**Практична робота №1**

**Застосування системного підходу для рішення математичних задач**

**Мета практичної роботи:** Ознайомитись з поняттям системного підходу для аналізу та синтезу складних систем. Навчитись застосовувати методологію системного підходу до створення складних систем. Вивчити логіку побудови структурних схем в середовищі моделювання Simulink Matlab та побудувати модель для рішення системи лінійних рівнянь.

**Інформаційна частина**

В даній роботі наведено основні принципи та методологія системного підходу для аналізу та побудови складних систем. Опис цілей, в’язків між локальними цілями та побудова структури дій для досягнення цілей.

**Ознаки складних систем.**

Часто розділяють поняття «великі» та «складні» системи. Отже введемо деяке пояснення цих визначень. Складність полягає в тому, що чіткої границі між цими термінами немає. Так в теорії систем великою (складною, системою великого масштабу, Large Scale Systems) називають таку, яка складається з великої кількості взаємозв’язаних та взаємодіючих елементів та може виконувати складні функції. Однією з ознак складної системи управління є показник живучості: при відмові певної кількості елементів система продовжує виконувати свої функції, хоча і з меншою ефективністю (проста система може бути лише в двох станах: працездатності та відмови).

*Складна система* – впорядкована множина структурно взаємозв’язаних та функціонально взаємодіючих різнотипних систем, які об’єднані структурно в цілісний об’єкт функціонально різнорідними взаємозв’язками для досягнення заданих цілей в певних умовах.

Велика система об’єднує різнотипні складні системи.

Можна виділити різні ознаки складних систем, але головними є ті, що характеризують як деякі кількісні сторони, наприклад, кількість елементів, так і якісні. Відомий підхід, коли складною системою (СС) називають таку, математичні моделі якої можна описати принаймні двома способами (детерміновані та стохастичні, теоретикой-мовірнісні і т.д.).

**Аналіз ієрархічних структур систем.**

Нехай є завдання створення, вдосконалювання, експлуатації або розширення функцій складної системи. Будемо виходити з того, що процес її рішення може бути розділений на деякі елементарні акти, які назвемо процедурами. Очевидно, що процедури повинні бути певним чином зв'язано один з одним. Зокрема, необхідно визначити їхній порядок, передачу інформації, умови початку й закінчення виконання. Виконання сукупності процедур приводить до того, до чого кожна окрема процедура привести не може, – до рішення поставленого завдання. Таким чином, у наявності всі ознаки системи, у якій у якості окремих елементів виступають процедури. Це означає, що процес рішення завдання може розглядатися як деяка система процедур, що володіє внутрішньою організацією, структурою, ієрархією, керуванням. Така система ставиться до класу цілеспрямованих систем з явно сформульованою метою – рішенням поставленого завдання. У системі процедур модулем буде група процедур, що володіє певною цілісністю й відносною незалежністю. Для такої групи процедур введемо термін «операція».



Рис. 1.Приклад розподілу завдання на операції й процедури

Отже, операція завжди складається з окремих процедур, але, як і модуль, може складатися з операцій більш низького рівня. Така ієрархія (різномасштабність) поняття операції, яка на практиці може досягати трьох чотирьох рівнів, як правило, не викликає незручностей, а, навпаки, дозволяє акцентувати увагу те на єдності, то на подільності розглянутої сукупності. Наведемо приклад. Операція придбання нового обладнання поділяється на операції його замовлення, оплати, перевезення, установки, налагодження. Кожна із цих операцій також розкладається на більш елементарні акти. Скажемо, перевезення буде складатися з упакування, однієї або декількох завантажень, розвантажень і транспортувань. Якщо немає необхідності вникати в ці акти більш глибоко, то для нас вони будуть процедурами. Якщо ж треба для якихось цілей розглядати, із чого полягає завантаження, то вона в нашому дослідженні буде операцією. На цьому прикладі добре видне, що погляд на даний акт як на елементарний при необхідності може замінятися більш деталізованим – як на складальний (операцію).

Для випадку деревоподібної ієрархії в процесі рішення завдання (що далеко не завжди має місце) розподіл на операції й процедури може мати вигляд, зображений на рис. 1.

Наведемо практичний приклад ієрархічного розподілу завдання на операції й процедури.

Надалі для зручності подання, коли не буде необхідності відрізняти один від одного процедуру й операцію, будемо використовувати для них єдиний термін «дія». Процес рішення завдання при цьому буде являти собою систему дій.

Відповідно до загального виду системи (1) організація процесу рішення завдання формально може бути записана як:

Σ:{{*M*}, {*x*}, *F*}, (1)

де Σ – система; {*M*} – сукупність елементів в системі; {*x*} – безліч зв'язків; *F* – функція (нова властивість) системи.

*R* :{{*M*}, {*x*}, *F*}, (1.1.2)

де {*M*} – безліч дій за рішенням завдання; {*x*} – безліч зв'язків між діями; *F* – формулювання поставленого завдання (ціль).

Кортеж (1.1.2) має всі особливості запису довільної системи. Він також умовний у тому розумінні, що спосіб опису мети, дій і зв'язків, їх приналежність певним класам повинні бути конкретизовані окремо. У самому загальному виді зі складовими кортежами {*M*} і {*x*} можливі лише ті операції, які припустимі з безлічами довільної природи, наприклад: доповнення, поділ, перетинання й ін.

Також звернемо увагу на те, що елемент *М* у (1.1.2) визначений як дія, а не як найпростіший акт рішення – процедура. Це зроблене для більшої варіативності формального запису. Так, якщо {*M*} – це операції верхнього ієрархічного рівня, те (1.1.2) являє собою цілком доступну для огляду, хоча й грубу схему рішення завдання. Саме такі схеми найчастіше будуть фігурувати в якості прикладів у даній книзі. Якщо ж {*M*} – це всі процедури в рішенні, то для досить складного завдання розшифрування всіх елементів *М* і *х* може бути досить об'ємним. Таке розшифрування, наприклад, потрібно при передачі (тиражуванні) способу рішення якого-небудь завдання. Документація на стандартизований опис навіть середнього по складності програмного засобу може займати до сотні сторінок тексту й позначень. Документація ж, пов'язана з описом усіх процедур по будівництву літака або ракети, досягає у ваговім вираженні десятків тонн паперу.

У складній системі, крім уведених понять процедури, операції, дії, вживаються й інші терміни. Так, великий комплекс дій, що приводить до виконання в певному змісті відособленої важливої частини всього рішення завдання, називають «напрямком робіт». Для розподілу дій у часі вживають терміни «стадія» і «етап». Однак основним системним поняттям у даній главі залишиться операція. Розкладання процесу рішення тільки на напрямки робіт або стадії й етапи практично завжди є ще недостатнє поглиблення в суть завдання.

**Зв’язки між локальними цілями та способи їх досягнення.**

При аналізі СС, з використанням методології системного підходу завжди виконують оцінку зв’язків між елементами та цілями, які ними досягаються. Цілі, які досягають елементи системи поділяються на локальні то глобальні. *Локальними* – називаються цілі, які досягаються окремими елементами, чи групами елементів системи. *Глобальними* – називається цілі, які досягаються сукупністю всіх елементів системи, тобто всією системою. В загальному випадку структура зв'язків між локальними цілями має довільний характер. Як крайні ситуації назвемо:

1. випадок, коли виконання будь-якої мети пов'язане з виконанням кожної з наступних (при відсутності ієрархії);
2. випадок повної незалежності досягнення локальних цілей: кожна мета виконується самостійно і їх зв'язок один з одним проявляється лише в тому, що виконані всі разом вони вирішують поставлене завдання.

Однак найпростіших і одночасно основних типів зв'язків між цілями всього три. Зручніше за все пояснити їх на прикладі розподілу мети операції на дві локальні цілі:

{*g*1, *g* 2} →*GJ*,

(індекс *J* у локальних цілей для простоти пропущений). При цьому можливо:

1. *послідовне виконання* – тільки досягнення однієї із цілей дає можливість виконати іншу;
2. *паралельне виконання* – цілі можуть виконуватися незалежно;
3. *циклічне виконання* – часткове виконання однієї із цілей дозволяє частково виконати іншу, що, в свою чергу, дозволяє повернутися до виконання першої, і так до повного виконання обох цілей (рис. 4.).

Цілі на рис. 4. називаються також зв'язаними (*а*), незв'язаними (*б)* і складно зв'язаними (*в)*. Останнє з погляду кібернетики являє приклад системи із зворотним зв'язком.



Рис. 4. Способи досягнення двох цілей

Типовим прикладом циклічного способу (рис. 4., *в)* є організація циклу програмним способом. У цьому випадку перед ЕОМ ставляться дві локальні цілі: перебрати всі параметри циклу (ціль *g*1) і виконати для кожного значення параметра певні дії (*g*2). Циклічне виконання цілей досить багатообразне і за межами області програмування. За цією схемою представимо, наприклад, будь-яке керування, що вимагає постійного виконання команд: ціль *g*1 – визначення керуючого впливу, ціль *g*2 – виконання цього впливу. Будівництво можна розглядати як цикл: завезення матеріалу та механізмів і власне будівельні роботи. Процес навчання для студента переслідує дві циклічні цілі – засвоєння знань і здачу заліків.

Спосіб досягнення кожної із цілей *g*1 і *g*2 окремо може бути дискретним (порціями, стрибками) і безперервним. У першому випадку схему рис. 4, *в* ще називають ітеративною, а кожний перехід від мети *g*1 до *g*2 і назад – ітерацією, кроком, циклом.

Для більш ніж двох локальних цілей зв'язок між ними буде комбінуванням наведених вище типів. Схематичні приклади деяких з них для випадку трьох локальних цілей зображені на рис. 5.



Рис. 5. Деякі способи досягнення трьох цілей

Нерідке виконання однієї локальної мети може ускладнити, навіть виключити виконання іншої. Такі цілі (дві й більш) називають антагоністичними. У складних системах практично не вдається позбутися того чи іншого ступеня антагоністичності локальних цілей. Проблема є найбільш гострою для цілей того самого ієрархічного рівня. В цьому випадку завдання прийнято називати багатоцільовим або багатокритеріальним. Цей вид завдань є досить актуальним і на даний момент активно вивчається. Звичайно це робиться в строго формалізованій постановці, що переводить центр ваги дослідження на математичний апарат.

Відштовхуючись від типових порядків виконання локальних цілей, проаналізуємо найпоширеніші на практиці схеми досягнення глобальної цілі.

***Типова схема 1*.** Загальна характеристика: у завданні явно присутні стадійність, етапи, почерговість операцій або процедур.

***Приклади*:** послідовність операцій по обробці деталі на верстаті; етапи виготовлення проектної документації; вирощування рослин; почерговість вивчення розділів підручника, тощо.

У схемі 1 переважає послідовний перехід від однієї локальної цілі до іншої. Стадійність рішення звичайно забезпечує простоту умов переходу від однієї мети до іншої, таким чином, полегшуючи їхнє узгодження.

***Типова схема 2*.** Загальна характеристика: завдання має щільно зв'язані, і одночасно значно відрізняючи сторони та аспекти, які повинні розглядатися одночасно.

***Приклади*:** проектування складного технічного об'єкта, що вимагає участі різних дослідницьких колективів і застосування більших наукових, інженерних і організаційних зусиль; завдання наземного супроводу космічного польоту; конвеєрне складання, у якому одночасно бере участь висококваліфікований персонал різних спеціальностей; завдання екологічного захисту; завдання керування суспільством. Основою рішення таких завдань є паралельне досягнення набору локальних цілей в умовах їх тісного зв'язку і антагоністичності. Виділення цілей тут, як правило, досить очевидно за функціональною ознакою. Але узгодження є серйозною, часто важко вирішувальною проблемою. Вдалий (тобто добре узгоджений) вибір локальних цілей у завданні, скажемо, створення складної автоматизованої системи може бути значним досягненням, що мають характер винаходу або навіть відкриття.

***Типова схема 3*.** Загальна характеристика: більш однорідне завдання, підлягає розподілу на частини в зв'язку з її громістким, значним обсягом вхідної інформації або обмежену в часі для рішення.

*Приклади*: розподіл термінового замовлення по заводах (в основі рішення задачі – паралельне виконання слабо пов’язаних цілей); створення розгалуженої інформаційної системи на основі мережі ЕОМ (в основі – паралельне виконання щільно пов’язаних між собою цілей); пошук інформації в банку даних (в основі – послідовний перехід вниз по деревовидній системі ознак); математична декомпозиція лінійної системи рівнянь великої розмірності (в основі – циклічне рішення систем меншої розмірності). Із прикладів видно, що локальні цілі тут зв'язуються з виділенням досить однорідних частин. У випадку виконання цілей одним колективом (однією людиною, однієї ЕОМ) приймає послідовне або послідовно-циклічне досягнення цілей (типу рис. 5, *б)*. Якщо цілі виконуються різними колективами (людьми, машинами), то переважає паралельне або паралельно-циклічне досягнення цілі (типу рис. 5, *в)*.

У цій типовій схемі складним може бути як виділення, так і узгодження цілей.

**Застосування методології системного підходу до створення складних систем.**

***Методологія*** – сукупність прийомів дослідження в науці.

Системний підхід при створенні складних структур управління проявляється в таких підходах:

1. Будь-яка система на першому етапі розглядається з урахуванням лише формальних зв’язків між різними факторами та оцінки характеру їх зміни під впливом зовнішніх умов. Теорія систем своїм об’єктом досліджень має не фізичну реальність, а загальні властивості та зв’язки між різними факторами. При зростанні складності СУ для їх аналізу та синтезу необхідно застосовувати знання з різних наук.
2. Система завжди досліджується в умовах невизначеності (мети, характеристик зовнішнього середовища та поведінки оператора). Важливо забезпечити в системі адаптацію та можливість розвитку.
3. Складність систем управління, їх інформаційна потужність вимагає залучення деяких спеціальних прийомів, наприклад, декомпозиції та агрегування.
4. У складних системах управління (ССУ) завжди використовуються структурні перетворення.
5. В загальній теорії систем повинні використовуватись терміни, методи, поняття та прийоми, які є зрозумілими для інших наукових дисциплін (інформатика та автоматика, вони повинні спілкуватись спільною мовою).
6. В теорії систем застосовуються уніфіковані поняття, які дають можливість охарактеризувати як систему будь-якої складності, так і будь-яку її частину.

**Практична частина**

Знаючи матеріал з таких дисциплін, як фізика, математика, теорія автоматичного управління та ідентифікація систем автоматизації, можна стверджувати, що існує багато процесів та комплексів, які можна описати лінійними рівняннями з декількома змінними. А параметри обмеження процесу можна описати з використанням нерівностей. Тому розглянемо систему, задану у вигляді лінійних нерівностей з двома змінними.

В процесі виконання практичної роботи у середовищі Simulink потрібно побудувати модель знаходження двох невідомих змінних заданої системи нерівностей.

Для побудови моделі потрібно:

1. Привести нерівності до виду алгебраїчних рівнянь.
2. Визначити послідовність дій для знаходження невідомих значень.
3. Скориставшись блоками бібліотеки математичних операцій (Math Operation) у середовищі Simulink, скласти модель (структурну схему дій) для виконання заданого розрахунку.
4. Скориставшись блоками бібліотеки каналів і підсистем (Ports & Subsystems) проаналізувати, чи належать отримані змінні до заданого діапазону і відповідний результат вивести у блок Display.

**Опис деяких блоків з бібліотеки** **Math Operation.**

***Блок обчислення суми – Sum.***

Призначення: виконує обчислення суми поточних значень сигналів.

Параметри:

1. Icon shape – форма блоку. Вибирається зі списку.

– Round – коло,

– Rectangular – прямокутник.

2. List of sign – Список знаків. У списку можна використовувати такі знаки: «+» (Плюс), «–» (мінус) і «|» (роздільник знаків).

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

Кількість входів і операція (додавання або віднімання) визначається списком знаків параметра List of sign, при цьому мітки входів позначаються відповідними знаками. У пункті List of sign можна також вказати число входів блоку. У цьому випадку всі входи будуть сумуючими.

Якщо кількість входів блоку перевищує 3, то зручніше використовувати блок Sum прямокутної форми.

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

Якщо кількість входів блоку більше, ніж один, то блок виконує поелементні операції над векторними і матричними сигналами. При цьому кількість елементів в матриці або векторі має бути однаковим.

Якщо як список знаків вказати цифру 1 (один вхід), то блок можна використовувати для визначення суми елементів вектора. Приклади використання блоку Sum показані на рис. 6.

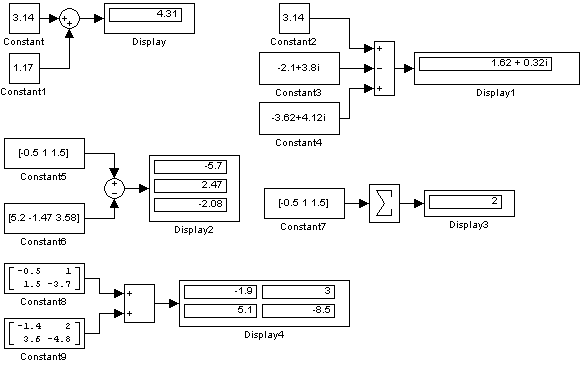


Рис. 6.Приклади використання блоку Sum

***Блок множення Product.***

Призначення: виконує обчислення твори поточних значень сигналів.

Параметри:

* 1. Number of inputs – Кількість входів. Може задаватися як число або як список знаків. У списку знаків можна використовувати знаки «\*» (помножити) і «/» (розділити).
  2. Multiplication – Спосіб виконання операції. Може приймати значення (зі списку):

– Element-wise – Поелементний.

– Matrix – Матричний.

* 1. Saturate on integer overflow (прапорець) – Запобігання переповнення цілого. При встановленому прапорці обмеження сигналів цілого типу виконується коректно.

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

Блок може використовуватися для операцій множення або ділення скалярних векторних або матричних сигналів. Типи вхідних сигналів блоку повинні збігатися. Якщо в якості входів вказати цифру 1 (один вхід), то блок можна використовувати для визначення добутку елементів вектора. Подібний до нього блок Divide так само являється модулем добутку. Приклади використання блоку Product при виконанні скалярних і поелементних операцій показані на рис. 7.

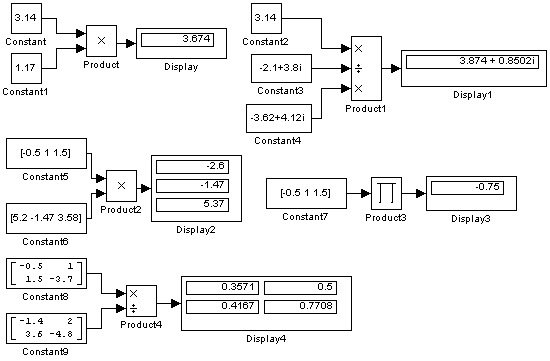


Рис. 7. Приклади використання блоку Product при виконанні скалярних і поелементних операцій

При виконанні матричних операцій необхідно дотримуватися правила їх виконання. Наприклад, при множенні двох матриць необхідно, щоб кількість рядків першої матриці дорівнювало кількості стовпців другої матриці. Приклади використання блоку Product при виконанні матричних операцій показані на рис. 8. У прикладі показані операції формування зворотної матриці, розподіл матриць, а також множення матриць.

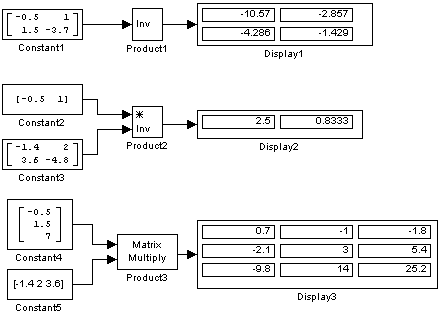


Рис. 8. Приклади використання блоку Product при виконанні матричних операцій

***Блок визначення мінімального чи максимального значення*** ***MinMax.***

Призначення: визначає максимальне або мінімальне значення з усіх сигналів, що надходять на його входи.

Параметри:

* Function – Вихідний параметр. Вибирається зі списку:
* Min – мінімальне значення.
* Max – Максимальне значення.
* Number of input ports – Кількість вхідних портів.

Вхідні сигнали блоку можуть бути скалярними або векторними. Блок визначає максимальне або мінімальне значення з усіх скалярних сигналів, що надходять на його входи. Якщо вхідні сигнали є векторними, то блок виконує поелементну операцію пошуку мінімального або максимального значення. У цьому випадку розмірності векторів повинні збігатися. Якщо кількість вхідних портів блоку задано рівним 1, то блок може використовуватися для знаходження мінімального чи максимального значення у вхідному векторі. Приклади використання блоку MinMax показані на рис. 9.1, 9.2.

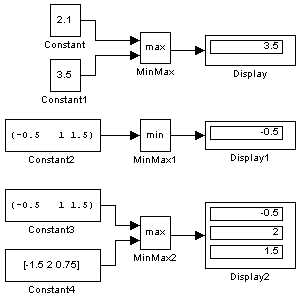


Рис. 9.1. Приклад використання блоку MinMax

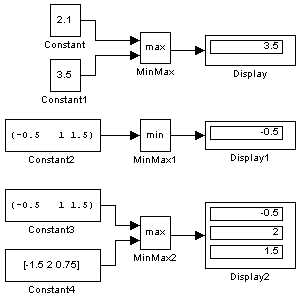


Рис. 9.2. Приклад використання блоку MinMax

***Блок алгебраїчного контуру Algebraic Constraint.***

Призначення: виконує пошук коренів алгебраїчних рівнянь.

Параметри:

* Initial guess – Початкове значення вихідного сигналу.

Блок знаходить таке значення вихідного сигналу, при якому значення вхідного сигналу стає рівним нулю. При цьому вхідний сигнал повинен бути прямо чи опосередковано пов'язаний з вихідним сигналом. На рис. 10 показаний приклад рішення системи нелінійних рівнянь виду:



Оскільки дана система рівнянь має два рішення, то початкові значення блоків Algebraic Constraint задані у вигляді векторів. Для першого (верхнього) блоку початкове значення задано вектором [1 -1], а для другого (нижнього) блоку – вектором [-1 1].

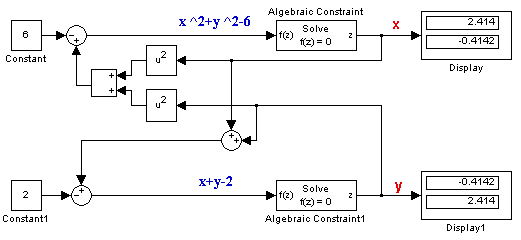


Рис. 10. Приклад використання Algebraic Constraint

Блок Algebraic Constraint може використовуватися також і для вирішення нелінійних матричних рівнянь. На рис. 11 показаний приклад вирішення нелінійного матричного рівняння виду:

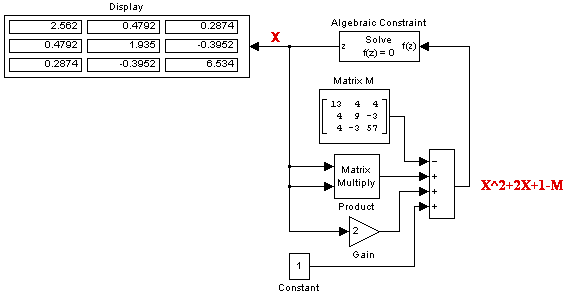
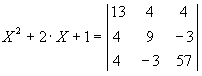


Рис. 11. Приклад використання блоку Algebraic Constraint для вирішення нелінійного матричного рівняння

**Опис деяких блоків бібліотеки Subsystem (підсистеми).**

Підсистема це фрагмент Simulink-моделі, оформлений у вигляді окремого блоку. Використання підсистем при складанні моделі має такі позитивні сторони:

* 1. Зменшує кількість одночасно відображаються блоків на екрані, що полегшує сприйняття моделі (в ідеалі модель повністю повинна відображатися на екрані монітора).
  2. Дозволяє створювати і налагоджувати фрагменти моделі окремо, що підвищує технологічність створення моделі.
  3. Дозволяє створювати власні бібліотеки.
  4. Дає можливість синхронізації паралельно працюючих підсистем.
  5. Дозволяє включати в модель власні довідкові кошти.
  6. Дає можливість пов'язувати підсистему з яких-небудь m-файлом, забезпечуючи запуск цього файлу при відкритті підсистеми (нестандартне відкриття підсистеми).

Використання підсистем і механізму їх блоків дозволяє створювати блоки, які не поступаються стандартним за своїм оформленням (власне вікно параметрів блоку, піктограма, довідка тощо).

Кількість підсистем в моделі не обмежена, крім того підсистеми можуть включати в себе інші підсистеми. Рівень вкладеності підсистем один в одного також не обмежений.

Зв'язок підсистеми з моделлю (або підсистемою верхнього рівня ієрархії) виконується за допомогою вхідних (блок Inport бібліотеки Sources) і вихідних (блок Outport бібліотеки Sinks) портів. Додавання в підсистему вхідного або вихідного порту призводить до появи на зображенні підсистеми мітки порту, за допомогою якої зовнішні сигнали передаються всередину підсистеми або виводяться в основну модель. Перейменування блоків Inport або Outport дозволяє змінити мітки портів, які відображаються на піктограмі підсистеми зі стандартних (In і Out) на ті, які потрібні користувачеві.

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

Підсистеми можуть бути також керованими або некерованими. Керовані підсистеми завжди є монолітними. Керовані підсистеми мають додаткові (управителі) входи, на які надходять сигнали активізують дану підсистему. Керуючі входи розташовані зверху чи знизу підсистеми. Коли керована підсистема активізована - вона виконує обчислення. У тому випадку якщо керована підсистема пасивна, то вона не виконує обчислення, а значення сигналів на її виходах визначаються налаштуваннями вихідних портів.

Для створення в моделі підсистеми можна скористатися двома способами:

Скопіювати потрібну підсистему з бібліотеки Subsystem в модель.

Виділити за допомогою миші потрібний фрагмент моделі і виконати команду Create Subsystem з меню Edit вікна моделі. Виділений фрагмент буде поміщений в підсистему, а входи і виходи підсистеми будуть забезпечені відповідними портами. Даний спосіб дозволяє створити віртуальну некеровану підсистему. Надалі, якщо це необхідно, можна зробити підсистему монолітною, змінивши її параметри, або керованою, додавши керуючий елемент з потрібної підсистеми знаходиться в бібліотеці. Скасувати угруповання блоків в підсистему можна командою Undo.

***Віртуальна і монолітна підсистеми Subsystem і Atomic Subsystem.***

Доступ до вікна параметрів підсистеми здійснюється через меню Edit командою Block Parameters.

Параметри:

* Show port labels - Показати мітки портів.
* Treat as atomic unit (прапорець) - Вважати підсистему монолітною. Таким чином, блоки віртуальною і монолітною підсистем - це один і той же блок, що відрізняється значенням даного параметра.
* Access - Доступність підсистеми для змін. Вибирається зі списку:
* ReadWrite - Користувач може відкривати та змінювати підсистему.
* ReadOnly - Користувач може відкривати підсистему тільки для перегляду.
* NoReadOrWrite - Користувач не може відкривати та змінювати підсистему.
* Name of error callback function - Ім'я функції використовуваної для обробки помилок виникають у даній підсистемі.

Інші параметри підсистеми доступні при розробці програм з використанням Real-Time Workshop і розглянуті в документації на цю програму.

Знаходиться в бібліотеці блок Subsystem (або Atomic Subsystem) містить вхідний і вихідний порти і лінію зв'язку між ними.

Після того як блок підсистеми скопійований з бібліотеки в модель, він стає доступним для редагування. Приклад використання блоку Subsystem показані на рис. 12.

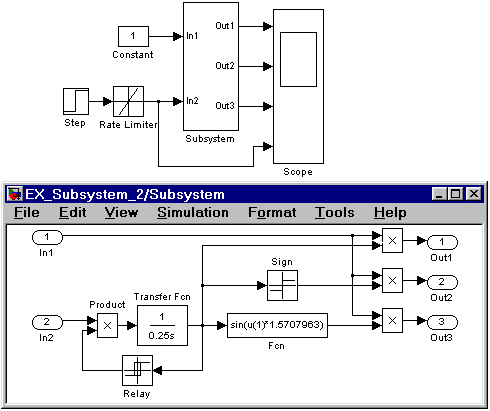


Рис. 12. Приклад використання блоку Subsystem

***Блок умовного оператора If.***

Призначення: забезпечує формування керуючих сигналів для підсистем If Action

Subsystem. Блок є аналогом оператора if-else мови програмування C.

Параметри:

* Number of inputs – Кількість входів.
* If expression – Умовний вираз. Умовний вираз може включати в себе наступні знаки: <. <=, ==, ~ =,>,> =, &, |, [], А також унарний мінус. Якщо записане умовний вираз істинний, то на вихідному If-порту блоку формується керуючий сигнал.
* Elseif expressions – Одне або список альтернативних умовних виразів розділених комами, обчислюються, якщо умовний вираз If expression помилково. Кожному умовного виразу, записаного в списку Elseif expressions відповідає вихідний Elseif-порт на якому формується керуючий сигнал, якщо відповідне умовне вираження істинне. При цьому алгоритм обчислення альтернативних умовних виразів такий, що якщо одне з альтернативних умовних виразів виявиться істинним, то наступні у списку вирази не перевіряються. Альтернативне умовний вираз може містити в собі ті ж знаки, що і вираз If expression.
* Show else condition (прапорець) – Показати Else-порт. На Else-порту формується керуючий сигнал, якщо умовний вираз і всі альтернативні умовні вирази помилкові.

На піктограмі блоку відображаються умовні вирази, записані в його параметрах. Додавання кожного нового альтернативного умовного виразу призводить до появи нового Elseif вихідного порту.

Якщо вхідні сигнали блоку є скалярами, то для їх позначення у виразах використовується запис виду u1, u2, u3 і т.д. Якщо вхідні сигнали векторні, то для позначення елементів вектора використовуються вирази виду u1 (1), u1 (2), u2 (1), u2 (2) і т.д.

На рис. 13 показаний приклад використання блоку If спільно з підсистемами If Action Subsystem. У прикладі першої підсистема пропускає через себе вхідний сигнал якщо вхідний сигнал блоку If більше 1, друга - якщо вхідний сигнал менше -1 (мінус один), і третя - якщо вхідний сигнал лежить в інтервалі від -1 до +1.

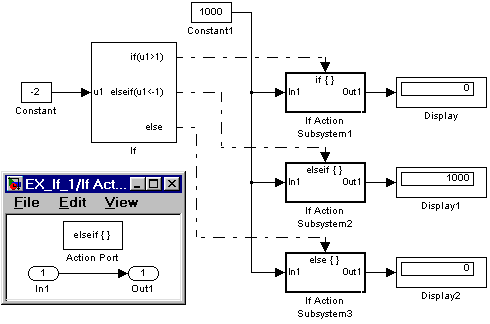


Рис. 13. Використання блоку If спільно з підсистемами

If Action Subsystem

***Блок перемикача Switch Case.***

Призначення: забезпечує формування керуючих сигналів для підсистем Case Action Subsystem. Блок є аналогом оператора Switch мови програмування C.

Параметри:

* 1. Case conditions – Список значень вхідних сигналів (ціле число). Кожному значенню відповідає окремий вихідний Case-порт. Якщо значення вхідного сигналу, що надходить на вхід блоку Switch Case, збігається з яким або значенням із списку, то на відповідному виході блоку формується керуючий сигнал. Якщо вхідний сигнал не є цілим, то його дробова частина відкидається. У виразі Case conditions можна використовувати квадратні дужки, якщо необхідно виробляти керуючий сигнал на будь-якому порту для кількох значень вхідного сигналу. Наприклад, вираз {1, [7,9]} задає два вихідних Case-порту. На першому з них керуючий сигнал формується, якщо вхідний сигнал блоку дорівнює 1, а на другому, - якщо вхідний сигнал дорівнює 7 або 9. У виразі Case conditions можна використовувати також діапазони значень. Наприклад, вираз {1:5} визначає, що для єдиного вихідного Case-порту вихідний сигнал буде вироблятися, якщо вхідний сигнал блоку дорівнює 1, 2, 3, 4 або 5.
  2. Show default case (прапорець) – Показати default case-порт. На виході default case-порту формується керуючий сигнал, якщо вхідний сигнал блоку не збігається з жодним значенням, перелічених у списку Case conditions.

На рис. 14 показаний приклад використання блоку Switch Case спільно з підсистемами Switch Case Action Subsystem. У прикладі першої підсистема пропускає через себе вхідний сигнал, якщо вхідний сигнал блоку Switch Case дорівнює 1, друга - якщо вхідний сигнал дорівнює -1 (мінус один), і третя - якщо вхідний сигнал не дорівнює ні -1 ні +1.

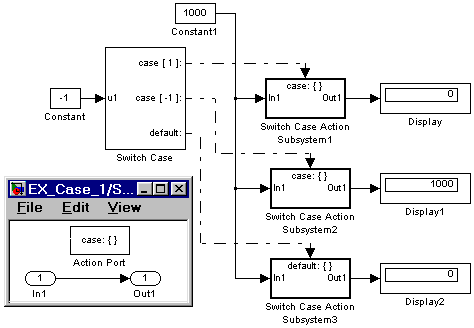


Рис. 14. Використання блоку Switch Case спільно з підсистемами

Switch Case Action Subsystem

**Методичний приклад**

Задачу досягнення цілей задано у вигляді системи двох лінійних нерівностей виду:



В середовищі Simulink потрібно побудувати модель рішення даної задачі таким чином, щоб при задані будь-яких значень коефіцієнтів рівнянь, модель автоматично розраховувала значення змінних, параметри яких задовольняли б заданій умові.

Для побудови даної моделі необхідно виконати аналіз зв’язків коефіцієнтів та змінних відповідно до результату, а також потрібно визначити структуру дій для складання правильної схеми.

Попередньо задамо навмання вибрані значення коефіцієнтів *a*=1, *b*=4, *c*=3, *d*=2.

Використовуючи блоки з бібліотеки Math Operation побудуємо структурну схему моделі (рис. 15.1), попередньо задавши значення коефіцієнтів у робочому просторі Matlab:

clear all;

a=1;

b=4;

c=3;

d=2;

Дані значення, які збережені в області Workspace, до редактора моделі завантажуються автоматично, коли ми прописуємо відповідні змінні у блоках структурної схеми.

Для перевірки результатів побудованої моделі, для початку, виконаємо аналітичний розв’язок даної системи рівнянь:





Добудуємо нашу модель, додавши до неї блок аналізу отриманих коренів рівнянь до заданого діапазону значень. Замаскуємо модель аналізу за допомогою блоку Subsystem (рис. 15.2).

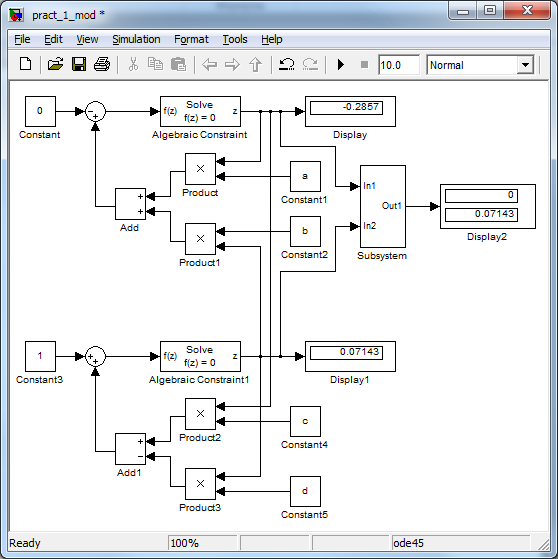


Рис. 15.1. Структурна схема в середовищі Simulink

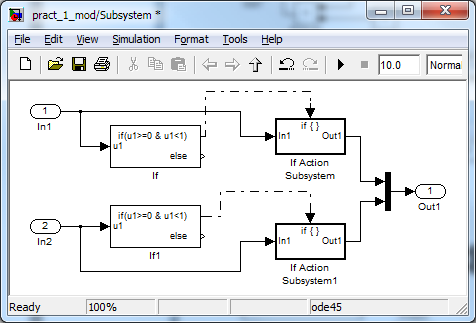


Рис. 16.2. Структурна схема замаскованої моделі у блоці Subsystem

Як ми бачимо з рис. 15.2. блок аналізу If можна записувати не повністю, лише по дійсному значенню (відповідність), протилежне значення (невідповідність) просто не використовується і може залишатись пустим.

Таким чином, побудувавши математичну модель у вигляді структурної схеми з функціональних блоків середовища Simulink, ми створили складну систему для рішення системи лінійних нерівностей заданого виду. Побудована структура дій складається з послідовних, паралельних та зворотних зв’язків.

**Завдання**

Відповідно до зданої умови, для виконання практичної роботи необхідно:

* 1. Скласти схему виконання дій для досягнення локальних цілей (рішення алгебраїчних нерівностей) з описом взаємозв’язків елементів, відповідно до загальної методології системного дослідження.
  2. Визначити послідовність дій для знаходження невідомих значень.
  3. Побудувати модель для рішення системи лінійних нерівностей в середовищі Simulink.
  4. Зробити висновки.
  5. Оформити звіт.

**Варіанти завдань**

*Таблиця 1*

***Параметри завдання***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Завдання | №  п/п | Завдання |
| 1 | 2 | 3 | 4 |
| 1. |  | 7. |  |
| 2. |  | 8. |  |
| 3. |  | 9. |  |
| 4. |  | 10. |  |
| 5. |  | 11. |  |
| 6. |  | 12. |  |

*Продовження таблиці 1*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 13. |  | 22. |  |
| 14. |  | 23. |  |
| 15. |  | 24. |  |
| 16. |  | 25. |  |
| 17. |  | 26. |  |
| 18. |  | 27. |  |
| 19. |  | 28. |  |
| 20. |  | 29. |  |
| 21. |  | 30. |  |

**Контрольні запитання**

1. Що виступає елементарним актом в рішенні завдання при дослідженні складних систем.
2. Охарактеризуйте загальний вигляд системи.
3. Дайте визначення поняттю складної системи.
4. Опишіть три основні характеристики дії при дослідженні складних систем.
5. Опишіть три випадки розподілу дій при дослідженні складних систем.
6. Поясніть поняття локальної та глобальної цілі.
7. Опишіть зв’язки, які можуть бути між локальними цілями.
8. Охарактеризуйте способи досягнення локальних цілей.
9. В чому полягає сутність системного підходу?
10. Опишіть етапи проектування технологічних об’єктів?