymbol{\Omega}) \\ \dfrac{d\mathbf{q}}{dt} = \dfrac{1}{2}[q_0(\boldsymbol{\omega} - \boldsymbol{\Omega}) + (\mathbf{q}\times)(\boldsymbol{\omega} + \boldsymbol{\Omega})] \end{cases} \tag{4.5}$$

У цих рівняннях величини $\mathbf{q}$, $\boldsymbol{\omega}$ и $\boldsymbol{\Omega}$ є векторами-стовпцями з проекцій, відповідно, векторної частини кватерніону, вектора абсолютної кутової швидкості КА на осі зв'язаної з корпусом КА системи координат, і вектора кутової швидкості орбітальної системи координат на її ж осі; q_0 - скалярна частина кватерніона повороту; $(\mathbf{q}\times)$ - кососиметрична матриця, складена з проекцій вектора $\mathbf{q}$.

Створимо М-файл S-функції, що здійснює інтегрування цих кінематичних рівнянь. Нижче наведений текст М-файла за ім'ям S_KUqwat.

```
function [sys,x0,str,ts] = S_KUqwat(t,x,u,flag,OM0,Qw0)
 % S-функція S_KUqwat Кінематичні Рівнянь орієнтації у кватерніонах
 % Реалізує перехід від заданого вектора абсолютної
 % кутової швидкості орбітального космічного апарату
 % до кватерніону поворота КА відносно орбітальної системи координат
 % ВХІД блоку:
 %     u = [omx,omy,omz]- вектор проекцій абсолютної кутової
 %         швидкості КА на осі СК, жорстко звязаної з ним
 % ВИХІД блоку:
 %     y=[qw0,qw1,qw2,qw3] - вектор компонентів кватерниона повороту
 %         відносно орбітальної декартової системи координат
 % Вхідні ПАРАМЕТРи S-функції:
 %     OM0 - орбитальна кутова швидкість;
 %     Qw0 - вектор початкового кватерніона повороту

 % Лазарев Ю.ф. Україна 28-07-2009
switch flag,
  case 0
    [sys,x0,str,ts] = mdlInitializeSizes(Qw0);
  case 1,
    sys = mdlDerivatives(t,x,u,OM0);
 case 3,
```

```
  sys = mdlOutputs(x);
 case 9
  sys = [];
end
 % Кінець процедури
        %
        %==============================================
         % Далі йдуть тексти внутрішніх процедур
        %==============================================
        %
        function [sys,x0,str,ts] = mdlInitializeSizes(Qw0)
        sizes = simsizes;
        sizes.NumContStates  = 4; sizes.NumDiscStates  = 0;
        sizes.NumOutputs     = 4;       sizes.NumInputs      = 3;
        sizes.DirFeedthrough = 1;  sizes.NumSampleTimes = 1;
        sys = simsizes(sizes);            x0  = Qw0;       str = [];       ts  = [0 0];
         % Кінець процедури mdlInitializeSizes

        %==============================================
        function z = mdlDerivatives(t,x,u,OM0)
        % ВХІДНИЙ вектор "u" - вектор проекцій абсолютної кутової швидкості КА
        % Вектор змінних стану X - складові кватерніона повороту
        %   x(1)=qw0 - скалярная часть кватерниона;
        %   x(2:4)=qw(1:3) - векторная часть кватерниона;
        % Вектор похідних змінних стану
        %   z(1)=d(qw0)/dt; z(2:4)=d(qw)/dt; ;
           % Формування векторів кутових швидкостей и кватерниона
        om=u;       OM  = [0 OM0 0]';   v=x(2:4);
        omMOM=om-OM;    omPOM=om+OM;
        z(1)=-(v'*omMOM)/2; % уравнения скалярной части кватерниона
        z4=(x(1)*omMOM+cross(v,omPOM))/2;% уравнения векторной части кватерниона
        z(2:4)=z4;
        % Кінець процедури mdlDerivatives

        %==============================================%
        function y = mdlOutputs(x)
        y=x;
         % Кінець процедури mdlOutputs
```

За аналогією з попереднім утворимо S-блок за ім'ям ***S_KUqwat***. Його входом є вектор проекцій абсолютної кутової швидкості КА, а виходом – вектор з чотирьох компонентів кватерніона поворота КА відносно орбітальної системи координат. Першій компонент являє собою скалярну частину, решта три – проекції векторної частини цього кватерніона. В результаті отримаємо вікно бібліотеки у виді, поданому на рис. 4. 27.

Рис. 4.27. Вид бібліотеки з доданим блоком S_KUEquat

Нарешті, досить важливо утворити S-блок, який здійснював би операцію векторного множення двох векторів, аналогичну М-функції **cross**.

Для цього зручніше використати інший стандартний блок ***SubSystem*** з поділу *Ports & Subsystems*. Перетягнемо його мишкою у вікно нової бібліотеки (рис. 4.28).

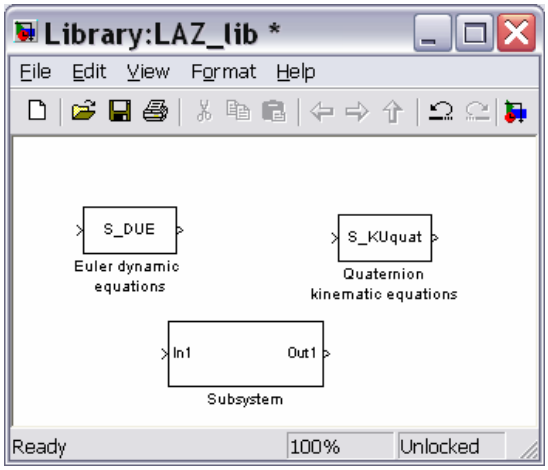


Рис. 4.28. Включення у бібліотеку блоку Subsystem

Двічі клацнувши на зображенні цього блоку, одержимо порожнє вікно, у якому складемо блок-схему підсистеми, наведену на рис. 4.29.

В ній використаний блок ***MATLAB Function*** з поділу *User-Defined Functions*, вікно настроювання якого подано на рис. 4.30. Саме він, власне, й виконує операцію векторного множення двох вхідних векторів, використовуючи для цього стандартну функцію **cross** системи MatLab.

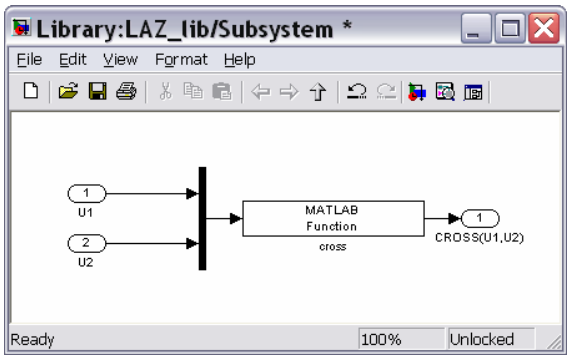


Рис. 4.29. Блок-схема підсистеми векторного добутку

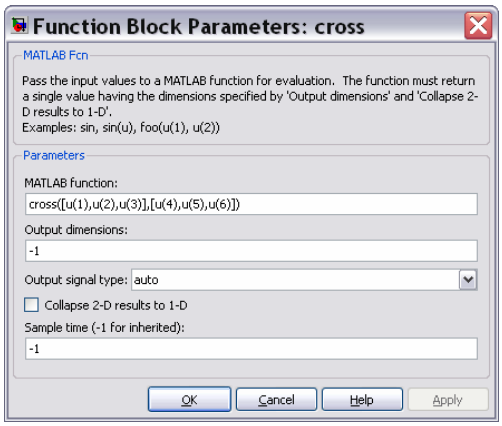


Рис. 4.30. Вікно настроювання блоку Cross (MATLAB Function)

У підсумку нами створена бібліотека, яка складається з трьох нових власних S-блоків. Запишемо її як **LAZ_lib** (рис. 4.31).

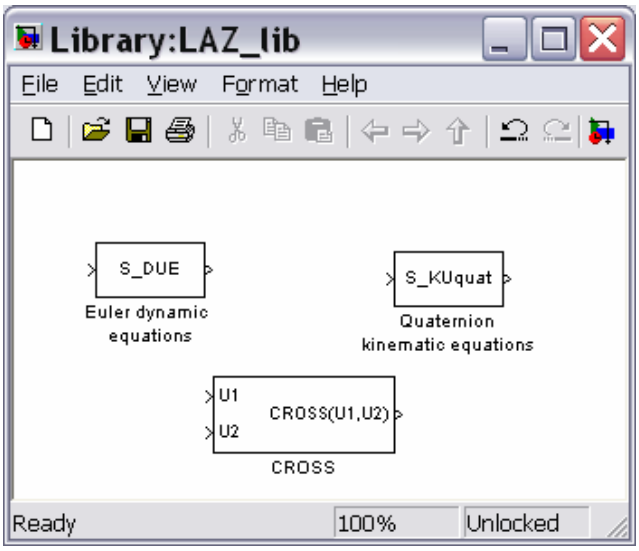


Рис. 4.31. Бібліотека користувача LAZ_lib

4.2.2. Утворення вікна настроювання (маски) блоку

Розглянемо процес створення вікна настроювання новоутвореного S-блоку. Це вікно називається по іншому маскою блоку і є зручним засобом встановлення і змінювання параметрів блоку.

Перш за все, потрібно виділити у бібліотеці той блок, для якого бажано утворити маску. Нехай це буде блок ***S_DUE*** нової бібліотеки. Потім слід виконати команду *Edit* > *Mask S-Function* (Редагування > Маскування S-функції) вікна бібліотеки, де міститься виділений блок. На екрані виникне вікно *Mask editor* редактора маски, подане на рис. 4.32.

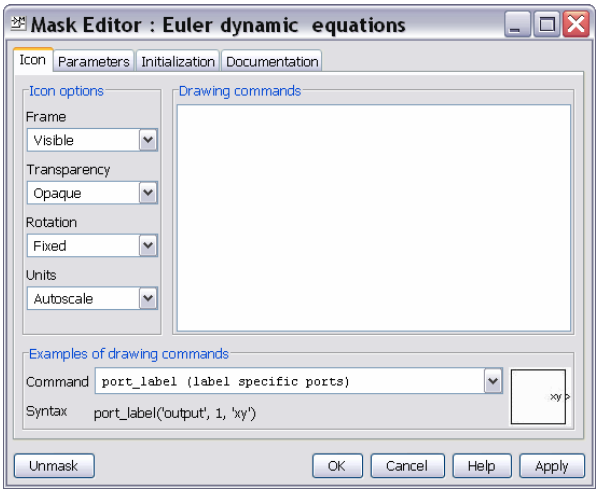


Рис. 4.32. Вікно Mask Editor

Примітка. Можливо, що при повторному виклику бібліотеки команда *Edit* > *Mask S-Function* буде не досяжною (не активованою). У цьому випадку зверніть увагу на команду *Unlock Library* (Розблокувати

бібліотеку) у тому ж меню – вона має бути активною. Після клацання на ній команда *Mask S-Function* має також стати активною.

Вікно Mask Editor (рис. 4.32) має чотири вкладинки:

Icon	для створення і редагування зображень на блоці;
Parameters	для створення і редагування діалогової частини (введення параметрів) вікна настроювання;
Initialization	для введення деяких команд середовища MatLab при ініціалізації блоку;
Documentation	для оформлення і редагування текста і довідкової частини маски.

Вікна настроювання (маски) мають, в загальному випадку, три частини. Верхня частина містить інформацію про призначення блоку, його головних параметрах і правилах, яких слід дотримуватися при встановленні параметрів блоку і цього застосуванні. У середній частині маски містяться поля введення параметрів блоку, написи над ними пояснюють сенс параметрів. Нижня частина вікна настроювання містить стандартні кнопки *OK, Cancel, Help* і *Apply.*

Редактор маски призначений для оформлення перших двох частин вікна настроювания, а також утворення довідки, яка викликається при клацанні на кнопці *Help* .

Перейдемо до вкладинки *Documentation* (рис. 4.33). У вікно *Mask type* (Тип маски) слід ввести ім'я блоку. У вікно *Mask description* (Опис маски) записується інформація, яка має бути відображеною у довідковій верхній частині вікна настроювання, а у полі *Mask help* (Довідка маски) –додаткова довідкова інформація, яка має виникати після клацання на кнопці *Help* .

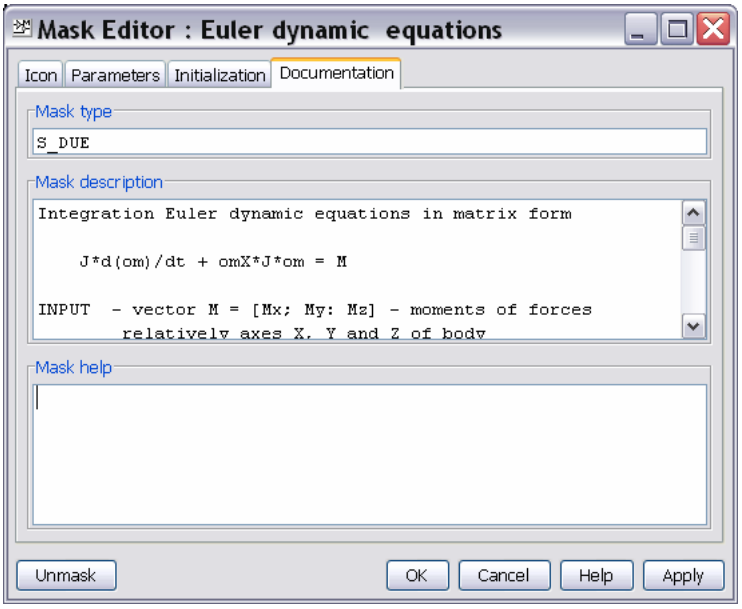


Рис. 4.33. Вкладинка Documentation вікна Mask Editor

Увага! Поява у тексті документації маски знаків кириліці зазвичай приводить до порушень роботи маски і блоки. Тому бажано при написан-

ні тексту документів маски застосовувати виключно знаки латиниці.

За допомогою вкладинки Parameters (Параметри) (рис. 4.34) можна сконструювати найважливішу частину маски – діалогову. Як видно з рисунку, головний простір цієї вкладинки займає область (поки що недосяжная) Dialog Parameters (Параметри діалога). В ній вводяться параметри, які визначають кількість полів введення у вікні настроювання, написи над ними і ймення, за якими вони фігуруватимуть у блоці. Ліворуч від вказаної області знаходиться єдина досяжна (активна) кнопка (вгорі) з зображенням стрілки. Це кнопка Add (Додати). Клацання на цій кнопці приводить до активізації області Dialog Parameters – в ній виникає рядок, в який слід ввести параметри.

У блоці ***S_DUE*** є два параметри, значення яких потрібно вводити у діалоговому режимі, - матриця моментів інерції тіла J розміром 3×3 і вектор UgSk0 початкових значень трьох проекцій кутової швидкості тіла. Тому потрібно створити два поля для введення значень вказаних параметрів і зробити написи до них.

Додавання чергового поля у маску здійснюється через клацання мишкою на кнопці Add (Додати). Сам запис виконується в області *Dialog Parameters* (Параметри діалога).

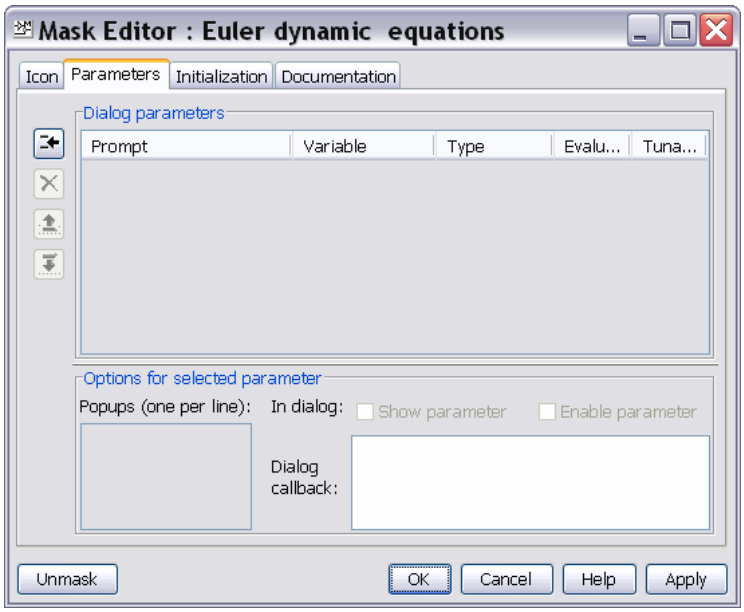


Рис. 4.34. Вкладинка Parameters вікна Mask Editor

У стовпець Prompt (Підказка) записується текст напису, що міститься над полем введення, а у стовпець Variable – ім’я, під яким введена величина фігуруватиме у блоці (рис. 4.35).

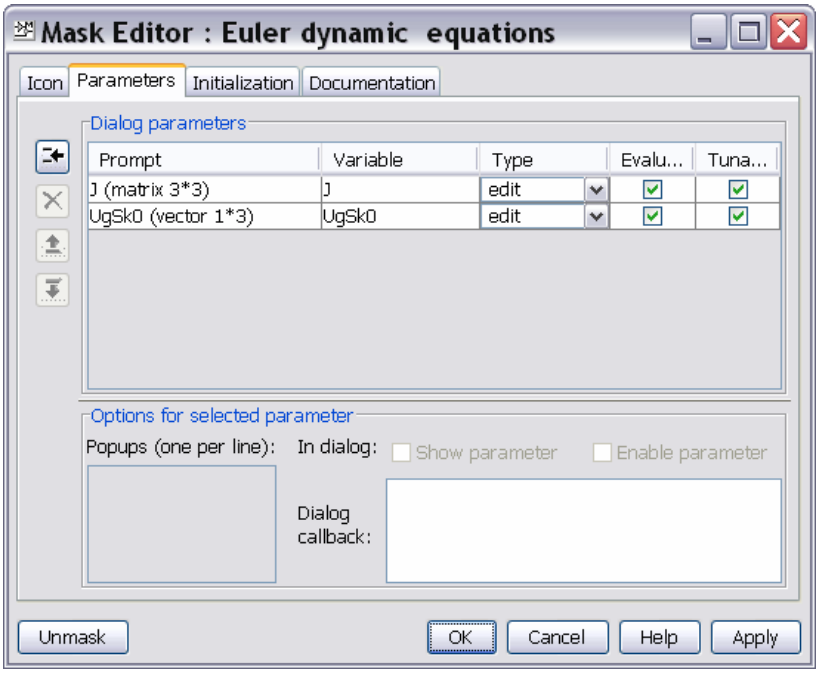


Рис. 4.35. Введення діалогових параметрів маски блоку S_DUE

Для завершення процесу створення маски, клацніть на кнопці ОК у вікні редактора маски, перейдіть у вікно бібліотеки і двічі клацніть на зображенні блоку ***S_DUE*** . На екрані виникне вікно настроювання цього блоку (рис. 4.36).

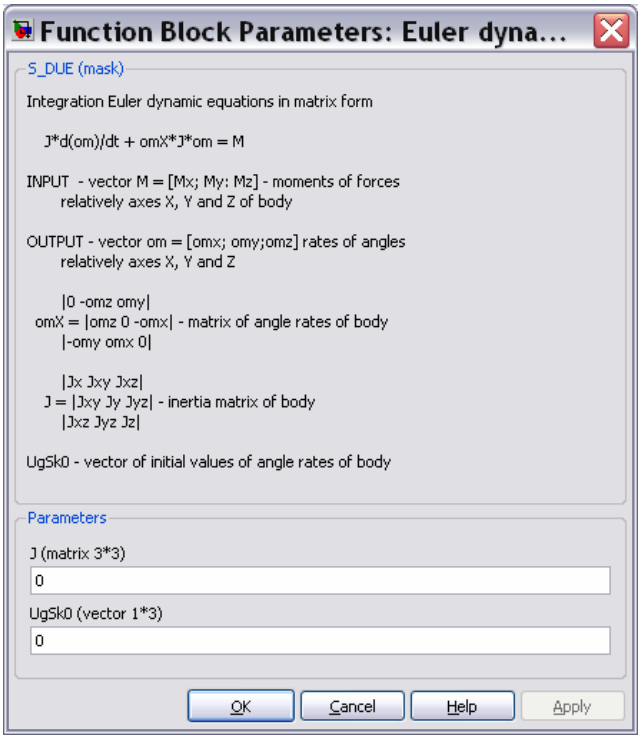


Рис. 4.36. Вікно настроювання (маска) блоку S_DUE

У такий самий спосіб утворюється вікно настроювання блоку ***S_KUquat*** (рис. 4.37)

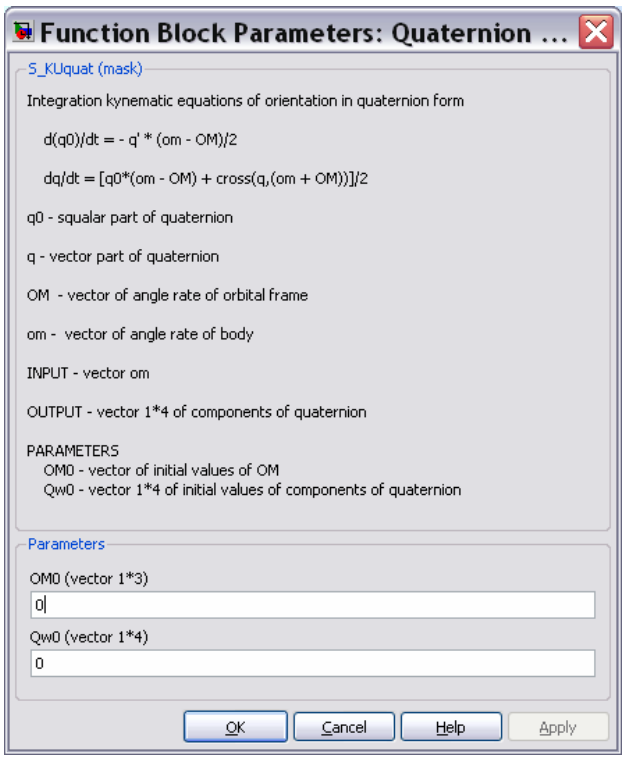


Рис. 4.37. Вікно настроювання (маска) блоку S_KUquat

4.3. Приклади застосування користувацької бібліотеки

4.3.1. Орієнтування космічного апарату

Як приклад застосування блоків власної бібліотеки розглянемо процес утворення S-моделі і комплекса М-програм, призначених для моделювання процесу автоматичного керування ориентацией космічного апарату (КА), зокрема, штучного супутника Землі (ШСЗ).

Задля спрощення будемо припускати космічний апарат твердим тілом, яке обертається навколо Землиі за замкненою орбітою. Керування орієнтацією (тобто кутовим положенням відносно орбітальної системи координат) здійснюється за допомогою трьох маховичних двигунів, осі яких збігаються з осями декартової системи координат, жорстко зв'язаної з корпусом космічного апарату. Маховичні двигуни, з одного боку, виконують розгін відповідного ротора у відповідності до рівнянь

$$\frac{dH_k}{dt} = M_k, \qquad (k = x, y, z\ ); \tag{4.6}$$

а, з іншого боку, накладають відповідний моменту сил (але у протилежному напрямку) навколо осі обертання ротора на корпус самого космічного апарату. Останній момент викликає змінювання кутового руху КА навколо цієї осі, тобто здійснює керування орієнтацією.

Змінювання кутової орієнтації космічного апарату у просторі підпорядковується основним законам механіки, з яких випливають рівняння, воплощенные в блоках **S_DUE** и **S_KUqwat**.

Складемо блок-схему S-моделі системи орієнтації і збережемо її у файлі ***SUO_KA.mdl*** (рис. 4.38). Она приведена на рис. 4.38.

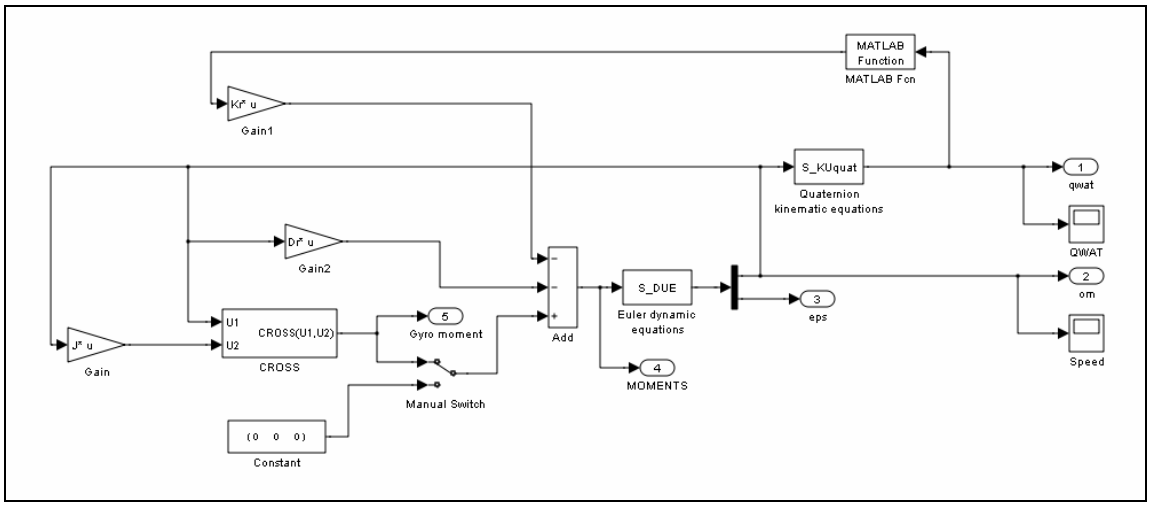


Рис. 4.38. Блок-схема S-моделі системи керування орієнтацією КА

Блок-схема складається з послідовно з'єднаних блоків **S_DUE** і **S_KUquat**, охоплених трьома ланцюгами зворотного зв'язку, які забезпечують керування

космічним апаратом по кватерніону, кутовій швидкості і компенсацію гіроскопичного моменту, що виникає при поворотах КА. Отримане на виході блоку ***S_KUquat*** значення кватерніону поворота надходить до входу блоку ***MATLAB Function***, який виділяє векторну частину цього кватерниону, необхідну для формування вектора моменту керування кутовим положенням КА (рис. 4. 39).

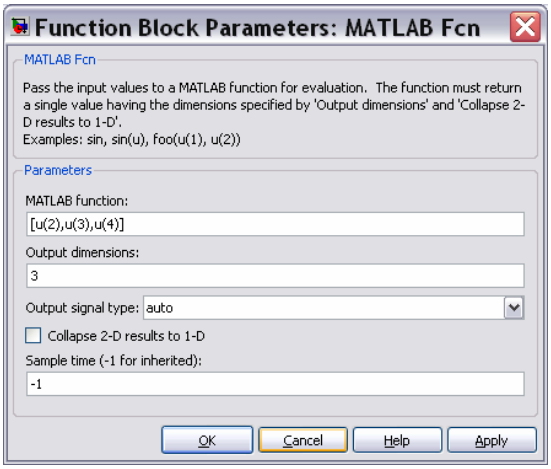


Рис. 4.39. Вікно настроювання блоку MATLAB Fcn

У цілому момент сил керування формується за наступним матричним законом:

$$\mathbf{M} = -K_r \cdot \mathbf{q} - D_r \cdot \boldsymbol{\omega} + (\boldsymbol{\omega}\times)\cdot(J\cdot\boldsymbol{\omega}). \qquad (4.7)$$

У ньому можна розрізняти три складові:

- складову, пропорційну векторній частині кватерниона відхилення поточного положення КА від заданого його положення (у цій програмі задане положення відповідає нульовому значенню векторної частини кватерніона); саме ця складова моменту керування змушує КА наближатися до заданого кутового положення;

- складову, пропорційну вектору кутової швидкості КА; вона забезпечує демпфірування процесу наближення КА до заданого положення;

- третя складова моменту вводиться для того, щоб компенсувати виникаючий при кутовому русі КА гіроскопічний момент, який прагне повернути КА навколо осі, перпендикулярної осі дії моменту сил керування; введення цієї складової змушує корпус КА повертатися до заданного положення за найкоротшим шляхом, тобто зменшує енергетичні витрати на переорієнтацію КА.

Матриці K_r и D_r визначають закон керування і мають бути відомими заздалегідь.

Формування першої складової моменту керування на блок-схемі здійснюється верхнім зворотним ланцюгом з матричним підсилювачем Kr. Друга складовая формується ланцюгом з матричним підсилювачем Dr. Третя складова забезпечується третім зворотним ланцюгом, у склад якого входять матричний підсилювач J і утворений раніше блок ***CROSS*** (см. рис. 4. 38).

Ці три складові підсумовуються на суматорі і надходять до входу блоку ***S_DUE***. У такий спосіб утворюється замкнена система керування.

Поставимо задачу промоделювати поводження системи орієнтації КА, керованого за компонентами кватерніона при різних законах регулювання у відповідності з даними, наведеними у статті [6], тобто за наступних значень матриці інерції КА

```
J=[1200 100 -200;100 2200 300;-100 300 3100]
J =
      1200       100      -200
       100      2200       300
      -100       300      3100
```

матриці моментів демпфірування

```
Dr=0.315*diag([1200 2200 3100])
Dr =
  378.0000         0         0
         0  693.0000         0
         0         0  976.5000
```

і за чотирьох значень матриці позиційного керування:

1) матриця керування є пропорційною оберненій матриці моментів інерції

```
Kr=k/J=diag([201,110,78]);
Kr =
  201    0    0
    0  110    0
    0    0   78
```

2) матриця керування є пропорційною одиничній матриці E

```
Kr=k*E=diag([110,110,110]) ;
Kr =
  110    0    0
    0  110    0
    0    0  110
```

3) матриця керування являє собою комбінацію матриць інерції і одиничної

```
Kr=(αJ+βE) =diag([72,110,204]);
Kr =
   72    0    0
    0  110    0
    0    0  204
```

4) матриця керування є пропорційною матриці J моментів інерції

```
Kr=k*J=diag([60,110,155])
 Kr =
   60    0    0
    0  110    0
    0    0  155
```

Будемо припускати, що орбіітальна кутова швидкість дорівнює нулеві, а початкове відхилення положення КА від заданого визначається кватерніоном

```
Qw0=[0.159 0.57 0.57 0.57]
```

що відповідає початковому відхиленню від потрібного положення, рівному $161{,}7^{o}$.

Щоб почати моделювання, потрібно присвоїти первісні значення усім параметрам. По закінченні моделювання необхідно на основі отриманих даних побудувати ряд графіків, які відбивали би процес переорієнтації КА. Виконаємо

ці процедури (а також саме моделювання) за допомогою спеціального М-файла ***SUO_KAupr.m***. Його текст наведений нижче

```
 % SUO_KAupr.m
 % Керуюча програма  для запуска моделі SUO_KA.mdl

 % Лазарєв Ю.Ф.  18-12-2001
 % Останні змінювання 5-08-2009
clear all,  clc
K=[3,1,2];      OM0=0;
 % Введение значений матрицы инерции
J=[1200 100 -200; 100 2200 300; -200 300 3100]
 % Введение начальных значений:
 % 1) проекций угловой скорости тела
UgSk0=[0 0 0];
 % 2)_компонентов кватерниона поворота
Qw0=[0.159,0.57,0.57,0.57];   qw0=Qw0(1),  qw=Qw0(2:4)
U0=2*acos(qw0); Cs0=qw/sin(U0/2);   Zk=[1 0 0 0];
 % Матрицы моментов УПРАВЛЕНИЯ
Dr=0.315*diag(diag(J))
V=[201,110,78;110 110 110;72 110 204;60 110 155]
for k=1:4
   % Установление параметров моделирования
  Kr=diag(V(k,:))
  options=simset('Solver','ode45','RelTol',1e-6);
  sim('SUO_KA',[0,100],options);        % МОДЕЛИРОВАНИЕ на S-модели
   % Формирование данных для вывода ГРАФИКОВ
  tt=tout;
  q0=yout(:,1);   qx=yout(:,2);   qy=yout(:,3);     qz=yout(:,4);
  omx=yout(:,5); omy=yout(:,6); omz=yout(:,7);
  Mx=yout(:,11); My=yout(:,12);  Mz=yout(:,13);
  if k==1
    t1=tt;
    q01=q0;   qx1=qx;  qy1=qy;  qz1=qz;
    omx1=omx;  omy1=omy; omz1=omz;
    Mx1=Mx;  My1=My; Mz1=Mz;
  elseif k==2
    t2=tt;
    q02=q0;   qx2=qx;  qy2=qy;  qz2=qz;
    omx2=omx;  omy2=omy; omz2=omz;
    Mx2=Mx;  My2=My; Mz2=Mz;
  elseif k==3
    t3=tt;
    q03=q0;       qx3=qx;       qy3=qy;       qz3=qz;
    omx3=omx;  omy3=omy; omz3=omz;
    Mx3=Mx;       My3=My;      Mz3=Mz;
  elseif k==4
    t4=tt;
    q04=q0;       qx4=qx;       qy4=qy;       qz4=qz;
    omx4=omx;  omy4=omy; omz4=omz;
    Mx4=Mx;       My4=My;      Mz4=Mz;
  end
  clear tt q0 qx qy qz omx omy omz Mx My Mz
end
A=180/pi;
D1=2*acos(q01); D2=2*acos(q02); D3=2*acos(q03); D4=2*acos(q04);
dt1=[0;diff(t1)];
dHx1=Mx1.*dt1; dHy1=My1.*dt1; dHz1=Mz1.*dt1;
domx1=[0;diff(omx1)]; domy1=[0;diff(omy1)]; domz1=[0;diff(omz1)];
DEx1=cumsum(abs(dHx1.*domx1));  DEy1=cumsum(abs(dHy1.*domy1));
DEz1=cumsum(abs(dHz1.*domz1));
```

```
d1=DEx1+DEy1+DEz1;  dt2=[0;diff(t2)];
dHx2=Mx2.*dt2; dHy2=My2.*dt2; dHz2=Mz2.*dt2;
domx2=[0;diff(omx2)]; domy2=[0;diff(omy2)]; domz2=[0;diff(omz2)];
DEx2=cumsum(abs(dHx2.*domx2));  DEy2=cumsum(abs(dHy2.*domy2));
DEz2=cumsum(abs(dHz2.*domz2));
d2=DEx2+DEy2+DEz2;  dt3=[0;diff(t3)];
dHx3=Mx3.*dt3; dHy3=My3.*dt3; dHz3=Mz3.*dt3;
domx3=[0;diff(omx3)]; domy3=[0;diff(omy3)]; domz3=[0;diff(omz3)];
DEx3=cumsum(abs(dHx3.*domx3));  DEy3=cumsum(abs(dHy3.*domy3));
DEz3=cumsum(abs(dHz3.*domz3));
d3=DEx3+DEy3+DEz3;  dt4=[0;diff(t4)];
dHx4=Mx4.*dt4; dHy4=My4.*dt4; dHz4=Mz4.*dt4;
domx4=[0;diff(omx4)]; domy4=[0;diff(omy4)]; domz4=[0;diff(omz4)];
DEx4=cumsum(abs(dHx4.*domx4));  DEy4=cumsum(abs(dHy4.*domy4));
DEz4=cumsum(abs(dHz4.*domz4));  d4=DEx4+DEy4+DEz4;
 % Графики проекций компонентов кватерниона на плоскост

subplot(2,2,1)
plot(qx1,qy1, qx2,qy2,':',qx3,qy3,'--',qx4,qy4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('          Проекції КОМПОНЕНТІВ кватерніонів')
xlabel('Q_x'),  set(gca,'FontName','Helvetica'), ylabel('Q_y')

subplot(2,2,2)
plot(qy1,qz1,qy2,qz2,':',qy3,qz3,'--',qy4,qz4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('  на координатні площини              ')
xlabel('Q_y'), set(gca,'FontName','Helvetica'), ylabel('Q_z')

subplot(2,2,3)
plot(qx1,qz1, qx2,qz2,':',qx3,qz3,'--',qx4,qz4,'.'), grid
set(gca,'FontSize',14')
xlabel('Q_x'), set(gca,'FontName','Helvetica'), ylabel('Q_z')
legend('Kr = k/J','Kr = kE','Kr = k/(\alphaJ+\betaE)','Kr = kJ',4)

subplot(2,2,4)
c1=D1./sin(D1/2)*A;
cx1=c1.*qx1; cy1=c1.*qy1; cz1=c1.*qz1;   c2=D2./sin(D2/2)*A;
cx2=c2.*qx2; cy2=c2.*qy2; cz2=c2.*qz2;   c3=D3./sin(D3/2)*A;
cx3=c3.*qx3; cy3=c3.*qy3; cz3=c3.*qz3;   c4=D4./sin(D4/2)*A;
cx4=c4.*qx4; cy4=c4.*qy4; cz4=c4.*qz4;
plot3(cx1,cy1,cz1,cx2,cy2,cz2,':',cx3,cy3,cz3,'--',cx4,cy4,cz4,'.'),grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Вектор ЕЙЛЕРОВОГО повороту у просторі')
xlabel('E_x (градуси)')
ylabel('E_y (градуси)'), set(gca,'FontName','Helvetica'),   zlabel('E_z')
 % Графики зависимостей компонентов кватерніону от времени
figure
subplot(2,2,1)
plot(t1,qx1,t2, qx2,':',t3,qx3,'--',t4,qx4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('        Залежність КОМПОНЕНТІВ кватерніонів від ЧАСУ')
xlabel('Час (с)'), set(gca,'FontName','Helvetica'), ylabel('Q_x')

subplot(2,2,2)
plot(t1,qy1,t2,qy2,':',t3,qy3,'--',t4,qy4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
 xlabel('Час (с)'), set(gca,'FontName','Helvetica'), ylabel('Q_y')

subplot(2,2,3)
plot(t1,qz1,t2,qz2,':',t3,qz3,'--',t4,qz4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
```

```
 xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('Q_z')

subplot(2,2,4)
plot(t1,D1*A,t2,D2*A,':',t3,D3*A,'--',t4,D4*A,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Поворот навколо осі Ейлера')
xlabel('Час (c)'),  set(gca,'FontName','Helvetica'), ylabel('Kut (gradus)')
legend('Kr = k/J','Kr = k','Kr = k/(\alphaJ+\betaE)','Kr = kJ')
 % Графики зависимостей проекций момента сил от времени
figure
subplot(2,2,1)
plot(t1,Mx1,t2, Mx2,':',t3,Mx3,'--',t4,Mx4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Залежність проекцій МОМЕНТУ КЕРУВАННЯ від часу')
xlabel('Час (c)'),set(gca,'FontName','Helvetica'), ylabel('M_x'),

subplot(2,2,2)
plot(t1,My1,t2,My2,':',t3,My3,'--',t4,My4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('M_y')

subplot(2,2,3)
plot(t1,Mz1,t2,Mz2,':',t3,Mz3,'--',t4,Mz4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
xlabel('Час (c)'), set(gca,'FontName','Helvetica'), ylabel('M_z')

subplot(2,2,4)
plot(t1,d1,t2,d2,':',t3,d3,'--',t4,d4,'.'), grid
set(gca,'FontSize',14,'FontName','MS Sans Serif')
title('Загальні витрати ЕНЕРГІЇ')
xlabel('Час (c)'), set(gca,'FontName','Helvetica')
ylabel('\Sigma\Delta H\Delta\omega')
legend('Kr = k/J','Kr = k','Kr = k/(\alphaJ+\betaE)','Kr = kJ',0)
```

Запуск цього М-файла приводить до результатів, поданих на рис. 4.40-4.42.

На рис. 4.40 наведені проекції траєкторій вектора кватерніона на усі три координатні площини, а також траєкторії у просторі вектора ейлерового повороту.

На рис. 4.41 показані графіки залежностей від часу компонентів кватерниону, а також проекцій вектора ейлерова повороту.

Рис 4.42 подає залежності проекцій моменту керування від часу, а також загальні (сумма по трьох ортогональних осях) прирости кинетичних моментів двигунів-маховиків, які забезпечують виконання цих розворотів КА. Останні характеризують у певній мірі витрати енергії на поворот КА.

Розглядаючи отримані графіки, можна дійти висновку, що найменші витрати енергии забезпечує керування за законом, коли матриця позиційного керування є пропорційною матриці моментів інерції КА. Цей закон дозволяє зекономити больш ніж у два рази у порівнянні з випадком, коли матриця коефіцієнтів позиційного керування є обернено пропорційною матриці моментів інерції.

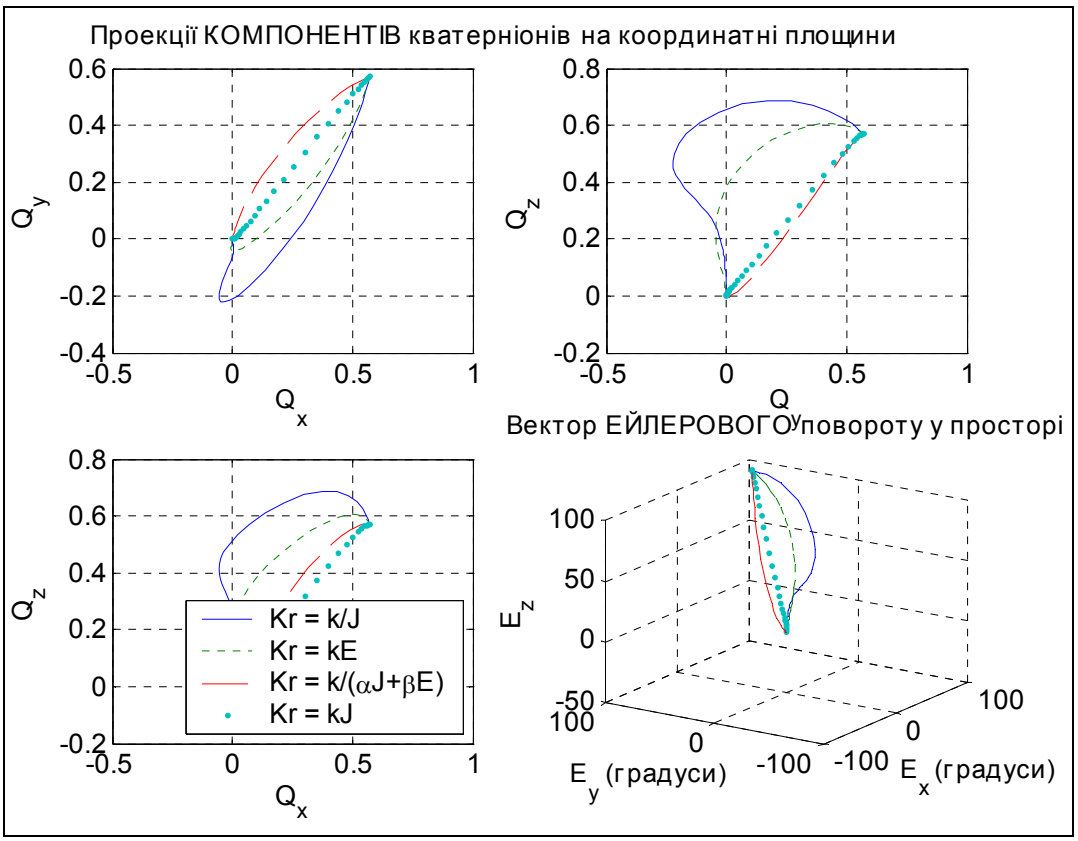


Рис. 4.40. Проекції кватерніону повороту КА на координатні площини

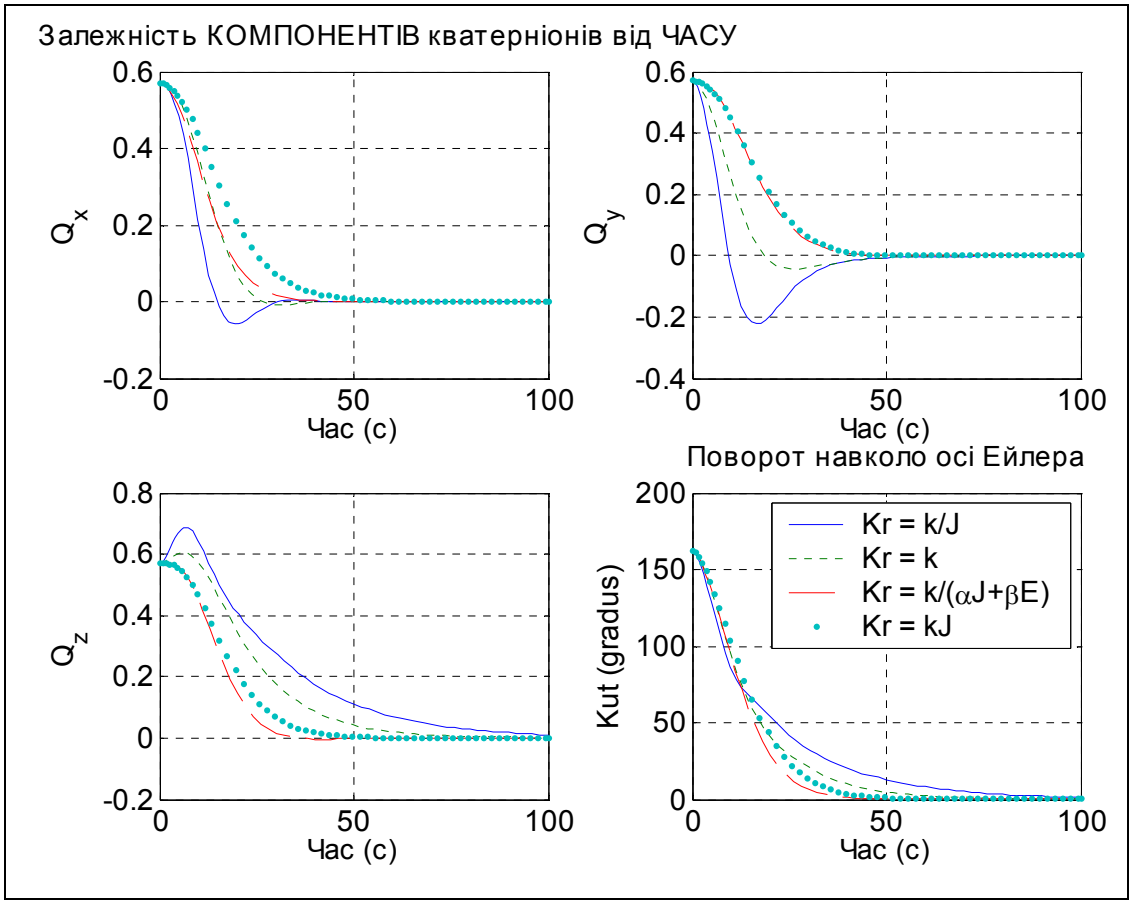


Рис. 4.41. Залежності компонентів кватерниона повороту КА від часу

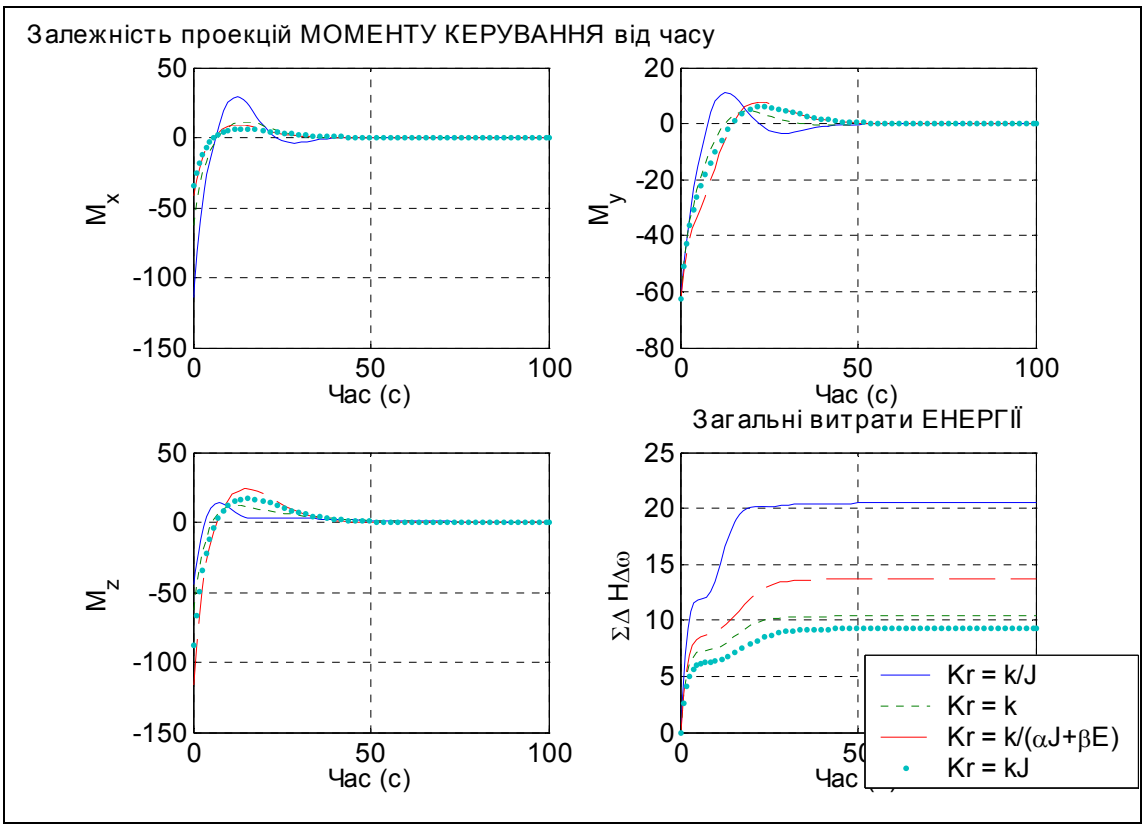


Рис. 4.42. Залежності компонентів моменту керування орієнтацією КА від часу

До того самого висновку можна дійти й сугубо теоретичним шляхом, якщо підставити вираз (8.7) моменту керування у рівняння (8.1) руху КА з врахуванням останньої залежності матриці *Kr* від матриці *J*. Якщо припустити також, що й матриця демпфірування *Dr* є також пропорційною матриці *J* с коефіцієнтом пропорційності *f*, то неважко впевнитися, що векторне рівняння руху у цьому випадку матиме вид:

$$\frac{d\boldsymbol{\omega}}{dt} + f \cdot \boldsymbol{\omega} + k \cdot \mathbf{q} = 0,$$

і воно розпадається на три однакові незалежні рівняння руху КА відносно трьох його координатних осей.

4.3.2. Маятник під дією сил сухого тертя

Як відомо, основні властивості сили сухого тертя, що виникає при відносному переміщенні двох тіл, що труться один по одному, є наступними:

- сила тертя завжди скерована у бік, протилежний відносній швидкості руху тіл;
- величина сили тертя не залежить від величини цієї відносної швидкості.

Зазначені властивості достатньо добре описуються математично, якщо скористуватися сигнум-функцією:

$$F_{tr} = -F_T \cdot sign(V),$$

де F_T - деяка додатна величина, що дорівнює величині сили сухого тертя, а V - швидкість тіла, з боку якого діє сила тертя, відносно тіла, на яке ця сила діє.

Однак нам відома ще одна властивість сили сухого тертя:

- якщо тіла, що труться, нерухомі одне відносно одного, то прикладання зовнішньої сили до одного з них, не призведе до відносного руху тіл доти, поки діюча сила (назвемо її "активною" - F_a) не перебільшить за величиною так звану силу тертя спокою $F_p > F_T > 0$.

У цьому випадку величина сили тертя вже визначається не величиною і напрямком швидкості, а величиною і напрямком прикладеної активної сили, приймаючи таке значення і напрямок, що вона повністю компенсує дію цієї сили:

$$F_a + F_{tr} = 0, \text{ якщо } V = 0 \text{ і } |F_a| \le F_p.$$

Ця особливість сил сухого тертя зумовлює цілу низку дивних властивостей систем, в яких діють подібні сили. Це, зокрема, таке явище, як "захоплювання" або "зчеплення" одного тіла з другим, коли обидва тіла починають рухатися як одне, залишаючись нерухомими одне відносно одного.

Теоретичне досліджування цієї властивості сил сухого тертя пов'язане зі значними труднощами, оскільки залежність сили тертя від швидкості має розривний характер, а також існує складна залежність сили чсухого тертя від швидкості і активної сили.

Сформулюємо задачу опису руху механічної системи, яка знаходиться під дією сил сухого тертя. Нехай q - узагальнена координата (це може бути лінійне переміщення або кут при обертальному русі), яка відповідає відносному переміщенню тіл, що труться. Складемо узагальнене рівняння руху з цієї координати і виділимо у ньому три частини.

Узагальнена сила інерції. До цієї частини віднесемо лище члени рівняння, які є пропорційними відносному узагальненому прискоренню. Цю частину можна подати у наступному виді: $(-M_q\ddot{q})$, де M_q має сенс узагальненої маси і може залежати від узагальненої координати q.

Узагальнена сила тертя. Віднесемо до цієї частини усі члени рівняння, які визначають вплив сил сухого тертя: $Q_{tr}(\dot{q}, Q_a)$.

Активна узагальнена сила Q_a. До цієї сили віноситимемо усі решту членів рівняння.

Тоді рівняння руху з цієї координати можна подати у наступному виді:

$$\ddot{q} = \frac{1}{M_q}\left[Q_a + Q_{tr}(\dot{q}, Q_a)\right]. \tag{4.8}$$

Лише після цієї операції можна сформувати математичну залежність узагальненої сили сухого тертя від усіх чинників, що впливають на неї у відповідностями з встановленими властивостями тертя:

$$Q_{tr}(\dot{q}, Q_a) = \begin{cases} -Q_{Td}, & якщо \quad \dot{q} > 0 \\ Q_{Td}, & якщо \quad \dot{q} < 0 \\ -Q_a, & якщо \quad \dot{q} = 0 \quad і \quad |Q_a| < Q_{Tp} \\ Q_{Tp}, & якщо \quad \dot{q} = 0 \quad і \quad |Q_a| \ge Q_{Tp} \end{cases} \tag{4.9}$$

де постійні додатні величини Q_{Td} і Q_{Tp} визначають величини узагальнених сил тертя руху і спокою відповідно. При цьому зазвичай виконується співвідношення $Q_{Tp} \geq Q_{Td}$.

Як бачимо, повністю описати усі вказані особливості сил сухого тертя можна лише після того, як задані (відомі) рівняння руху і виділена так звана активна сила.

Створимо універсальний блок, що здійснює однократне інтегрування рівняння (4.8). Входом цього блоку має бути поточне значення активної сили, а виходом – поточне значення узагальненої швидкості $\dot{q}$. Параметри блоку приймемо наступні:

- *Mq* - узагальнена маса M_q;
- *TrDvig* - величина узагальненої сили тертя руху Q_{Td};
- *TrPoc* - величина узагальненої сили тертя спокою Q_{Tp};
- *qt*0 - початкове значення узагальненої швидкості $\dot{q}_0$.

Оформимо блок у виді підсистеми (рис. 4. 43) і назвемо його ***Suhoe Trenye***.

Головний елемент блоку – интегратор (***Integrator***). Він здійснює інтегрування сигнала відносного прискорення, що надходить до нього, видаючи сигнал, що дорівнює поточному значенню узагальненої швидкості. Початкове значення узагальненої швидкості задається зовнішнім блоком ***IC***, до якого надається постійний сигнал з блоку ***Constant***, що дорівнює встановленому в ньому початковому значенню узагальненої швидкості.

Сигнал узагальненого прискорення формірується у такий спосіб. Спочатку на суматорі (блок Sum0 активна сила сумується з силою тертя. Результат подається на блок Gain, який здійснює ділення сумарного сигнала на узагальнену масу. Блоки, що формірують силу тертя, розташовані у нижній частині блок-схеми. Якщо швидкість $\dot{q}$ не дорівнює нулеві, то перемикач у блоці Kluch знаходиться у нижньому положенні, і сигнал узагальненої швидкості проходить через блок Sign (нижня права частини схеми), помножується на постійний коефіцієнт *TrDvig* і передається до суматора як сила тертя з протилежним знаком. У такий спосіб реалізуються перші два співвідношення (4.9)

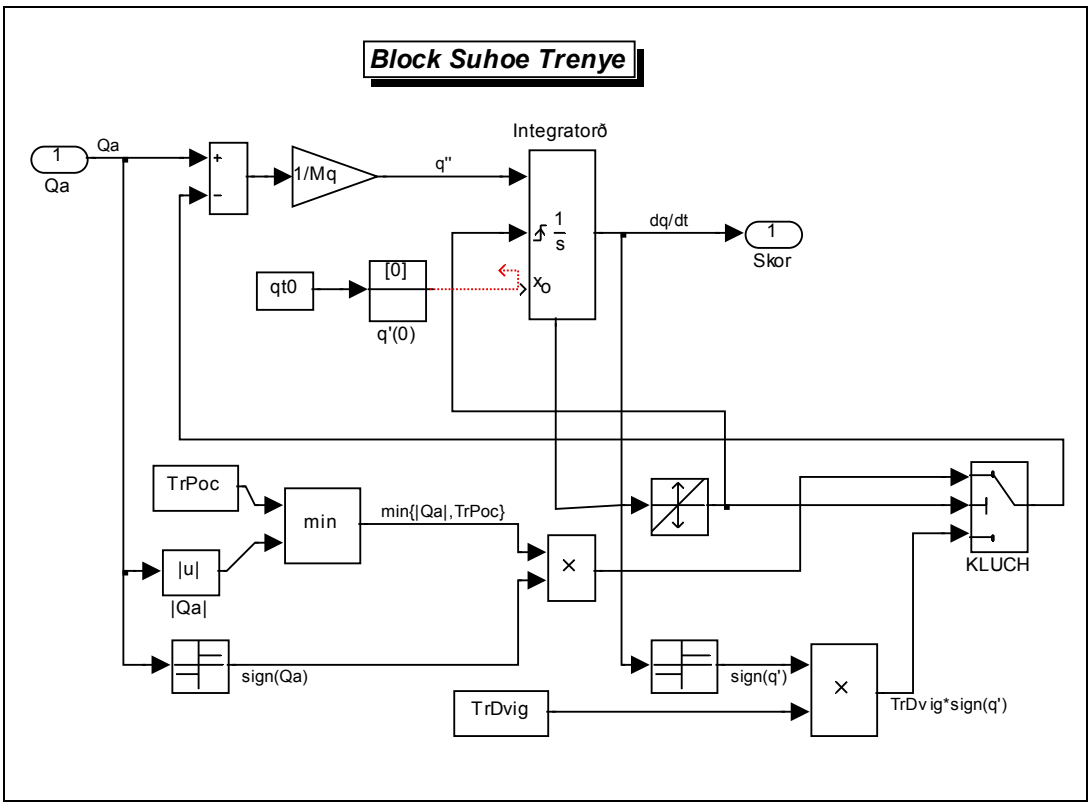


Рис. 4.43. Блок-схема блоку Suhoe Trenye

Значно складніше виконати дві останні умови (8.9). Для цього перш за все потрібно якнайточніше визначити момент часу,коли відносна швидкість проходить через нуль. Необхідно зробити наступне.

1. У блоці інтегратора потрібно відкрити порт стану *Show state port*. При цьому на зображенні блоку внизу виникне додатковий вихід – порт стану. Окрім того, потрібно підключити зовнішнє керування роботою інтегратора, встановивши для параметра *External reset* значення *rising* (рис. 4.44). З лівого боку блоку виникне зображення ще одного вхідного порту (керуючого).
2. Порт стану інтегратора потрібно з'єднати зі входом блоку ***Hit Crossing***, який здійснює фіксування точного моменту переходу швидкості через нуль і видає у цей момент часу керуючий одиничний сигнал.
3. Вихід блоку ***Hit Crossing*** слід з'єднати зі керуючим входомблоку ***Swith*** (другий вхід). При цьому у якості порогу (параметр *Threshold*) цього блоку потрібно встановити значення 0,5. Окрім того, вихід блоку ***Hit Crossing*** необхідно з'єднати з портом керування блоку ***Integrator***.

Сукупність описаних блоків працює у такий спосіб. Якщо значення швидкості не проходить через нуль, вихідний сигнал блоку ***Hit Crossing*** дорівнює нулеві. Він менше за поріг блоку ***Swith*** (0,5). Тому перемикач з'єднає з виходом третій (нижній) вхід, і на суматор надається сила тертя руху. Як тільки блок ***Hit Crossing*** зареєструє перетинання швидкістю нуля, на його виході сигнал стає рівним одиниці, він стає більшим за поріг блоку ***Swith***, який у цьому

випадку перемикає на суматор гілку блок схеми, що формує тертя спокою (ліва нижня частина блок-схеми). Одночасно сигнал блоку ***Hit Crossing*** надається до керуючого входу блоку ***Integrator***. За ним інтегратор починає інтегрування заново з моменту перетинання швидкістю нуля з початковою умовою, встановленим у блоці ***IC*** (у нашому випадку $\dot{q}_0 = 0$). Якщо при подальшому інтегруванні значення $\dot{q}$ залишається рівним нулю, то стан системи залишається незмінним. Якщо ж величина $\dot{q}$ на деякому кроці набуде значення, відмінного від нуля, блок ***Hit Crossing*** скине значення свого вихідного сигнала до нуля, перемикач перекине "рубильник" у нижне положення, і знову "запрцюють" сили тертя руху

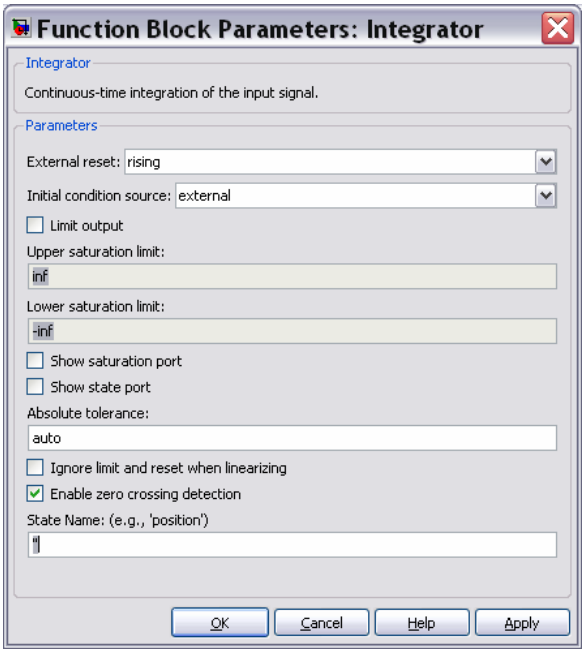


Рис. 4.44. Вікно настроювання блоку Integrator

.

Гілка, що формірує сили тертя спокою, здійснює наступні функції. Спочатку визначається модуль активної силию подім він порівнюється зі значенням сили тертя спокою. Визначається менша з цих двох додатних величин. Потім їй присвоюється знак активної сили (блоки ***Sign*** і ***Product***). Отримана величина й складає силу тертя спокою, вона спрямовується на перший вхід перемикача.

Розглянемо тепер задачу дослідження руху фізичного маятника, на який діє момент сил сухого тертя в опорах його осі обертання.

Позначимо через α кут повороту маятника відносно осови. Тоді рівняння руху (обертання навколо його осі) маятника можна записати так:

$$\varphi'' + \sin\varphi = \mu_{tr}(\alpha'), \qquad (4.10)$$

де, як і раніше, φ - кут відхилення маятника від вертикалі. Величина $\mu_{tr}(\alpha')$ являє собою безрозмірний момент сил тертя, тобто відношення моменту сил тертя до опорного маятникового моменту маятника mgl.

Позначимо кут повороту основи навколо осі обертання маятника через ϑ. Тоді три кути ϑ, α і φ будуть пов'язані один з одним співвідношенням

$$\alpha = \varphi - \vartheta . \tag{4.11}$$

Запишемо рівняння (4.10) з врахуванням цього у виді:

$$\alpha'' = -\vartheta'' - \sin(\alpha + \vartheta) + \mu_{tr}(\alpha') . \tag{4.12}$$

Координата α характеризує відносне (кутове) переміщення маятника і основи, які труться один по одній. Тому у розглядуваному випадку можна вважати, що

$$q = \alpha ; \qquad M_q = 1 ; \quad Q_a = -\vartheta'' - \sin(\alpha + \vartheta) = -\vartheta'' - \sin\varphi .$$

Блок-схема S-моделі ***FM_Suh_Tr***, яка реалізує інтегрування рівняння (4.12), наведена на рис. 4.45.

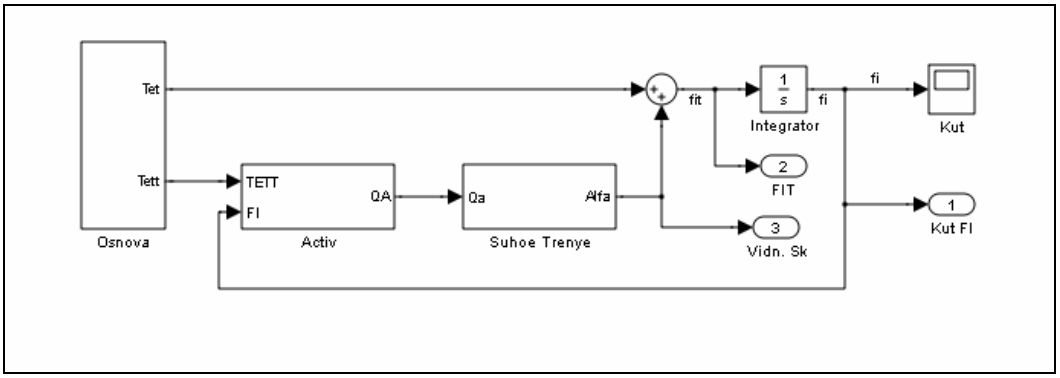


Рис. 4.45. Блок-схема S-моделі FM_Suh_Tr

Блок ***Osnova*** у цій моделі (рис. 4.46) формує сигнали кутової швидкості (Tet) і кутового прискорення (Tett) обертання основи.

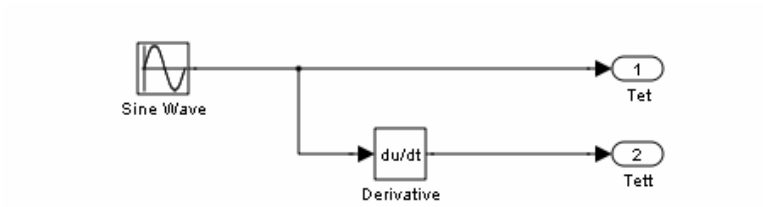


Рис. 4.46. Блок-схема підсистеми Osnova

Як бачимо, кутова швидкість основи формується за законом

$$\vartheta'(\tau) = \vartheta'_0 + \vartheta'_m \sin(\omega_\theta \tau + \varepsilon_\vartheta) .$$

Значення констант, які використовуються, вводяться у вікні настроювання блоку Sine Wave (рис. 4.48):

Tet0 – стала складова кутової швидкості основи ϑ'_0;

Tetm – амплітуда кутової швидкості ϑ'_m;

omt – частота змінювання кутової швидкості ω_θ;

et – початкова фаза кутової швидкості ε_ϑ.

Блок-схема третьої підсистеми Activ, яка формує сигнал активної сили, є вельми простою (рис. 4. 48).

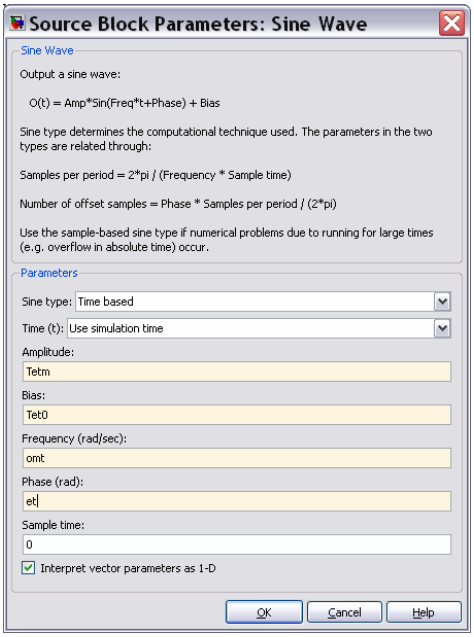


Рис. 4.47. Вікно настроювання блоку Sine Wave

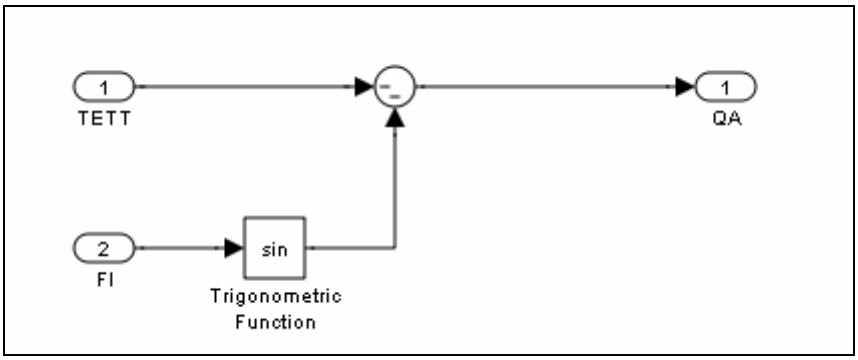


Рис. 4.48. Блок-схема підсистеми Activ

Як і раніше, створимо керуючу М-програму FM_Suh_Tr_upr. m, яка виконуватиме наступні функції:

- введення значень усіх параметрів, що визначають рух системи;
- запуск S-моделі FM_Suh_Tr. mdl до моделювання;
- виведення результатів моделювання у графічне вікно Matlab.

Нижче наведений текст програми.

```
% FM_Suh_Tr_upr
% Керуюча програма  для запуска моделі FM_Suh_Tr.mdl

% Лазарєв Ю.Ф. 4-08-2009
clear all, clc
      % 1. Задання маси і характеристик тертя
Mq=1;   TrPoc=0.2;  TrDvig=0.01;
      % 2. Задання параметрів обертання основи
      %    Teta' = Tet0+Tetm*sin(omt*t+et)
Tetm=0; Tet0=0; omt=0;      et=0;
      % 3. Задання початкових умов
fi0=30*pi/180;  fit0=0;
      % 4. Розрахунок початкової відносної швидкості
qt0=fit0-Tet0-Tetm*sin(et);
      % 5. Запуск моделі на моделювання
```

```
sim('FM_Suh_Tr');      % МОДЕЛЮВАННЯ на S-моделі
      % 6 Формування вихідних масивів
Fl=yout(:,1)*180/pi;   Flt=yout(:,2);   ALt=yout(:,3);  t=tout;
      % 7. Виведення графіків
subplot(2,2,1), plot(Fl,Flt,'.',Fl,ALt), grid, set(gca,'fontsize',12)
xlabel('Кут (градуси)'), ylabel('Кутова швидкість (б/р)')
legend('віднocн.','абсолютн.',0)
set(gca,'fontsize',14), title('Фазовий портрет')
subplot(2,2,[3 4]), plot(t,Fl), grid, set(gca,'fontsize',12)
xlabel('Час (б/р)'),    ylabel('Кут (градуси)')
set(gca,'fontsize',14), title('Кут відхилення від вертикалі')
subplot(2,2,2), axis('off')
h=text(0,1,'Маятник з сухим тертям','fontsize',16);
h=text(-0.2,0.8,'Обертання основи:    Teta''(t)=Tet0+Tetm*sin(omt*t+et)','fontsize',12);
h=text(-0.2,0.7,['де:      ',...
      sprintf('Tet0 = %g; ',Tet0),sprintf('Tetm = %g;',Tetm),...
      sprintf('omt = %g;   ',omt),...
      sprintf('et =% g градусів',et*180/pi)]);
h=text(-0.2,0.5,'Характеристики тертя','fontsize',12);
h=text(-0.1,0.4,[sprintf('Тертя спокою = %g;    ',TrPoc),...
      sprintf('Тертя руху = %g;   ',TrDvig)]);
h=text(-0.2,0.2,sprintf('Початкова абс. кут. швидк. = %g;',fit0),...
        'fontsize',12);
h=text(-0.2,0.0,'-----------------------------------------');
h=text(-0.2,-0.1,'Програма FM-Suh-Tr-upr   4-08-2008  Лазарев Ю. Ф.');
h=text(-0.2,-0.2,'-----------------------------------------');
```

Результат виконання цієї програми при нерухомій основі поданий на рис. 4.49.

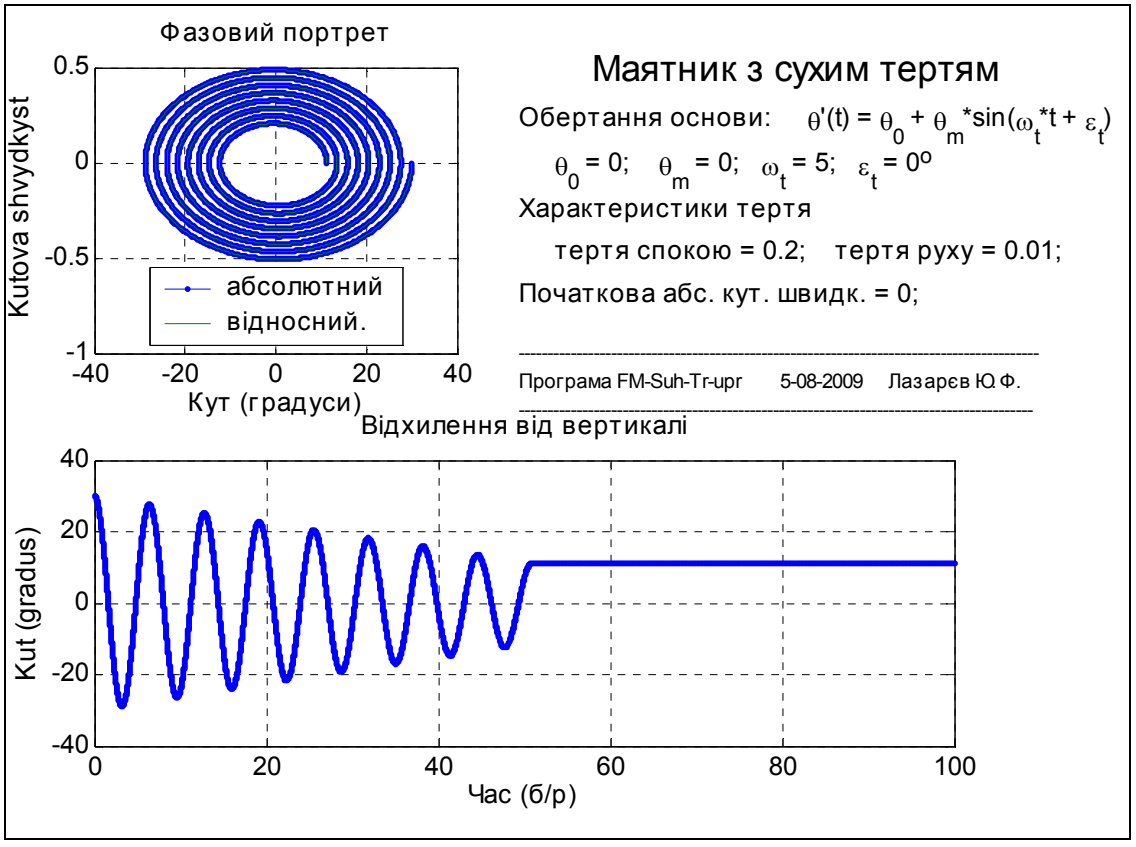


Рис. 4.49. Вільні коливання маятника під дією сил сухого тертя

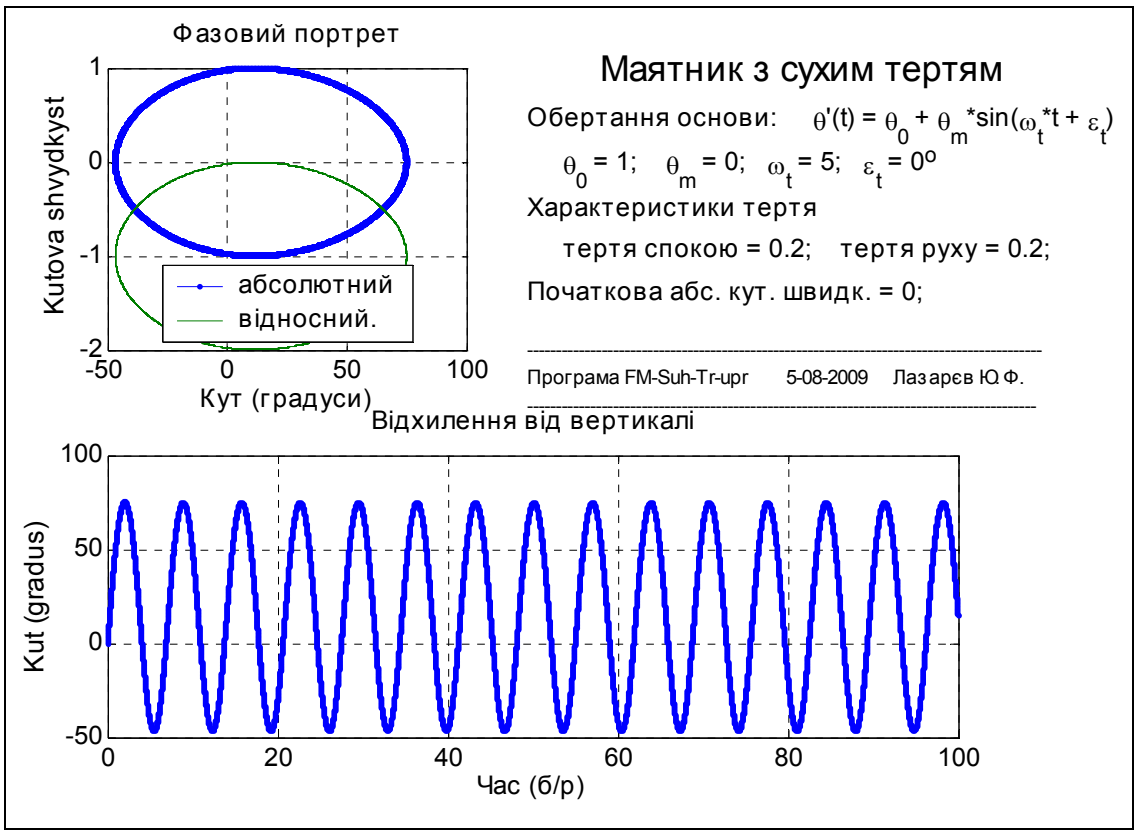


Рис. 4.50. Маятник Фроуда при великій кутовій швидкості основи

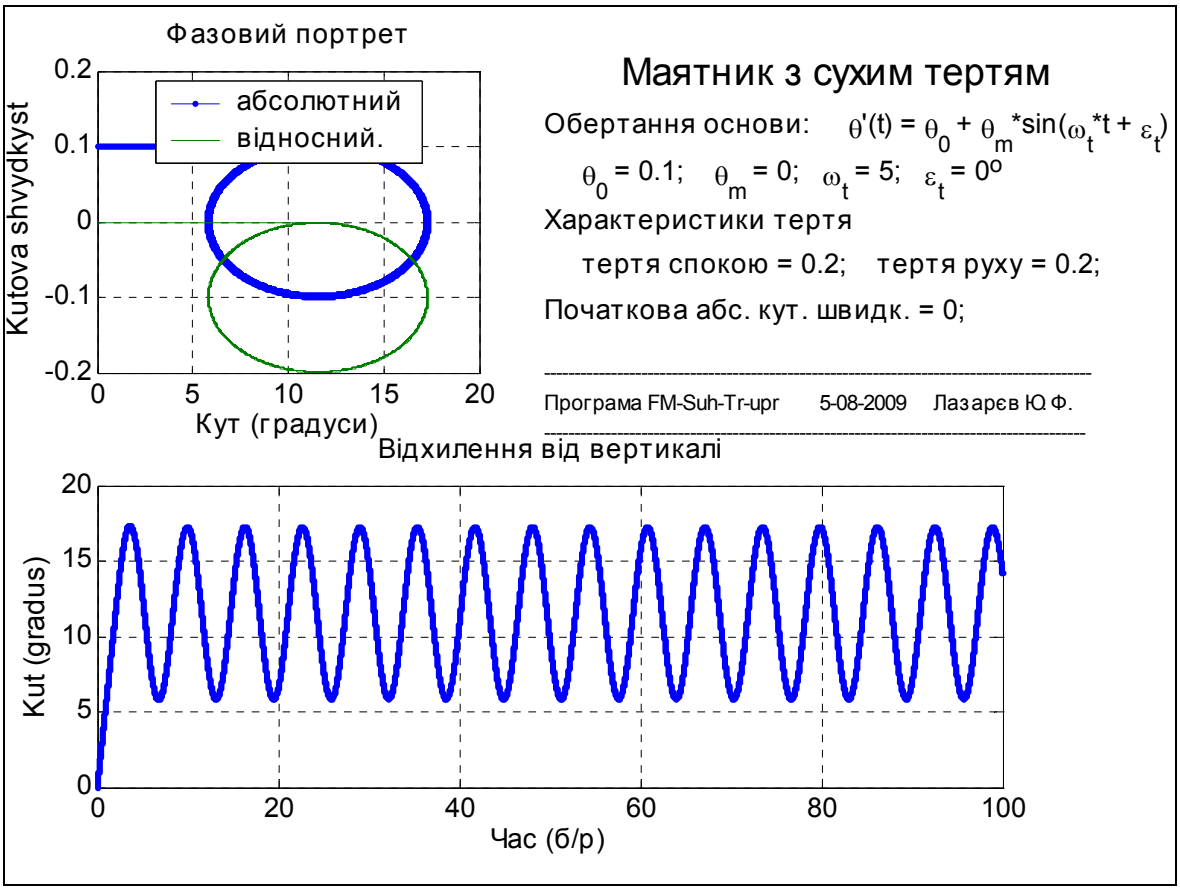


Рис. 4.51. Маятник Фроуда при малій кутовій швидкості основи

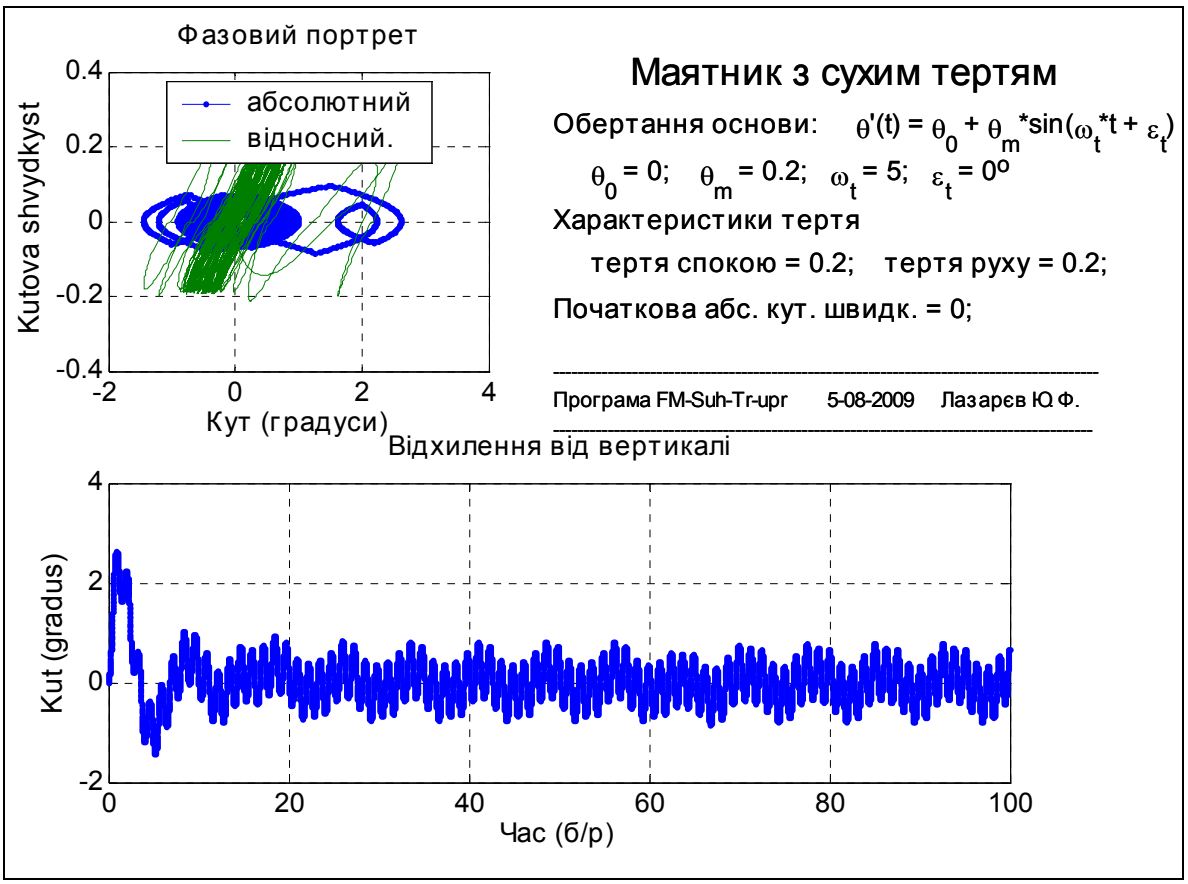


Рис. 4.52. Маятник при коливаннях основи з великою амплітудою

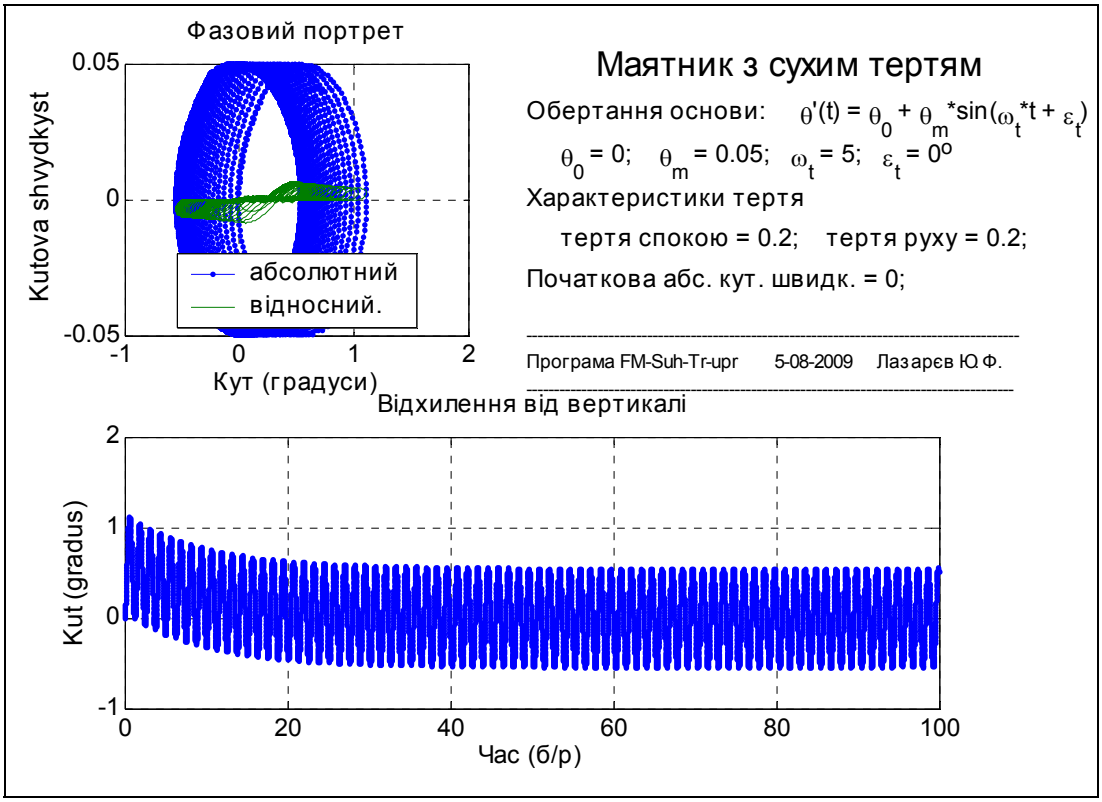


Рис. 4.53. Маятник при коливаннях основи з малою амплітудою

Прослідковуються три основні нелінійні властивості маятника:

- обвідна вільних коливань являє собою пряму лінію;

- коливання загасають за кінцевий час;
- маятник зупиняється у зміщеному відносно вертикалі положенні.

Розглянемо поводження маятника при рівномірному обертанні основи навколо осі маятника (такий маятник називають маятником Фроуда).

При обертанні основи з постійною кутовою швидкістю під дією сил сухого тертя маятник здійснює коливання, зміщеного відносно вертикалі на $11^{o},5$ у бік обертання основи. Від величини кутової швидкості обертання основи залежить лише амплітуда цих коливань (рис. 4.50, 4.51).

Вплив коливань основи навколо осі маятника з різною амплітудою показаний на рис. 4.52, 4. 53. Помітна цікава особливість. Амплітуда вимушених коливань маятника практично не залежить від амплітуди коливань основи. Окрім того, за значних амплітуд коливань основи на вимушені коливання накладаються незагасаючи власні коливання маятника.

4.4. Контрольні запитання

1. Як всередині блоків позначаються вхідні величини, вихідні величини блоку і його змінні стану?
2. Що таке виявлення перетинання нуля, для чого ця процедура прислуговується і якими блоками використовується?
3. Якими засобами забезпечується передавання даних з середовища Matlab у S-модель і навпаки?
4. У чому полягає головна перевага блоку S-function у порівнянні з усіма іншими блоками бібліотеки Simulink, що дозволяють користувачеві створювати власні блоки?
5. Чи модливо забезпечити одночасне інтегрування кількох процесів одним блоком ***Integrator***?
6. Що таке S-функції, для чого вони призначені і як їх створити?
7. Як забезпечити запуск S-моделі з програми Matlab?
8. Як забезпечити запуск програми Matlab з S-моделі?
9. Як створити вікно настроювання блоку?
10. Як створити власну бібліотеку S-блоків?

4.5. Література

1. Гультяев А. К. MatLAB 5.2. Имитационное моделирование в среде Win-dows: Практич. пособие. - СПб. : КОРОНА принт, 1999. - 288 с.
2. Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс. - СПб. : "Питер", 2000. - 430 с.
3. Дьяконов В.П. Справочник по применению системы PC MatLAB. - М.: Физматлит, 1993. - 113с.
4. Дьяконов В. П., Абраменкова И. В. MATLAB 5.0/5.3. Система символьной математики. - М.: Нолидж, 1999. - 640с

5. Краснопрошина А. А., Репникова Н. Б., Ильченко А. А. Современный анализ систем управления с применением MATLAB, Simulink, Control System: Учебное пособие. - К.: "Корнійчук", 1999. - 144 с.
6. Лазарєв Ю.Ф. Початки програмування у середовищі MatLAB: Навч. посібник. - К.: "Корнійчук", 1999. - 160 с.
7. Лазарев Ю. Ф. MatLAB 5.x. - К.: "Ирина" (BHV), 2000. - 384 с.
8. Мартынов Г. Г., Иванов А. П. MATLAB 5.x, вычисления, визуализация, программирование. - М.: "Кудиц-образ", 2000. - 332 с.
9. Медведев В. С., Потемкин В. Г. Control System Toolbox. MatLAB 5 для студентов. - М.: ДИАЛОГ-МИФИ, 1999. - 287 с.
10. Потемкин В. Г. Система MatLAB: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1997. - 350 с.
11. Потемкин В. Г. MatLAB 5 для студентов: Справ. пособие. - М.: ДИАЛОГ-МИФИ, 1998. - 314 с.
12. Потемкин В. Г., Рудаков П. И. MatLAB 5 для студентов. - 2-е изд., испр. и дополн. - М.: ДИАЛОГ-МИФИ, 1999. - 448 с.
13. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т.Том 1. - М.: ДИАЛОГ-МИФИ, 1999. - 366 с.
14. Потемкин В. Г. Система инженерных и научных расчетов MatLAB 5.x: - В 2-х т. Том 2. - М.: ДИАЛОГ-МИФИ, 1999. - 304 с.
15. Рудаков П. И., Сафонов В. И. Обработка сигналов и изображений. MATLAB 5x. - М.: "ДИАЛОГ-МИФИ", 2000. - 413 с.