В. М. Дубовой, М. С. Юхимчук, Ю. Я. Лещенко

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ В СИСТЕМІ SCILAB/XCOS

Міністерство освіти і науки України
Вінницький національний технічний університет
Кафедра Комп'ютерних систем управління

В. М. Дубовой, М. С. Юхимчук, Ю. Я. Лещенко

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ В СИСТЕМІ SCILAB/XCOS

Електронний навчальний посібник
комбінованого (локального та мережного) використання

Видання 2-е, перероблене та доповнене

Вінниця
ВНТУ
2024

УДК 519.876.2:004.94
Д79

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 6 від 28.11.2023 р.)

Рецензенти: Р. Н. Квєтний, доктор технічних наук, професор,
А. Я. Кулик, доктор технічних наук, професор,
П. Д. Лежнюк, доктор технічних наук, професор

Дубовой, В. М.

Д79 Імітаційне моделювання в системі Scilab/xcos: електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс]. – [Вид. 2-е, переробл. та доповн.]. / Дубовой В. М., Юхимчук М. С., Лещенко Ю. Я. – Вінниця : ВНТУ, 2024. – 119 с.

Навчальний посібник з використання безкоштовного пакета Scilab/xcos призначений для першого знайомства і набуття навичок імітаційного моделювання. Знання цього інструменту моделювання необхідно фахівцям в усіх сферах науки і техніки. Навчальний посібник призначений для студентів денної і заочної форм навчання технічних та економічних спеціальностей.

УДК 519.876.2:004.94

Зміст

ВСТУП

Пакет **Scilab** призначений для виконання інженерних і наукових обчислень та математичного моделювання систем. За своїми можливостями пакет **Scilab** можна порівняти з відомим математичним пакетом MATLAB. Однак пакет **Scilab** – програма вільного розповсюдження (безкоштовна). Існують версії **Scilab** для різних операційних систем: Linux, Windows, MacOS. Останню версію пакета завжди можна скачати на офіційному сайті програми **www.scilab.org**.

Scilab дозволяє здійснювати такі інженерні та наукові обчислення:

• розв’язання нелінійних рівнянь і систем;

• розв’язання задач лінійної алгебри;

• розв’язання задач оптимізації;

• диференціювання та інтегрування;

• обробка експериментальних даних (інтерполяція і апроксимація, метод найменших квадратів);

• розв’язання звичайних диференціальних рівнянь і систем.

Крім того, **Scilab** надає широкі можливості зі створення і редагування різних видів графіків і поверхонь.

Аналогічно тому, як у MATLAB є додаток для імітаційного моделювання систем Simulink, так у **Scilab** є додаток **Xcos**, який містить бібліотеку універсальних блоків для побудови моделі та здійснення імітаційного моделювання систем.

Під час роботи з **Xcos** користувач має можливість модернізувати бібліотечні блоки, створювати свої власні, а також складати нові бібліотеки блоків за допомогою підпрограм, написаних мовами **C++,** Fortran.

Під час моделювання користувач може вибирати метод розв'язання диференціальних рівнянь, а також спосіб зміни модельного часу (з фіксованим або змінним кроком). У процесі моделювання є можливість стежити за процесами, які відбуваються в системі. Для цього використовують пристрої спостереження, які входять до складу бібліотеки **Xcos**. Результати моделювання можуть бути подані у вигляді графіків або таблиць.

Цінність **Xcos** полягає у великій бібліотеці компонентів (блоків). Вона містить джерела сигналів з практично будь-якими часовими залежностями, масштабувальні, лінійні і нелінійні перетворювачі з різноманітними формами передавальних характеристик, блоки інтегрування і диференціювання тощо. Як програмний засіб **Xcos** – типовий представник візуально-орієнтованого засобу програмування. Програма автоматично генерується в процесі введення обраних блоків, їх з'єднань і задавання параметрів. Для опису процесів, що протікають в системах, можуть використовуватися різні типи об'єктів за

характером зміни в часі (дискретні, безперервні, циклічні). Практично для всіх блоків існує можливість індивідуального налаштування.

Система **Scilab** є відкритою. Аналогічно системі Linux, вона розвивається зусиллями ентузіастів-користувачів. Незважаючи на те, що система **Scilab** містить достатню кількість вбудованих команд, операторів і функцій, відмінна її риса – це гнучкість. Користувач може створити будь-яку нову команду або функцію, а потім використовувати її нарівні з вбудованими. До того ж, система має досить потужну власну мову програмування високого рівня, що говорить про можливість вирішення нових завдань. Якщо в процесі роботи користувача ним буде розроблена група блоків, які дозволяють розв'язувати певний клас задач, які не передбачені у **Scilab**, то він може надіслати їх через сайт **www.scilab.org** і вони можуть бути включені у нову версію пакета.

Значна частина посібника запозичена з Інтернет-джерел.

У другому виданні посібника враховано досвід його застосування у навчальному процесі, виправлено виявлені неточності, додано пояснення деяких особливостей застосування імітаційного моделювання. Також додано розділ з прикладами імітаційних моделей.

1 ТЕОРЕТИЧНІ ОСНОВИ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

1.1 Концепція імітаційного моделювання

Імітаційне моделювання – це метод дослідження, який полягає у відтворенні властивостей реальних об'єктів за допомогою віртуальних об'єктів. Всі розрахунки в комп'ютерній моделі виконуються в так званому системному часі, який відповідає реальному часу функціонування об'єкта дослідження або системи у певному масштабі. Отже, імітаційним моделюванням називають відтворення на комп'ютері розгорнутого в часі процесу функціонування системи, що складається з віртуальних об’єктів, з урахуванням її взаємодії із зовнішнім середовищем.

Імітаційне моделювання – найбільш універсальний метод дослідження і оцінення ефективності систем, поведінка яких залежить від випадкових факторів. Моделі є хорошим засобом прогнозування поведінки об'єктів і систем. Моделювання дозволяє проводити контрольовані експерименти в ситуаціях, коли проведення експериментів на реальних об'єктах є недоцільним, небезпечним, неможливим або досить дорогим.

До імітаційного моделювання вдаються, коли:

– дорого або неможливо експериментувати на реальному об'єкті;

– неможливо побудувати аналітичну модель, тому що в системі є складні логічні причинно-наслідкові зв'язки, нелінійні динамічні блоки, стохастичні (випадкові) впливи;

– необхідно дослідити поведінку системи впродовж часу.

Основна задача імітаційного моделювання полягає у декомпозиції системи на відносно прості блоки (підсистеми), аналітичні моделі яких відомі або можуть бути легко отримані; встановленні зв’язків між цими блоками; поданні вхідних впливів на систему у вигляді послідовності числових значень, які надходять з певним інтервалом (інтервалом дискретизації процесів у часі); здійсненні їх перетворень послідовно та відповідно до математичних залежностей, які описують послідовність і зміст перетворення впливів, сигналів і даних у реальній системі.

Імітаційне моделювання суттєво спрощує процес отримання результатів через можливість використання агрегатного принципу (тобто декомпозиції системи на блоки і їх об’єднання у агрегати). Це означає, що відпадає необхідність розв’язування складних систем рівнянь, які описують функціонування складних замкнених систем керування.

Пояснимо ці переваги на прикладі. Нехай необхідно здійснити моделювання нелінійної динамічної системи, структурна схему якої зображено на рис.1.1.

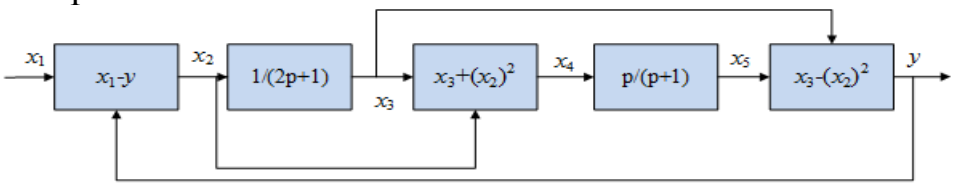


Рисунок 1.1 – Приклад нелінійної динамічної системи

Наведена система описується системою рівнянь

$$\begin{cases} x_2(t) = x_1(t) - y(t) \\ 2\frac{dx_3(t)}{dt} + x_3(t) = x_2(t) \\ x_4(t) = x_3(t) + x_2^2(t) \\ \frac{dx_5(t)}{dt} + x_5(t) = \frac{dx_4(t)}{dt} \\ y(t) = x_3(t) - x_2^2(t) \end{cases}.$$

Отримати модель системи загалом або розв’язати систему рівнянь важко, а інколи неможливо, через її нелінійність, а завдяки агрегатному принципу та імітаційному моделюванню результати можна отримати досить просто.

Найбільш поширеним способом дослідження імітаційної моделі є *статистичне моделювання*.

Методика статистичного моделювання містить в собі ряд послідовних етапів:

− моделювання на комп’ютері випадкових впливів на систему у вигляді числових послідовностей із заданою кореляцією і законом розподілення ймовірностей;

− моделювання перетворення сигналів;

− статистична обробка результатів моделювання.

1.2 Генерування вхідних впливів

У системі Xcos передбачено генерування випадкових некорельованих впливів, розподілених за рівномірним або нормальним законом розподілу (рівномірний або нормальний білий шум). Генерування впливів з іншими розподілами ймовірностей є доволі складною задачею.

Задача дещо спрощується за генерування нормальної послідовності із заданою кореляційною функцією, оскільки отримання заданої кореляційної

функції виконується шляхом лінійної фільтрації, яка не призводить до втрати нормальності розподілу послідовності даних.

Методика генерування нормальної послідовності із заданою кореляційною функцією передбачає використання початкової послідовності типу «нормальний білий шум» – таку послідовність звичайно генерують стандартні генератори. Спектральна щільність потужності такої послідовності є константою $G_{xx} = D_x/2\pi$. Методика складається з таких кроків:

1. Підготовчий етап:

1.1. Знаходження спектральної щільності необхідної послідовності як перетворення Фур'є заданої кореляційної функції

$$G'_{xx}(j\omega) = \frac{1}{2\pi}\int_0^{\infty} R_{xx}(\tau)e^{-i\omega\tau}d\tau;$$

1.2. Знаходження амплітудно-частотної характеристики необхідного фільтра $A(\omega) = |W(j\omega)| = \frac{|G'_{xx}(j\omega)|}{D_x/2\pi}$;

1.3. Знаходження відповідного дискретного перетворення.

2. Власне моделювання:

2.1. Генерування початкової послідовності типу нормального білого шуму з заданою дисперсією;

2.2. Перетворення її за знайденою формулою дискретної фільтрації – отримання вхідної послідовності для імітаційної моделі;

2.3. Імітація перетворення вхідних даних системою.

3. Обробка результатів імітаційного моделювання.

Окремим випадком є генерування числової послідовності з нормальним розподілом ймовірностей і експоненціальною автокореляційною функцією. Цей вид випадкової послідовності є найпоширенішим і найлегшим для генерування. Генерування здійснюється у два етапи:

1. Генерування некорельованої рівномірно розподіленої послідовності $\{x_i\}$ *за допомогою будь-якої програми-генератора, які наразі є в усіх мовах програмування;*

2. Перетворення на задану послідовність $\{x'_j\}$ *методом підрахунку ковзного середнього*

$$x'_j = \frac{1}{m}\sum_{i=j}^{j+m-1} x_i.$$

За $m \geq 6$ через дію умов центральної граничної теореми теорії ймовірностей розподіл послідовності $\{x'_j\}$ буде близьким до нормального. Крім того, елементи цієї послідовності під час утворення мають різну кількість спільних доданків, наприклад:

за $m = 6$ маємо $x_1' = x_1 + x_2 + \cdots + x_6,$
$x_2' = x_2 + \cdots + x_6 + x_7,$
$x_3' = x_3 + \cdots + x_6 + x_7 + x_8,$
тобто x_1' має 5 спільних доданків з x_2', 4 спільних доданки з x_3' і т. д., що приводить до поступового зменшення зв'язку між елементами отриманої послідовності. Це приблизно відповідає експоненціальному характеру кореляційної функції. Очевидно, інтервал кореляції цієї послідовності буде $m - 1$.

1.3 Обробка результатів імітаційного моделювання

Обробка результатів імітаційного моделювання здійснюється за тими самими правилами і методиками, що й обробка результатів реальних натурних експериментів. Зокрема, якщо моделями вхідних сигналів є випадкові числові послідовності, то для обробки результатів використовуються методи дисперсійного, кореляційного і регресійного аналізу.

1.4 Оцінювання необхідного обсягу тестів та трудомісткості імітаційного моделювання

Основою для визначення необхідного обсягу тестових даних є оцінка похибки імітаційного моделювання. Залежно від вигляду статистичних характеристик вихідних сигналів імітаційної моделі, які необхідно оцінити у конкретній задачі, похибки оцінки обчислюються за формулами:

– *за оцінювання розподілу ймовірностей* способом підраховування кількості даних, які потрапляють у кожен інтервал розбиття діапазону можливих значень (спосіб побудови гістограми) відносна похибка розраховується за формулою

$$\varepsilon_{p^*} = \sqrt{(1 - P_A)/N},$$

де P_A– ймовірність потрапляння даного у окремий інтервал;

N– обсяг вибірки.

Під час оцінювання розподілу ймовірностей на основі визначення відносного часу перебування у заданому інтервалі

$$\varepsilon_{p^*} = \sqrt{\frac{2(1-P_A)\tau}{TP_A}},$$

де T – загальний час спостереження;

τ – інтервал надходження дискретних даних;

– *за оцінювання середнього значення* усієї множини результатів моделювання і відсутності кореляції між окремими результатами

$$\varepsilon_{m^*} = \frac{1}{m_x}\sqrt{\frac{D_x}{N}} = \frac{\sigma_x}{m_x\sqrt{N}},$$

де m_xі σ_x – математичне сподівання і с.к.в. результатів.

Під час оцінювання поточного середнього значення методом ковзного середнього на основі рекурентної формули $m_x = \frac{m_x(n-1)+x}{n}$ і відсутності кореляції між окремими результатами

$$\varepsilon_{m^*} = \sqrt{\frac{1}{n}\left(\frac{m_{x^2}}{m_x^2} - 1\right)} = \frac{\sigma_x}{m_x\sqrt{n}};$$

– *під час оцінювання дисперсії* нормально розподіленої послідовності і відсутності кореляції між окремими результатами

$$\varepsilon_D = \frac{\sigma_D}{D_x} = \sqrt{\frac{2}{N-1}};$$

– *під час оцінювання коефіцієнта кореляції* на основі зіставлення часу перевищення деякого рівня *U* у двох корельованих послідовностях

$$\varepsilon_\rho^2 = \frac{D[\rho_x^*(\tau)]}{U^2\rho_x^2(\tau)} = \frac{1}{(\beta_\rho^*)} = \frac{[1-\rho_x^2(\tau)]}{N\rho_x^2(\tau)},$$

де $\beta_\rho^* = U/\sigma_\rho^*(\tau)$.

З наведених співвідношень видно, що в усіх випадках похибка прямо чи опосередковано (через час проведення моделювання) залежить від кількості отриманих даних. Тому необхідна кількість даних може бути знайдена шляхом розв'язування нерівності

$$\varepsilon(N) \le \varepsilon_{max},$$

де $\varepsilon_{\max}$ – максимально допустима похибка моделювання.

Наприклад, для оцінювання закону розподілу ймовірностей шляхом побудови гістограми скористаємось нерівністю на основі формули

$$\sqrt{(1-P)/N} \le \varepsilon_{max}.$$

Якщо гістограма будується на 10 інтервалах розбиття діапазону можливих значень результатів і попередній гіпотезі, що цей закон близький до трикутного, розрахуємо мінімальну ймовірність (з формули видно, що кількість даних обернена до ймовірності)

$$P_{min} = \frac{1}{2}\cdot(0{,}1D_0)\cdot(0{,}2h),$$

де D_0– діапазон;

h – висота трикутного розподілу.

Враховуючи, що $\frac{1}{2}D_0h = 1$, отримуємо $P_{min} = 0{,}02$.

Розв'язуючи нерівність відносно N, знаходимо, що для отримання відносної статистичної похибки не гірше 0,1 (тобто 10% – досить неточний експеримент!) на кожному інтервалі необхідно мати $N \geq (1 - 0{,}02)/0{,}1^2$, тобто не менше 98 даних, а усього для 10 інтервалів імітаційна вибірка має містити не менше 980 даних.

За попередньої гіпотези про нормальність закону розподілу результатів кількість даних має бути значно більшою, оскільки на кінцях діапазону значення ймовірності для нормального закону менше, ніж для трикутного. Очевидно, найменша кількість даних необхідна за умови гіпотези про рівномірний характер розподілу результатів.

Оскільки заздалегідь невідомо, виправдається висунута гіпотеза чи ні, то найчастіше імітаційний експеримент проводять у два етапи: спочатку генерують мінімальну кількість даних, проводять експеримент, попередньо визначають тип розподілу ймовірностей результатів, а потім уточнюють розрахунок необхідної кількості даних і проводять додаткове моделювання.

1.5 Масштаби і масштабні співвідношення

В імітаційній моделі змінні відображаються відповідними «модельними змінними». Для переходу від вихідних математичних співвідношень до модельних змінних і для зворотного переходу кожної зі змінних використовується відповідний масштаб.

Масштабом або масштабним коефіцієнтом математичної змінної x називається множник μ_x.

$$\mu_x = \frac{x_M}{x} = \frac{\text{Значення реальної модельної змінної}}{\text{Значення змінної в системі рівнянь}}.$$

З урахуванням масштабу імітаційні моделі отримуються з математичних співвідношень реальної моделі підстановкою

$$x = \frac{x_M}{\mu_x}.$$

Наприклад, імітаційна модель закону всесвітнього тяжіння $F = \gamma \frac{M \cdot m}{r^2}$ $F = \gamma \frac{M \cdot m}{r^2}$ отримується підстановкою:

$$F = {F_M}/{\mu_F};$$
$$\gamma = {\gamma_M}/{\mu_\gamma};$$
$$M = {M_M}/{\mu_M};$$
$$m = {m_M}/{\mu_m}.$$

Тоді імітаційна модель матиме вигляд

$$F_{M}/_{\mu_F} = \gamma_{M}/_{\mu_\gamma} \frac{{}^{M_M}/_{\mu_M} \cdot {}^{m_M}/_{\mu_m}}{({}^{r_M}/_{\mu_r})^2}.$$

Звідки, виділивши окремо усі масштаби, отримуємо імітаційну модель

$$F_M = \frac{\mu_F \cdot \mu_r^2}{\mu_\gamma \cdot \mu_M \cdot \mu_m} \cdot \gamma_M \frac{M_M \cdot m_M}{r_M^2}.$$

Перевагою використання масштабів є можливість приведення змінних до близьких діапазонів. Це дозволяє спостерігати процеси у одних осях. У розглянутому прикладі маси Землі M і фізичного тіла m відрізняються на багато порядків, а використання різних масштабів дозволяє у імітаційній моделі зробити їх порівнянними.

Найчастіше в імітаційному моделюванні застосовується масштабування часу, оскільки реальні процеси відбуваються, як правило, занадто швидко або занадто повільно для людини-спостерігача. Масштабування часу дозволяє прискорити або уповільнити моделювання.

Якщо позначити через τ час в процесі імітаційного моделювання, то зв'язок цього часу з часом подання реального процесу t виражається в такий спосіб:

$\tau = t$ або $\mu_t = \frac{\tau}{t} = 1$ – натуральний масштаб часу;

$\tau > t$ або $\mu_t = \frac{\tau}{t} > 1$ – сповільнений масштаб часу;

$\tau < t$ або $\mu_t = \frac{\tau}{t} < 1$ – прискорений масштаб часу.

Розглянемо для прикладу модель у вигляді диференціального рівняння

$$\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 2ty = rnd.$$

Для підвищення точності розв’язку воно приводиться до інтегрального рівняння

$$z = rnd - 3\int z \cdot dt - 2t \iint z \cdot dt$$

шляхом заміни $\frac{d^2y}{dt^2} = z$.

Відповідну імітаційну модель і результат моделювання $y(t)$ показано на рис. 1.2.

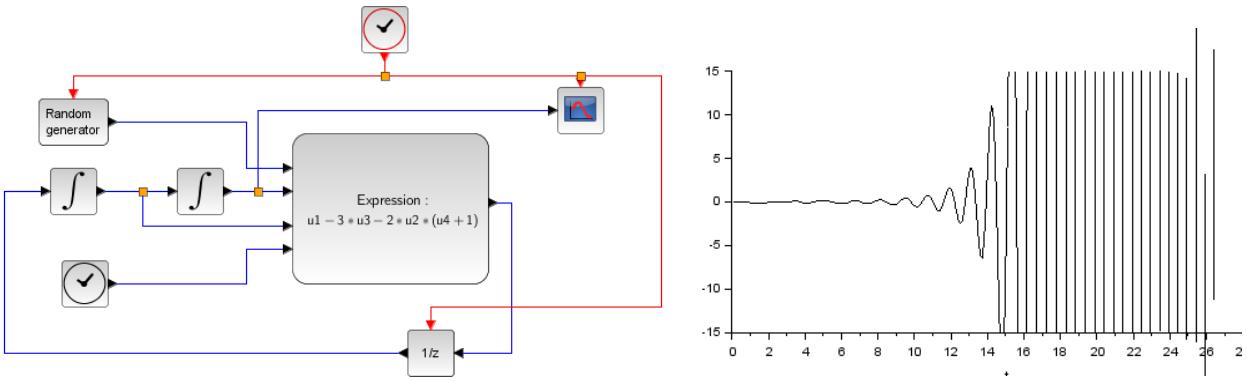


Рисунок 1.2 – Модель і результат моделювання у реальному часі

За необхідності прискорити або уповільнити процес моделювання вводимо віртуальний (модельний) час $\tau = mt$.

Тоді $t = \tau/m$; $dt = \frac{1}{m}d\tau$; і $m^2\frac{d^2y}{d\tau^2} + 3m\frac{dy}{d\tau} + \frac{2}{m}\tau y = rnd$.

Відповідно, інтегральне рівняння для моделювання матиме вигляд

$$z = \frac{1}{m^2}(\text{rnd} - 3\text{m}\int z \cdot d\tau - \frac{2}{m}\tau \iint z \cdot d\tau),$$

а імітаційну модель з масштабом часу показано на рис. 1.3. Значення масштабу часу задається у вікні «Оточення» (Simulation – Set Context). Це зручно, оскільки масштабу часу, введений для одного блока, має застосовуватися для усієї моделі системи.

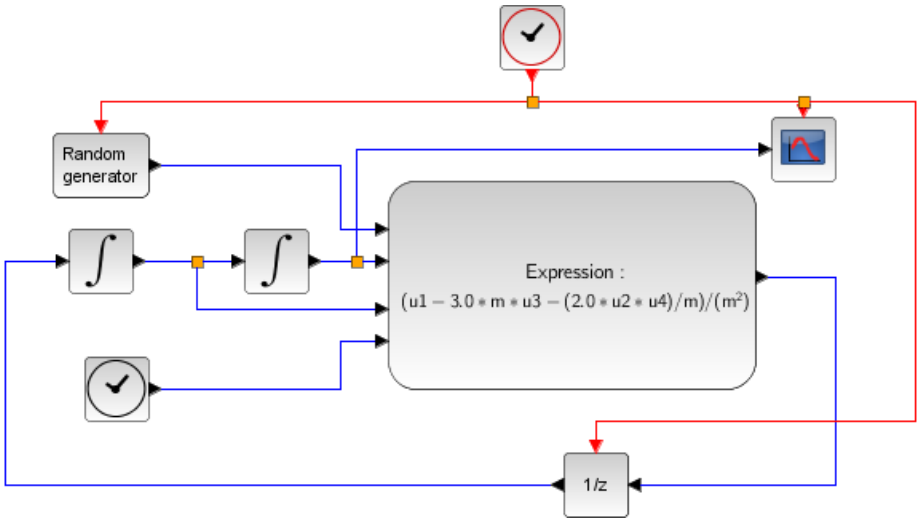


Рисунок 1.3 – Модель з масштабом часу

Зауважимо, що визначення змінної оточення (у цьому випадку m) необхідно зробити ***до запису*** її у налаштуваннях блоків.

Залежно від значення m моделювання може бути пришвидшене, як на

рис. 1.4, а) – тут $m < 1$ і уповільнене, як на рис. 1.4, б) – тут $m > 1$.

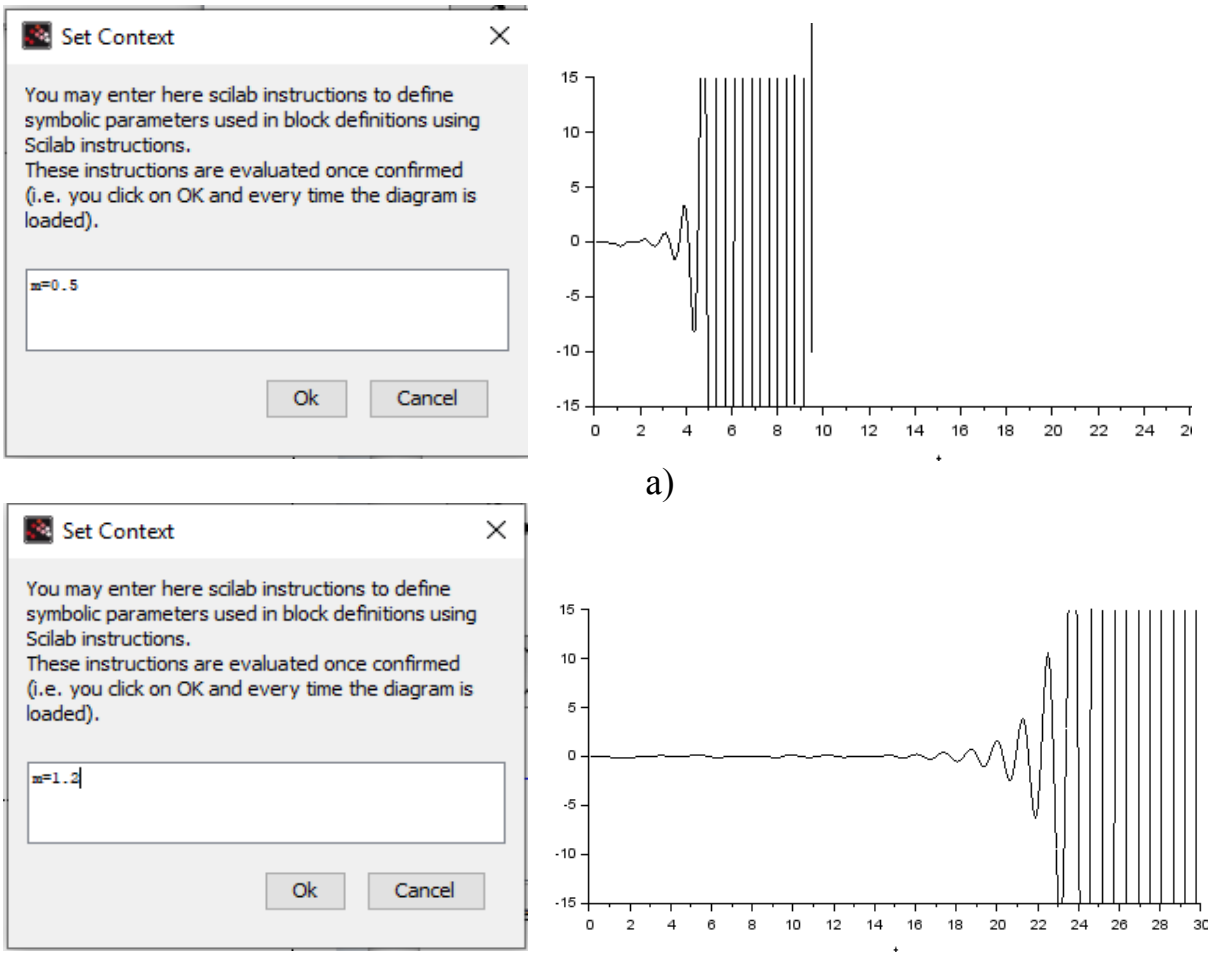


Рисунок 1.4 – Застосування масштабу часу

2 ПОЧАТОК РОБОТИ З Scilab

2.1 Установка Scilab на ПК під керуванням Windows

Версію пакета, що вільно розповсюджується, разом з повною документацією англійською мовою можна отримати на сайті програми **www.scilab.org**. Крім стандартного пакета інсталяції, можна також скачати й встановити велику кількість додаткових інструментів (Toolboxes) з сторінки **https://atoms.scilab.org/** для розв'язування різних типів задач.

Для того, щоб встановити **Scilab** на ПК, необхідно запустити одноіменний виконуваний файл, після чого починає свою роботу **Майстер встановлення**. В першому вікні **Майстра встановлення** потрібно вибрати мову пакета і натиснути кнопку **OK** для продовження установлення. Надалі рекомендується погодитися з усіма пропозиціями **Майстра** (просто натискати **Next**).

Наступне вікно є інформаційним. Користувач отримує повідомлення про те, що на його комп'ютер буде встановлено пакет **Scilab** і рекомендацію закрити інші додатки перед установленням. Для переходу до третього вікна **Майстра встановлення** використовують кнопку **Next**. В цьому вікні потрібно прийняти умови ліцензійної угоди (*I accept the agreement*) і натиснути клавішу **Next** для продовження. На наступному етапі користувачеві буде запропоновано обрати шлях для установлення пакета. За замовчуванням це тека **C:\Program Files\scilab**. Крім того, в цьому вікні виводиться інформація про кількість місця на вибраному диску, необхідного для установлення стандартного набору компонентів системи. Натиснення кнопки **Next** призведе до появи діалогового вікна, в якому буде запропоновано обрати один з чотирьох типів установлення:

- установлення за замовчуванням (*Installation Default*);
- повне установлення (*Full installation*);
- компактне установлення (*Compac tinstallation*);
- установлення обраних компонентів (*Custom installation*).

Рекомендується установлення за замовчуванням.

Далі буде наведено список компонентів, що відповідають обраному типу установлення. Перехід до наступного вікна майстра налаштування параметрів здійснюється за допомогою кнопки **Next.**

Наступне вікно **Майстра встановлення** повідомляє користувачеві про те, що після установлення в **меню Пуск** буде створено ярлик, призначений для запуску **Scilab**. **Next** призведе до появи наступного вікна, в якому **Майстер встановлення** запропонує список додаткових завдань, доступних

після установлення. Для переходу до наступного кроку також необхідно вибрати **Next**.

Після установлення пакета головне меню системи містить команди, призначені для роботи з файлами, налаштування середовища, редагування команд поточної сесії і отримання довідкової інформації. Крім того, за допомогою головного меню можна створювати, редагувати, виконувати налагодження та запускати на виконання так звані файли-сценарії **Scilab**, а також працювати з графічними додатками пакета.

Також після встановлення пакета Ви можете обрати мову інтерфейсу. У наступному тексті будуть використовуватися як українські, так і англійські назви елементів інтерфейсу. У своїй практичній роботі рекомендуємо віддавати перевагу англійському інтерфейсу, оскільки це полегшить використання розроблених моделей у статтях і інших публікаціях Ваших результатів. Мову інтерфейсу можна переключати за допомогою меню  – **General – Language settings**.

Для полегшення роботи наводимо відповідність основних назв українською та англійською мовами.

Таблиця 2.1 – Відповідність українських і англійських назв елементів Scilab

Main menu	**Головне меню**
File	Файл
Edit	Зміни
Control	Керування
Applications	Програми
Xcos	Xcos
	
General	Загальні
Language settings	Параметри мови
?	Довідка
Menu Xcos	**Меню Xcos**
File	Файл
New diagram	Створити діаграму
Open	Відкрити
Recent files	Недавні файли
Close	Закрити
Save	Зберегти
Save as	Зберегти як
Export	Експортувати
Edit	Змінити
Undo	Вернути

Redo	Повторити
Cut	Вирізати
Copy	Копіювати
Paste	Вставити
Delete	Вилучити
Select all	Позначити все
Selection to superblock	Позначене у суперблок
View	Перегляд
Palette browser	Перегляд палітри
Recently Used Blocks	Нещодавно використані блоки
Continuous time systems	Неперервні динамічні системи
Discontinuities	Розриви
Discrete time systems	Дискретні динамічні системи
Lookup tables	Визначення табличних значень
Event handling	Обробка подій
Mathematical operations	Математичні операції
Matrix	Операції над матрицями
Electrical	Електрика
Integer	Ціле
Port & Subsystem	Порт і підсистема
Zero crossing detection	Визначення переходів через нуль
Signal Routing	Маршрутизація сигналів
Signal Processing	Обробка сигналів
Implicit	Неявний
Annotations	Примітки
Sinks	Пристрої реєстрації
Sources	Джерела сигналів
Thermo-Hydraulics	Термогідравлічні блоки
Demonstration blocks	Приклади блоків
User-Defined Functions	Функції, визначені користувачем
Simulation	Моделювання
Setup	Налаштування
Final integration time	Загальний час інтегрування
Real time scaling	Динамічне масштабування
Integrator absolute tolerance	Абсолютна похибка

	інтегрування
Integrator relative tolerance	Відносна похибка інтегрування
Tolerance on time	Похибка за часом
Max integration time interval	Максимальний час інтегрування
Solver kind	Тип розв’язувача
Maximum step size	Максимальний розмір кроку
Set Context	Вказати контекст
Format	Формат
Tools	Інструменти
?	Довідка

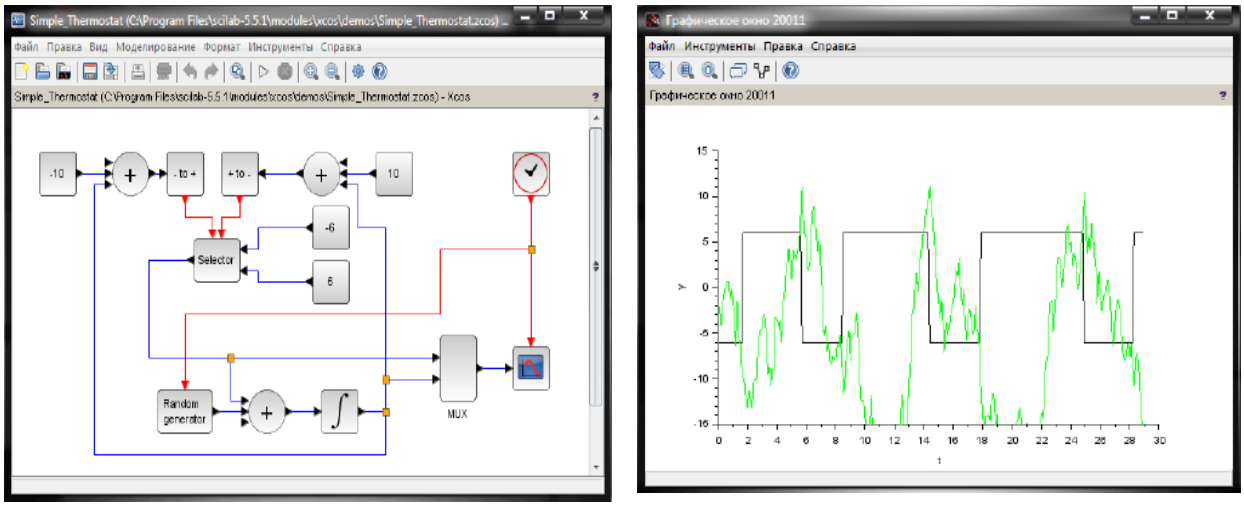
2.2 Загальні відомості про додаток для імітаційного моделювання Xcos

Xcos – це програма, яка входить до складу системи математичного моделювання **Scilab** і надає розробникам можливості проектування систем в галузі автоматики, механіки, гідравліки й електроніки, а також систем масового обслуговування. **Xcos** є графічним інтерактивним середовищем, в основі якого лежить блочне моделювання. Додаток призначений для вирішення завдань динамічного і ситуаційного моделювання систем, процесів, пристроїв, а також тестування та аналізу цих систем. В такому випадку модельований об'єкт (система, пристрій, процес) подається графічно своєю функціональною схемою, що містить блоки елементів системи і зв'язки між ними. Приклад основних вікон **Xcos** показано на рис. 2.1.

Пакет **Xcos** вважається одним з кращих пакетів з відкритим кодом для моделювання блочно заданих (агрегатних) моделей динамічних систем і є графічним інтерактивним середовищем, за допомогою якого можна робити імітацію, тестування, аналіз динамічних систем.

За допомогою **Xcos** можна створювати системи керування, системи обробки сигналів, системи зв'язку, моделі будь-яких динамічних систем. **Xcos** створений розробниками **Scilab** і відрізняється від **Scilab** тим, що **Scilab** – це насамперед високорівнева мова програмування з відповідною оболонкою, а **Xcos** – це система графічного моделювання на основі блок-схем, яка складається з попередньо скомпільованих бібліотек. Моделі **Xcos** зберігаються в файлах з розширенням ***.zcos*** або ***.xcos***.

Для спостереження за процесами і сигналами в **Xcos** можна використовувати різні блоки візуалізації.



а) Приклад Xcos-моделі б) Результати її роботи

Рисунок 2.1 – Робочі вікна Xcos

У **Xcos** модельований об'єкт (система, пристрій, процес) подається блок-схемою, яка містить блоки елементів системи і зв'язки між ними. Функціональні блоки елементів модельованої системи можуть бути складними підсистемами (агрегатами) зі своєю організацією, утворюючи ієрархічні структури.

Блоки, що вносяться в створювану модель, можуть бути пов'язані один з одним за фізичними впливами, даними та керівними сигналами. Інформаційними з'єднаннями передаються дані щодо сигналів і впливів, а керівними – сигнали активації. Блоки також можуть мати інформаційні та керівні входи і виходи. Як правило, інформаційні входи і виходи блоків розташовуються зліва і праворуч від зображення блока, а керівні – зверху і знизу. Тип зв'язку залежить від блока і логіки роботи моделі. Дані, якими обмінюються блоки, можуть бути скалярними величинами, векторами або матрицями довільної розмірності.

2.3 Запуск Xcos

Для запуску програми необхідно попередньо запустити пакет **Scilab** у режимі **Desktop**. Основне вікно пакета **Scilab** показано на рис. 2.2. Там само показано підказку, що з'являється у вікні під час наведення покажчика миші на ярлик **Xcos** в панелі інструментів.

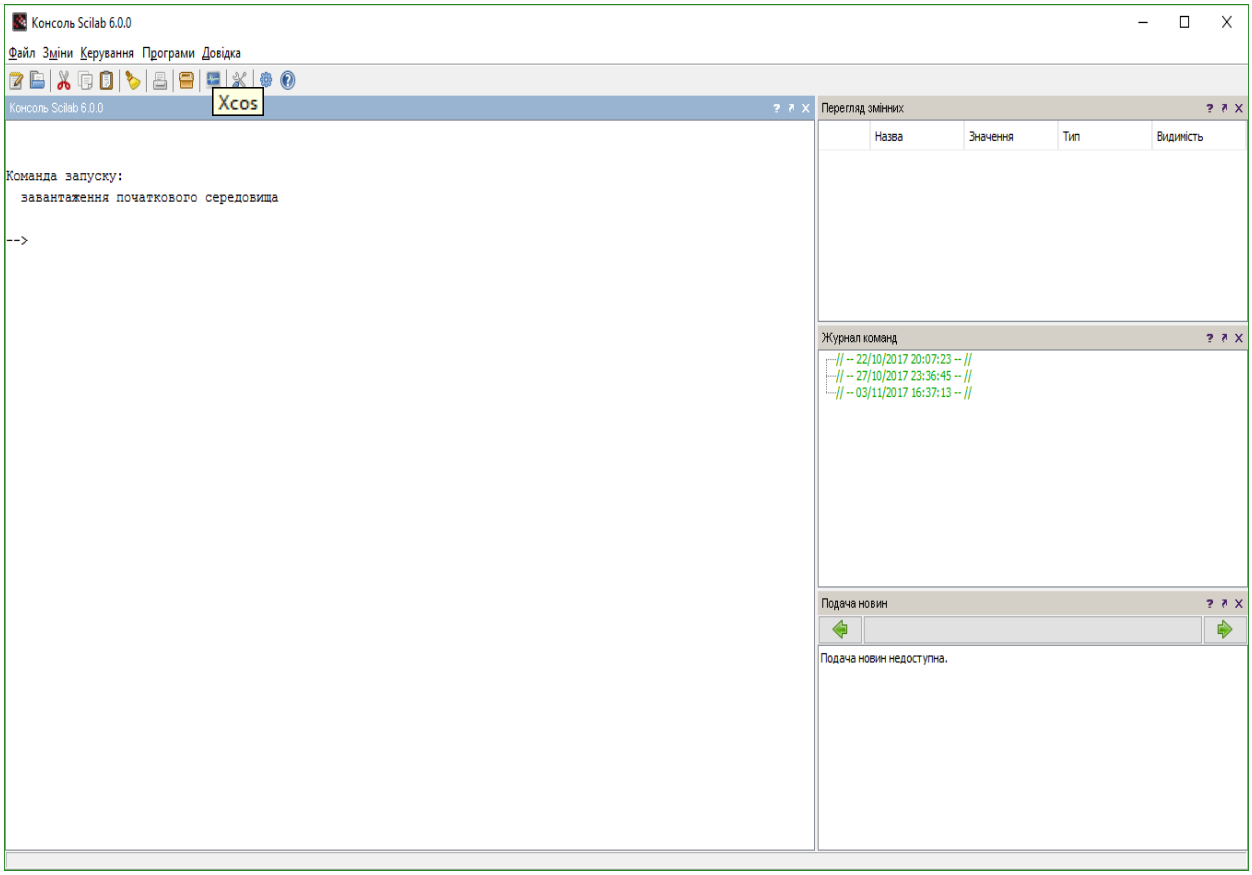


Рисунок 2.2 – Командне вікно Scilab

Після запуску **Xcos** зазвичай відображаються два вікна (рис. 2.3):

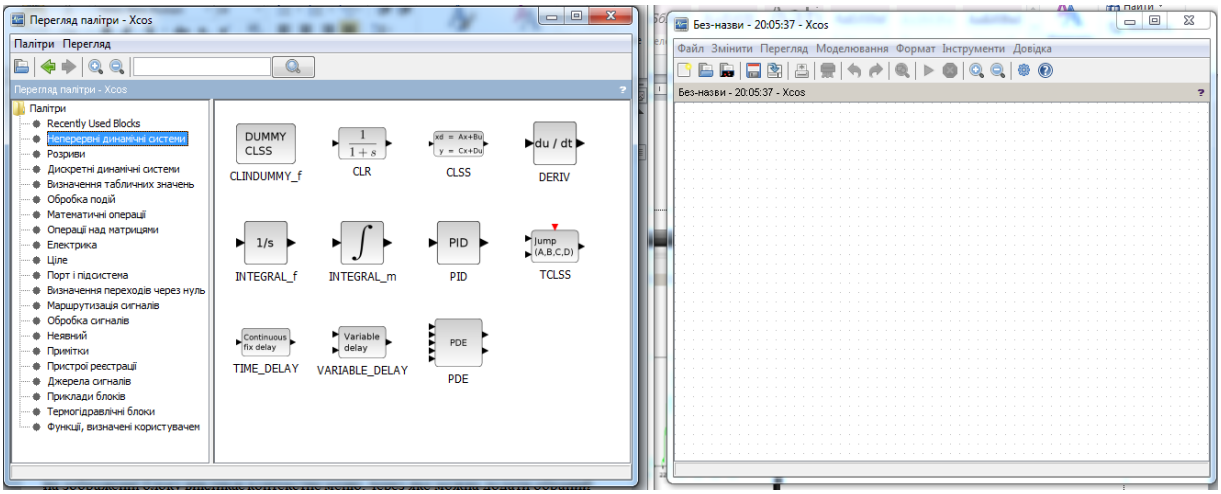
- ✓ вікно ***Перегляд палітри блоків;***
- ✓ вікно ***Графічного редактора***.

Якщо вікно ***Перегляд палітри блоків*** відсутнє, його необхідно відобразити, вибравши ***Перегляд → Палітри блоків*** в головному меню вікна ***Графічного редактора*** **Xcos**.

У вікні ***Перегляд палітри блоків*** подано групи блоків, з яких будується діаграма **Xcos**.

Виділивши потрібну групу лівим кліком мишки (ЛКМ), побачимо графічні зображення блоків, які входять до неї. Правий клік миші (ПКМ) на зображенні блока викликає контекстне меню, через яке можна додати обраний блок до діаграми або викликати довідку за цим блоком. Додати вибраний блок до діаграми можна просто перетягнувши його мишею.

Зверніть увагу, що певний блок може бути присутнім у декількох палітрах. Це зроблено для спрощення побудови схеми моделювання (зменшується необхідність переходів між палітрами.



а) Перегляд палітри блоків б) Вікно редактора

Рисунок 2.3 – Графічний редактор Xcos

2.4 Основи моделювання в Xcos

Будь-яка діаграма **Xcos** містить два типи з'єднань: інформаційні (чорні) і керівні (червоні). По інформаційних з'єднаннях передаються сигнали даних, а по керівних – сигнали активації. Блоки так само можуть мати інформаційні та керівні входи і виходи. Як правило, інформаційні входи і виходи блоків розташовуються зліва і праворуч від зображення блока, а керівні – зверху і знизу.

Як основне джерела сигналів активації використовується лічильник часу ***SampleCLK***. Його особливість полягає в тому, що всі такі лічильники всередині однієї діаграми синхронізовані.

Якщо блок має керівний вхід, то він «спрацьовує» кожен раз, коли на нього надходить сигнал активації. Поведінка блока, що не має керівного входу, визначається його внутрішніми параметрами.

Блок може успадковувати сигнал активації від попереднього блока, тобто спрацьовувати у разі надходженні на його інформаційний вхід сигналу даних. Також, блок може бути активним завжди (наприклад, генератор гармонічного сигналу).

Вихідні значення константних блоків не змінюються ніколи, як часто б до них не зверталися.

Зміна параметрів блока

Подвійний клік ЛКМ на блоці у вікні графічного редактора викликає

вікно параметрів. Це вікно дозволяє змінювати параметри блока (якщо блок допускає зміну параметрів).

Час моделювання

Для встановлення часу моделювання вибрати пункт ***Моделювання → Налаштування*** в головному меню графічного редактора (рис. 2.4) і встановити параметр ***Загальний час інтегрування*** таким, що дорівнює потрібному значенню. Найчастіше встановлюють 30, оскільки такий параметр відповідає кількості точок відображення графіків за замовчуванням, проте можна встановити інший час за умови узгодження відповідних параметрів блоків.

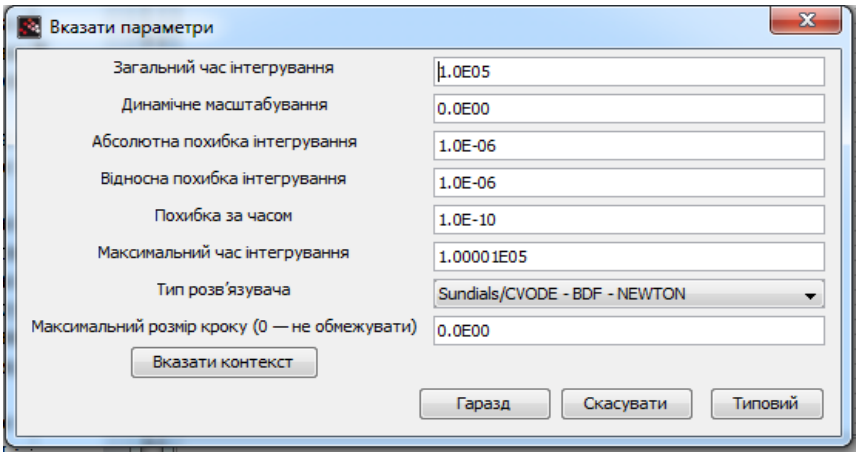


Рисунок 2.4 – Вікно встановлення загальних параметрів моделювання

Інший спосіб задання кінцевого часу моделювання: використати блок ***ENDBLK*** у палітрі ***Обробка «подій»***.

Розглянемо параметри моделювання, які користувач має задати перш ніж почати моделювання. Зауважимо, що за традицією, яка походить від аналогових обчислювальних машин (АВМ), в яких більшість обчислювальних операцій виконувалась інтеграторами на основі операційних підсилювачів, у назвах параметрів часто використовується термін «інтегрування», який у більшості випадків потрібно розуміти як «обчислення» або «моделювання».

- ***Час закінчення роботи (Загальний час інтегрування***): цей час за замовчуванням становить 10^5 (100000 секунд), час початку роботи завжди дорівнює нулю. *Рекомендовано встановити значення 30-50.*
- ***Відносна похибка інтегрування*** задає помилку відносно величини кожного стану, тобто відсоток від величини. Наприклад, 10^{-6} означає, що точність обчисленого стану в межах 10^{-4}%. Абсолютне відхилення дає поняття про величину прийнятної помилки. Його величина за замовчуванням 10^{-4}.

- ***Максимальний розмір кроку***: максимальний крок задає найбільший крок інтегрування, який може вибрати вирішувач. Ця установка важлива тим, що перешкоджає вибору вирішувачем занадто великого кроку.

- ***Максимальний час інтегрування***: максимальний часовий інтервал для кожного виклику вирішувача. *Він має бути зменшений, якщо надходить повідомлення «**too many calls**» (занадто багато запитів).*

Масштаб часу (швидкість процесу)

Виділимо параметр налаштування, який відповідає за швидкість процесу моделювання.

Динамічне масштабування – за його допомогою можна забезпечити зручну для спостереження швидкість відтворення процесу на екрані. За замовчуванням для цього параметра встановлене значення 0, тобто максимальна швидкість моделювання. *Для наочності процесів рекомендується встановити 0.1.*

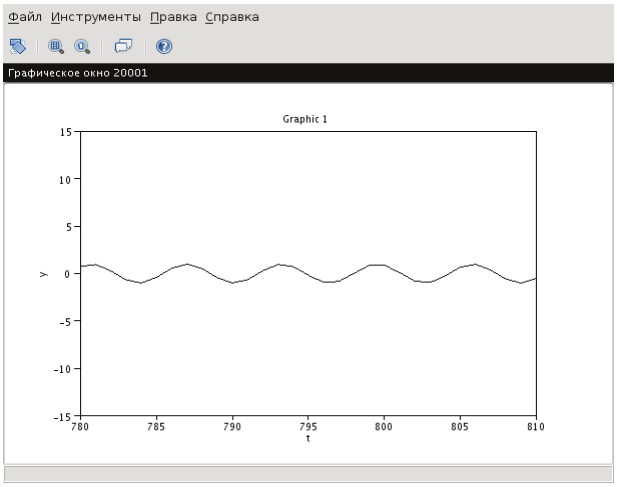
Збереження і завантаження

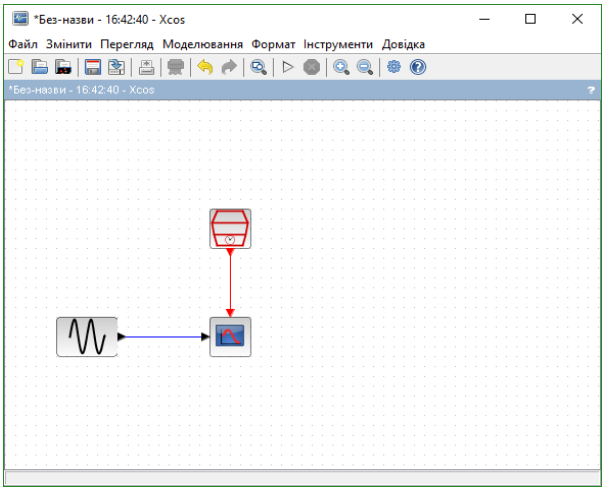
Зберігайте поточну діаграму частіше! Якщо у діаграмі робилися будь-які зміни, то вона може зберегтися неправильно. Тому відкрийте вікно створення нової діаграми: ***Файл → Створити діаграму***, скопіюйте наявну діаграму у нове вікно (зафіксуйте ЛКМ над лівим верхнім кутом діаграми і потягніть її донизу направо, потім натисніть Ctrl-Insert, перейдіть у нове вікно і натисніть Shift-Insert). Збережіть діаграму, вибравши ***Файл → Зберегти як***. Завжди зберігайте діаграми тільки в окремій теці, призначеній для конкретної задачі! У цій теці будуть зберігатися також і результати – експортовані графіки і дані.

Завантажити збережену діаграму можна через ***Файл → Відкрити*** або ***Файл → Недавні файли***.

2.5 Найпростіша діаграма

Розпочинати знайомство з пакетом **Xcos** варто з найпростішої діаграми (рис. 2.5).





а) Діаграма б) Результат моделювання

Рисунок 2.5 -Найпростіша діаграма

Виберіть палітру ***Джерела сигналів і впливів*** і перетягніть у вікно діаграми блоки 

(генератор синусоїди) і 

(лічильник часу). Потім перейдіть до палітри ***Пристрої реєстрації*** та додайте до діаграми блок 

(осцилограф).

З'єднайте вихід генератора з чорним входом осцилографа, а вихід лічильника з червоним входом осцилографа. Лічильник використовується для періодичної активації осцилографа із заданим часовим інтервалом. Сполучні лінії проводяться від виходу до входу (або навпаки) за натиснутої ЛКМ. Дозволені з’єднання підсвічуються зеленим. Для видалення сполучної лінії виділіть її та натисніть Delete.

Для створення відгалуження від лінії роблять на потрібному місці два кліки ЛКМ і тягнуть лінію в потрібне місце (рис. 2.6). На вже виділеному зв'язку можна зробити клік один раз. Для видалення ще не завершеної лінії зв'язку роблять один клік ПКМ.

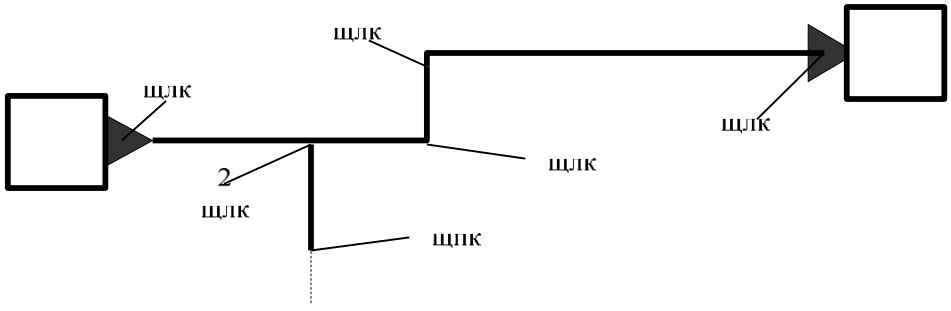


Рисунок 2.6 – Створення розгалуження ліній

Для запуску моделювання вибирають ***Моделювання → Виконати*** в головному меню редактора або просто натискають на відповідну кнопку в панелі інструментів. Для зупинення моделювання виберіть ***Моделювання → Завершити*** або ж скористайтеся відповідною кнопкою в панелі інструментів.

Встановіть частоту генератора ***2***, інтервал дискретизації – ***0.1*** і кінцевий час моделювання – ***30***.

Відкрийте вікно «***Встановлення параметрів***» для осцилографа на діаграмі (рис. 2.7). Змініть значення змінних ***Ymin*** і ***Ymax***, встановивши їх такими, що дорівнюють -2 і 2, відповідно.

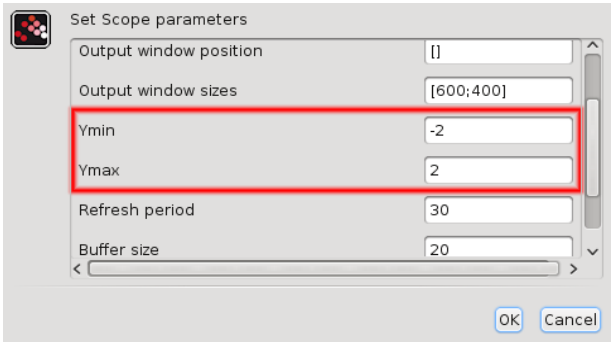


Рисунок 2.7 – Вікно зміни параметрів осцилографа

Запустіть моделювання.

Зупиніть моделювання і зверніть увагу на графік сигналу. Синусоїда на ньому виглядає «рваною». Щоб зробити її більш гладкою, треба зменшити інтервал дискретизації. Для цього зверніться до лічильника часу і змініть параметр ***Sampletime*** (***Інтервал дискретизації***), зробивши його таким, що дорівнює 0.1. Запустіть моделювання.

3 ВИКОРИСТАННЯ ОСНОВНИХ БЛОКІВ Xcos

3.1 Осцилографи

Для графічного відображення сигналу як функції часу в **Xcos** використовуються блоки **CSCOPE** і **CMSCOPE** з палітри ***Пристрої реєстрації***.

Блок **CSCOPE** має один вхід і відображає один сигнал (скаляр) або множину сигналів (вектор) в єдиній системі координат.

Додайте до діаграми, що складається з генератора, осцилографа і лічильника часу, блок **INTEGRAL_f** 

з палітри ***Неперервні динамічні системи*** і блок **MUXMUX** 

(мультиплексор) з палітри ***Маршрутизація сигналів***.

Діаграму моделі і результат моделювання наведено на рис. 3.1.

Мультиплексор в цьому прикладі об'єднує два скаляра на своїх входах в один вектор з двох елементів. Осцилограф відображає їх зміну у часі у вигляді двох графіків в одних осях.

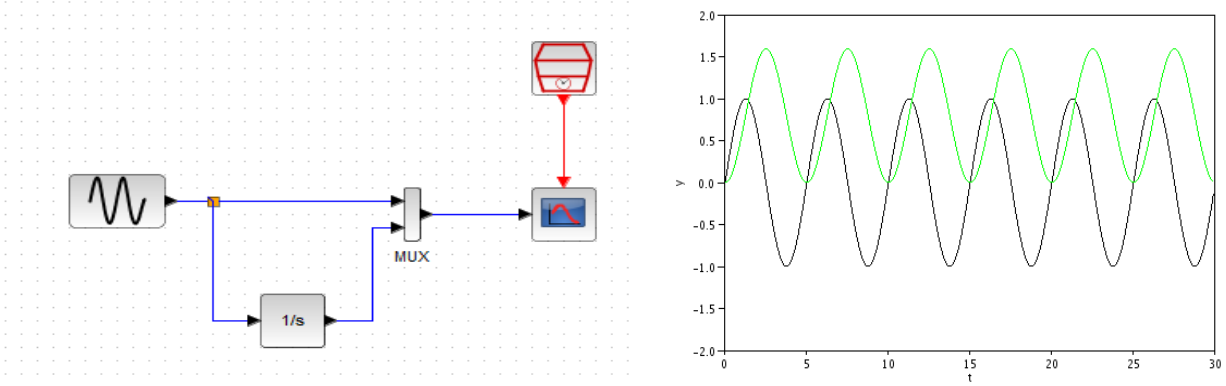


Рисунок 3.1 – Використання **ESCOPE**

Блок **CMSCOPE** має два і більше входів і відображає сигнали в окремих системах координат в єдиному графічному вікні.

Додайте до діаграми блок **CMSCOPE**  і блок **ABS_VALUE** 

(модуль) з палітри ***Математичні операції***. Діаграму моделі і результат моделювання наведено на рис. 3.2.

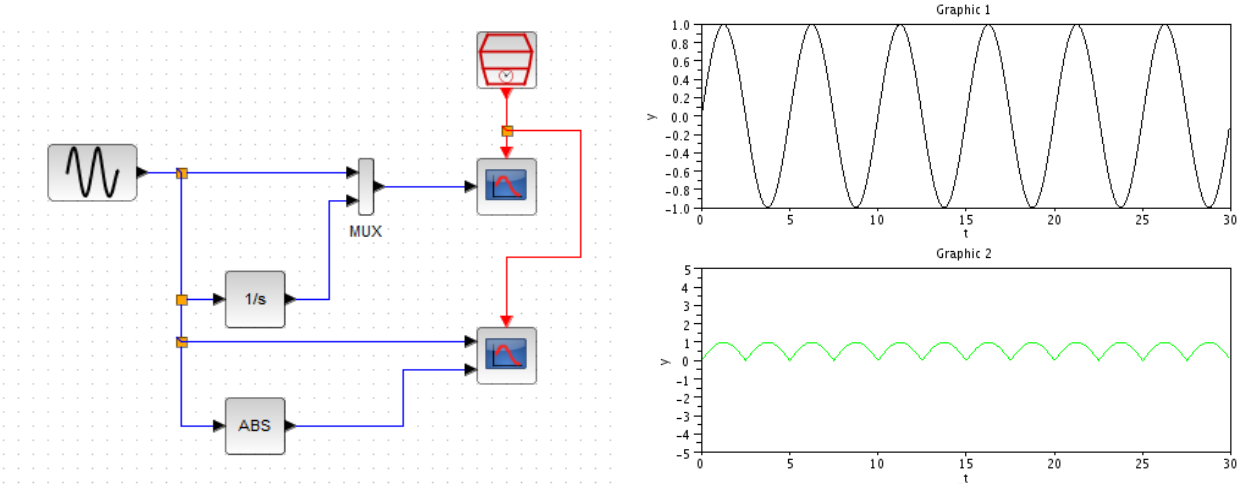


Рисунок 3.2 – Використання **CMSCOPE**

Межі по осі ***y*** задаються в параметрах блока змінними **Ymin vector** і **Ymax vector**. Перший елемент вектора відноситься до першого графіка, другий – до другого. Задайте межі (-2,2) для першого графіка і (0,2) для другого.

Ще одну діаграму моделі і результат моделювання наведено на рис. 3.3.

Інтервал оновлення осцилографа (розмір відображуваного проміжку осі часу ***t***) задається в параметрах блока змінною **Refreshperiod** (**інтервал оновлення**). Для блока **CSCOPE** це скаляр, для **CMSCOPE** – вектор, перший елемент якого відноситься до першого графіка, другий – до другого.

Блок **CMSCOPE**, аналогічно **CSCOPE**, відображає векторний вхід у вигляді множини сигналів в одній системі координат. Однак, на відміну від **CSCOPE**, для нього потрібно явно вказати розмірності кожного з входів. Розмірності входів задаються в параметрах блока змінною **Inputportsizes** (розмірності вхідних портів) вектором, перший елемент якого відноситься до першого графіка, другий – до другого.

Задайте розмірність першого входу 2. Підключіть до першого входу осцилографа сигнал з виходу мультиплексора. Запустіть моделювання.

Змінна **Inputportsizes** має ще одне важливе значення: Її розмірність визначає кількість входів осцилографа. Зміна розмірності **Inputportsizes** тягне за собою відповідну зміну розмірності змінних **Yminvector**, **Ymaxvector** і **Refreshperiod**, які задаються для кожного входу окремо.

Додайте третій елемент до **Inputportsizes**, який дорівнює 1. Встановіть для нового графіка межі по осі ***y*** і **Інтервал оновлення**. Підключіть до третього входу осцилографи сигнал з виходу генератора.

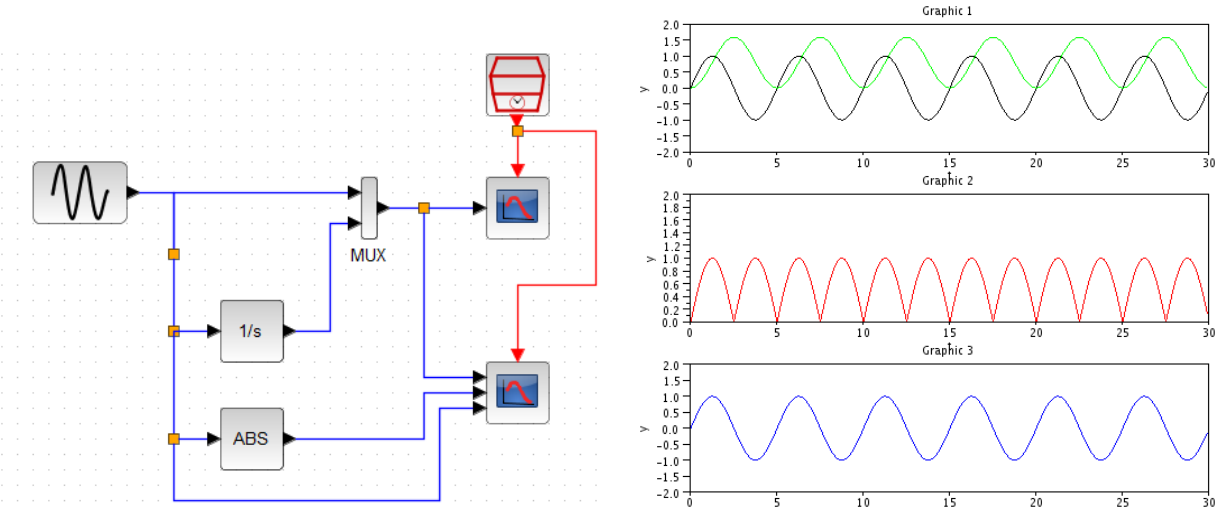


Рисунок 3.3 – Розширене використання **CMSCOPE**

Колір графіків функцій задається в параметрах блока змінної

Drawingcolors вектор, елементи якого відповідають номеру кольору в стандартній палітрі. Перший елемент визначає колір першої кривої, другий колір другої кривої і т. д. Якщо вказати значення кольору зі знаком мінус, то замість кривих на графіку будуть відображатися позначки.

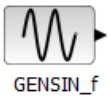
3.2 Джерела сигналів

Джерела сигналів знаходяться в палітрі ***Джерела сигналів і впливів***.

3.2.1 Константа

Блок **CONST_m**  використовується для формування постійної величини. Він має один параметр: **ConstantValue** – значення константи.

3.2.2 Генератор синусоїди

Блок **GENSIN_f** 

використовується для отримання сигналів синусоїдальної форми. Він є активним завжди.

Параметри блока:

- **Magnitude** – амплітуда;
- **Frequency**(rad/s) – частота(рад/с);
- **Phase**(rad) – фаза(рад).

Створіть нову діаграму (***Файл → Нова діаграма***). Додайте до неї два генератори синусоїди. Встановіть фазу першого генератора $\pi/2$ (**%pi/2,** де $\%pi$ – константа π у Scilab, отримаємо косинусоїду), а другого залиште «нуль» – синусоїда (рис. 3.4). Перегляньте сигнали з обох генераторів в одному вікні осцилографа.

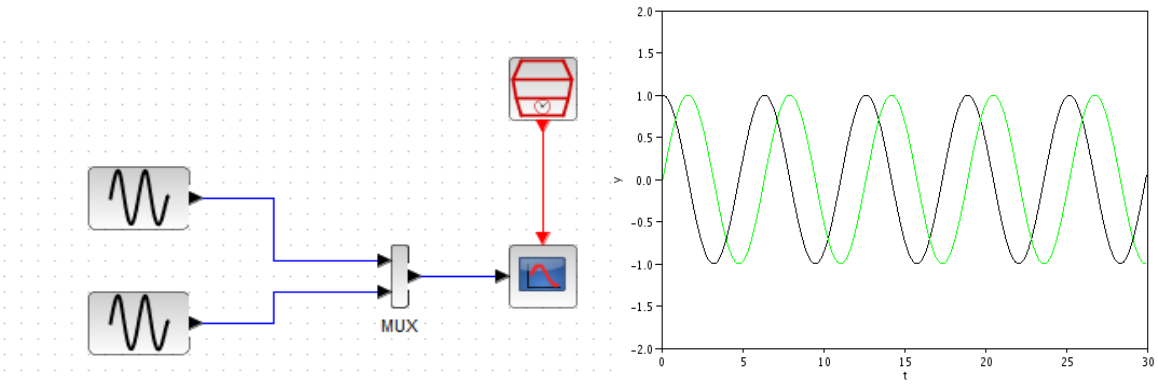


Рисунок 3.4 – Використання **GENSIN_f**

3.2.3 Генератор прямокутних імпульсів

Блок **GENSQR_f** 

використовується для отримання послідовності прямокутних імпульсів (відеоімпульсів). Блок має один керівний вхід і один інформаційний вихід.

Параметр **Amplitude** задає амплітуду імпульсів. Тривалість імпульсів визначається інтервалом надходження на керівний вхід сигналів активації.

Створіть нову діаграму (рис. 3.5). Додайте до діаграми генератор прямокутних імпульсів і лічильник часу. Встановіть інтервал дискретизації – 5. З'єднайте керівний вихід лічильника з керівним входом генератора. Перегляньте сигнал генератора на екрані осцилографа.

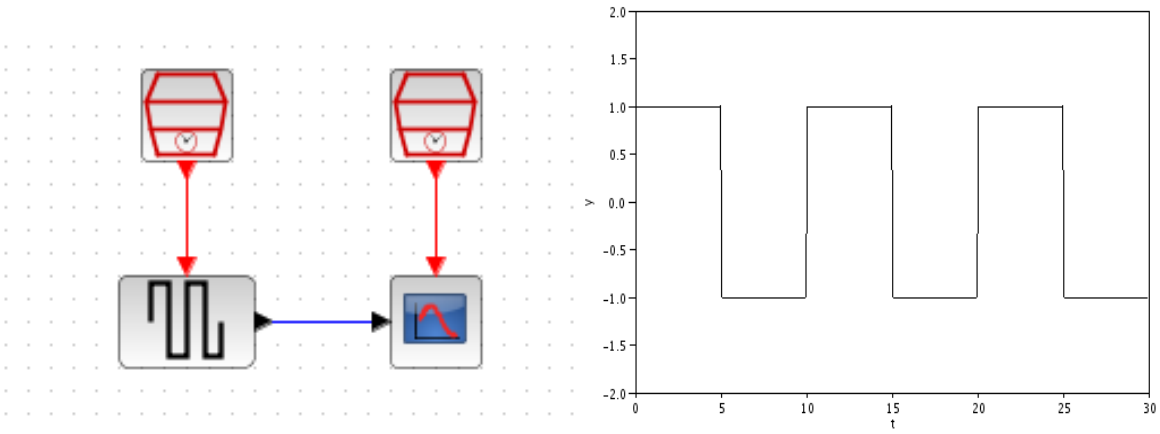
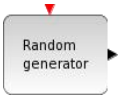


Рисунок 3.5 – Використання **GENSQR_f**

3.2.4 Генератор випадкових чисел

Блок **RAND_m** 

використовується для отримання випадкових чисел, розподілених за нормальним або рівномірним законом. Блок має один керівний вхід і один інформаційний вихід. Параметри блока:

• **Datatype** (тип вихідних даних): 1 дійсні числа, 2 комплексні;

• **Flag**: прапор, який визначає вид закону розподілу: 0 – рівномірний, 1 – нормальний (гаусівський);

• **A** і **B**: для рівномірного розподілу величина **A** визначає мінімальне значення, а величина **A** + **B** – максимальне. Для нормального розподілу **A** визначає математичне сподівання, а **B** – середнє квадратичне відхилення (СКВ).

• **SEED**: числа, використовувані для ініціалізації машинного генератора псевдовипадкових чисел. Перше значення відноситься до дійсної, а друге – до уявної частини вихідного сигналу. Два генератори з однаковим параметром **SEED** видаватимуть два ідентичних псевдовипадкових сигнали. Також під час повторних запусків генератора з однаковим **SEED** будуть генеруватися однакові послідовності чисел.

Створіть діаграму і додайте до неї генератор випадкових чисел. Встановіть параметри генератора таким чином, щоб отримати на виході випадкові числа, розподілені за нормальним законом з математичним сподіванням 0 і СКВ, що дорівнює 1. Виведіть сигнал генератора у вікно осцилографа з інтервалом дискретизації 0.1.

Випадковий процес на виході генератора (рис. 3.6) є білим гаусівським шумом.

За допомогою блока **RAND_m** можна отримати випадковий синхронний телеграфний сигнал (СТС), що імітує передане двійкове повідомлення.

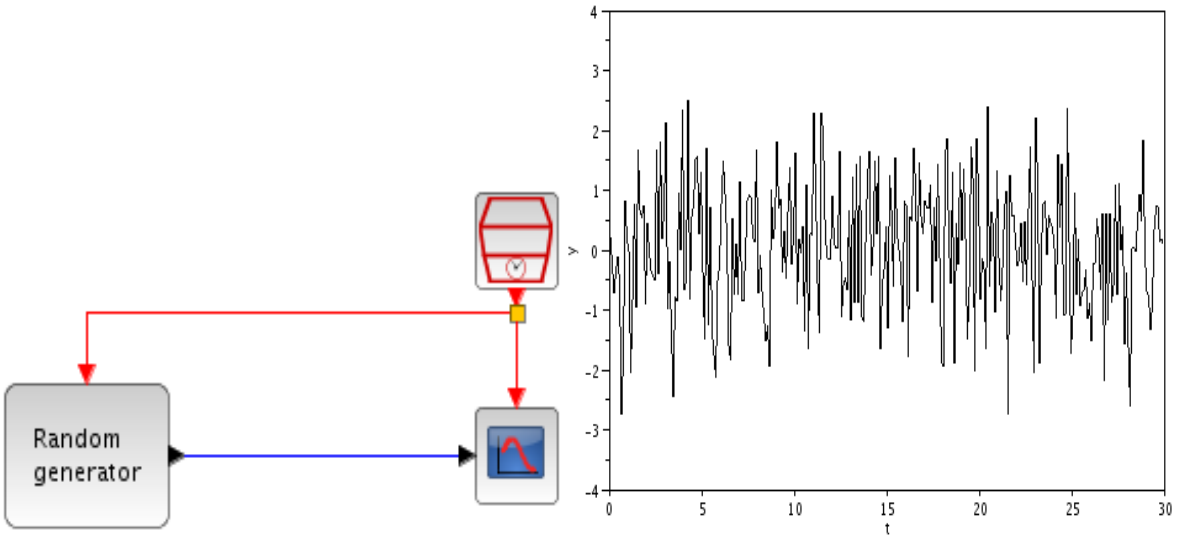


Рисунок 3.6 – Білий шум

Встановіть параметри генератора таким чином, щоб отримати числа, рівномірно розподілені в діапазоні (-1,1).

Додайте до виходу генератора блок **SIGNUM** 

з палітри ***Математичні операції***. На керівний вхід генератора подайте сигнал від лічильника часу з інтервалом 2.

Виведіть вихідний сигнал блока **SIGNUM** на екран осцилографа з інтервалом дискретизації 0.1 (рис. 3.7).

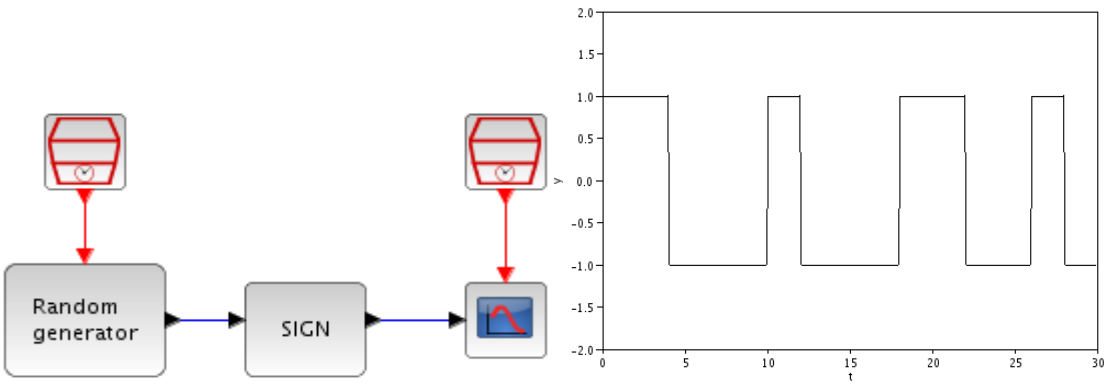


Рисунок 3.7 – Випадковий синхронний телеграфний сигнал

3.2.5 Функція вмикання

Блок **STEP_FUNCTION** 

генерує ступінчасту функцію (вмикання). Параметри блока:

- **Steptime** – момент часу увімкнення;
- **Initialvalue** – початкове значення;
- **Finalvalue** – кінцеве значення.

Створіть діаграму, яка містить генератор функції вмикання з параметрами: час вмикання – 10, початкове значення – 0, кінцеве значення – 1. Відобразіть сигнал з виходу генератора в вікні осцилографа (рис. 3.8).

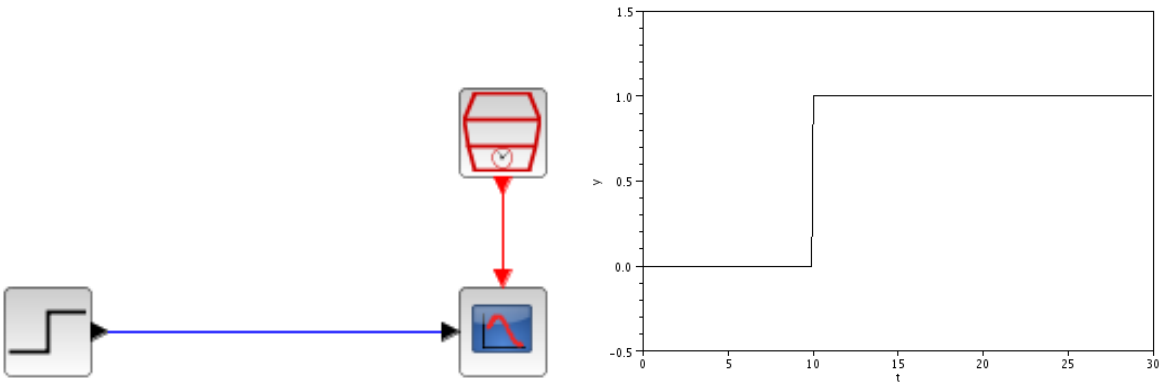


Рисунок 3.8 – Функція вмикання

Використовуючи пару генераторів функції вмикання і блок **BIGSOM_f** (суматора) з палітри ***Математичні операції***, можна отримати одиночний прямокутний імпульс. Додайте до діаграми ще один блок **STEP_FUNCTION** і блок **BIGSOM_f**. Встановіть час вмикання другого генератора – 15 і кінцеве значення – (-1). З'єднайте блоки, як показано на рис. 3.9.

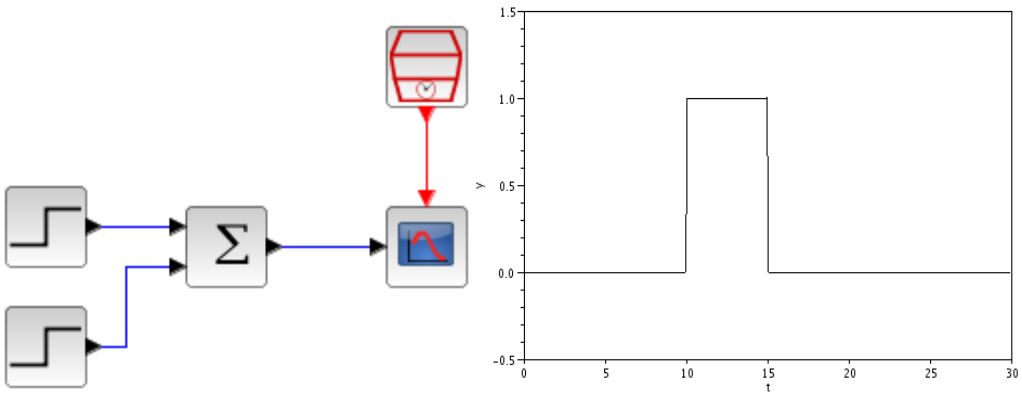


Рисунок 3.9 – Одиночний прямокутний імпульс

Блок **BIGSOM_f**  (суматор) має один параметр: **Inputportssigns / gain** – вектор вагових коефіцієнтів вхідних портів. Розмірність цього вектора визначає кількість портів. Вихідний сигнал суматора дорівнює зваженій сумі вхідних сигналів. За замовчуванням блок має два входи з вагами 1, тобто просто підсумовує вхідні сигнали.

Тривалість отриманого імпульсу визначається різницею часу вмикання генераторів. Зменшуючи цю різницю до як завгодно малого значення, будемо отримувати сигнал, що наближається до дельта-функції. Практично мінімальна різниця визначається найменшим часом дискретизації з усіх лічильників на діаграмі.

3.3 Маршрутизація сигналів

З палітри ***Маршрутизація сигналів*** нам знадобляться три основних блоки:

- **MUX**  – мультиплексор;
- **ISELECT_m**  – селектор;
- **NRMSOM_f**  – шина.

Мультиплексор MUX  об'єднує вхідні скаляри в один вихідний вектор. Кількість входів задається змінною Numberofinputports в параметрах блока.

Селектор ISELECT_m  використовується для розбиття вхідного

потоку на кілька вихідних. Кількість керівних входів дорівнює кількості інформаційних виходів. Кожен керівний вхід відповідає одному виходу: за надходження на перший керівний вхід сигналу активації вхідний потік виводиться на перший вихід, за надходження сигналу активації на другий керуючий вхід – виводиться на другий вихід. Параметри блока:

Datatype – тип даних: 1 – дійсні, 2 – комплексні і т. д. (відповідають стандартним типам даних Scilab);

Numberofoutputs – кількість виходів;

Initialconnectedoutput – номер початково підключеного виходу.

Нехай є синхронний тактовий сигнал (СТС) з тактовим інтервалом, що дорівнює 1. Необхідно розділити сигнал від джерела на два потоки, перший з яких містить посилки з парними, а другий – з непарними номерами.

Створіть нову діаграму. Складіть схему джерела СТС. Додайте до діаграми селектор і два лічильники часу. З'єднайте лічильники з керівними входами селектора. Встановіть параметр **Initialconnectedoutput** селектора 2. Задайте інтервали дискретизації лічильників на входах селектора такими, що дорівнюють 2 та встановіть затримку (**offset**) лічильника на першому вході селектора 1. Додайте до схеми осцилограф **CMSCOPE**. Виведіть у вікна осцилографа сигнал з виходу генератора СТС і сигнали з виходів селектора (рис. 3.10).

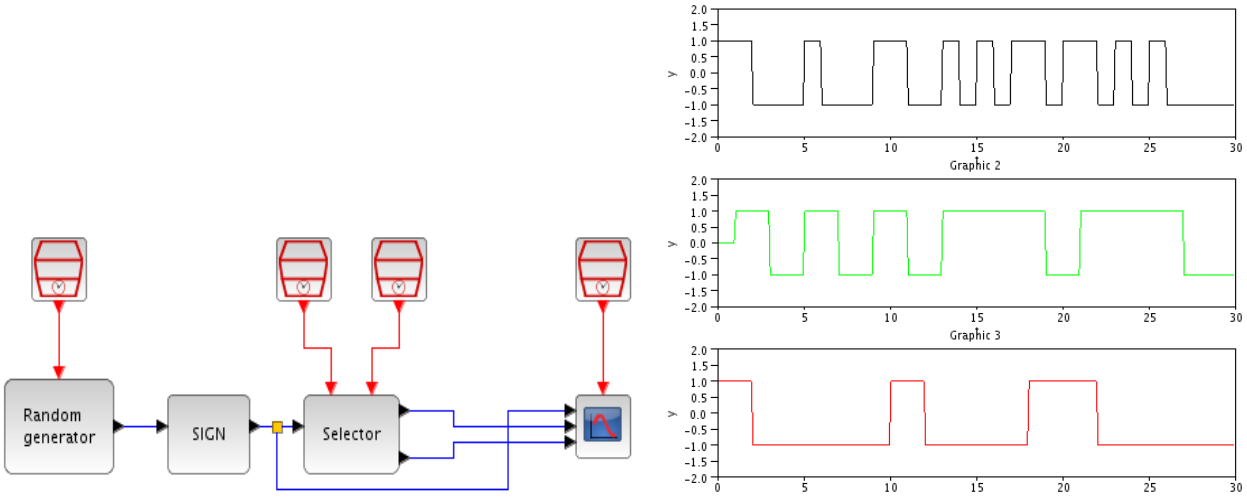


Рисунок 3.10 – Використання селектора

Шина NRMSOM_f  об'єднує велику кількість вхідних потоків в один вихідний. Кількість вхідних потоків задається змінною numberofinputs в параметрах блока. Об'єднайте потоки парних і непарних посилок з попереднього прикладу в один. Додайте до діаграми блок NRMSOM_f. Подайте на перший вхід блока сигнал з першого виходу селектора, на другий

– сигнал з другого виходу селектора. Додайте до діаграми осцилографи CMSCOPE. Виведіть у вікна осцилографів сигнал з виходу генератора СТС і сигнал з виходу шини (рис. 3.11).

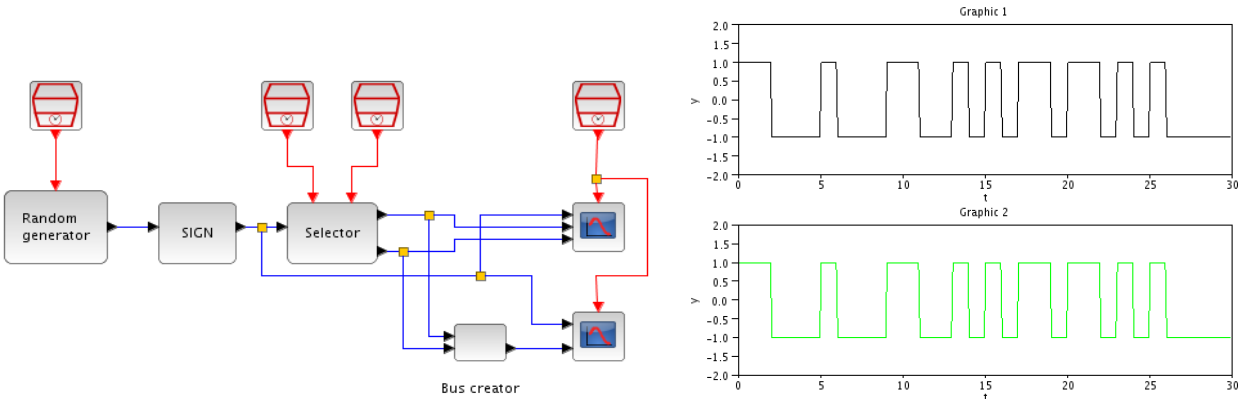


Рисунок 3.11 – Використання шини

При оновленні значення сигналу на будь-якому з входів шини воно перенаправляється на вихід.

3.4 Блок затримки

Блок **TIME_DELAY** 

з палітри ***Неперервні динамічні системи*** реалізує затримку вхідного сигналу в часі. Величина затримки визначається змінною **Delay** в параметрах блока. Змінна **initialinput** задає початкове значення вихідного сигналу, а змінна **Buffersize** – розмір буфера, в якому зберігаються відліки затриманого вхідного сигналу. Розмір буфера має бути не меншим, ніж число відліків сигналу за час затримки.

Створіть нову діаграму. Додайте до діаграми генератор синусоїди і блок затримки. Встановіть час затримки 5. Виведіть сигнал з генератора і затриманий сигнал в одному вікні осцилографа (рис. 3.12).

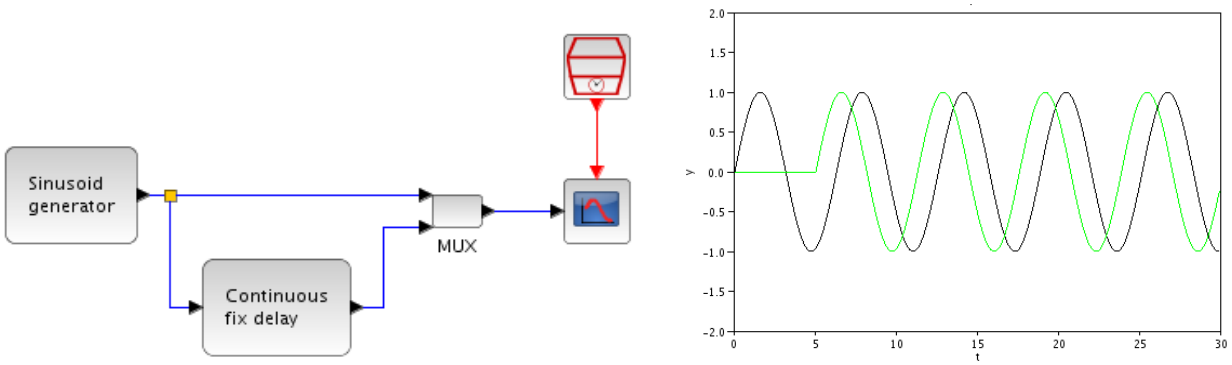


Рисунок 3.12 – Використання блока затримки

3.5 Перехід через нульовий рівень

Застосування блоків перетину нуля (блоки палітри ***Визначення переходів через нуль***) найчастіше є необхідним в гібридних моделях для непередбачуваних «подій». Наприклад, під час моделювання системи контролю рівня рідини в резервуарі в разі припинення, як тільки рівень перевищить певну величину. Також ці блоки визначають різкі зміни (скачки) сигналів. Ці явища можуть викликати помилки у разі використання методів ітерації. **Xcos**, завдяки механізму виявлення моментів переходу через нуль, може виявити такі «події».

Для звичайних випадків в **Xcos** використовується блок перетину нуля «**Zcross»**. У деяких випадках, на додаток до встановлення факту перетину, необхідно знати напрямок перетину. З цією метою в Xcos є два інших блоки: блок «**+ to –**» і блок «**– to +**», які фіксують не тільки факт перетину, але і враховують напрямок цього перетину (рис. 3.13).

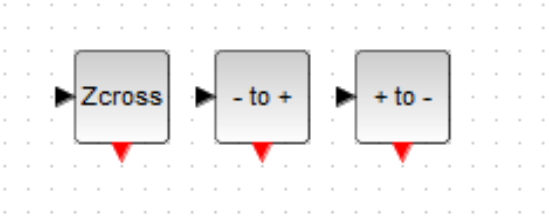


Рисунок 3.13 – Блоки перетину нуля

Покажемо простий приклад діаграми **Xcos** щоб проілюструвати використання блоків нульового перетину.

Вище розглядалися годинники активації **EventClock**, які генерували послідовність рівномірно розташованих в часі «подій». Вони були використані, щоб активізувати **SCOPE**. Відкриємо порожнє вікно Xcos. Створимо таку модель, як на рис. 3.14.

Блок «**+ to –**» знаходимо в палітрі ***Визначення переходів через нуль***, **S/H**-блок (**sampleandhold**, збереження) – в палітрі ***Дискретні динамічні системи*** і **cos**-блок – в палітрі ***Математичні операції***. Відзначимо, що блок **CMSCOPE** має 3 входи. Кількість входів **CMSCOPE** є параметром блока; його потрібно встановити перш, ніж входи будуть приєднані.

Блок «**+ to –**» генерує «подію» кожен раз, коли вхідний сигнал перетинає нульовий рівень, змінюючись від плюса до мінуса. Вхідний сигнал має бути неперервною функцією часу. Ці «події» активізують блок **S/H**, який копіює вхідний сигнал на свій вихід. Сигнал на виході не змінюється до наступної активації блока.

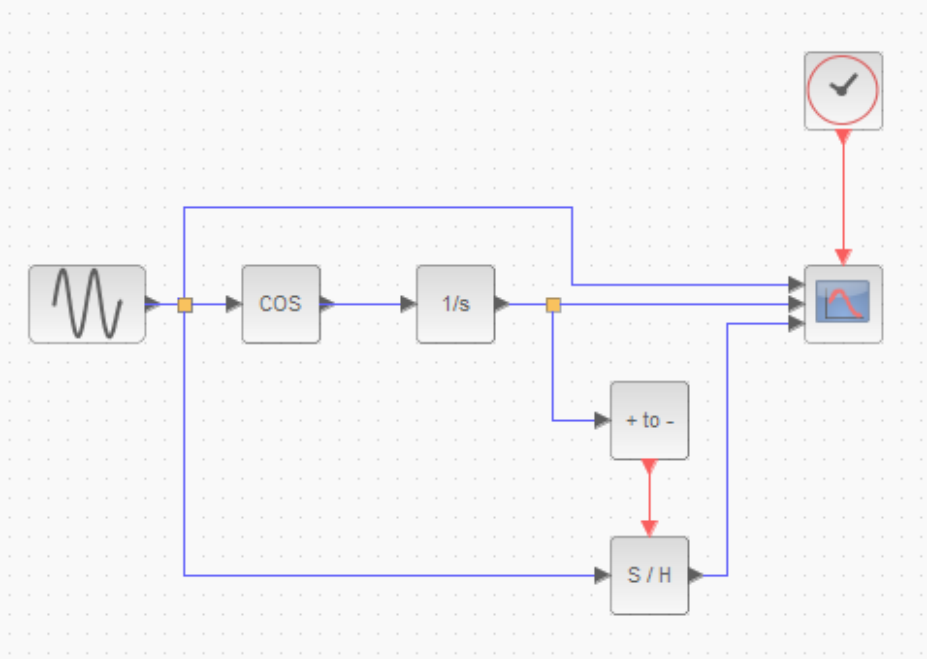


Рисунок 3.14 – Приклад моделі з Event Clock

Результат рахунку показано на рис. 3.15.

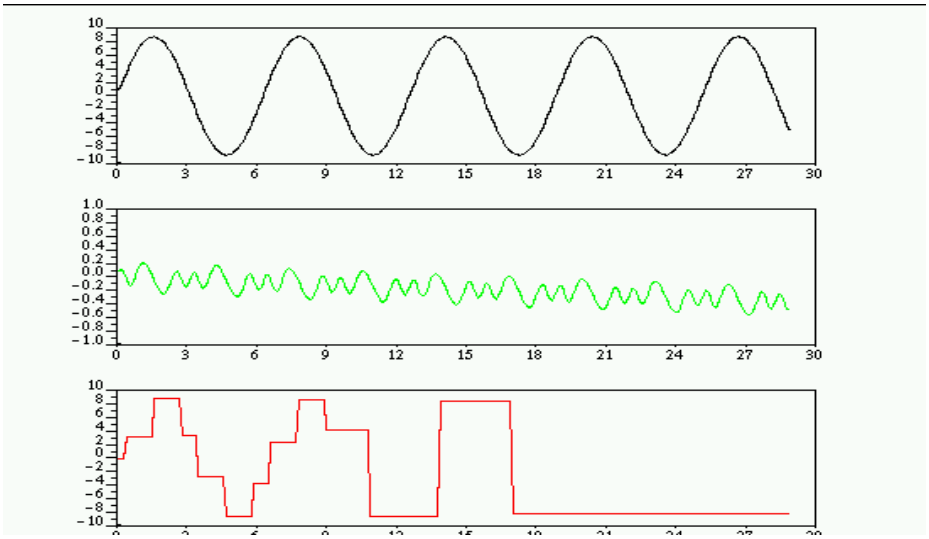


Рисунок 3.15 – Результати моделювання

Параметр амплітуди блока генератора **GENSIN_f** встановлено 8. «Події» з виходу блока перетину нульового рівня були використані для генерації дискретних сигналів. Ці дискретні сигнали можуть бути використані для керування безперервними компонентами. Простий приклад наведено на рис. 3.16.

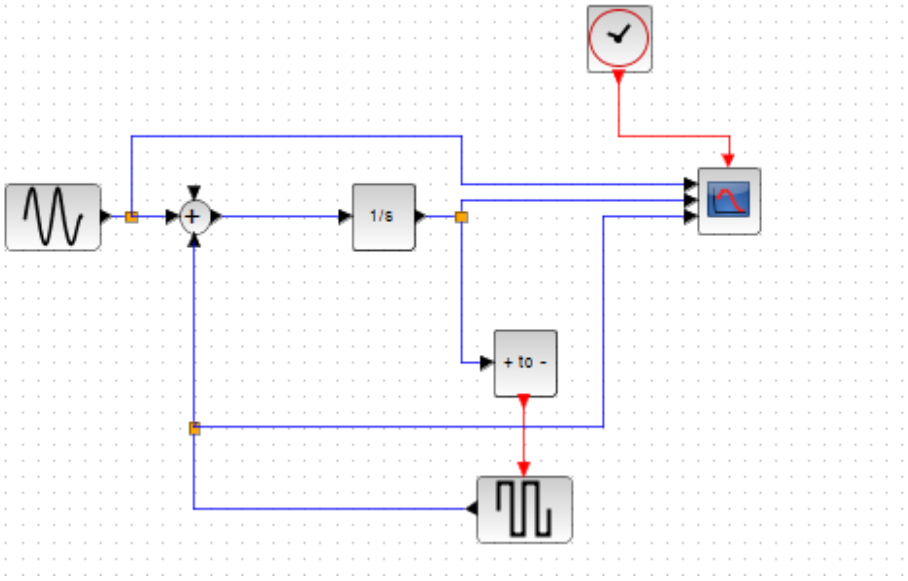


Рисунок 3.16 – Приклад моделі системи керування

Дискретний сигнал повертається в безперервну частину схеми. Відзначимо, що у генератора прямокутних хвиль вихідний порт зліва (в палітрі ***Джерела сигналів*** вихід у цього блока праворуч). Це зроблено за допомогою використання команди ***Віддзеркалити*** в меню ***Формат***. Цей блок видає на вихід 0 або 1, перемикання проводиться сигналом активації. Результат моделювання показано на рис. 3.17.

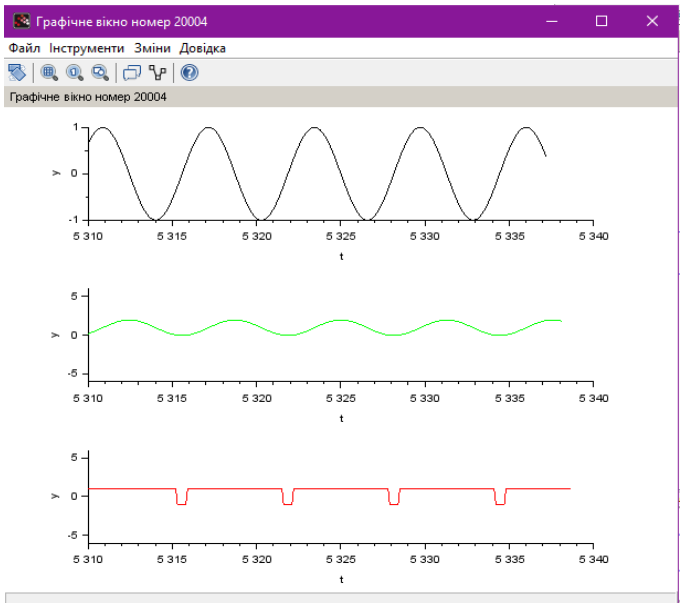


Рисунок 3.17 – Результати моделювання

3.6 Блоки з умовою

Основні функціональні блоки виконують задані обчислювальні функції, тоді як блоки з умовою управляють порядком обчислень в моделі. Ці блоки знаходяться у палітрі ***Обробка «подій»***. Вони можуть активізувати інші блоки в моделі **Xcos** при виконанні деяких умов. Найвживаніші блоки перевірки умов показані на рис. 3.18.

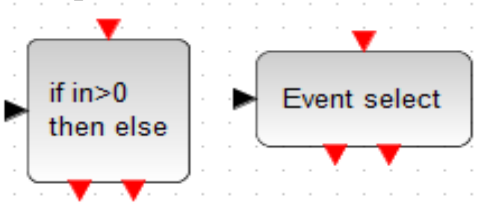


Рисунок 3.18 – Приклад блоків з умовою

Кожного разу, коли ці блоки отримують сигнал активації, вони в цей самий момент формують синхронний сигнал активації на вихідному порту залежно від рівня сигналу на інформаційному вході. Сигнали на виходах цих блоків взаємовиключні.

Щоб показати застосування блоку "**If-Then- Else**" розглянемо схему на рис. 3.19.

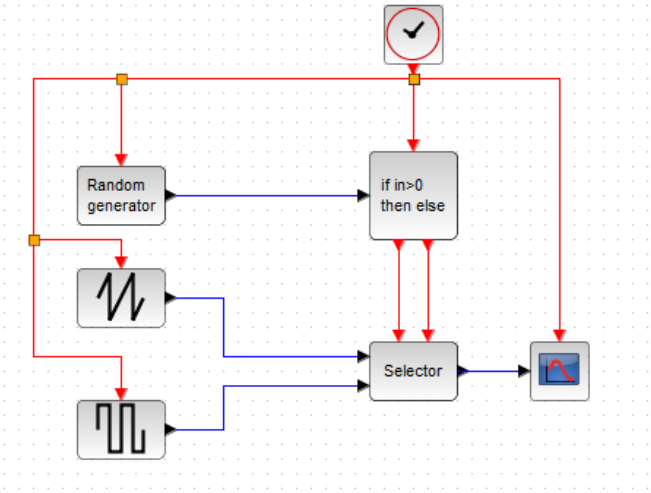


Рисунок 3.19 – Приклад моделі з використанням блока "**If-Then-Else**"

У блока **Selector** є два вхідних порти активації. Блок може бути активованим в певний момент тільки одним з них. Коли **Selector** активований за першим чи другим входом, вхідний сигнал приймається на вхід також з першого або другого інформаційного входу, відповідно. Таким чином, на виході **Selector** присутній пилкоподібний сигнал, якщо сигнал

Randomgenerator додатний, і прямокутний хвильової сигнал, якщо сигнал **Randomgenerator** від'ємний.

Якщо блок, «**If-Then-Else»** не має вхідного порту активації, він керує блоками безперервного сигналу, які отримують активацію по інших колах. В цьому випадку, чисельний вирішувач бачить тільки ті блоки, які знаходяться в активній гілці блока «**If-Then-Else**».

Блок «**Eventselect**» також вважається синхронним блоком. У нього є вбудований індикатор перетину нуля. Блок має один вхід і в будь-який момент активізований тільки один вихід.

3.7 Дискретні системи

Блок є дискретним, якщо він активізований «подіями» і між подіями генерує постійний сигнал. «Події» не обов'язково мають бути періодичними. Схема може бути сформована і виключно на дискретних блоках. Ці блоки знаходяться у палітрі **Дискретні динамічні системи.**

Розглянемо приклад на рис. 3.20.

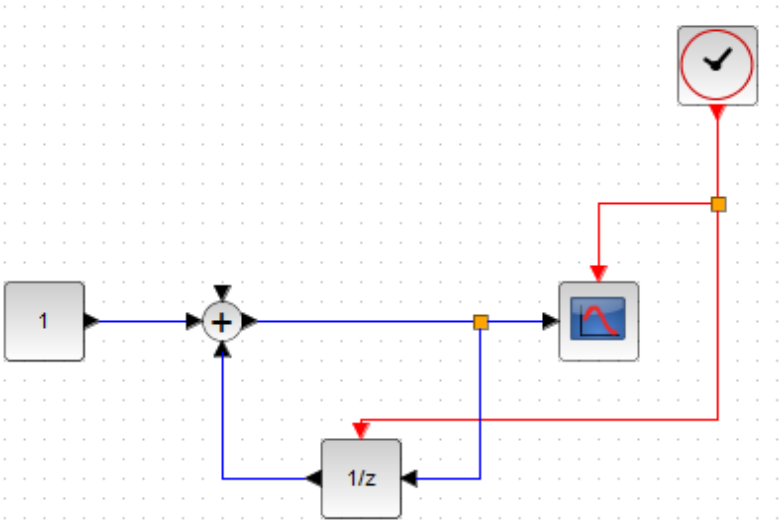


Рисунок 3.20 – Модель системи з дискретними блоками

Тут блок **1/z** (дискретна затримка) – регістр зсуву. Кожен раз, отримуючи сигнал активації, він відображає свій внутрішній стан на виході і змінює свій внутрішній стан відповідно до вхідного сигналу. Схема є лічильником. За кожним сигналом годинника стан збільшується на одиницю. В цьому випадку природно, що **Scope** відображає сигнал у вигляді точок, а не лінії. Це може бути зроблено за допомогою зміни параметрів **Scope**.

Результат моделювання для періоду годинника активації 2 секунди показано на рис. 3.21.

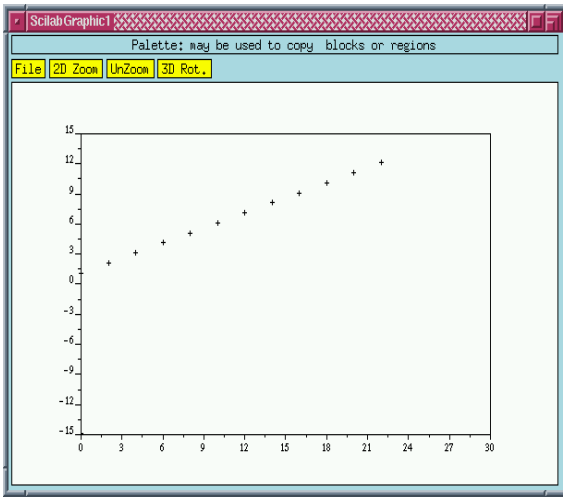


Рисунок 3.21 – Результати моделювання системи з дискретними блоками

Відзначимо, що блоки цієї схеми активізуються тільки під час «події» за винятком блока константи, який активізований постійно.

У загальному випадку блок без вхідних портів активації – або активний завжди, або успадковує час активації від вхідних сигналів. У цьому останньому випадку час активації є об'єднанням часу активації вхідних сигналів.

Блоки зі змінним в часі вихідним сигналом, навіть якщо у них є входи, не успадкують час активації; вони просто активні завжди.

Створимо нову схему, зображену на рис. 3.22.

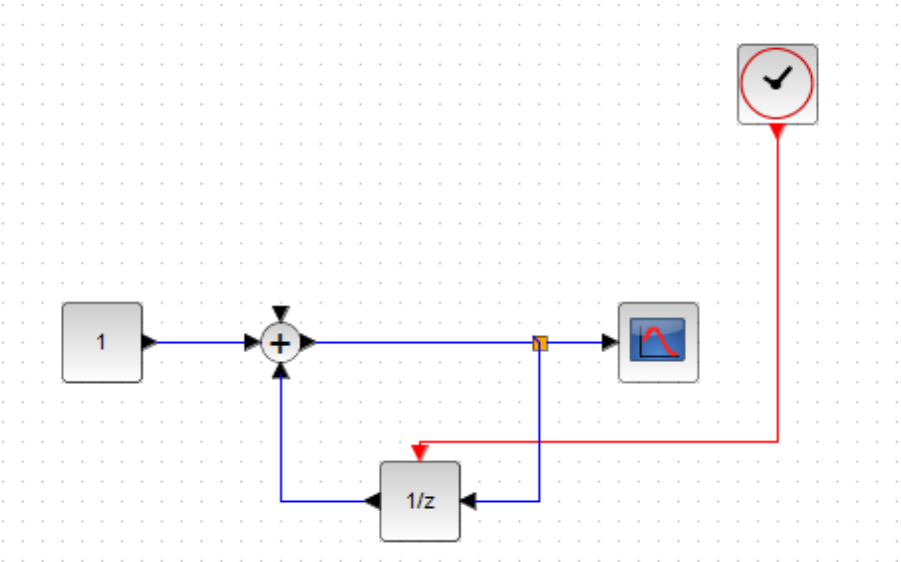


Рисунок 3.22 – Приклад моделі системи зі змінним в часі вихідним сигналом

Блок **Scope** може успадковувати час активації. Вхідний порт активації цього блока тому може бути видалений установленням відповідного блочного параметра. Результат в цьому випадку ідентичний попередньому.

Створимо нову схему (рис. 3.23). Для того, щоб побачити механізм успадкування в дії, в схему введено два незалежних джерела «подій». Період обох годинників дорівнює 2 с, але початковий час другого годинника встановлений 0.3 с (першого 0 с). Таким чином, у схемі є два незалежних генератори «подій».

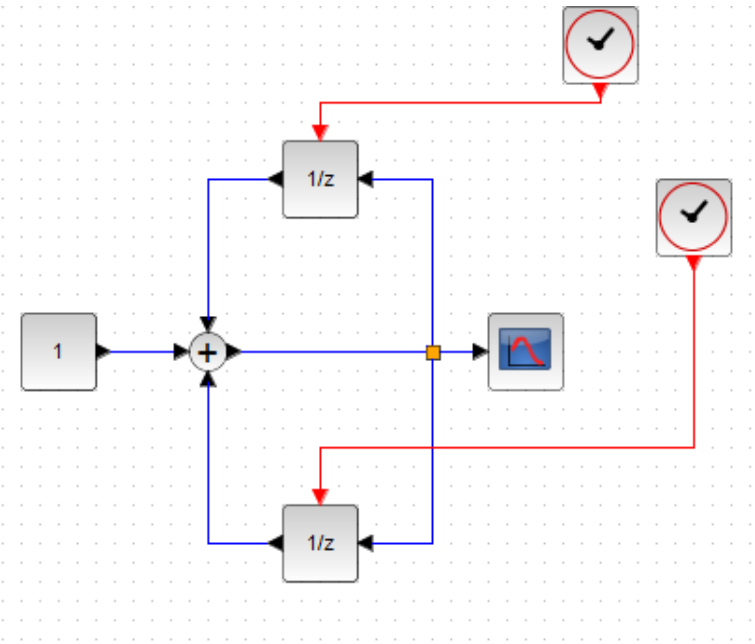


Рисунок 3.23 – Приклад моделі системи з двома незалежними джерелами «подій»

В цьому випадку немає ризику виникнення невизначеності, оскільки дві незалежних «події» ніколи не відбуваються в один і той самий час. Це обумовлено таким вибором параметрів годинників. У разі іншого вибору може виникнути невизначеність. Тому цей тип схем застосовувати не рекомендується.

Аналогічний результат може бути отриманий інакше, як це буде показано нижче.

Запустимо модель, отримаємо графік процесу (рис. 3.24).

Відзначимо, що час активації **Scope** є суміщенням часу активації двох регістрів. **Scope** активізується в моменти 2n і (2n + 0.3) с, n = 0, 1, 2 ...

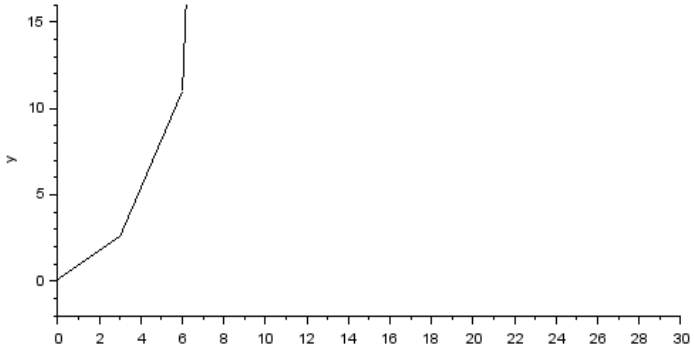


Рисунок 3.24 – Результати моделювання системи з двома незалежними джерелами «подій»

Розглянемо ще кілька простих схем. Як було сказано вище, якщо кожен блок активізується власним генератором «подій», наприклад, так, як це показано на рис. 3.25, то блоки несинхронні, навіть якщо обидва генератора «подій» налаштовані однаково.

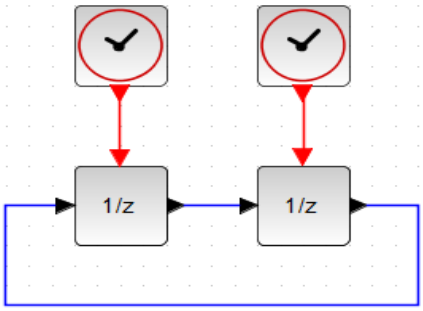


Рисунок 3.25 – Несинхронні блоки

Результат моделювання для цієї блок-схеми в Xcos непередбачуваний. Ця ж схема за використання **SampleCLK** виглядає так, як на рис. 3.26.

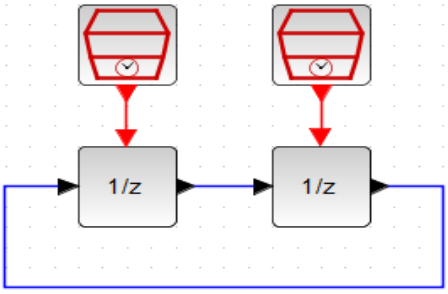


Рисунок 3.26 – Блоки **SampleCLK**

В цьому випадку компілятор **Xcos** починає виконувати необхідні

перетворення на блоках **SampleCLK** так, щоб знайти найповільніший годинник, який з урахуванням піддискретизації може замінити активізацію цих блоків.

У схемі, показаній на рисунку, обчислення тривіальні, оскільки обидва блоки мають ідентичні періоди. Еквівалентну схему наведено на рис. 3.27.

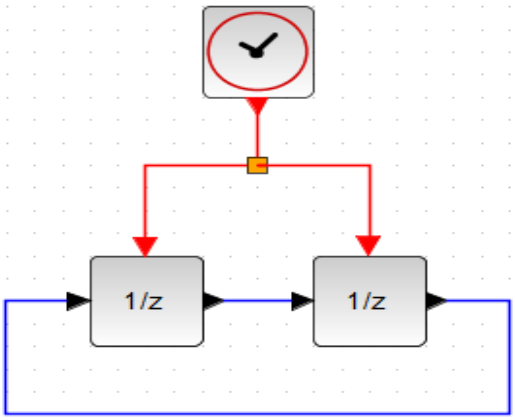


Рисунок 3.27 – Приклад блоків, які мають ідентичні періоди

У **Xcos** блок **1/z**, подібно до будь-якого іншого блока, може працювати без вхідного порту активації, успадковуючи її зі свого основного входу. Наприклад, як на рис. 3.28.

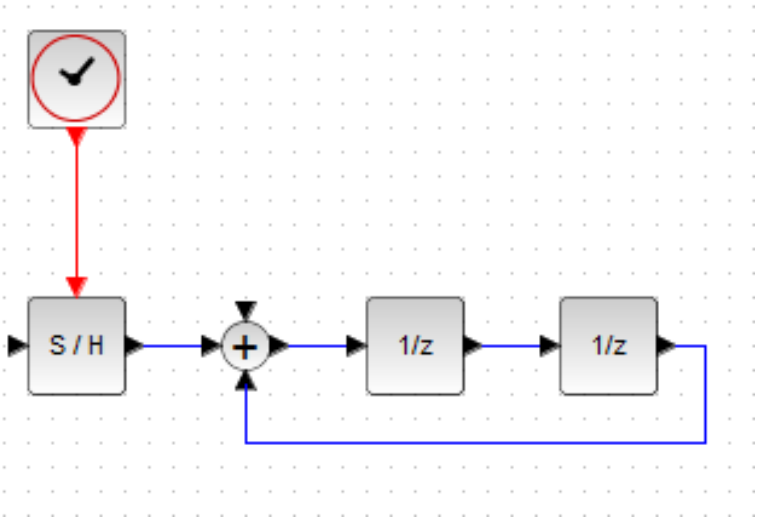


Рисунок 3.28 – Приклад блоків без вхідного порту активації

Наступна блок-схема (рис. 3.29) є узагальненням цього принципу

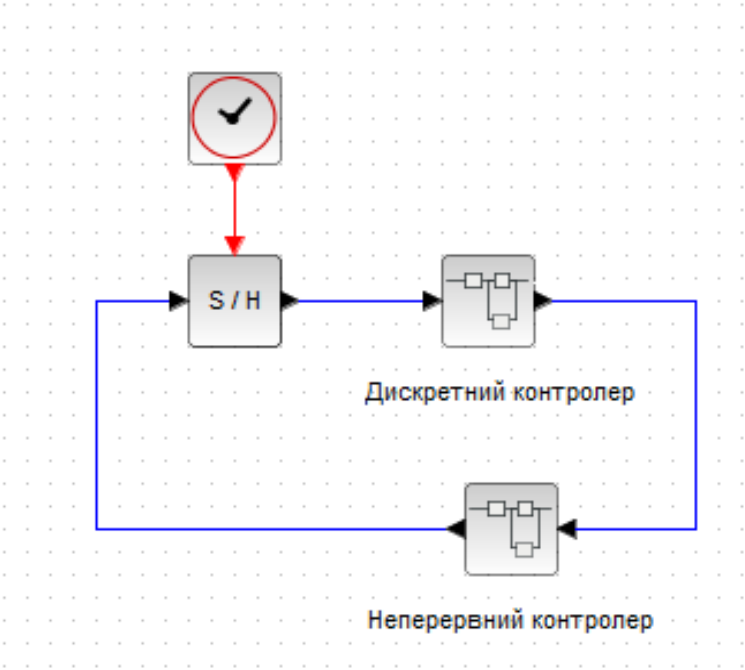


Рисунок 3.29 – Узагальнена схема блоків без вхідного порту активації

Неперервна частина не буде успадковувати ніякої активації, оскільки вона і так завжди активна. Крім того, потрібно зазначити, що **S/H** блок тут не потрібен, оскільки вихід дискретної частини і без цього зберігає вихідну величину до наступної активації.

3.8 Використання Synchro-блоків

Отже не всі блоки моделі Xcos мають активізуватися. Модель може мати два незалежних джерела активації, які керують різними блоками. Інша ситуація виникає, коли активація необхідна в багатьох точках схеми.

Коли два блоки активовані одним джерелом активації (наприклад, одним генератором «подій»), кажуть, що вони синхронізовані. У цьому випадку у них однакові часи активації, і якщо вихід кожного підключений до входу іншого, то компілятор знає, що блоки виконуються в правильному порядку. Два блоки, активовані двома незалежними годинами в один і той самий час, не синхронізовані, навіть з урахуванням того, що у цих годинників однакові параметри. Вони можуть бути активовані в будь-якому порядку.

З іншого боку, два сигнали активації можуть бути синхронними, але не бути одночасними. Наприклад, моменти однієї активація можуть бути підмножиною моментів іншої активації (активація з кратними частотами). В цьому випадку частина «подій», сформованих тактовим генератором з більш високою частотою, одночасна з «подіями» низькочастотного тактового

генератора. Але і в цьому випадку обидва сигнали (з високою і низькою частотами) мають бути породжені одним генератором, інакше вони будуть несинхронними. Отже необхідно переконатися, що низькочастотна активація отримана з високочастотної.

Розглянемо приклад (рис. 3.30). Тут блок синхронізації маршрутизує «події», що надійшли на вхідний порт активації між вихідними портами активації. Вибір вихідного порту активації залежить від величини сигналу на основному вході блока.

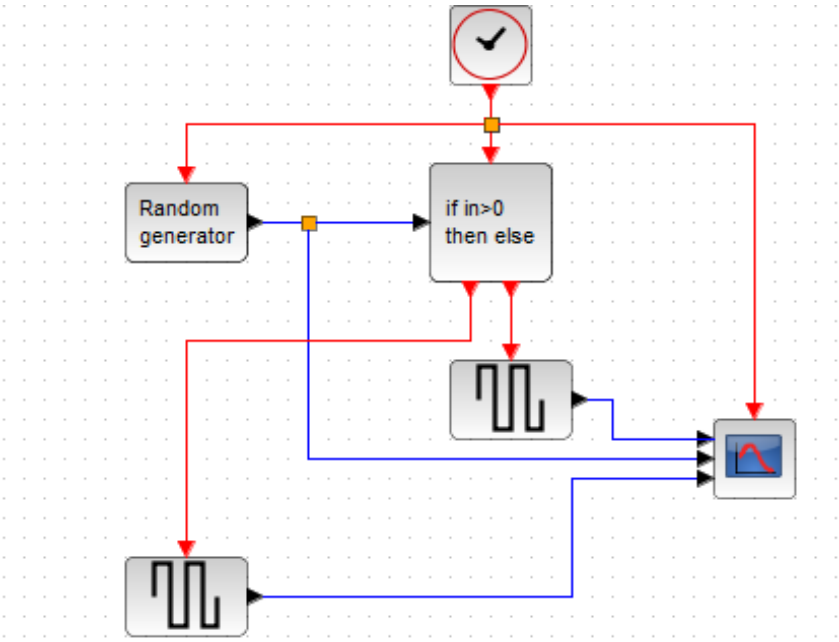


Рисунок 3.30 – Приклад системи з блоком синхронізації

Блок **If-then-else** направляє сигнал активації (в цьому випадку він надходить з годинника) на один зі своїх вихідних портів активації.

Якщо вхідний сигнал (тобто вихідний сигнал блока **Random generator**), додатний, сигнал активації від годинника надходить в перший вихідний порт активації, в іншому випадку – в другий.

Генератор шуму формує випадкову послідовність. Параметри блока визначають статистичні властивості випадкової змінної. Блок генератора прямокутних хвиль видає свій сигнал на вихід, змінюючи свій внутрішній стан в момент активації.

Виберемо опцію ***Gaussian*** для шумового генератора. Модифікуємо параметри **MScope**, щоб отримувати 3 входи.

Результат для періоду годинника 2с показано на рис. 3.31.

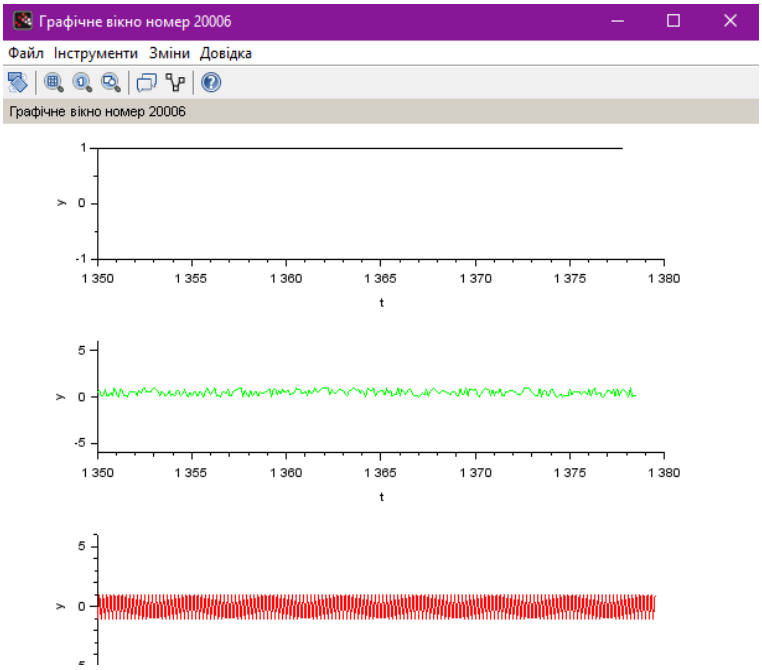


Рисунок 3.31 – Результат моделювання

Розглянемо тепер цикл «подій» з умовою. Наприклад, лічильник, який зупиняється за нульового результату. Тут є певні проблеми.

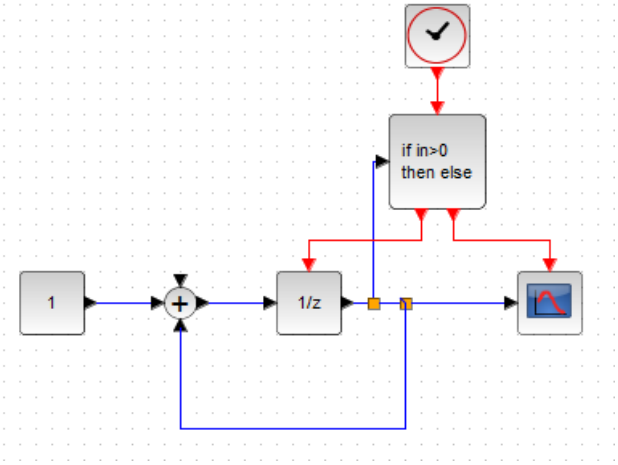


Рисунок 3.32 – Приклад циклу «подій» з умовою

Компілятор видає повідомлення **algebraic loop**. Ця модель є неприйнятною, оскільки для вирішення, на який порт блок **If-then-else** має направляти сигнал активації, що надходить, потрібно знати величину виходу **1/z**-блока. Але цей вихід залежить від надходження «події» (або ненадходження) з **If-then-else**-блока. Виникає невизначеність, яка називається алгебраїчним циклом (**algebraic loop**).

Може здатися, що **1/z**-блок – це просто деякою мірою блок затримки і він має перервати цикл. Але це не так. Причина в тому, що функції **1/z**-блока реалізуються таким чином.

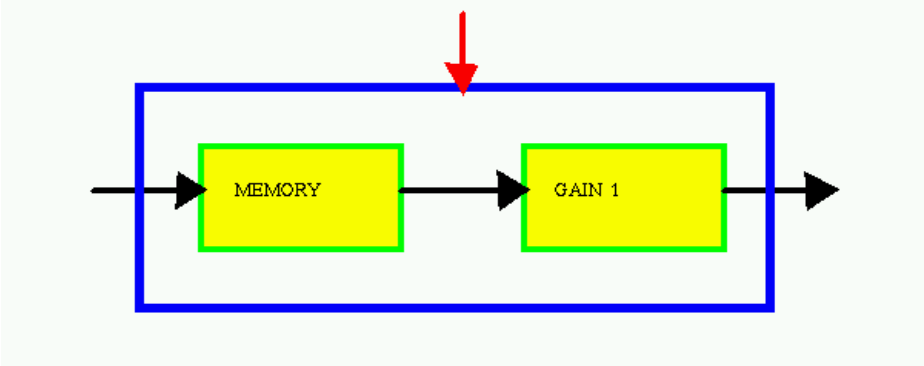


Рисунок 3.33 – Структура **1/z**-блока

Зі схеми видно, що коли **1/z**-блок активований, вміст пам'яті подається на вихід, а потім вхідний сигнал копіюється в пам'ять. Таким чином, немає безпосереднього зв'язку між входом і виходом, а є між вхідним сигналом активації і виходом. Але для того, щоб перервати цикл, нам потрібно мати вихід **MEMORY** доступним за межами блока. Це неможливо в цьому блоці. Тому в останніх версіях **Xcos** є блок **Counter.**

Блок **Event Clock** є суттєвим елементом будь-якої моделі **Xcos**. Це скомпільований суперблок.

Створимо таку схему (рис. 3.34). Блок **Delay** знаходиться в палітрі **Обробка «подій».**

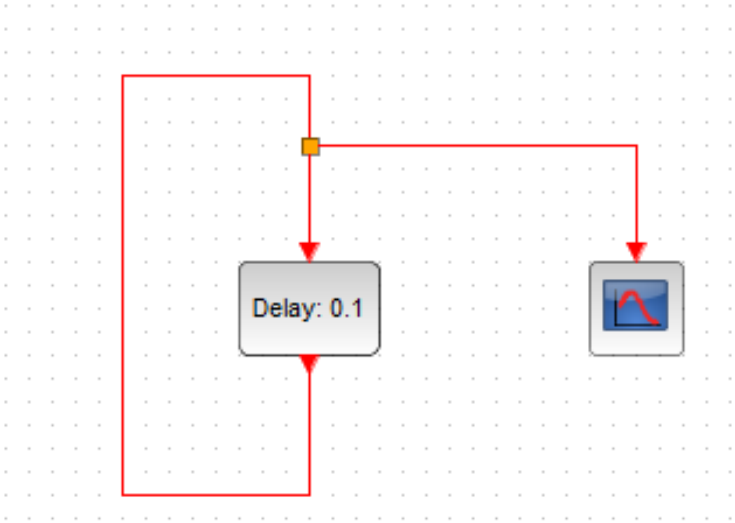


Рисунок 3.34 – Приклад системи з блоком **Delay**

Блок **Delay** має два параметри. Перший визначає затримку між «подіями», які надходять і виходять, другий – час початку генерації «подій». Якщо цей час від'ємний, то блок спочатку не генерує «події». Це означає, що модель залишається замороженою у часі. У багатьох ситуаціях установлення часу початку генерації не потрібно.

Встановимо параметри блока **Delay** (delay = 1 і initial firing time = 5) і запустимо модель. На рис. 3.35 видно, що зациклений блок затримки поводиться як **EventClock.**

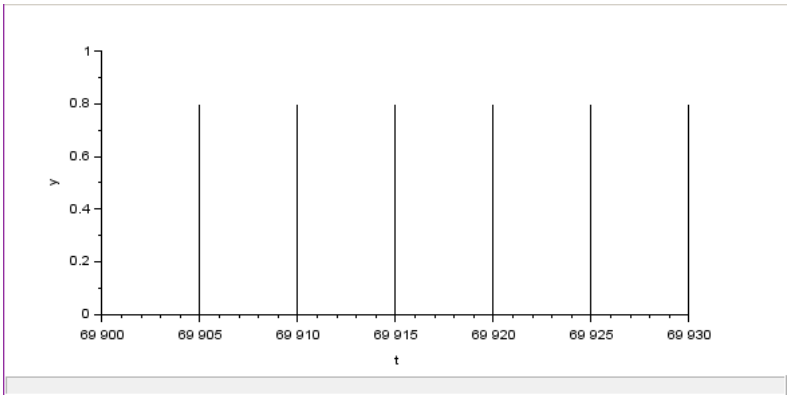


Рисунок 3.35 – Результат моделювання

3.9 Структуровані моделі (ієрархія і суперблоки)

Модель, що складається з дуже великої кількості блоків, незручна, її важко запам'ятати, робота з нею і її налагодження ускладнені. Для великих систем корисно використовувати засоби створення ієрархічної моделі.

Створимо модель, показану на рис. 3.36.

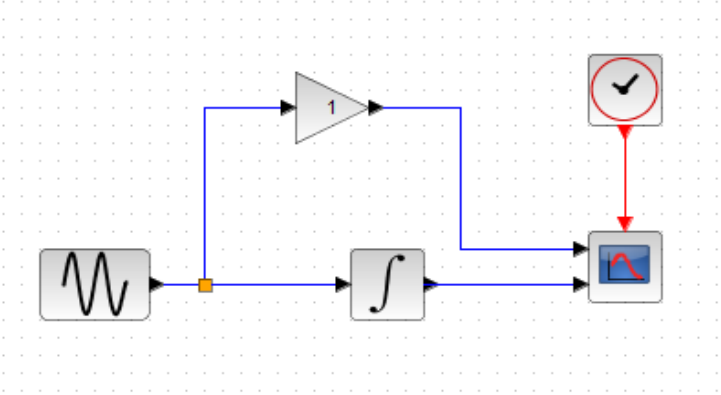
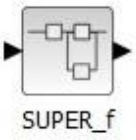
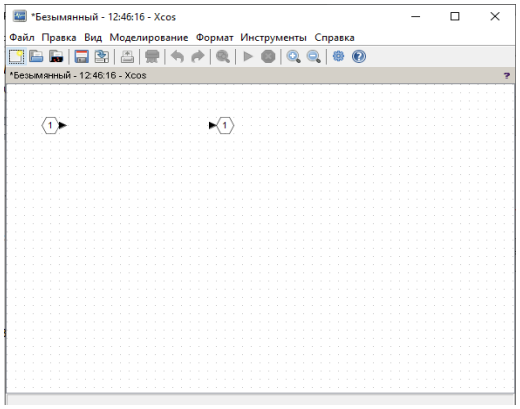


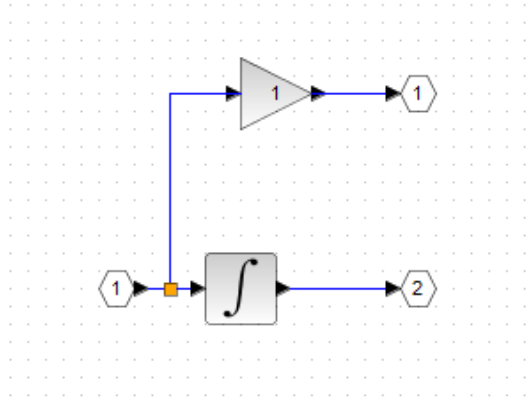
Рисунок 3.36 – Приклад простої моделі

Розмістимо інтегратор і підсилювач в суперблоці, як на рис. 3.37. Для цього можна перетягнути з палітри «**Функції користувача**» на робочу

область моделі блок 

. Відкрийте його (подвійний КЛК мишкою на ньому), з'явиться вікно, зображене на рис. 3.37, а).





а) б)

Рисунок 3.37 – Приклад суперблока

Виберіть у моделі область, яку потрібно помістити в суперблок, виріжте її – «*Правка-вирізати*», і вставте у суперблок – «*Правка-вставити*». Приєднайте входи і виходи блоків, які перенесені у суперблок, до портів  і  . Якщо портів не вистачає, перенесіть їх з відповідної палітри і встановіть для кожного параметр – номер порту. Отримаємо схему, зображену на рис. 3.37, б). Закрийте вікно суперблока. Тепер приєднайте входи і виходи суперблока до інших блоків моделі. Входи і виходи суперблока розташовані зверху донизу у послідовності нумерації, яка була їм надана під час створення суперблока. Для прикладу з рис. 3.36 отримаємо схему, зображену на рис. 3.38.

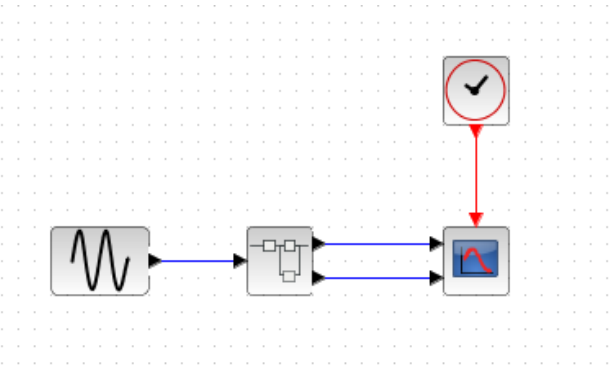


Рисунок 3.38 – Модель з суперблоком

Запустимо модель Xcos. Отримуємо такий результат.

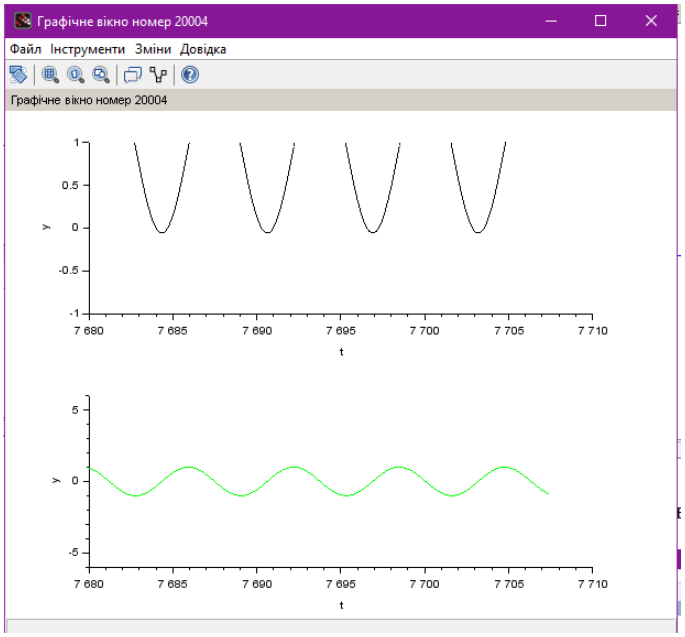


Рисунок 3.39 – Графіки роботи моделі з суперблоком

Створений суперблок має два вихідних сигнальних порти і один вхідний порт. Тепер суперблок може бути використаний подібно будь-якому іншому блоку.

3.10 Створення нових блоків Xcos

Для того, щоб створити новий блок, нам потрібні дві функції. По-перше, функція сполучення, яка визначає графічні властивості, параметри і под. та ім'я функції, яка виконує обчислення, необхідні при моделюванні. Функція сполучення повинна бути функцією **Scilab**, функція обчислювального блока може бути написана на **C**, **Fortran** або мовою **Scilab**.

Якщо розглядається блок простої дії (у нього немає стану) і він не має вхідних або вихідних портів активації, то він може бути реалізований з використанням **C** або **Fortran**-блока в палітрі **Функції користувача**.

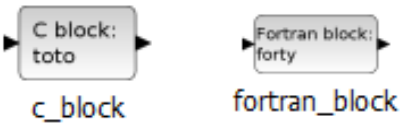


Рисунок 3.40 – Блоки додаткових функцій користувача

Блок **Fortran** допускає зовнішнє задання обчислювальної функції. Для того, щоб використовувати цей блок, необхідно мати компілятор **f2c.** Для MC Windows необхідний **Visual C ++**. Компілятор **f2c** є в пакеті **Scilab**.

В параметрах блока (рис. 3.41) можна задати розмірність вхідних та вихідних портів, параметри та ім'я обчислювальної функції.

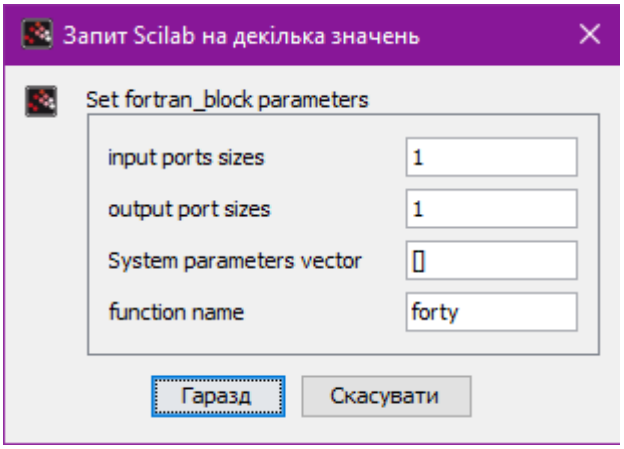


Рисунок 3.41 – Приклад відкриття блоку

Показаний на рисунку блок матиме три скалярних входи і один скалярний вихід. Іменем обчислювальної функції буде **forty**.

Тепер **Xcos** автоматично генерує шаблон функції forty (виклики і оголошення). У нього можна вписати необхідну підпрограму.

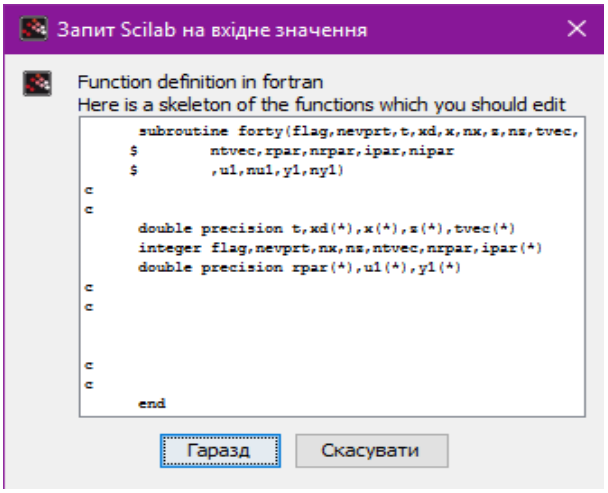


Рисунок 3.42 – Приклад шаблону функції forty

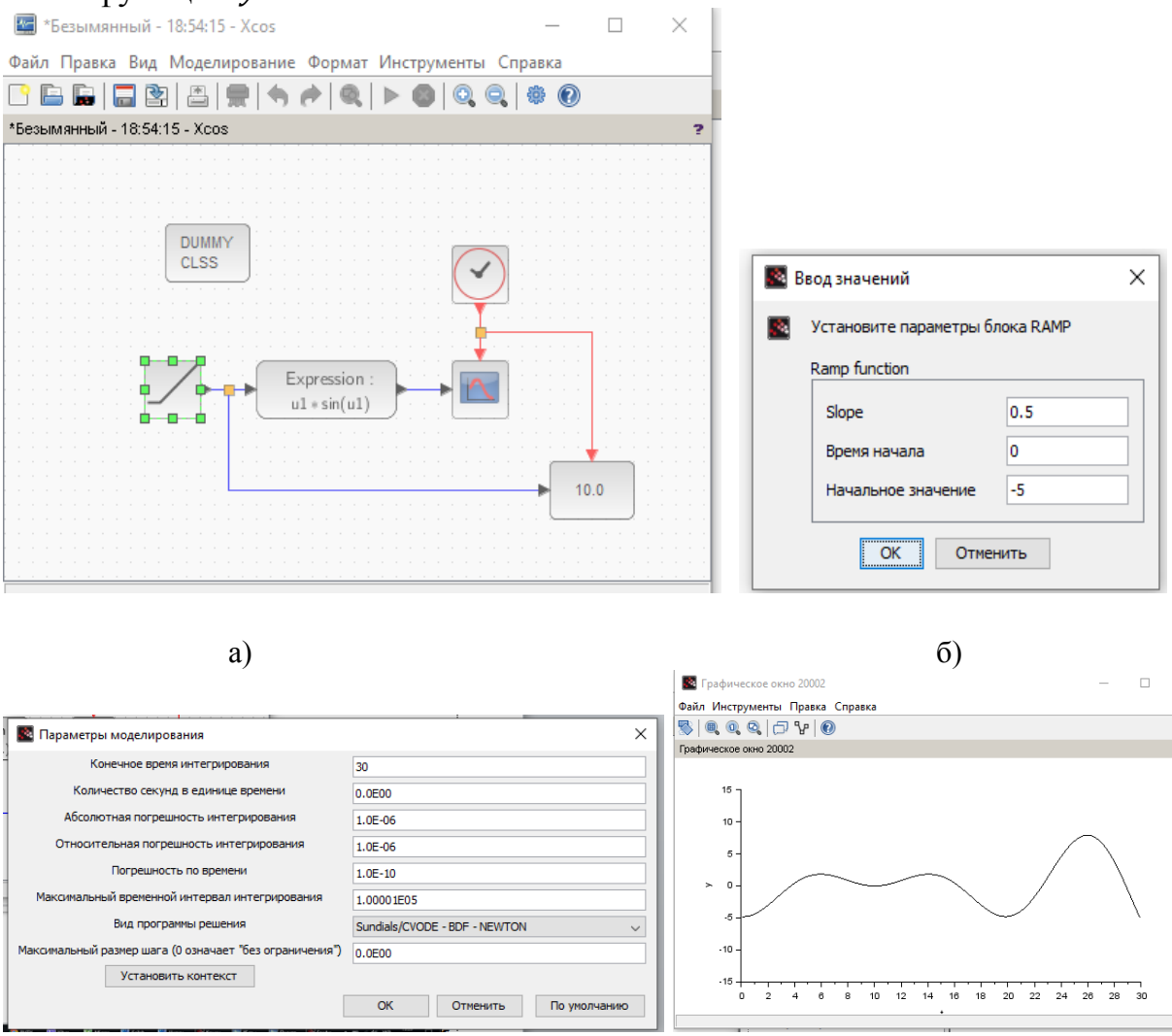
Блок може бути використаний під час створення моделей. Іконка блока може бути змінена через меню **Format**.

Блок мовою **C** створюється так само. Так само і для блока Scifunc, за винятком того, що використовуються вирази **Scifunc**. У разі **Scifunc** блок може мати як дискретні, так і неперервні стани.

4 ПРИКЛАДИ ТИПОВИХ ЗАДАЧ МОДЕЛЮВАННЯ

4.1 Побудова графіка функції

Для побудови графіка функції одного аргумента (2D-графік або графік пласкої кривої) створимо модель, зображену на рис. 4.1, а). Для прикладу взято функцію $y = x \cdot \sin x$.



а) б)

в) г)

Рисунок 4.1 – побудова 2D-графіка

Зміна аргументу x моделюється джерелом сигналу, який змінюється лінійно. Параметри налаштування блока (рис. 4.1, б) разом з налаштуванням загальних параметрів моделювання (рис. 4.1, в) визначають діапазон зміни аргументу – область визначення функції. У прикладі задано початкове

значення $x_{min} = -5$ («Начальное значение» на рис. 4.1, б), а кінцеве значення перевищує початкове на добуток часу моделювання («Конечное время интегрирования» на рис. 4.1, в) на приріст аргументу на кожному такті моделювання («Slope» на рис. 4.1, б): $x_{max} = x_{min} + 30 \cdot 0{,}5 = 10$. Таким чином, графік функції будується на інтервалі (-5, 10). Отриманий графік можна побачити на рис. 4.1, г).

За необхідності побудови декількох графіків у одних осях, вхідні дані об'єднують у вектор за допомогою блока MUX. Приклад моделі і результат показано на рис. 4.2.

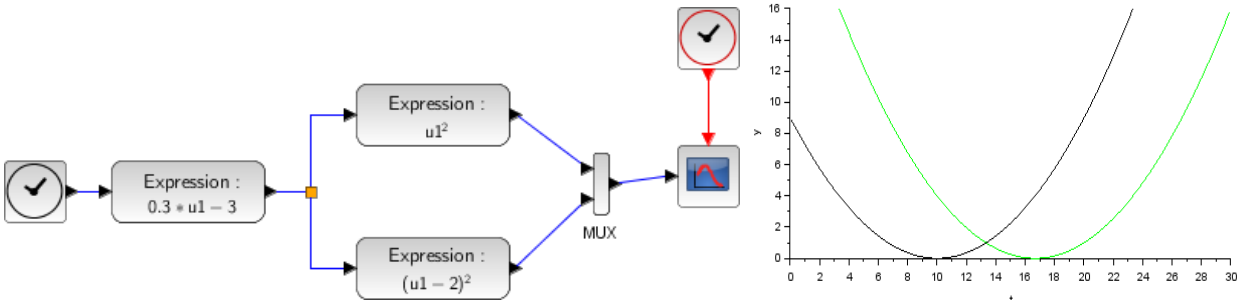


Рисунок 4.2 – Побудова декількох графіків у спільних осях

Якщо ж графіки повинні мати різні масштаби та/або різні діапазони, то їх будують у окремих системах координат за допомогою блоку CMSCOPE, як показано на рис. 4.3.

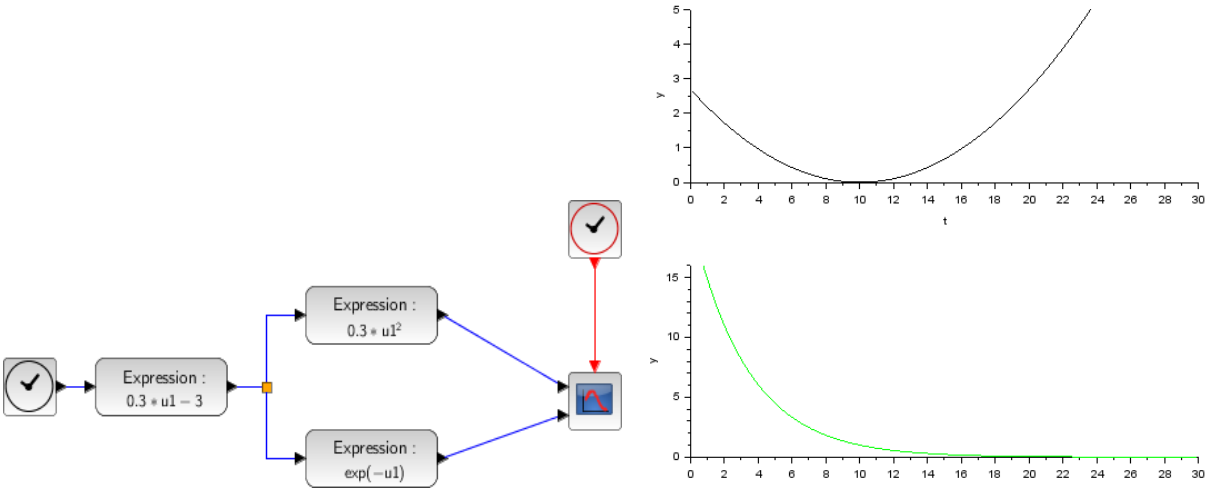


Рисунок 4.3 – Побудова синхронізованих графіків у різних осях

Для побудови двовимірних траєкторій, зокрема – фазових траєкторій, годографів систем керування, використовується блок CSCOPXY. Приклад такого графіка показано на рис. 4.4.

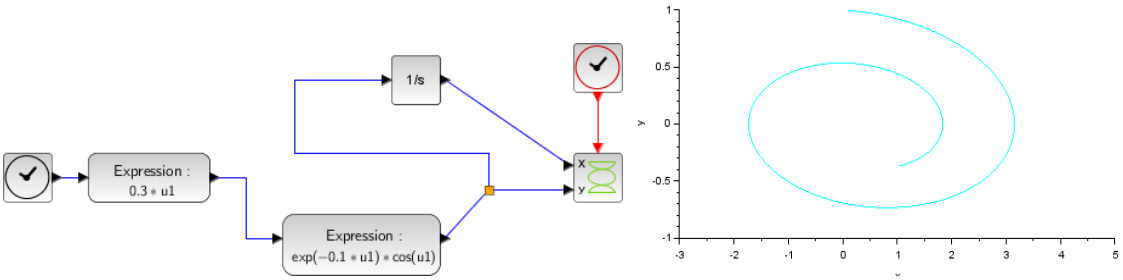
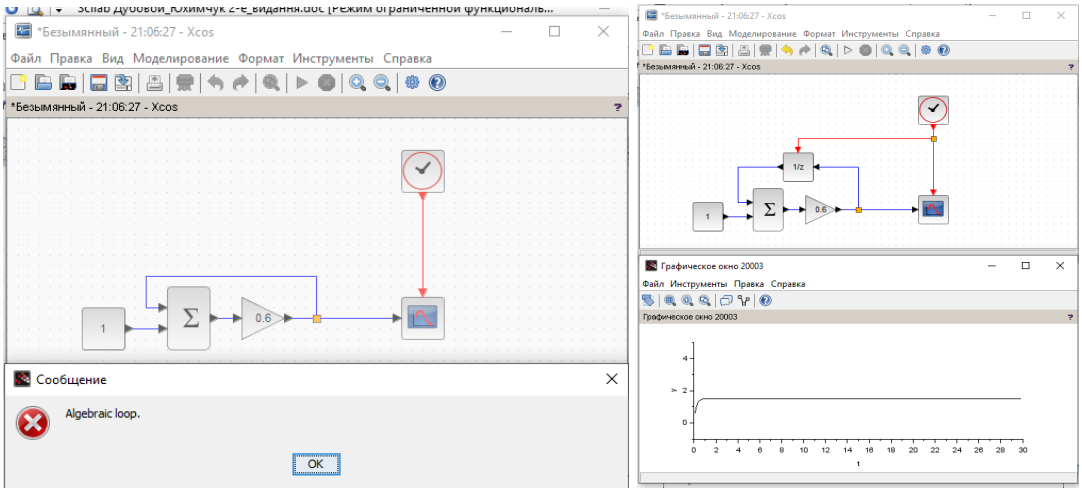


Рисунок 4.4 – Побудова графіка залежності між двома змінними

4.2 Особливості моделювання замкнених систем

Значна частина систем різного призначення мають замкнену структуру – різноманітні зворотні зв'язки. Під час моделювання замкнених систем потрібно пам'ятати, що блоки палітр Xcos є не фізичними об'єктами, а зображеннями певних програмних модулів, які виконуються цифровим процесором комп'ютера. Відповідно, замкнені системи моделюються циклічними програмами. Такі циклічні програми, як правило, містять рекурсію даних: наступний цикл використовує дані попереднього циклу. В імітаційній моделі Xcos для правильної послідовності використання даних необхідно явно відокремлювати у часі наступний цикл від попереднього. Для цього у замкнений цикл (як правило – у зворотний зв'язок) вставляють блок затримки на 1 такт моделювання 

(див. підрозділ 3.7). Приклад наведено на рис. 4.5. На рис. 4.5, а) модель без затримки, внаслідок чого система видала повідомлення про помилку «*Algebraic loop*». На рис. 4.5, б) додано затримку на такт і модель працює – є результат у вигляді графіка.



а) б)

Рисунок 4.5 – Уникнення помилки моделі замкненої системи за допомогою затримки

4.3 Розв’язування нелінійних алгебраїчних рівнянь

Нелінійні алгебраїчні рівняння переважно розв’язуються чисельними методами. Зокрема у Scilab/Xcos для цього найчастіше використовується ітераційний метод, оскільки імітаційне моделювання застосовується для динамічних процесів, а ітераційне розв’язання є динамічним процесом руху точки у напрямку кореня.

Відповідно до методу простої ітерації, розв’язок рівняння $f(x) = 0$ відшуковують за допомогою послідовного наближення $x_{i+1} = x_i - \lambda \cdot f(x_i)$, де λ – константа менша 1, знак якої дорівнює знаку похідної функції $f(x)$ поблизу шуканого кореня. Якщо функція має декілька коренів, то важливим є вибір початкового значення x_0, з якого починається рух до кореня, оскільки між двома коренями обов’язково є екстремум, де похідна дорівнює нулю і далі змінює знак. Ітераційна процедура не може «перескочити» через таку точку.

Приклад. Необхідно розв’язати рівняння

$$x^3 - 9x^2 + 23x - 15 = 0.$$

Відповідно до формули ітераційного розв’язування, будемо шукати корені за допомогою процедури

$$x_{i+1} = x_i - 0.01 \cdot sign(3x_i^2 - 18x_i + 23) \cdot (x_i^3 - 9x_i^2 + 23x_i - 15).$$

Процес розв’язання показано на рис. 4.6.

Для вибору початкових точок x_0 спочатку будуємо графік функції – вікно зверху зліва. Отриманий графік – вікно знизу зліва. На графіку видно, що функція має екстремуми у точках приблизно $x = 2$ і $x = 4$. Відповідно, будемо шукати корені, задавши початкові точки в інтервалах

$$x < 2;\ 2 < x < 4 \text{ і } x > 4.$$

Робимо схему для розв’язування – вікно зверху посередині. Початкове значення задаємо у параметрах дискретного інтегратора $1/z$, який також виконує функцію дискретної затримки на один такт, необхідної для правильного виконання циклічної процедури (див. підрозділ 4.2). Вікно встановлення параметра – посередині знизу.

На рисунку ми бачимо, що встановивши початкове значення (initial condition) 2.5, ми отримали корінь $x = 3$.

Аналогічно з початкового значення 0 ми отримаємо корінь 1, а з початкового значення 7 – корінь 5.

Отже, розв’язком рівняння будуть три корені (1; 3; 5).

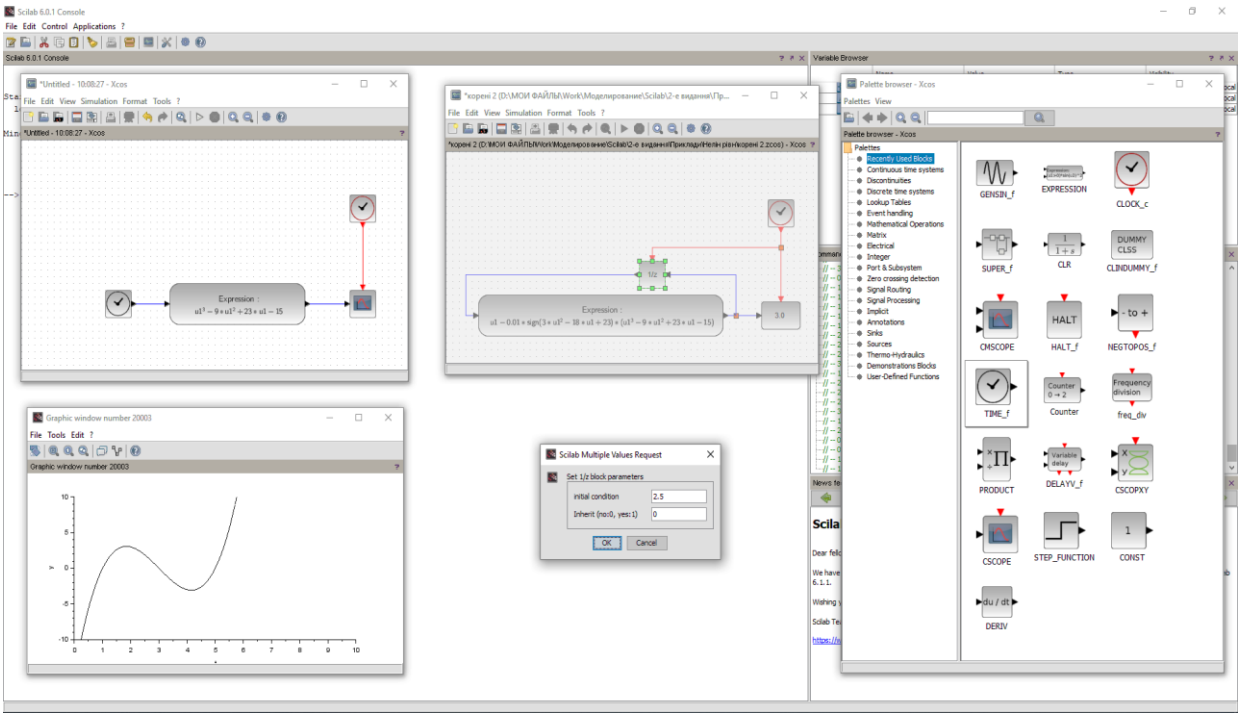


Рисунок 4.6 – Знаходження коренів нелінійного рівняння

4.4 Розв’язування диференціальних рівнянь

Scilab може використовуватися для вирішення звичайних диференціальних рівнянь, які поділяються на лінійні та нелінійні. В лінійних рівняннях змінні та їх похідні можуть бути лише в першому ступені, а коефіцієнти – або постійні, або є функціями незалежної змінної t. Наведемо кілька типових прикладів лінійних та нелінійних рівнянь.

Приклади лінійних рівнянь:

$$\frac{d^2x}{dt^2} + \sin t \frac{dx}{dt} + e^{-t}x = t^2.$$

$$\frac{dx}{dt} - x \ln t = 30.$$

Приклади нелінійних рівнянь:

$$\frac{d^2x}{dt^2} + \sin t \left(\frac{dx}{dt}\right)^2 + e^{-t}x = t^2.$$

$$x\frac{dx}{dt} - x \ln t = 30.$$

Лінійні та нелінійні диференціальні рівняння можуть мати постійну та змінну структуру. Рівняння, коефіцієнти яких постійні чи змінюються безперервно у функції часу або довільного аргументу, називатимемо постійної структури. Рівняння, коефіцієнти яких є функціями часу з

розривами, належать до рівнянь змінної структури. З їхньою допомогою описуються процеси у об'єктах у кілька етапів.

Розв’язання диференціального рівняння у Scilab/Xcos складається з етапів підготовки та отримання результатів.

Підготовка моделі до розрахунків:

- приведення рівнянь до вигляду, зручного для аналогового моделювання;
- розрахунок масштабів і коефіцієнтів передачі блоків;
- складання структурної схеми моделі;
- підготовка даних для контролю правильності розв’язання задачі;
- коригування масштабів і схем з'єднань за результатами пробних рішень.

При попередньому аналізі завдання складається завдання на моделювання, що включає такі положення:

- Систему диференціальних рівнянь;
- при початкових умов при n-му порядку системи рівнянь;
- тимчасовий інтервал розв'язання задачі;
- орієнтовні значення граничних значень змінних до розрахунку масштабів;
- перелік вихідних змінних, їх діапазон та інші характеристики;
- Спосіб реєстрації результатів (графіки, числові значення).

У процесі підготовки рівнянь до розв'язання проводиться заміна реальних змінних на машинні з введенням масштабів змінних та часу.

Приклад 1. Нехай необхідно розв’язати диференціальне рівняння з постійними коефіцієнтами (стаціонарне)

$$\frac{dy}{dt} + 2y = \sin(3t).$$

Розв’язання

Замінимо змінну на нотацію

$$\frac{dy}{dt} = x.$$

Тоді $$y = \int x \cdot dt$$

і рівняння буде в інтегральній формі

$$x + \int xdt = \sin(3t).$$

Виділяємо невідому змінну

$$x = \sin(3t) - \int xdt.$$

Діаграма аналогової імітаційної моделі в системі Scilab

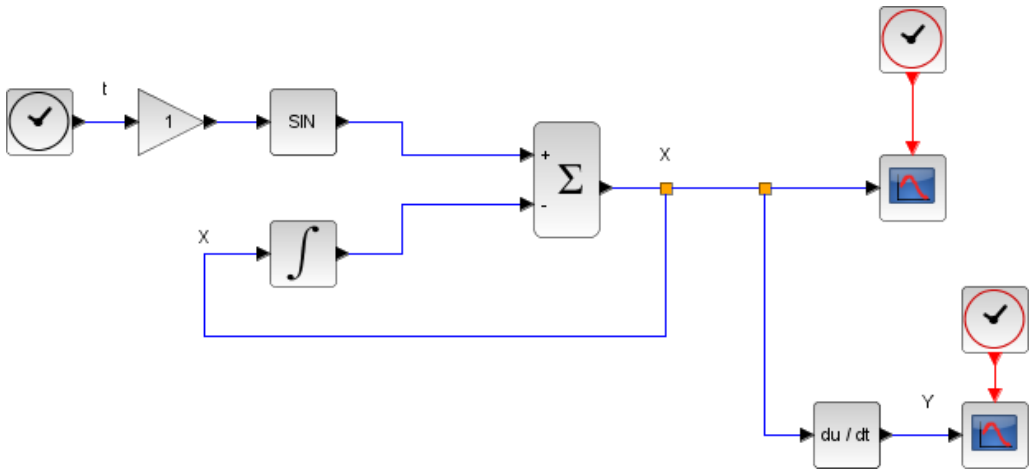


Рисунок 4.7 – Схема розв’язання диференціального рівняння прикладу 1

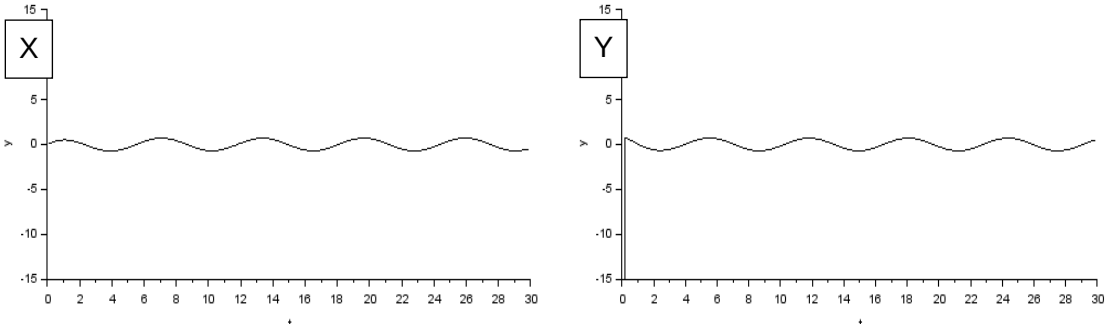


Рисунок 4.8 – Результати розв’язання диференціального рівняння прикладу 1

Приклад 2. Нехай необхідно розв’язати нелінійне нестаціонарне диференціальне рівняння з випадковим впливом на об’єкт (права частина рівняння):

$$\frac{d^2y}{dt^2} - 3y\left(\frac{dy}{dt}\right)^2 + 2ty = rnd.$$

Розв’язання

Замінимо змінну на нотацію

$$\frac{dy}{dt} = x,$$

$$\frac{d^2y}{dt^2} = \frac{dx}{dt} = z.$$

Тоді $x = \int z \cdot dt$ і $y = \int x \cdot dt.$

Таким чином, рівняння буде в інтегральній формі

$$z + 3yx^2 + 2t\int xdt = rnd$$

або $$z + 3(\iint z \cdot dt)(\int z \cdot dt)^2 + 2t \iint z \cdot dt = rnd.$$

Виділяємо невідому змінну

$$z = rnd - 3(\iint z \cdot dt)(\int z \cdot dt)^2 - 2t \iint z \cdot dt.$$

Діаграма аналогової імітаційної моделі в системі Scilab

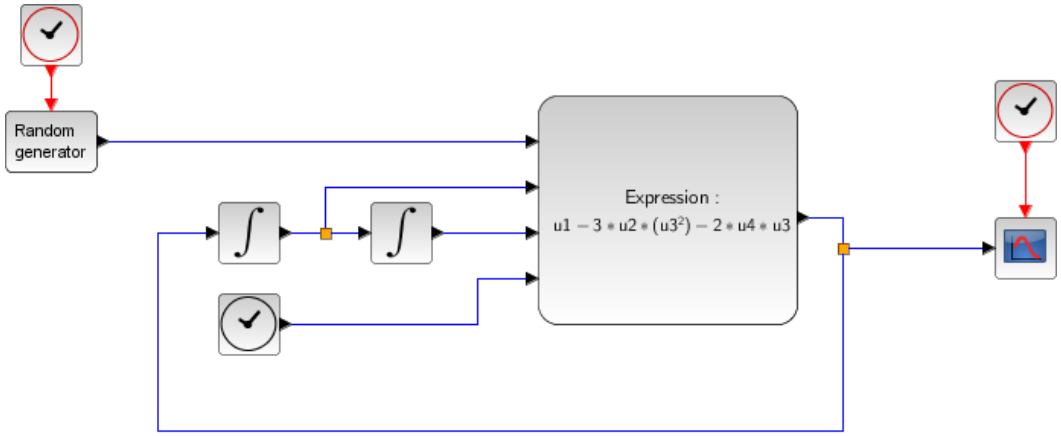


Рисунок 4.9 – Схема розв'язання диференціального рівняння прикладу 2

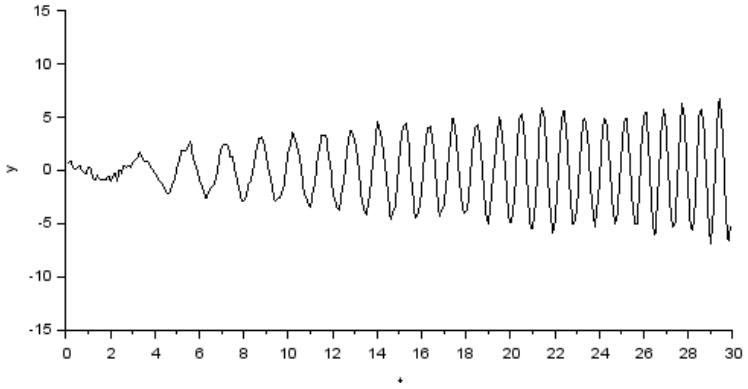


Рисунок 4.10 – Результат розв'язання диференціального рівняння прикладу 2

4.5 Оптимізація систем

Задачі і методи оптимізації систем дуже різноманітні. Принципово відрізняються задачі неперервної оптимізації і задачі дискретної оптимізації на графах. У цьому підрозділі розглянемо тільки неперервну однокритеріальну оптимізацію без обмежень. До такої задачі за допомогою різних прийомів частіше за все зводять інші задачі неперервної оптимізації.

У найпростішому випадку екстремум критеріальної функції $f(x)$ шукають за відомим з математики способом: знаходження коренів похідної

$f'(x) = 0$. Для прикладу з підрозділу 4.3 $f(x) = x^3 - 9x^2 + 23x - 15$, отже шукаємо екстремуми як корені рівняння

$$3x^2 - 18x + 23 = 0$$

за ітераційною формулою

$$x_{i+1} = x_i - 0.1 \cdot sign(6x_i - 18) \cdot (3x_i^2 - 18x_i + 23).$$

Оскільки критерій у цьому прикладі має два екстремуми, то, залежно від мети оптимізації (пошук максимуму або мінімуму), початкове наближення варто обирати за межами інтервалу, де знаходяться екстремуми, з того боку, де є необхідний екстремум – див. графік функції.

4.6 Моделі систем керування

Процеси в системах керування визначаються головними *принципами* їх функціонування. Виділяють декілька фундаментальних принципів:

- принцип розімкненого керування;
- принцип керування за відхиленням (принцип зворотного зв'язку);
- принцип оптимального керування;
- принцип адаптації.

Основна мета принципів керування – визначення способу переведення об'єкта керування у заданий стан з максимальною точністю в умовах дії збурення *f*. Узагальнені структурні схеми систем, які відповідають цим принципам, наведено на рис. 4.11.

Принцип розімкненого керування (рис. 4.11, а) є найпростішим. Суть розімкненого керування полягає у вимірюванні збурення *Z* і компенсації його впливу на об'єкт керування ОК через додавання відповідної корекції (-*v*), яка виробляється блоком корекції БК до керування *u*, яке надходить з регулятора *P*.

Принцип керування за відхиленням (рис. 4.11, б) ще називають принципом замкненого керування або принципом від'ємного зворотного зв'язку (ЗЗ). Це найфундаментальніший принцип керування у природі та техніці. Керування за цим принципом передбачає вимірювання стану *y* об'єкта керування за допомогою блока зворотного зв'язку БЗЗ і знаходження його відхилення $\delta = x - y$ від завдання *x*, на основі чого регулятор виробляє керівний вплив *u*.

Принцип комбінованого регулювання використовується одночасно з регулюванням за збуренням і за відхиленням (див. рис. 4.11, в), що забезпечує більш високу точність керування.

Принцип адаптації (див. рис. 4.11, г) полягає у пристосуванні параметрів, структури або алгоритму функціонування системи керування до зміни умов. Адаптація здійснюється за допомогою блока корекції закону керування БКЗ, який оцінює зміни умов (ідентифікує систему) і приймає

рішення щодо необхідних змін у законі керування, внаслідок чого змінюються параметри або взагалі вигляд оператора F_p.

Принцип оптимального керування (див. рис. 4.11, д) полягає у знаходженні такого закону керування u, який забезпечив би переведення об'єкта керування з початкового стану y_0 у кінцевий стан y_n з додатковою умовою – забезпечити досягнення мінімуму або максимуму певного показника Q (наприклад, часу керування, витрат енергії тощо), який називається критерієм або цільовою функцією. Можуть також задаватися обмеження на певні параметри процесу керування. Задача розв'язується за допомогою блока оптимізації О, який знаходить оптимальну траєкторію (бажану послідовність проміжних станів об'єкта), з якої регулятор формує керування u.

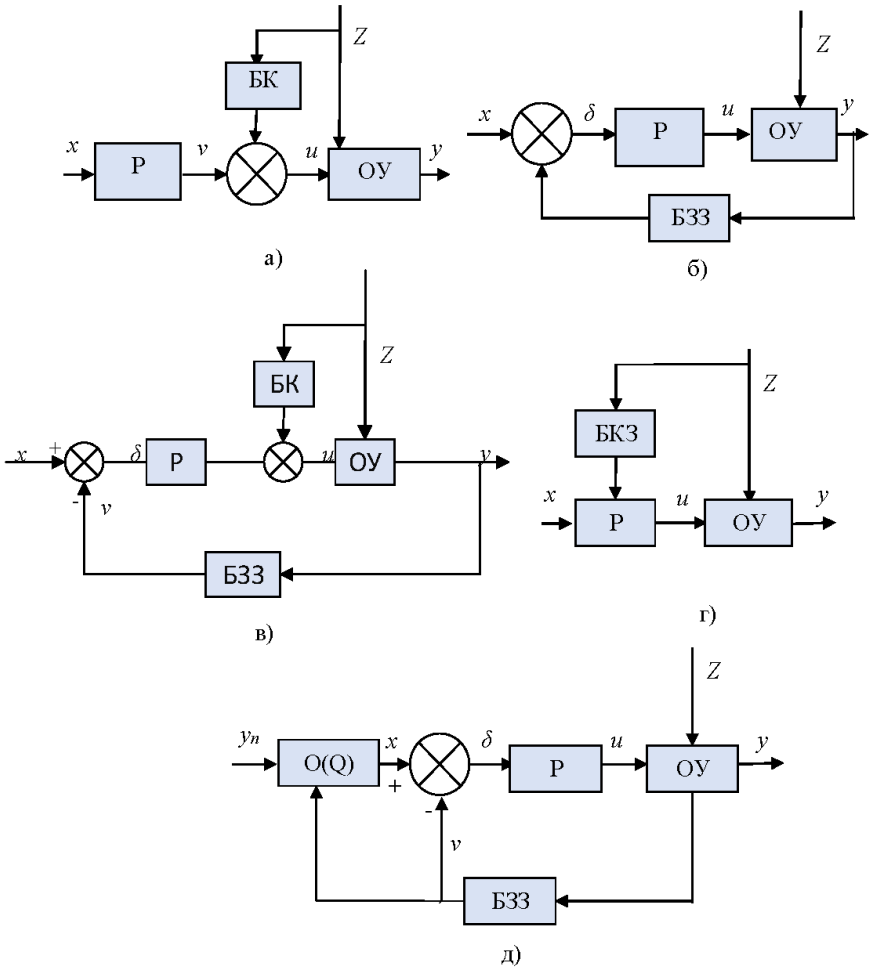


Рисунок 4.11 – Структурні моделі принципів керування

Закон керування – це правило обчислення керівного сигналу u на основі вставки (вхідного сигналу) x і параметрів стану об'єкта y. За комп'ютерної реалізації закон керування називають алгоритмом.

Закони керування розділяють на лінійні і нелінійні.

Лінійні закони використовуються в системах автоматичного керування технологічними процесами для керування лінійними об'єктами без обмежень. Вибір лінійного закону керування визначається необхідною точністю керування.

Найпоширенішими є типові закони керування:

- Пропорційний – **П**;
- Інтегральний – **І**;
- Диференціальний – **Д**;
- Пропорційно-інтегральний – **ПІ**;
- Пропорційно-диференціальний (форсувальний) – **ПД**;
- Пропорційно-інтегрально-диференціальний – **ПІД**.

Найпростішим є пропорційний закон

$$u = k(x - y), \tag{4.1}$$

тобто з передатною функцією регулятора

$$W_p = k. \tag{4.2}$$

Така система має значні похибки − як статичну, так і динамічну.

Для усунення статичної похибки використовується інтегральний закон керування

$$u = k \int_0^t (x - y)dt. \tag{4.3}$$

Передатна функція регулятора, що реалізує такий закон

$$W_p = \frac{k}{p}. \tag{4.4}$$

Усуваючи статичну похибку, інтегральний закон керування суттєво погіршує динамічні характеристики системи.

Для одночасного покращення як статичних, так і динамічних характеристик використовуються комбіновані закони керування.

Для зменшення динамічної похибки використовується пропорційно-диференціальний (форсувальний) закон

$$u = k_1(x - y) + k_2 \frac{d(x-y)}{dt}. \tag{4.5}$$

Передатна функція регулятора, що реалізує такий закон

$$W_p = k_1 + k_2 p. \tag{4.6}$$

Диференціальна складова зменшує динамічну та збільшує статичну похибку системи. Крім того, дещо погіршується завадостійкість.

Пропорційно-інтегральний закон

$$u = k_1(x - v) + k_2 \int_0^t (x - v)dt \tag{4.7}$$

зменшує статичну похибку, не погіршуючи динаміку системи.

Передатна функція регулятора, що реалізує такий закон,

$$W_p = k_1 + {}^{k_2}/_p. \tag{4.8}$$

Пропорційно-інтегральний закон усуває статичну похибку, не збільшуючи динамічну похибку системи.

Найкращим (але найскладнішим) є пропорційно-інтегрально-диференціальний закон

$$u = k_1(x - v) + k_2 \int_0^t (x - v)dt + k_3 \frac{d(x-v)}{dt}. \quad (4.9)$$

Передатна функція регулятора, що реалізує такий закон

$$W_p = k_1 + {}^{k_2}/_{p} + k_3 p. \quad (4.10)$$

Пропорційно-інтегрально-диференціальний закон усуває статичну похибку і зменшує динамічну похибку системи.

Усі елементи систем керування (об'єкти, регулятори, блоки зворотного зв'язку, коректори тощо) можуть працювати як у неперервному, так і імпульсному режимах (дискретно), бути лінійними, нелінійними, релейними або цифровими.

Уся ця різноманітність принципів, законів, режимів роботи тощо приводить до відповідної різноманітності систем керування. Ця різноманітність ще більше розширюється у випадку багатовимірних об'єктів (тобто об'єктів, які характеризуються декількома параметрами, як, наприклад, електричний генератор, що характеризується напругою, струмом і частотою вихідного сигналу). Тоді система керування стає багатоконтурною, причому контури регулювання можуть мати досить складні зв'язки один з одним.

Імітаційне моделювання дозволяє дослідити моделі будь-якої структури і складності.

Приклад 1. Необхідно створити імітаційну модель системи, зображеної на рис. 4.12.

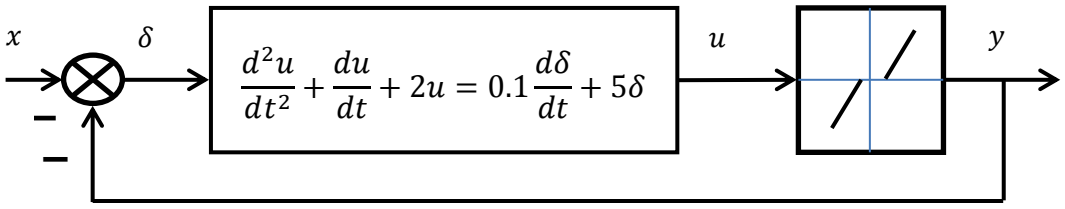


Рисунок 4.12 – Приклад 1 системи керування

До складу системи входять 3 блоки. Моделі блока порівняння (суматора) і релейного об'єкта можуть бути реалізовані готовими блоками, які є у палітрах Scilab/Xcos. А реалізація динамічного регулятора потребує додаткової підготовки.

Регулятор, функція якого задана диференціальним рівнянням, може бути реалізований двома способами:

1 спосіб. Перетворюємо ліву частину диференціального рівняння на інтегральне і включаємо модель-розв'язок отриманого рівняння до складу системи керування:

$$\frac{d^2u}{dt^2} = z;$$
$$\frac{du}{dt} = \int zdt;$$
$$u = \iint zdtdt;$$
$$z + 3\int zdt + 2\iint zdtdt = 0.1\frac{d\delta}{dt} + 5\delta;$$
$$z = 0.1\frac{d\delta}{dt} + 5\delta - 3\int zdt - 2\iint zdtdt.$$

Розв'язок рівняння знаходимо аналогічно підрозділу 4.4. Як результат отримуємо модель системи (рис. 4.13).

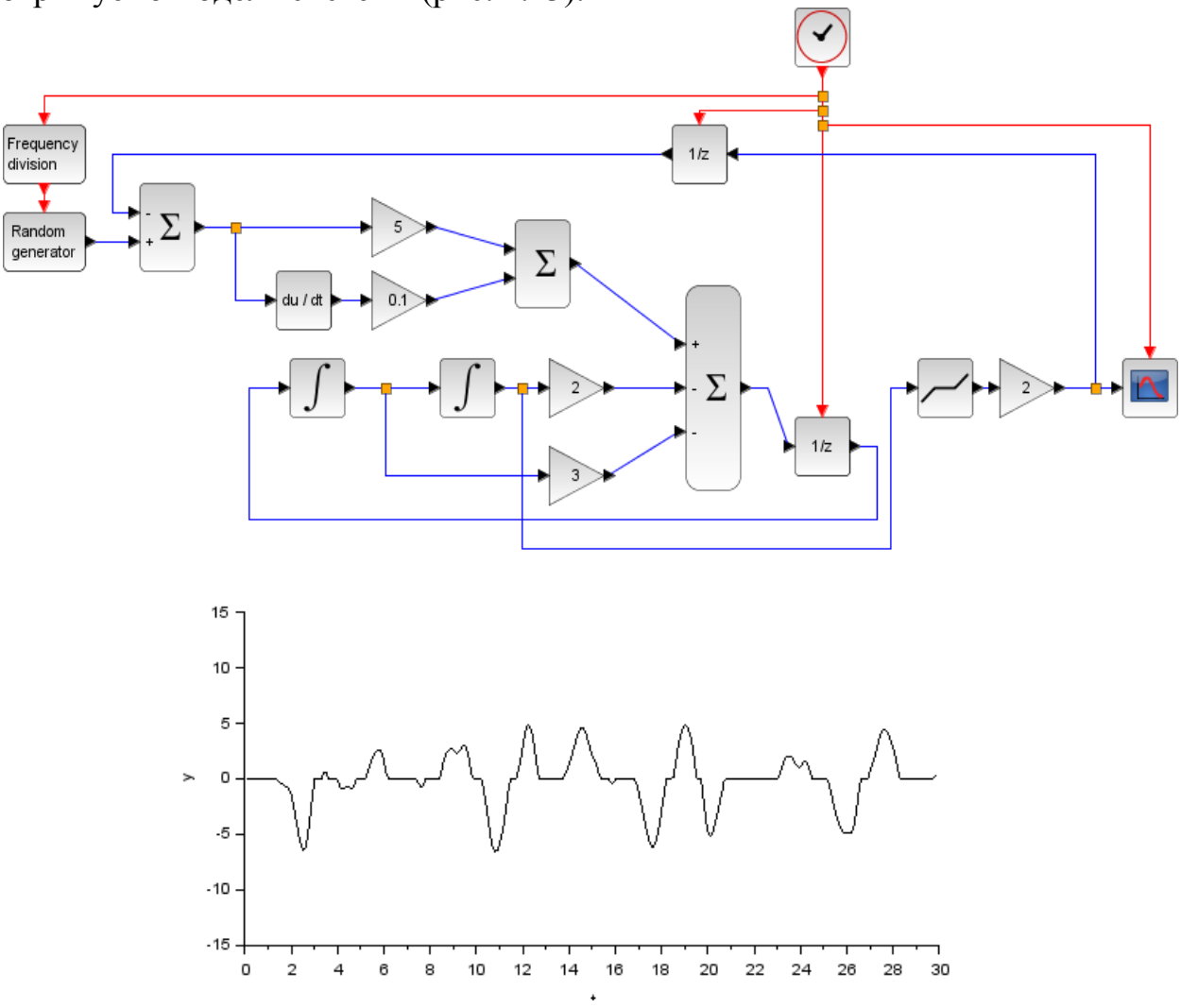


Рисунок 4.13 – Модель для прикладу 1 за способом 1

Аналогічний результат отримуємо другим способом.

2 спосіб. Перетворюємо диференціальне рівняння на операторну форму і знаходимо передатну функцію регулятора.

Заміняємо $\frac{d}{dt} \to s$. Отримуємо

$$s^2u + 3su + 2u = 0.1s\delta + 5\delta,$$

$$u(s^2 + 3s + 2) = \delta(0.1s + 5),$$

$$u = \delta \frac{0.1s + 5}{s^2 + 3s + 2}.$$

Відповідну модель системи показано на рис. 4.14.

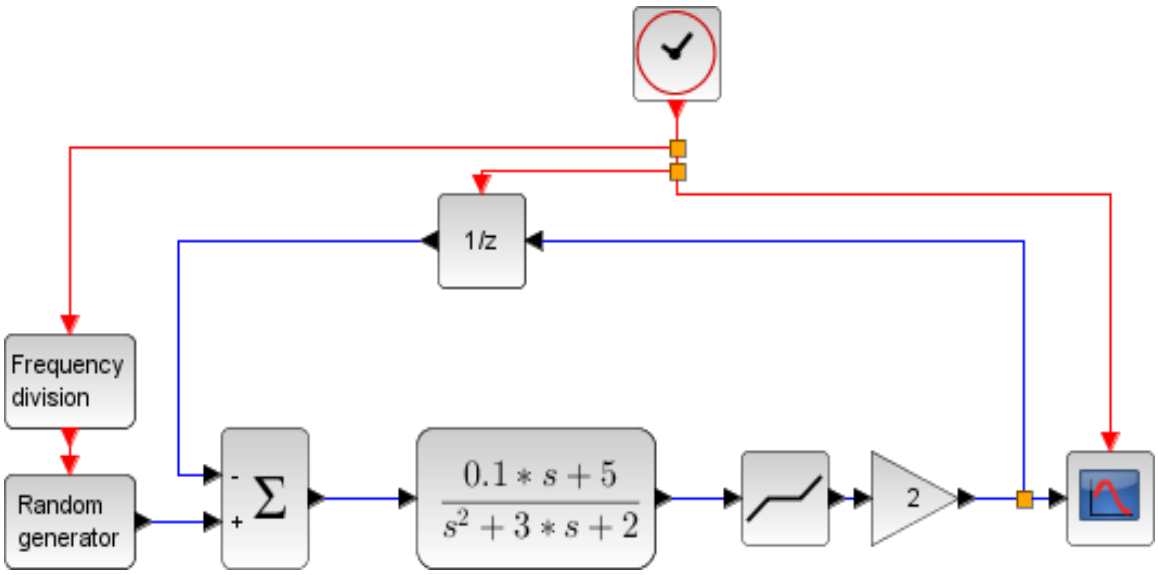


Рисунок 4.14 – Модель для прикладу 1 за способом 2

Спосіб 2 може застосовуватися тільки для лінійних диференціальних рівнянь, оскільки з нелінійних рівнянь неможливо знайти передатну функцію.

Приклад 2. Система має аналогічну структуру, проте регулятор описується нелінійним диференціальним рівнянням

$$\frac{d^2u}{dt^2} + 0.2(\frac{du}{dt})^2 + 5u = \frac{d\delta}{dt} + 3\delta$$

Відповідна імітаційна модель системи

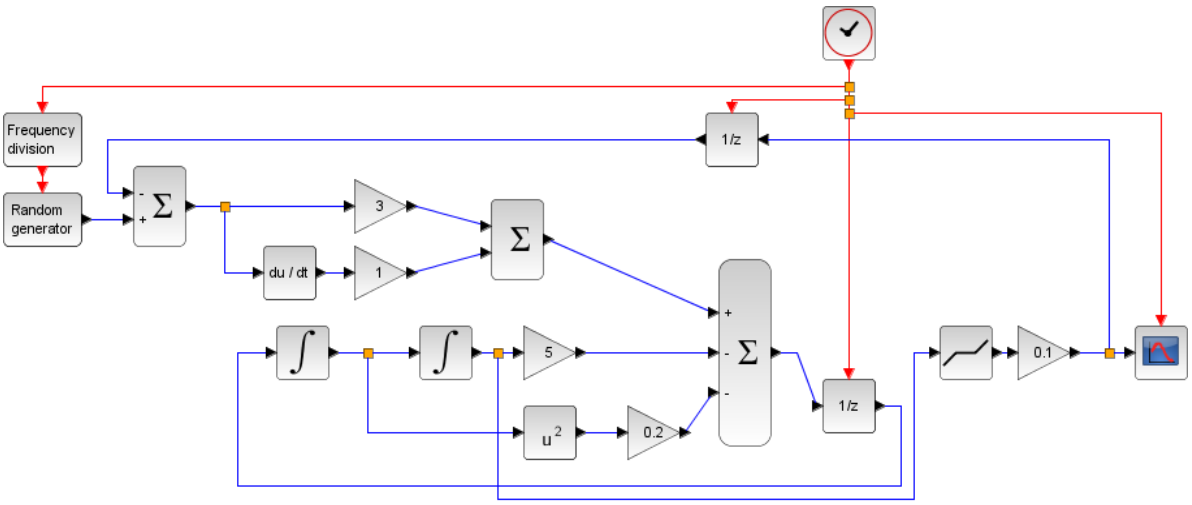
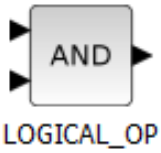


Рисунок 4.15 – Модель для прикладу 2 за способом 1

4.7 Моделі логічних і цифрових схем

Для імітаційного моделювання логічних схем в системі Scilab/Xcos використовуються блоки палітри «Ціле», зокрема блок базових логічних операцій

	Логічна операція Вид операції задається в параметрах: 0 – AND 1 – OR 2 – NAND 3 – NOR 4 – XOR 5 – NOT

Приклад моделі логічної схеми показано на рис. 4.16

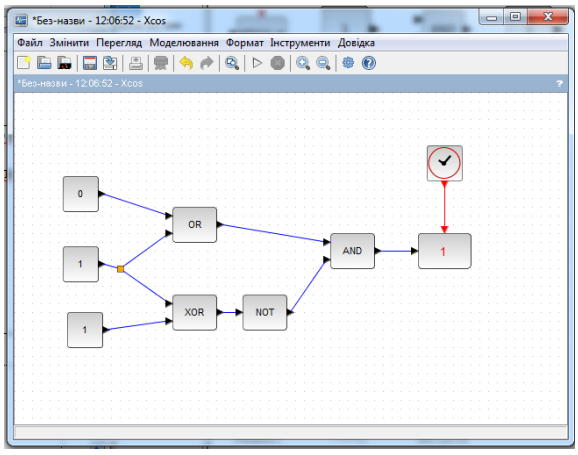


Рисунок 4.16 – Приклад моделі логічної схеми

4.8 Моделі електричних схем

Електричні (електронні) пристрої моделюються блоками палітри «Електрика». При складанні моделі слід уважно слідкувати за типом входів/виходів – електричні сигнали або цифрові дані.

Приклад простого амплітудного модулятора показано на рис. 4.17. Модуляція здійснюється шляхом керування коефіцієнтом пропускання фільтра.

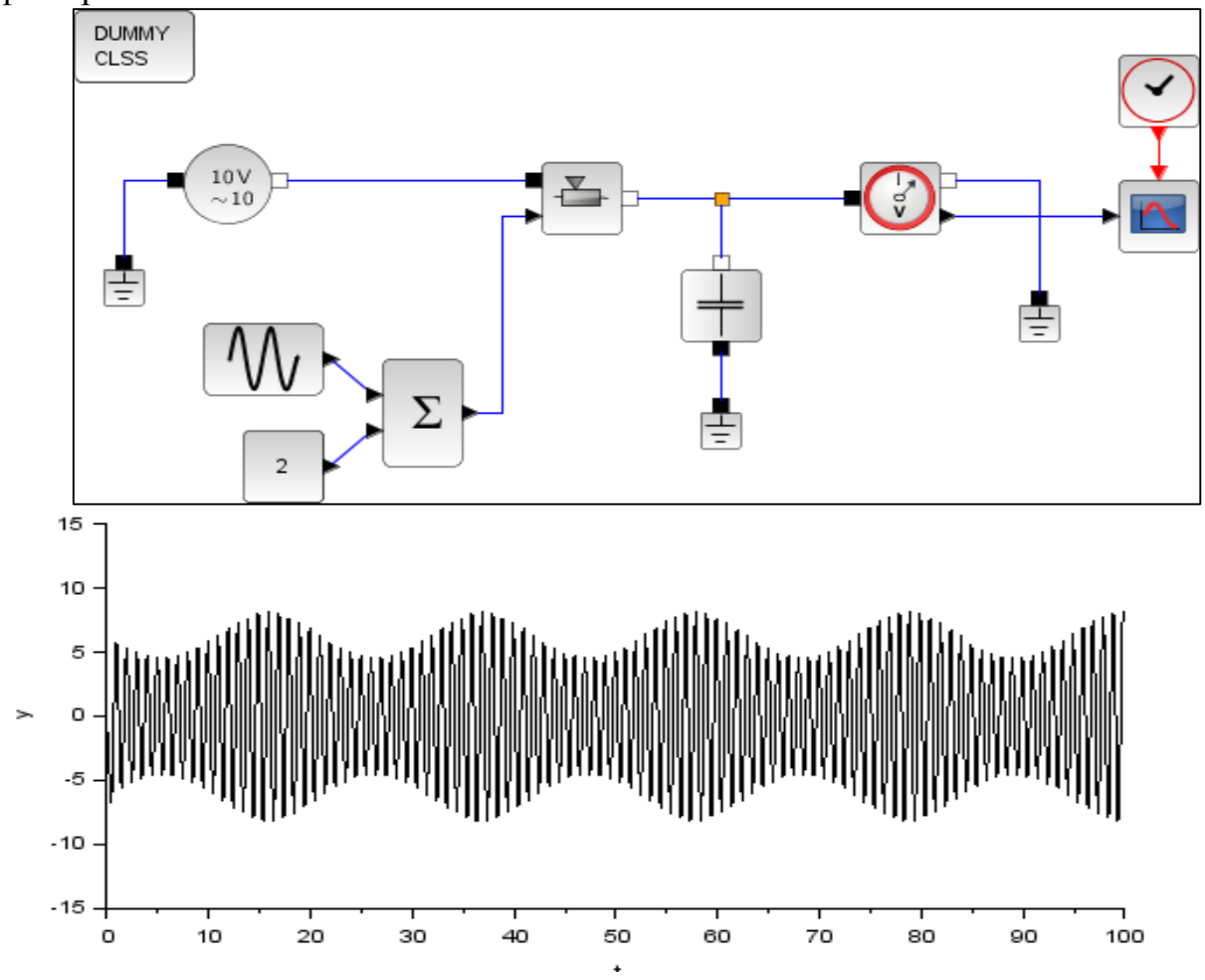


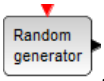
Рисунок 4.17 – Приклад моделі електричної схеми амплітудного модулятора

4.9 Статистичний аналіз випадкових процесів

У реальних умовах функціонування практично усіх систем відбувається під дією випадкових впливів. Отже і їх моделювання має враховувати цю дію. Відповідно виникає необхідність генерувати випадкові впливи на модель і здійснювати статистичний аналіз результатів.

Загальну схему дослідження систем в умовах дії випадкових впливів показано на рис. 4.18.

Рисунок 4.18 – Загальна схема дослідження перетворення випадкових процесів

Генерування випадкового впливу здійснюється за допомогою блока 

, проте цей блок генерує лише випадкові процеси з нормальним (гаусівським) або рівномірним розподілами ймовірності і майже нульовою кореляцією («білий шум»). Якщо ж для моделювання необхідні випадкові впливи з іншими статистичними характеристиками, то згенерований випадковий процес перетворюють на заданий. Отримання іншого розподілу ймовірностей досягається нелінійним статичним перетворенням, а отримання іншої кореляційної функції – лінійним динамічним перетворенням [3].

Статистичний аналіз полягає у визначенні розподілу ймовірностей результатів моделювання. При моделюванні розподіл отримують у вигляді гістограми, тобто набору значень ймовірностей потрапляння результатів моделювання у задані інтервали на числовій осі. На рис. 4.19 статистичний аналізатор показано у вигляді суперблока, числова вісь розбита на 8 інтервалів, причому перший і останній інтервали є напіввідкритими: $(-\infty, a)$ і $(b, +\infty)$. Значення ймовірностей відображаються у числовому вигляді в інтервалі $[0; 1]$ у блоках 

.

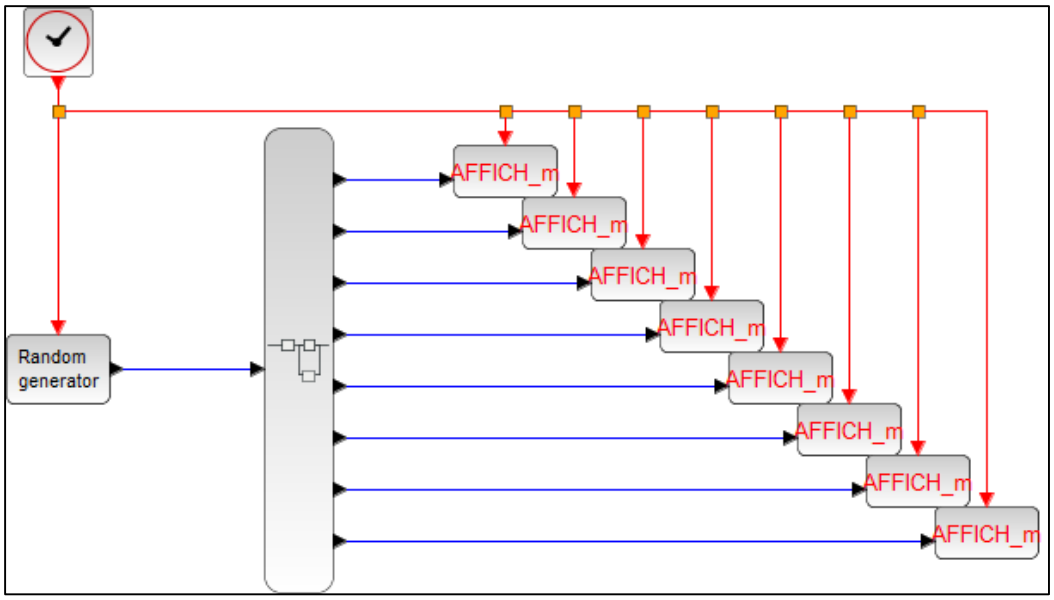
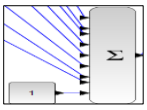


Рисунок 4.19 – Загальна модель аналізу розподілу ймовірностей.

Схема суперблока аналізатора розподілу ймовірностей показана на рис. 4.20. Підрахунок кількості потраплянь вхідного процесу у окремі інтервали здійснюється внутрішніми суперблоками. Кожному внутрішньому суперблоку задається номер інтервалу. Загальна кількість проаналізованих значень вхідного процесу підраховується блоком  . Ще додається одиниця для уникнення ділення на 0 у наступній операції. У цій наступній операції отримується значення відносної частоти потрапляння у заданий інтервал, яке є статистичною оцінкою відповідної ймовірності $p_i \approx \frac{n_i}{\sum_{i=1}^{N} n_i}$, де N – кількість інтервалів (у нашому прикладі $N = 8$).

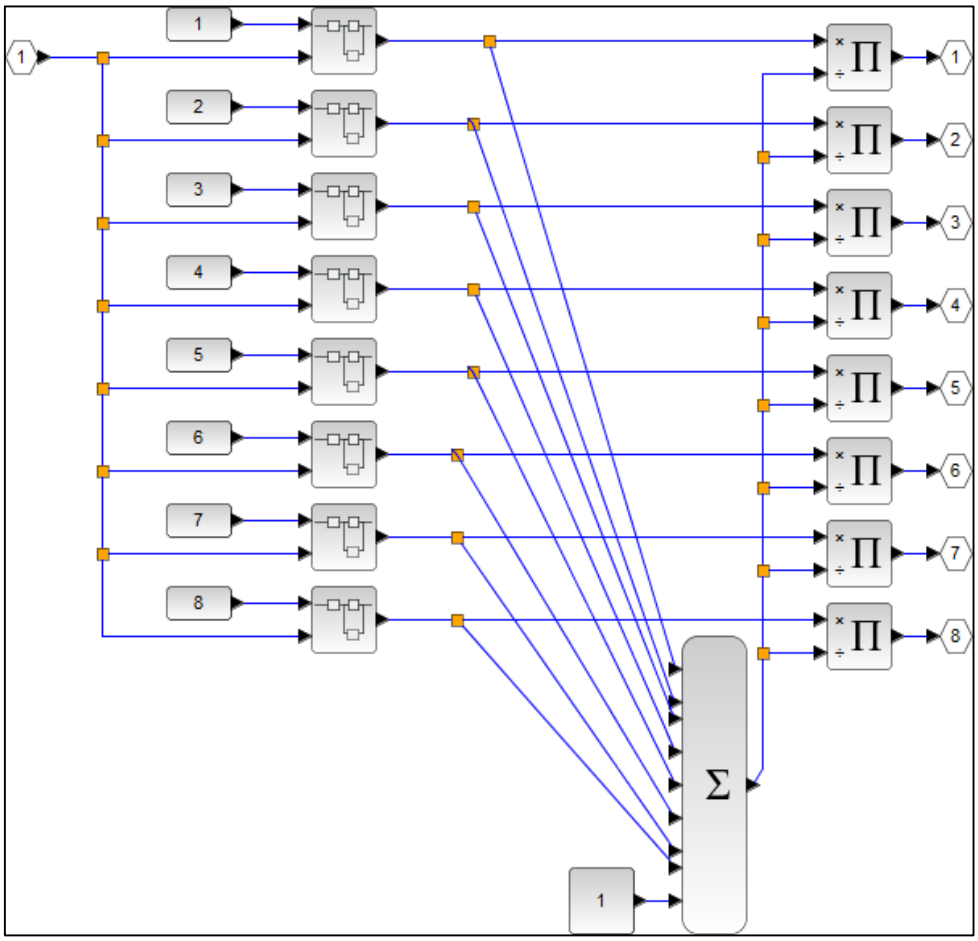
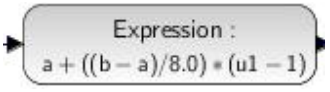


Рисунок 4.20 – Схема суперблока аналізатора розподілу ймовірностей

Схеми блоків підрахунку потраплянь вхідного процесу у середній, перший і останній інтервали показано на рис. 4.21, 4.22 і 4.23, відповідно. На вхід «1» цих блоків подається номер інтервалу, а на вхід «2» – вхідний

процес. У блоках 

і 

розраховуються нижня і верхня межі інтервалу, де $[a; b]$ – діапазон аналізу.

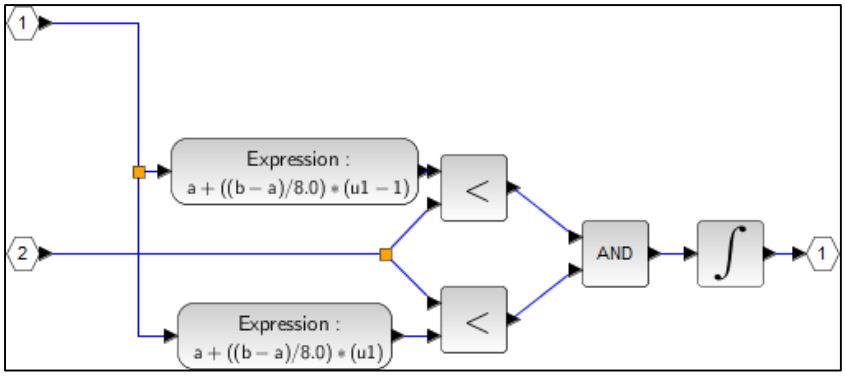


Рисунок 4.21 – Суперблок підрахунку кількості потраплянь у середній інтервал

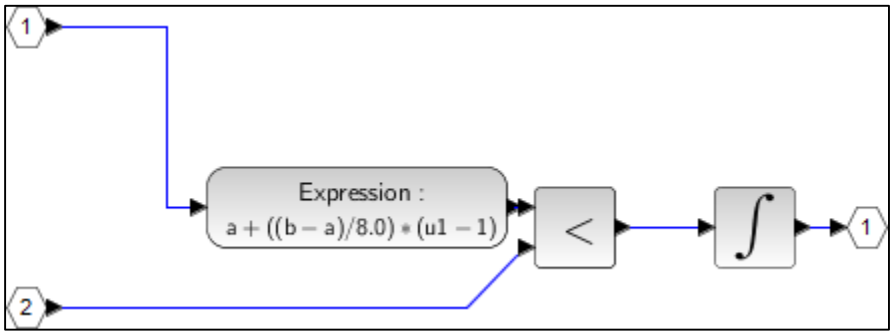


Рисунок 4.22 – Суперблок підрахунку кількості потраплянь у перший інтервал

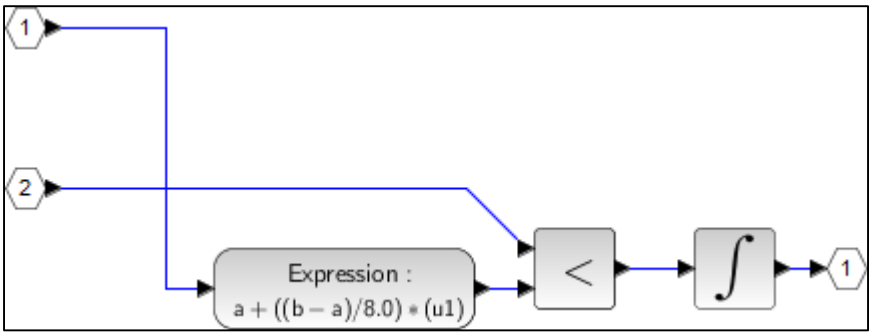


Рисунок 4.23 – Суперблок підрахунку кількості потраплянь у останній інтервал

Межі діапазону аналізу $[a; b]$ задаються як контекстні змінні (рис. 4.24). Для розподілу ймовірностей з параметрами, визначеними у вікні налаштувань генератора (рис. 4.25), нижче показано результат аналізу випадкового процесу на рис. 4.26.

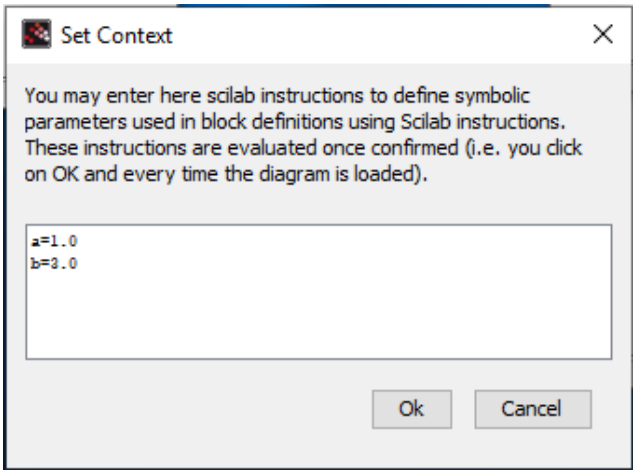


Рисунок 4.24 – Вікно задавання контекстних змінних

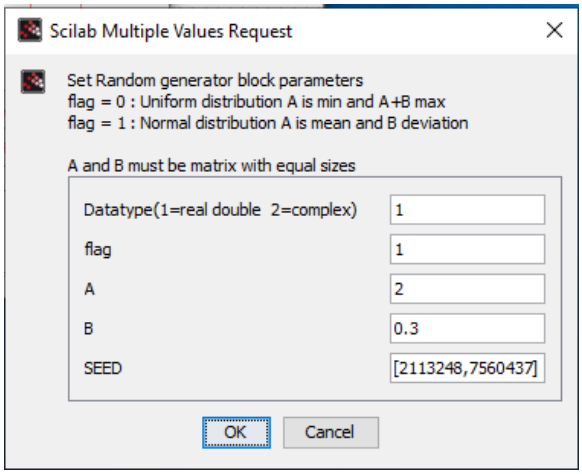


Рисунок 4.25 – Вікно налаштувань генератора випадкового процесу

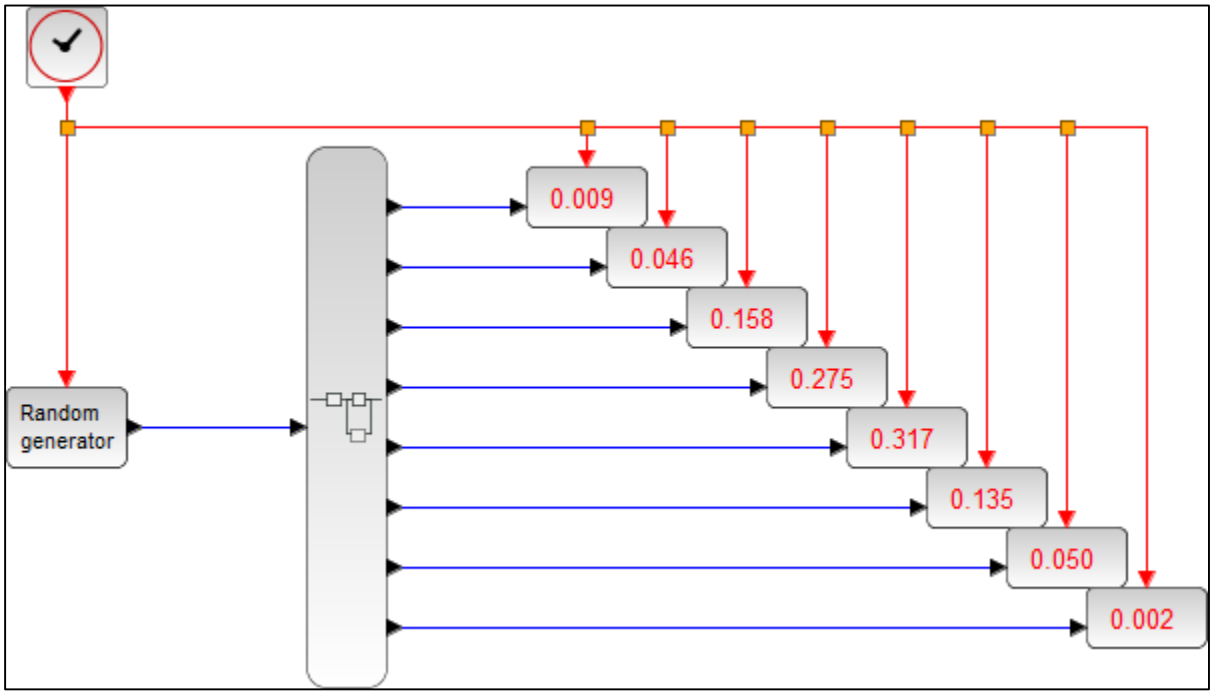


Рисунок 4.26 – Результат аналізу випадкового процесу

Якщо в результаті аналізу ми отримаємо великі значення ймовірності у першому або останньому інтервалах, то це означатиме неправильний вибір відповідної межі діапазону статистичного аналізу.

Для правильної інтерпретації результатів моделювання та аналізу часто корисно будувати графік розподілу ймовірностей $p(x)$, де $x(t)$ – досліджуваний вхідний процес. Особливість побудови такого графіка полягає в тому, що він будується не як функція часу (як у попередніх прикладах), а як функція

значень вхідного процесу, причому може бути побудований лише після закінчення статистичного аналізу. На рис. 4.27 показано схему статистичного аналізу з наступною побудовою графіка статистичного розподілу ймовірностей. Для цього весь процес розбитий на дві частини:

- протягом 100 тактів генерується і аналізується випадковий процес;
- протягом наступних 10 тактів будується графік.

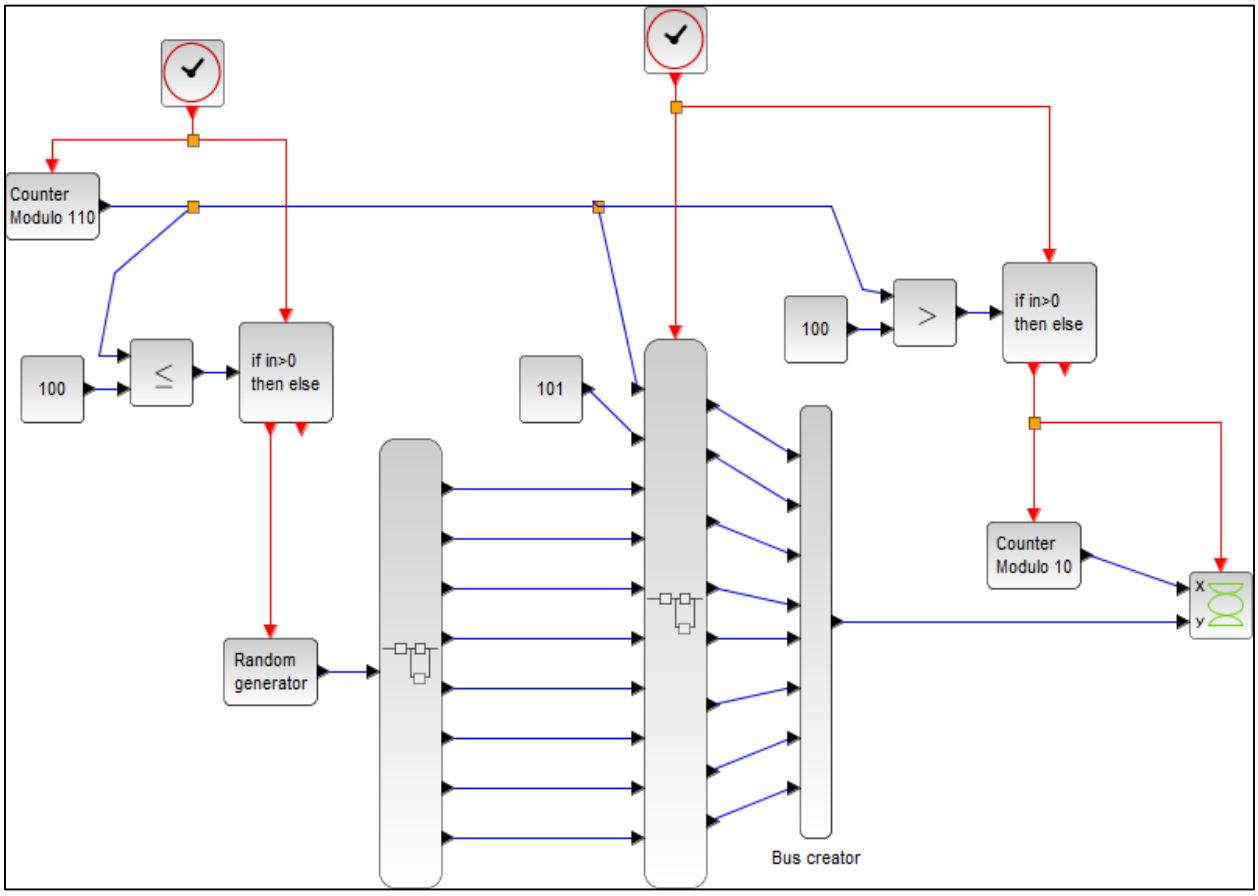
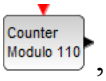
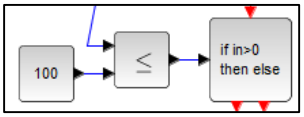
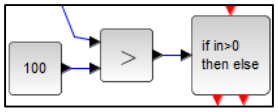


Рисунок 4.27 – Схема статистичного аналізу з наступною побудовою графіка

Такти підраховуються за допомогою лічильника 

, а перевірка кількості тактів і відповідний «дозвіл» на роботу двох частин схеми здійснюється блоками 

і 

, відповідно.

Для побудови графіка здійснюється перетворення паралельного набору значень ймовірностей, отриманого внаслідок статистичного аналізу, на послідовний набір за допомогою схеми опитування паралельних виходів на

рис. 4.28 і блока формування шини 

. На перший вхід схеми опитування подається номер поточного такту, на другий вхід – номер такту, з якого починається опитування. Структуру суперблоків перевірки номера такту і комутації показано на рис. 4.29.

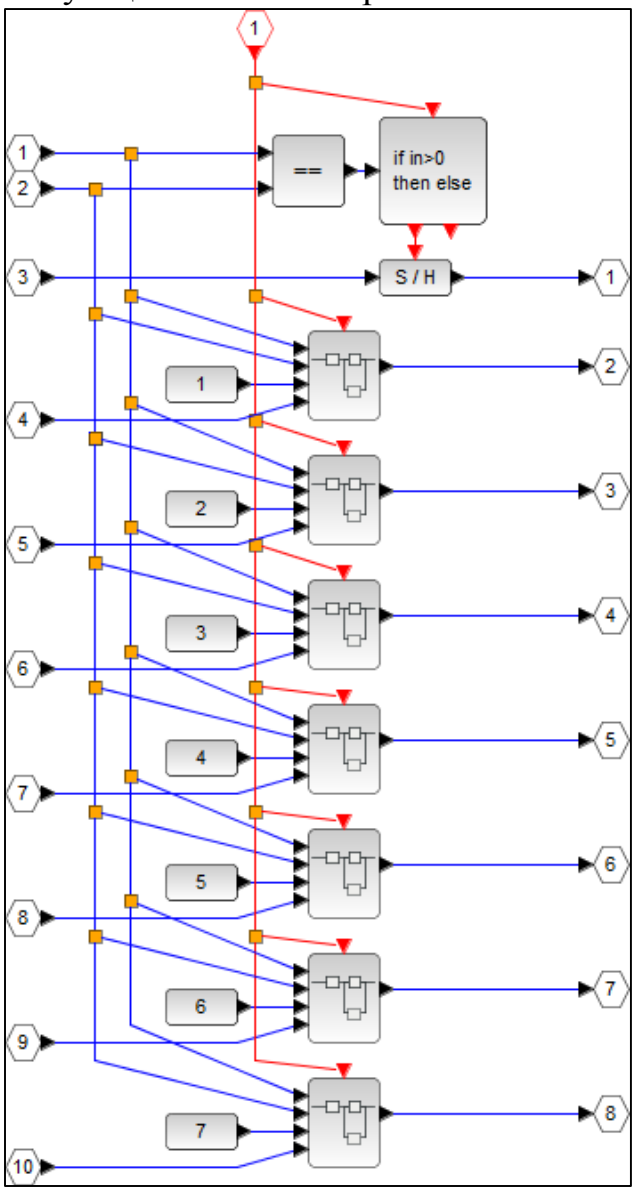


Рисунок 4.28 – Схема опитування паралельних виходів

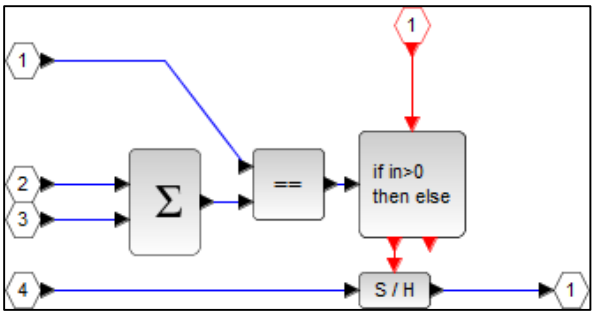


Рисунок 4.29 – Суперблок перевірки номера такту і комутації

Графік будується за допомогою блока 

.

Результати побудови графіка статистичного аналізу розподілу ймовірностей показано на рис. 4.30.

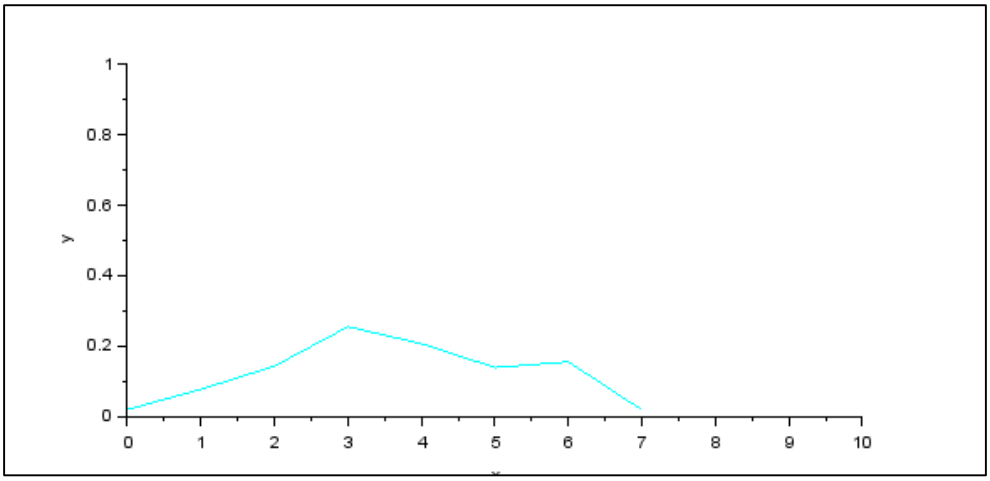


Рисунок 4.30 – Графік статистичного аналізу розподілу ймовірностей

Графік, хоча і нагадує нормальний розподіл, який генерував генератор, проте доволі суттєво відрізняється від теоретичної форми. Це зумовлено невеликою кількістю згенерованих даних – всього 100, що для аналізу розподілу ймовірностей дуже мало, та малою кількістю інтервалів розбиття діапазону значень.

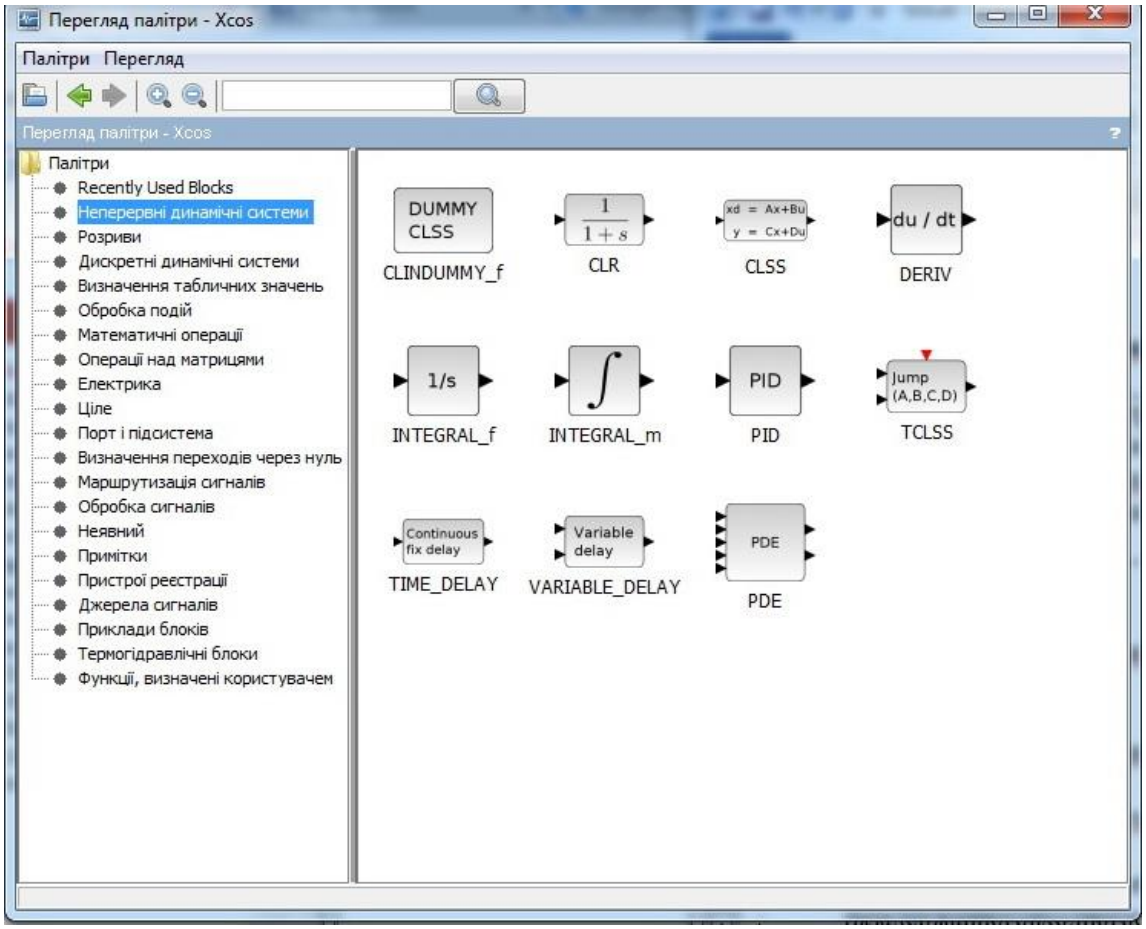
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

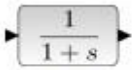
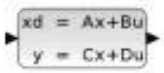
1. Моделювання об’єктів і систем : [лабораторний практикум] / В. М. Дубовой, О. Д. Никитенко, М. С. Юхимчук, А. В. Галущак. – Вінниця : ВНТУ, 2021. – 157 с.
2. Scilab/Xcos help pages.
3. Моделювання та оптимізація систем : [підручник] / В. М. Дубовой, Р. Н. Квєтний, О. І. Михальов, А. В. Усов. – Вінниця : ВНТУ, 2017. – 798 с.
4. Фетісов В. С. Математична система Scilab : навч.-метод. посібн. – 2-ге вид., перероб. і доп. – Ніжин : НДУ ім. М. Гоголя, 2022. – 82 с.
5. Campbell S. L. Modeling and Simulation in Scilab/Xcos with XcosLab 4.4, Second Edition. / Stephen L. Campbell, Jean-Philippe Chancelierand, Ramine Nikoukhah.// – Springer, 2010.
6. Alice Karpenko Моделирование систем в программной среде Scilab&Xcos 5.5.1 [Електронний документ] / Режим доступу: https://www.kv.by/node/10948
7. Nikoukhah R. Xcos: a dynamic systems modeler and simulator. / Ramine Nikoukhah. INRIA-Rocquencourt. Domaine de Voluceau, France.
8. Najafi M. The numerical solver for the simulation of the hybrid dynamical systems. / Masoud Najafi// Universite Paris, 2005, 237 с.

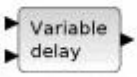
5 КОРОТКА ДОВІДКА

Зауваження щодо довідки:
1. Блоки моделей розділені на «палітри» відповідно до їх функцій. Проте досвід моделювання показав, що деякі блоки часто використовуються у сполученнях з блоками інших палітр. У таких випадках вони дублюються у різних палітрах. У подальших довідкових даних такі дубльовані блоки не показано.
2. Деякі блоки мають декілька варіантів реалізації, які відрізняються додаванням «_f» або «_m» до назви блока. Інколи різниця полягає у форматі вхідних/вихідних даних або параметрів налаштувань, але у більшості випадків їх використання еквівалентне.

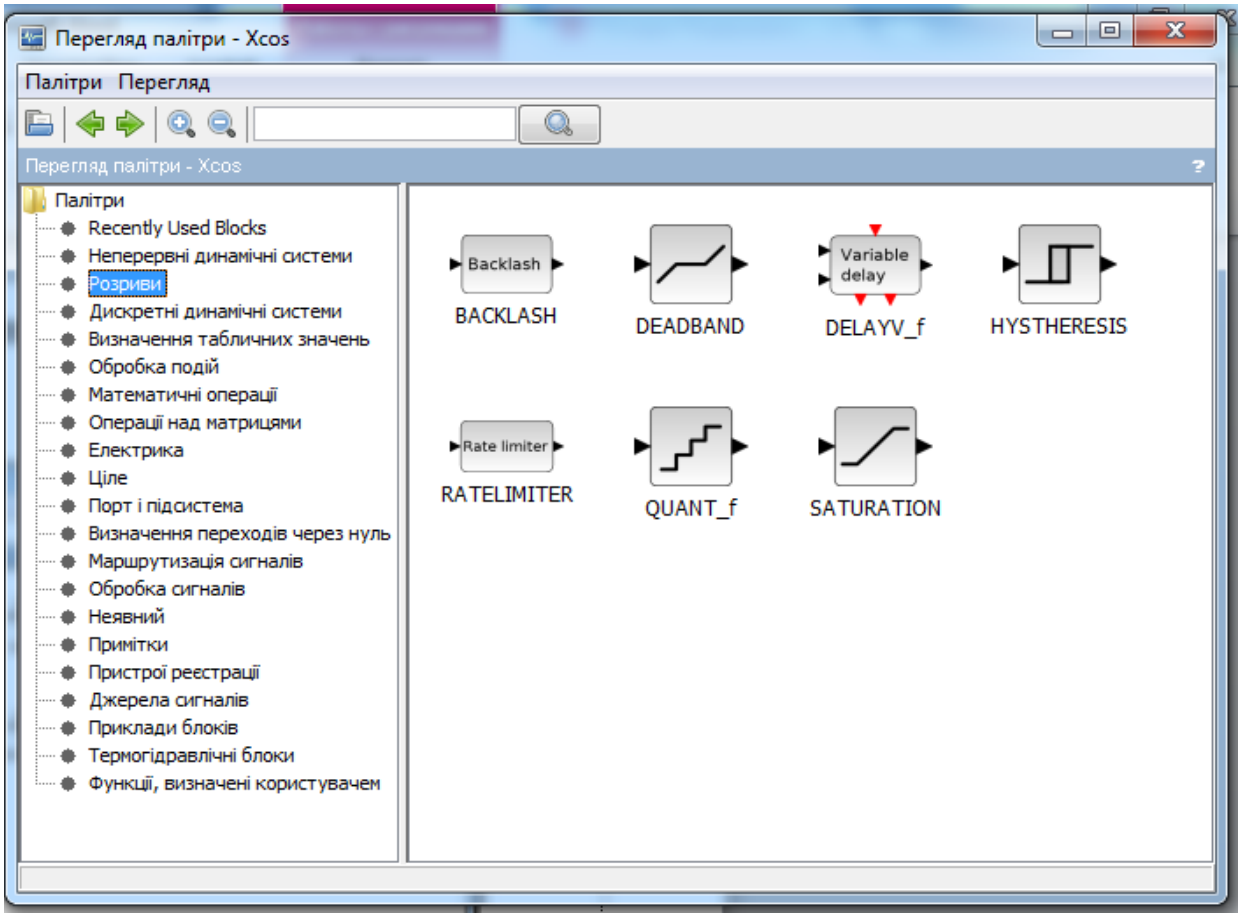
5.1 Continuous time systems (Неперервні динамічні системи)

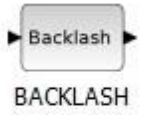
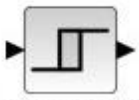
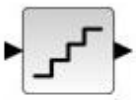


 CLINDUMMY_f	CLINDUMMY_f – блок має встановлюватися в будь-якій схемі з дискретними станами, яка містить блок з перетином нуля, але не в системі з безперервним станом.
 CLR	CLR – безперервна передавальна функція, подана раціональною функцією. Блок реалізує лінійну систему. Раціональна функція має бути належною (ступінь чисельника має бути меншим або дорівнювати ступеню знаменника).
 CLSS	CLSS – безперервна система в просторі станів. Мають бути задані матриці A, B, C, D і початковий стан x_0. x – це вектор змінних стану, u – вектор вхідних впливів, y – вектор вихідних змінних.
 DERIV	DERIV – блок похідної. Вона обчислюється за вхідним сигналом $\Delta u/\Delta t$ Початковий вихід для блока – 0.
 INTEGRAL_f	Інтегратор, визначений за допомогою передатної функції. Вихід є інтегралом входу. Є можливість встановлення початкового значення.
 INTEGRAL_m	INTEGRAL_m – інтегратор. Має набагато більше можливостей, ніж INTEGRAL_f, зокрема обмеження часу інтегрування, обмеження виходу.
 PID	ПІД-регулятор $y = k_p u + k_i \int_0^t u dt + k_d \frac{du}{dt}$ Закон ПІД-регулятора (алгоритм) містить в собі три параметри: коефіцієнти пропорційної складової k_p, інтегральної k_i та диференціальної k_d
 TCLSS	TCLSS – безперервна лінійна система зі стрибком Блок реалізує безперервну лінійну систему з можливістю стрибків у стані. Кількість входів до цього блока два. Перший вхід є звичайним входом лінійної системи, другий – це нове значення стану, яке копіюється в стан, коли «подія» надходить на порт «події» цього блока. Це означає, що стан системи переходить до значення, присутнього на другому вході. Система визначається матрицями (A, B, C, D) та початковим станом x_0. Розміри мають бути сумісними. Розмір входів та виходів регулюється автоматично.

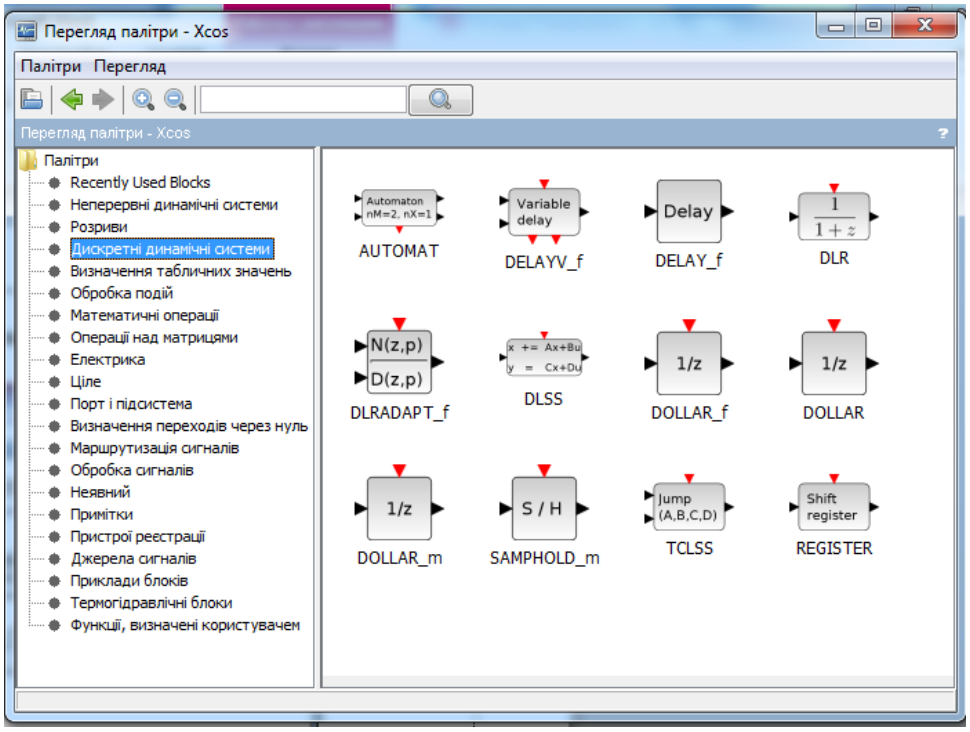
 TIME_DELAY	TIME_DELAY – постійна затримка за часом. На початку моделювання блок виводить параметр Initial input, доки час моделювання перевищить параметр Time delay, тоді блок починає створювати затримку входу. Параметр «Time delay» має бути не від'ємним.
 VARIABLE_DELAY	Блок Variable Transport Delay може бути використаний для імітації змінної затримки часу. Величина затримки задається значенням на другому вході.
 PDE	Блок є реалізацією кількох чисельних методів (кінцеві елементи, різниці і об’єми 1 і 2 порядку) розв’язання одновимірних диференціальних рівнянь в частинних похідних (ЧДУ) в Xcos. Математичні рамки обмежують ЧДУ лінійним скаляром, максимальним порядком два (час і простір). Система вибирає найбільш ефективний чисельний спосіб залежно від типу рівняння і управляє рішенням. Документація відсутня.

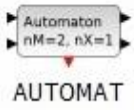
5.2 Discontinuities (Розриви)

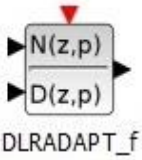
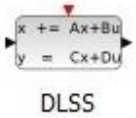
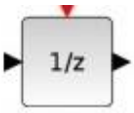


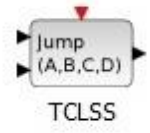
	BACKLASH – мертвий хід (гістерезис із пропорційними гілками)
	DEADBAND – зона нечутливості (забезпечує задану область нульового вихідного сигналу). Це цілий ряд входів, для яких вихідні дані залишаються незмінними. Зовні цього діапазону є лінійний зв'язок між входом мертвої зони u(k) і виходом мертвої смуги v(k)
	Гістерезис
	RATELIMITER-блок обмежує першу похідну сигналу, що проходить через нього. Сигнал на виході змінюється не швидше за задану межу
	QUANT_f – Квантизатор Блок виводить квантування вводу відповідно до вибору методів (типи квантування): • 1: округлення • 2: за верхньою межею • 3: за нижньою межею
	SATURATION – Обмежувач

5.3 Discrete time systems (Дискретні динамічні системи)

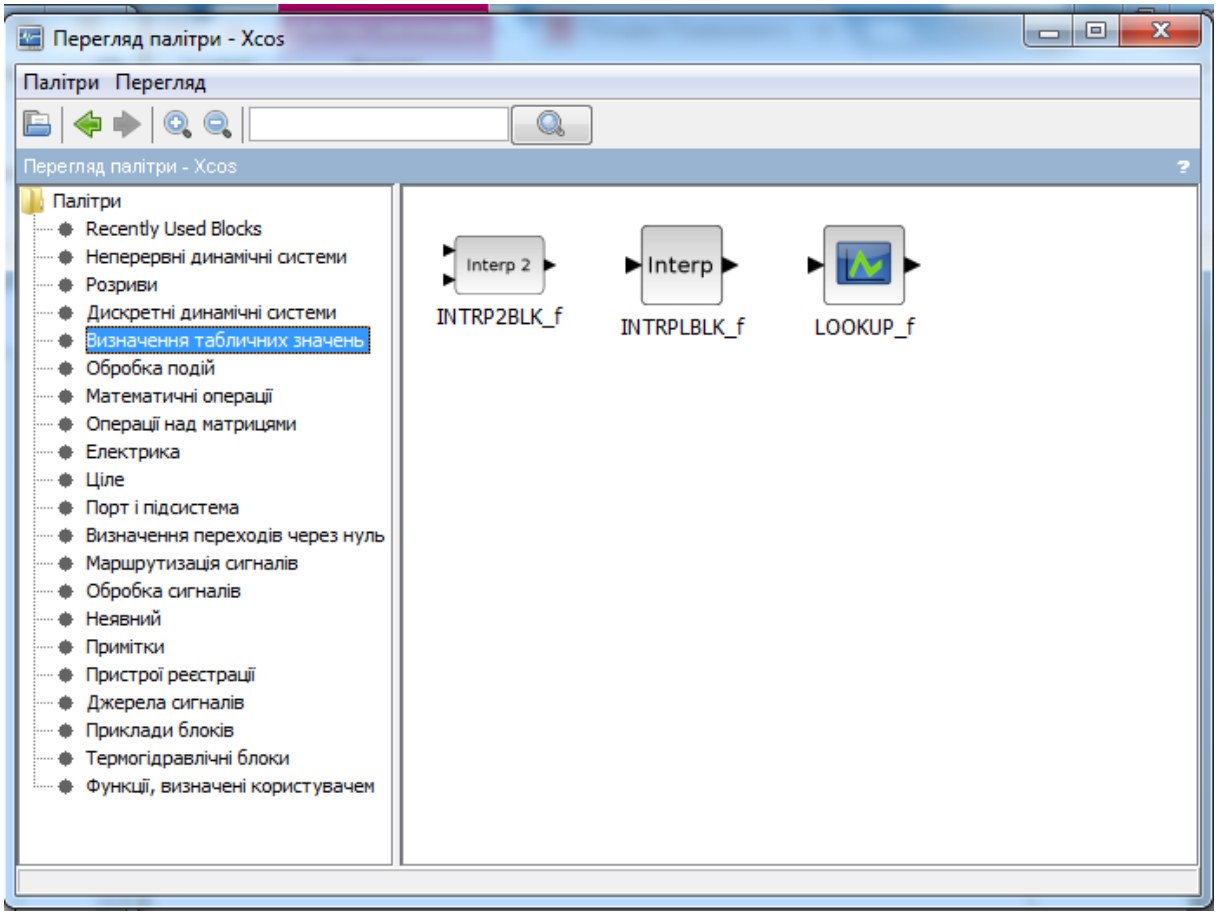


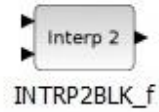
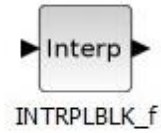
 AUTOMAT	Блок дає можливість будувати гібридні автомати, тобто гібридні системи, чия дискретна частина визначена способами і переходами між способами, а безперервна частина визначена через диференціальні рівняння алгебри. Блок забезпечує автоматичне перемикання між підсистемами. Підсистеми побудовані таким способом, що вони мають стаціонарний вхідний вектор і обчислюють плавну та стрибкоподібну функції (перетин нуля) й передають їх назад до блока автомата. Стаціонарні змінні визначені в блоці автомата, підсистеми – статичні функції.
 DELAYV_f	Блок Variable Delay може бути використаний для імітації змінної часу затримки між дією та її ефектом. (Аналогічний блок є у палітрі «Розриви»)
 DELAY_f	Блок реалізує дискретну затримку. Вона побудована з регістром зсуву та годинником. Значення затримки визначається часом дискретизації, помноженим на значення стану регістру.

	Дискретна передатна функція. Блок реалізує лінійну дискретну систему, подану раціональною функцією типу $(1 + a_1 z + a_2 z^2 + \cdots)/(1 + b_1 z + b_2 z^2 + \cdots)$
	Система, яка подана нулями та полюсами дискретної передавальної функції.
	Дискретна система в просторі станів. Мають бути задані матриці A, B, C, D і початковий стан x_0. Розміри мають бути сумісними. Після надходження вхідної «події» на порт «події», стан входу оновлюється
	Блоки еквівалентні оператору затримки дискретного часу 1/z. Блок приймає один вхід і генерує один вихід, який може бути або скалярним, або вектором. Якщо вхід є вектором, всі елементи вектора затримуються однаково. Вхідна величина подається на вихід за сигналом активації, а після цього на вході запам'ятовуються нові вхідні величини
	
	
	Відтворення на виході вхідного сигналу і зчитування нового вхідного за сигналом активації. Кожного разу, коли на блок надходить «подія», блок копіює свій вхід на виході і тримає до наступної «події». Для періодичної вибірки та утримання вхід «події» має бути згенерований годинником

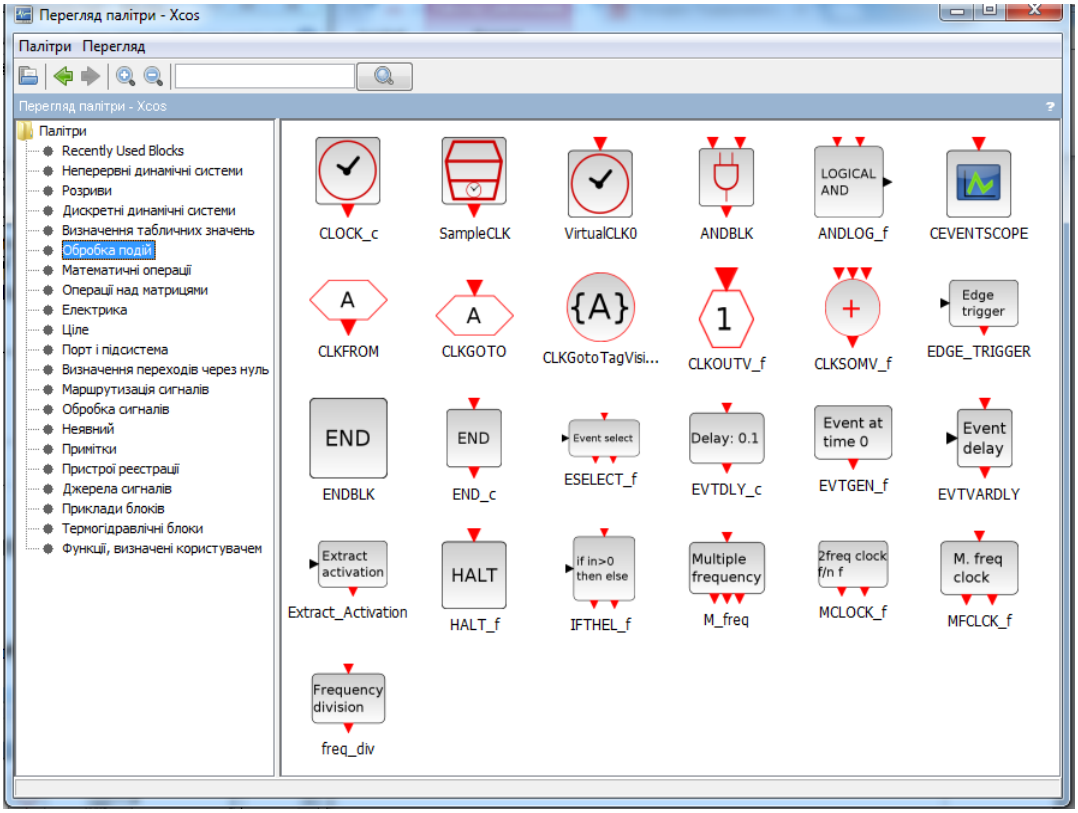
	Блок реалізує безперервну лінійну систему з можливістю стрибків у стані. Кількість входів до цього блока становить два. (Див. аналогічний блок у палітрі «Неперервні динамічні системи»)
	Зсувний регістр Блок реалізує регістр зсуву. На кожній вхідній «події» регістр зміщується на один крок

5.4 Lookup Tables (Визначення табличних значень)

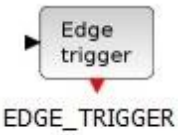
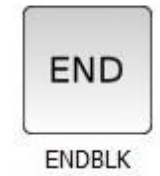
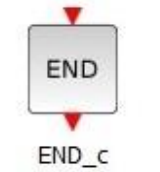
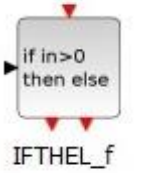


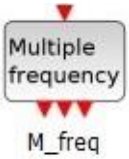
	2D Інтерполяція (інтерполяція поверхні)
	Інтерполяція функції однієї змінної. Вихід блока є функцією вхідних даних, отриманих шляхом лінійної інтерполяції. Блок має один скалярний вхід і один скалярний вихід
	Блок реалізує нелінійну функцію, задану точками за допомогою графічного редактора

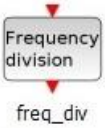
5.5 Event handling (Обробка подій)



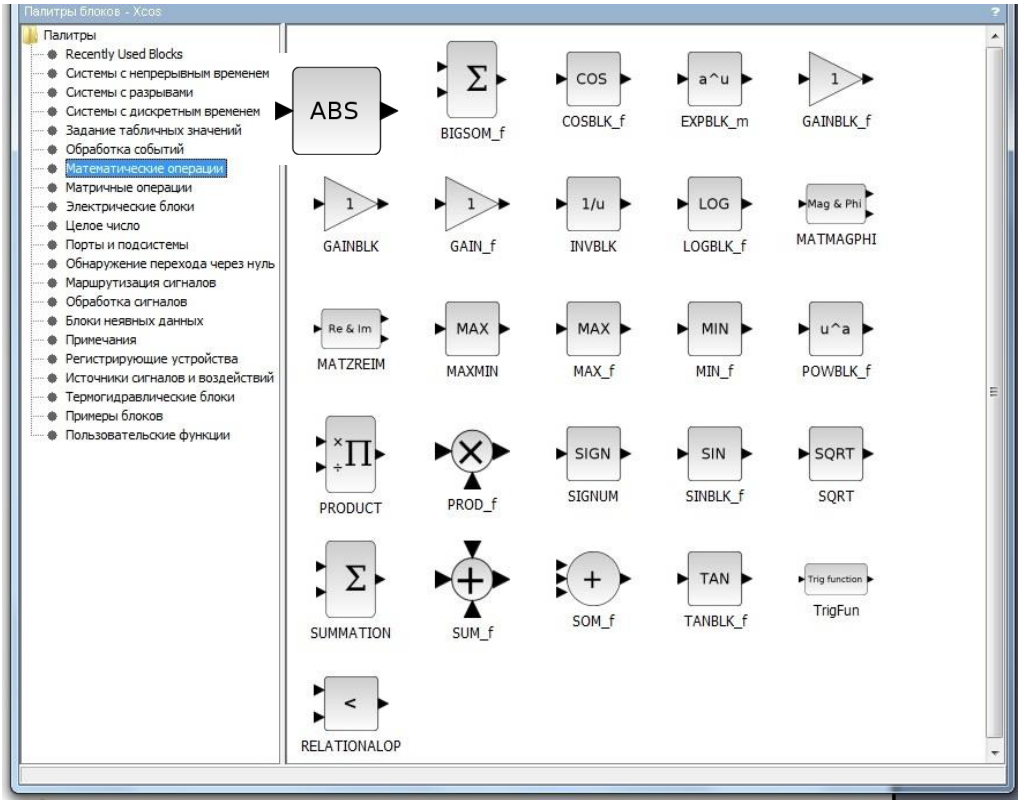
	Годинник активації. Встановлюється крок роботи і час початку роботи Вихід цього блока генерує регулярний потік «подій» з параметром Period в секундах. Початкову дату генерації «подій» можна встановити в секундах за допомогою параметра «Час ініціалізації».
	Годинник активації. Встановлюється крок роботи і час початку роботи Різниця між SampleCLK і CLOCK_c полягає в тому, що всі блоки SampleCLK в схемі є синхронними
	Блок вважається віртуальним блоком компілятора. Він використовується в SuperBlock (підсистеми) для запуску завжди активних блоків (наприклад, синусоїдального генератора) у SuperBlock і на рівні нижче
	Блок формує «подію» на виході за наявності на вході двох «подій» одночасно
	Блок формує +1, якщо «події» прийшли на обидва входи разом і -1, якщо прийшла тільки одна «подія»
	Перегляд подій (сигналу активації)

	CLKGOTO Блок використовується в разі «події». (Див. також GoTagVisibility блок палітри «Маршрутизація сигналів»)
	Блок підсумовує до трьох «подій». Вихід відтворює вхідні «події». Вхід і вихід синхронні
	Блок генерує «подію» на збільшення, зменшення або і те, і інше (див. установлення параметрів). *Він реагує тільки на стрибок, який збігається з «подією».* Вихідна «подія» синхронна з «подією», що викликала стрибок
	Блок може бути використаний для установлення кінцевого часу роботи моделі. В цьому випадку модель буде зупинена за даними цього блока, а не за установленням в ***Моделювання – Налаштування***. Цей параметр може бути числом або змінною в контексті Xcos
	Блок збільшує поточний час до остаточного часу моделювання
	Комутація «подій». Блок синхронізації If-Then-Else, тобто синхронізація за першим чи другим виходом залежно від виконання або невиконання умови. Вхідні та вихідні «події» синхронізовані
	

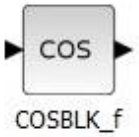
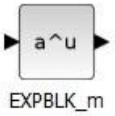
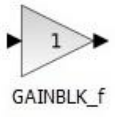
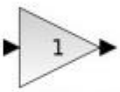
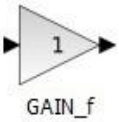
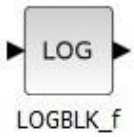
 EVTDLY_c	Затримка «подій» Одна «подія» створюється затримкою після того, як «подія» входить до унікального порту вхідних «подій». Початкова вихідна «подія» також визначається за допомогою параметрів
 EVTGEN_f	Генератор «події». Генерує одну «подію» у встановлений час
 EVTVARDLY	Змінна затримка «події». Після надходження «події» на вхід активації, вона затримується на величину, яка визначається сигналом, що надходить на сигнальний вхід. Блок може також генерувати початкову «подію» на виході
 Extract_Activation	Формування сигналу активації («події») з вхідного сигналу
 HALT_f	Блок має один вхідний порт «подій». «Події» моделювання зупиняються і активується головне вікно Xcos. Моделювання можна перезапустити або продовжити (кнопка «Виконати»). Використовується переважно для відлагодження
 M_freq	Блок створює «події» в певні моменти часу роботи моделі. Періоди задаються в полі «Sample Time», а часові зсуви в полі «Offset». Блок має один вхід, а кількість виходів залежить від кількості заданих моментів часу. Наприклад, якщо вектор часу [1; +1; 2] і вектор зсуву – [0; 0.5; 0], тоді блок має 7 виходів
 MCLOCK_f	Дільник частоти. На виході 1 – «події» з частотою f/n, на виході 2 – з частотою f. Параметр n задається у налаштуваннях

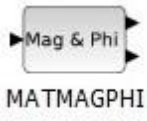
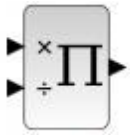
	Множник частоти
	Ділення частоти Вхід керується годинником «події»

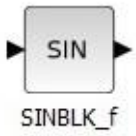
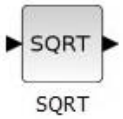
5.6 Mathematical Operations (Математичні операції)

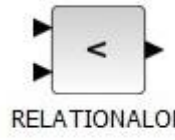
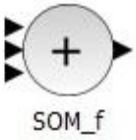


	Абсолютна величина

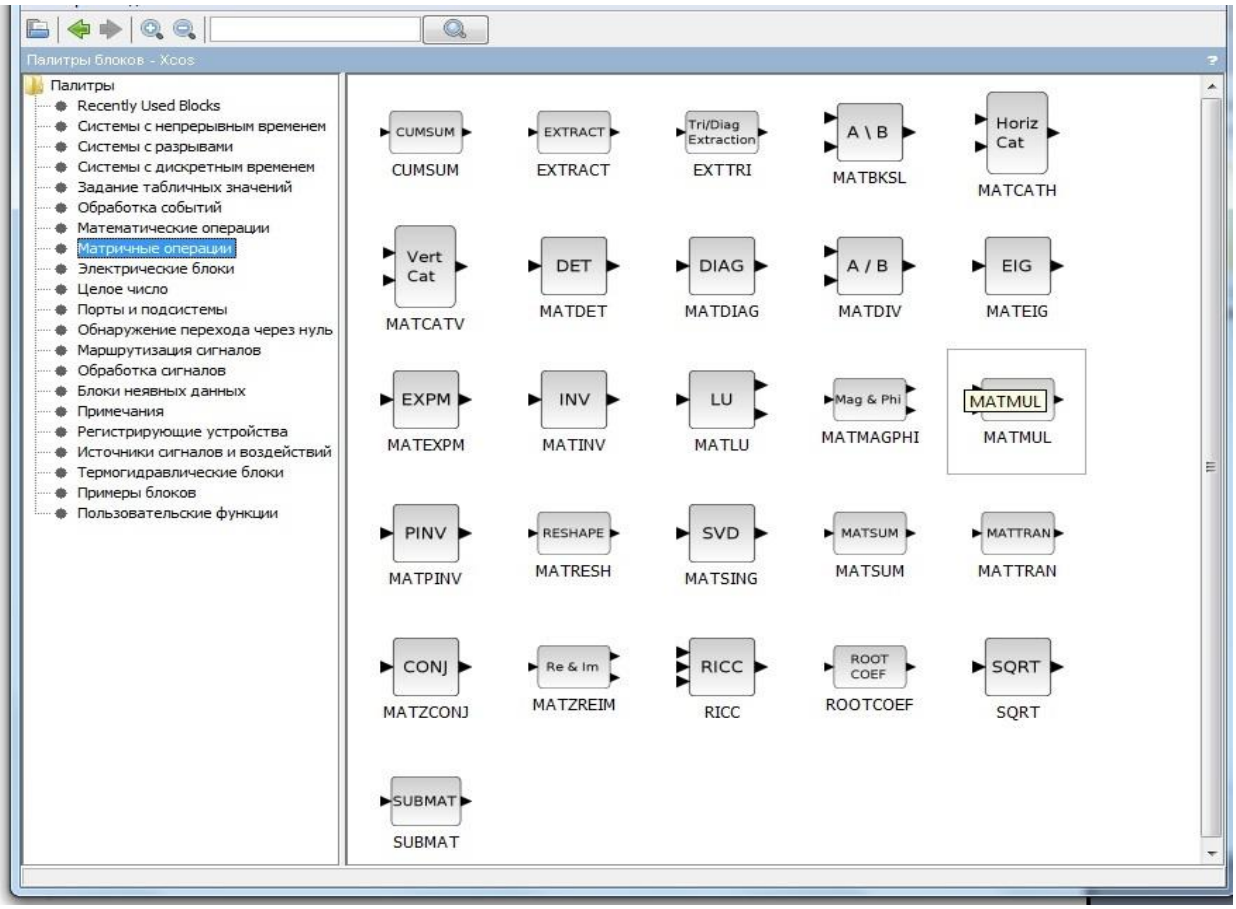
	Сума або різниця Користувач може регулювати коефіцієнт для кожного вводу за допомогою параметра: $y = k_1u_1 + k_2u_2 + k_3u_3 + \cdots$
	Косинус
	Експонента Виходом цього блока є вектор y з $y[i] = a^{u[i]}$, де параметр a є позитивним скаляром і u – вхідним вектором
	Множення на коефіцієнт Блок обчислює добуток квадратної матриці A на вхідний вектор U, де кількість рядків / стовпців A дорівнює кількості рядків U. Матриця A встановлюється параметром Gain
	
	
	Обернене значення Блок обчислює вихідний вектор 1/y, де y – вхідний вектор. Розмір вводу та виводу визначається контекстом
	Логарифм Користувач може встановити основу логарифма за допомогою параметра «Основи». За замовчуванням блок обчислює натуральний логарифм. Розмір вхідного та вихідного портів визначається контекстом

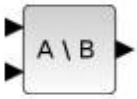
 MATMAGPHI	Блок виробляє два типи перетворення. 1. Блок перетворює комплексне число до модуля і фази в радіанах, в цьому випадку вхід комплексний, а вихід – два дійсних числа. Якщо вхід два дійсних числа, то вихід за кутом від 0 до π, а за величиною модуля – модуль вхідного числа. 2. Обернене перетворення
 MATZREIM	Комплексна композиція/декомпозиція Блок розкладає матрицю комплексних чисел (або окреме число), розділяючи дійсні та уявні частини, або складає матрицю комплексних чисел, об'єднуючи дві частини
 MAXMIN	Блок видає на вихід найбільший або найменший елемент, або елементи вхідного вектора. Функцію можна вибрати у параметрах
 MAX_f	Блок видає на вихід найбільший елемент або елементи вхідного вектора
 MIN_f	Блок видає на вихід найменший елемент або елементи вхідного вектора
 POWBLK_f	Степінь. Блок реалізує операцію $y[i] = (u[i])^a$
 PRODUCT	Множення або ділення Блок обчислює множення або ділення векторних входів. Кількість входів та операції вказуються за допомогою параметра «Кількість входів». Операція ділення чи множення визначається знаком у параметрі «кількість входів»: +1 (множити) або -1 (ділити) для кожного входу

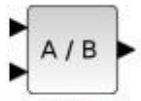
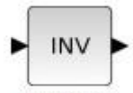
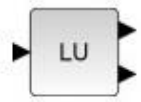
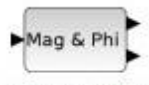
	Множення Вихід цього блока є поелементним добутком двох вхідних векторів
	Функція Signum Вихід блока є вектором знаків елементів вхідного вектора. Для кожного елемента вхідного вектора елемент вихідного вектора: • 1, коли елемент більше нуля; • 0, коли елемент дорівнює нулю; • -1, коли елемент менше нуля
	Синус
	Квадратний корінь Він підтримує комплексний тип даних
	Блок виконує підсумовування або віднімання вхідних сигналів. Блок може підсумовувати або віднімати скаляр, вектор або матриці. Число входів задається параметром Number of inputs. Цей параметр складається з плюс і мінус одиниць або це може бути додатна величина (скаляр). У першому випадку кількість одиниць вказує число входів, а знаки вказують, чи є це підсумовуванням або відніманням. У другому випадку – це блок підсумовування і величина вказує число входів
	Додавання Блок виконує додавання трьох його входів. Блок може додати скалярні або векторні входи
	Тангенс

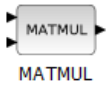
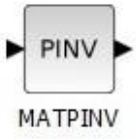
	Тригонометрична функція Тригонометричний функціональний блок виконує численні загальні тригонометричні функції
	Операція порівняння за величиною. Вид операції задається у налаштуваннях
	Не рекомендований до використання

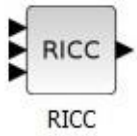
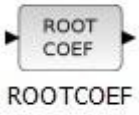
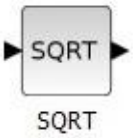
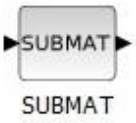
5.7 Matrix (Операції над матрицями)



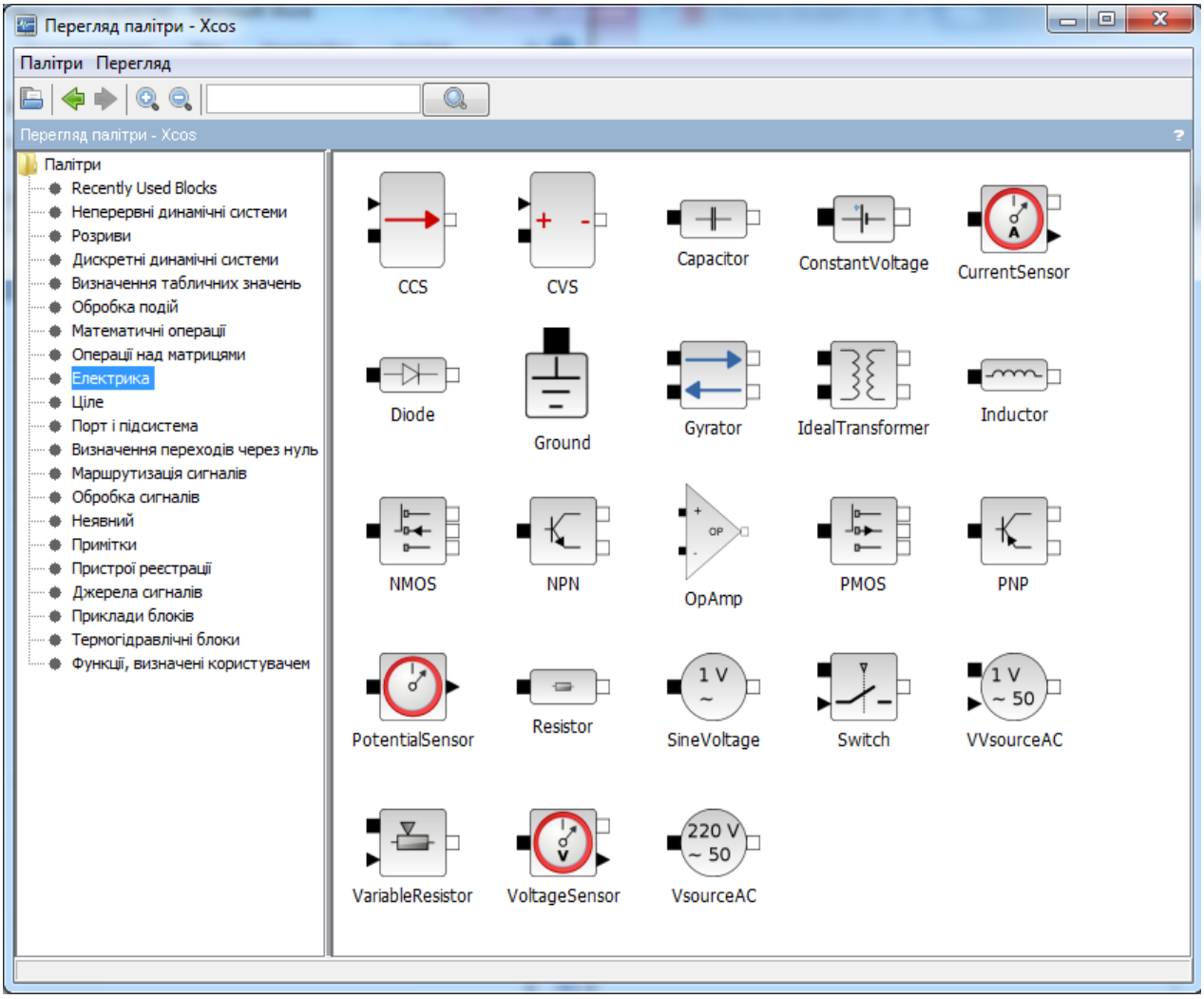
 CUMSUM	Кумулятивна сума Блок CUMSUM накопичувально підсумовує елементи вхідної матриці U розміром M × N по рядках, стовпцях або до першого не-одиночного виміру. Розмір виходу також M × N
 EXTRACT	Екстракція матриці Блок ЕКСТРАКТ витягує деякі елементи з вхідної матриці. Розмір виводу залежить від кількості рядків і кількості стовпців для екстракції. Елементи параметрів Lines to extract і Columns to extract визначають відповідно індекси рядків і стовпців, які потрібно видобути. Написання цих параметрів відповідає правилам вилучення, зокрема ви можете вказати ряд індексів з оператором діапазону
 EXTTRI	Трикутна або діагональна екстракція Залежно від параметра у вихідну матрицю копіюються елементи, що лежать вище, нижче або на головній діагоналі. Розмір вихідної матриці такий самий, як і вхідної матриці
 MATBKSL	Ліве матричне ділення Це розв’язання рівняння (A * x = B). Вищий вхід – це матриця A, нижній – матриця B, а вихід – x. Якщо A є матрицею M × N1, то B має бути матрицею M × N2, де N1 та N2 можуть бути різними або однаковими. Вихід x є матрицею N1×N2
 MATCATH	Горизонтальна конкатенація (горизонтальне об’єднання). Це також називається конкатенацією відповідно до колонок. Потрібно, щоб входи U1, U2, ..., Un мали однакову кількість рядків M. Користувач має встановити кількість вхідних матриць у параметрі «Кількість вхідних даних». Результат буде мати розмірність M×(N1+N2+…+Nn)
 MATCATV	Блок MATCATV виводить вертикальне об'єднання декількох матриць. Це також називається конкатенацією відповідно до рядків. Потрібно, щоб входи U1, U2, ..., Un мали однакову кількість стовпців N. Користувач має встановити кількість вхідних матриць у параметрі «Кількість вхідних даних». Результат буде мати розмірність (M1+M2+…+Mn) ×N
 MATDET	Детермінант матриці

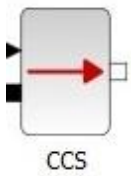
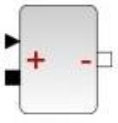
 MATDIAG	Створення діагональної матриці Блок MATDIAG створює діагональну матрицю з вектора. Якщо вхід є вектором M × 1, то вихід – матриця M × M
 MATDIV	Праве матричне ділення Це розв'язок рівняння (x * B = A). Вищий вхід – це матриця A, нижня – матриця B, а вихід – x
 MATEIG	Власна матриця Блок MATEIG обчислює власні значення та власні вектори квадратної вхідної матриці U. Коли тип розпаду встановлюється на: • 1, блок виводить власне значення в векторній формі, якщо вхід є матрицею M × M, вихід є вектором M × 1. • 2, блок виводить дві матриці. Для матриці вводу M × M перший вихід є діагональною матрицею M × M, складеною власними значеннями, а друга – матрицею M × M, де стовпці є власними векторами.
 MATEXPM	Експоненціальна матриця MATEXPM виводить експоненціальну матрицю квадратної вхідної матриці, обчислених апроксимацією Паде. Вихід являє собою квадратну матрицю з однаковим розміром входу
 MATINV	Обернення матриці Блок MATINV виводить обернену квадратну вхідну матрицю, яка розраховується за допомогою факторизації LU. Попередження виводиться, якщо вхід погано масштабується або майже однозначний
 MATLU	Факторизація Блок MATLU виводить дві матриці L і U, з поворотом рядків, з LU факторизацією квадратної вхідної матриці. Якщо A є вхідною матрицею, то: E * A = L * U, де E – матриця перестановки, U – верхня трикутна матриця, L – нижня трикутна матриця
 MATMAGPHI	Блок виконує два типи перетворення: 1. Блок перетворює елементи матриці комплексних чисел на модуль і кут у радіанах, у цьому випадку вхід комплексний, а вихід – два реальних числа. 2. Обернене перетворення

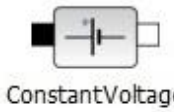
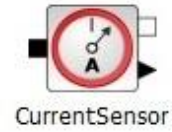
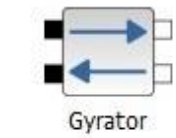
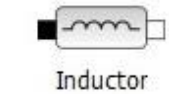
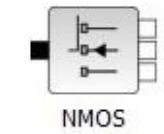
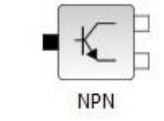
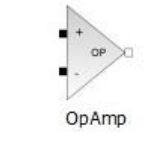
	Матричне множення
	Псевдообернення матриці
	Блок змінює розмірність матриці або вектора на задану в полі «output size desired». Вихідна розмірність може бути меншою або дорівнювати вхідній
	Блок MATSING розкладає матрицю A(M × N) на три матриці U, S та V такі, що A = U * S * V '
	Сума елементів матриці Блок MATSUM повертає суму елементів вхідної матриці / вектора. Коли параметр суми встановлено на: • 0 (все) блок виводить суму всіх елементів матриці. Вихід є скаляром; • 1 (рядки) блок – рядова сума. Вихід – це вектор рядка; • 2 (Стовпці) блок – це стовпчаста сума. Вивід – векторний стовпець
	Транспонування матриці
	Замінює комплексні елементи матриці на спряжені

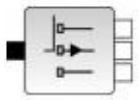
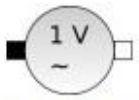
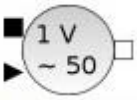
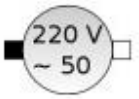
	Комплексна декомпозиція Блок розкладає матрицю комплексних чисел, розділяючи дійсні та уявні частини або складаючи матрицю комплексних чисел, об'єднуючи дві частини
	Розв’язання рівняння Ріккаті Блок обчислює розв'язок рівняння Ріккаті за допомогою різних методів і для безперервного та дискретного випадків
	Обчислення поліноміальних коефіцієнтів Блок вираховує коефіцієнти полінома з заданими його кореневими значеннями. Ці корені подаються на вхід у вигляді вектора-стовпця. Довжина вектора кореня має бути вказана в параметрі розміру рядка вхідних даних
	Квадратний корінь кожного елемента матриці. Див. «Математичні операції»
	SUBMAT – підматриця Блок виводить підматрицю вхідної матриці. Розміри вхідної матриці вказані в параметрах вводу. Користувач задає діапазон з чотирма параметрами: • параметри Starting Row Indicator та Ending Row Index визначають діапазон рядків, які потрібно видобути; • параметри Starting Column Index та Ending Column Index визначають діапазон стовпців, який потрібно видобути

5.8 Electrical (Електрика)

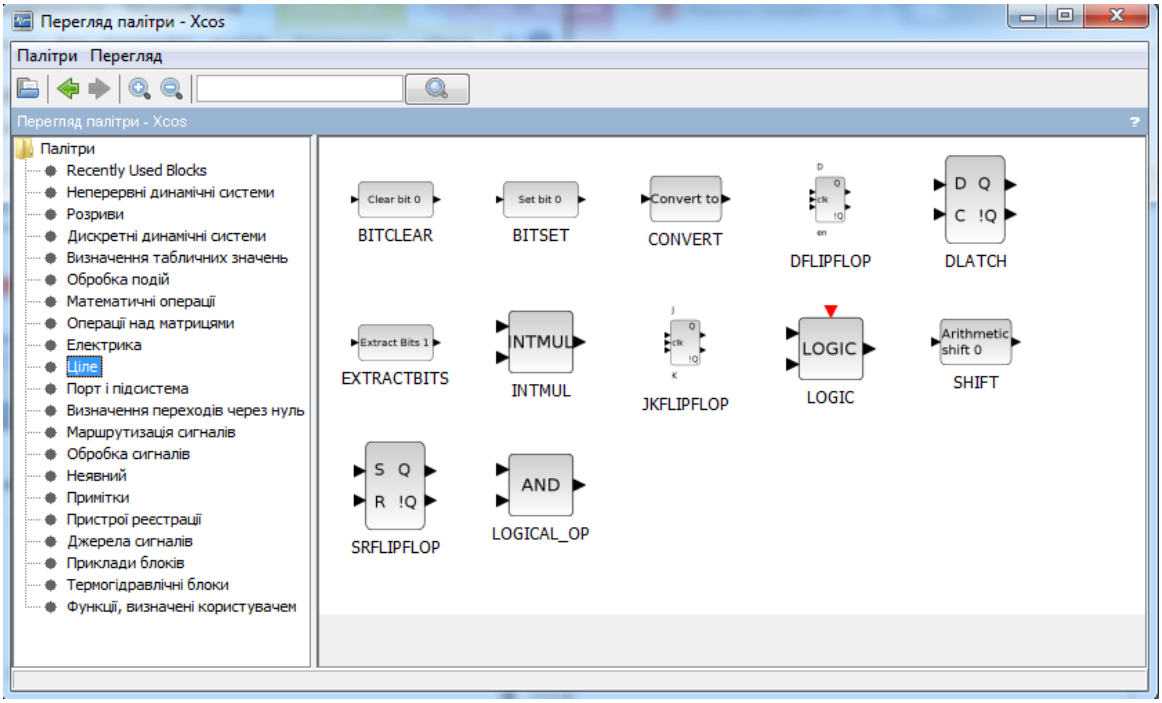


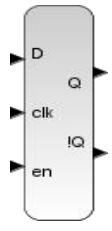
 CCS	Кероване джерело струму Блок є ідеальним джерелом струму. Поточне значення контролюється через вхід блока
 CVS	Кероване джерело напруги Блок є ідеальним джерелом напруги. Значення напруги регулюється через вхід.
 Capacitor	Електричний конденсатор

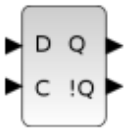
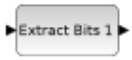
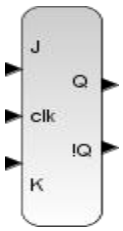
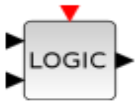
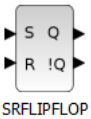
	Джерело постійної напруги Вихідна напруга цього джерела напруги постійного струму визначається користувачем. Чорний порт вказує на позитивний полюс. Опір цього джерела напруги дорівнює нулю
	Амперметр Напрямок від чорного до білого порту вважається позитивним. Опір цього блоку дорівнює нулю
	Діод
	Заземлення (точка нульового потенціалу) Кожна електрична схема має містити принаймні один заземлений елемент
	Фазообертач (Гіратор)
	Ідеальний трансформатор
	Індуктор електричний (котушка індуктивності)
	Транзистор Модель NMOS є простою моделлю n-канального польового транзистора
	Транзистор n-p-n
	Операційний підсилювач

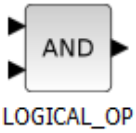
	Транзистор Модель PMOS є простою моделлю p-канального польового транзистора
	Транзистор p-n-p Ця модель являє собою просту модель біполярного транзистора
	Потенціалоскоп Блок використовується для вимірювання напруги відносно опорної напруги (земляний блок) у електричному колі
	Резистор
	Джерело синусоїдальної напруги. Частота і амплітуда задаються у налаштуваннях
	Неідеальний електричний ключ Якщо вхід стає позитивним, два входи з'єднані через резистор опору R_{ON}. В іншому випадку – два входи з'єднані через опір R_{OFF}
	Регульоване джерело змінної напруги Джерело регульованої напруги змінного струму. Амплітуда вихідної напруги регулюється входом даних, а частота визначається у налаштуваннях. Опір блока дорівнює нулю
	Електричний змінний резистор Опір (Rx) керується через вхідний порт
	Вольтметр Цей компонент використовується для вимірювання напруги між двома вузлами в електричній схемі.
	Електричне джерело змінної напруги Амплітуда та частота вихідної напруги встановлюються користувачем. Опір цього блока дорівнює нулю

5.9 Integer (Ціле)

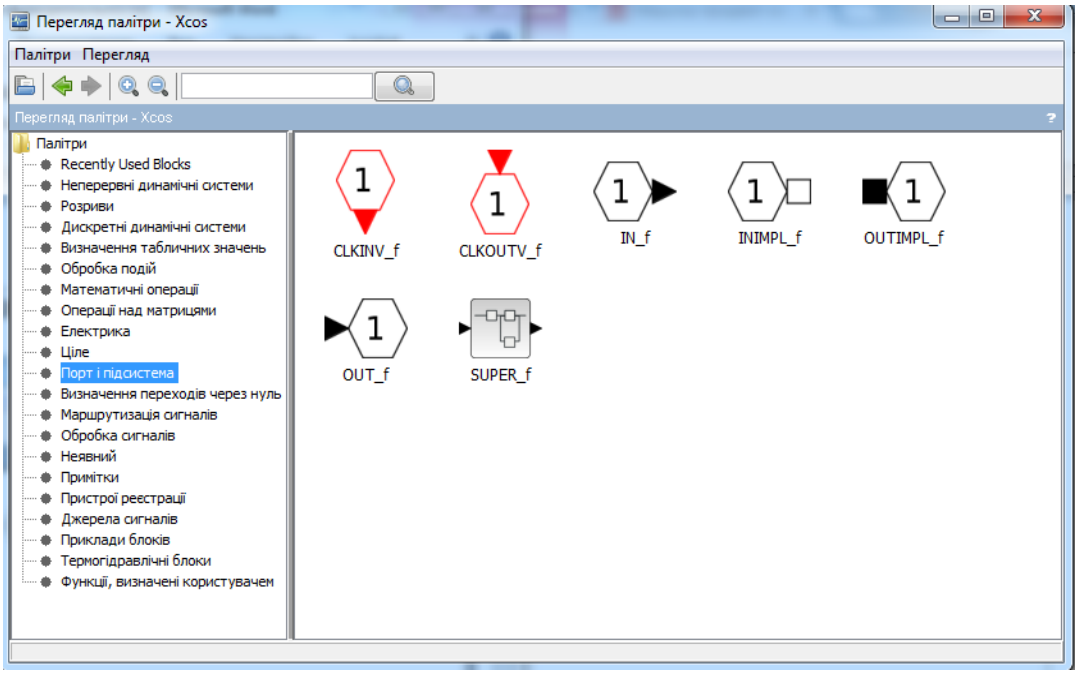


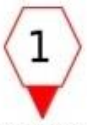
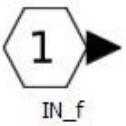
 BITCLEAR	Очищення бітів Блок задає значення 0 вказаного біта його вводу. Користувач дає індекс біта в поле Index of Bit
 BITSET	Установлення бітів Блок задає значення 1 вказаного біта його вводу. Користувач дає індекс біта в поле Index of Bit
 CONVERT	Перетворення типу даних Блок перетворює вхідний сигнал реального подвійного чи цілого типу даних у ціле чи дійсний подвійний тип даних
 DFLIPFLOP	D –Тригер з дозволом Блок виводить на Q його вхідний стан (D), коли увімкнено вхід (en) та на зростаючому фронті вхідного сигналу годинника (clk). Вихід !Q є інверсією Q. Цей тригер також відомий як затримка тригера, оскільки стан виходів змінюється лише на наступному зростаючому фронті годинника

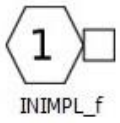
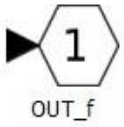
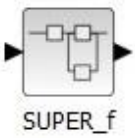
 DLATCH	D – тригер Блок копіює свій стан вводу (D) на виході (Q), коли вхідний вхід (C) є високим. Вихід !Q є інверсією Q
 EXTRACTBITS	Блок виводить набір суміжних бітів вхідного числа
 INTMUL	Матричне множення цілого Блок обчислює матричне множення двох матриць цілих чисел. Кількість рядків другої матриці має дорівнювати кількості стовпців першої матриці
 JKFLIPFLOP	JK -тригер Тригер JK є найуніверсальнішим з тригерів. Він має два входи, традиційно позначені як J (Set) та K (Reset). Зміна стану відбувається за синхросигналом
 LOGIC	Блок комбінованої логіки Блок реалізує задану таблицю істинності для моделювання цифрової схеми та інших логічних операцій
 SHIFT	Зсувний регістр Блок змінює біти вхідного сигналу. У цій операції цифри рухаються вправо або вліво. Користувач може вибрати правило, яке змінює біти. Кількість та напрямок зсувів встановлюються з кількістю бітів для переміщення вліво. Якщо цей номер додатний, вхід зміщений уліво, інакше його буде зміщено вправо
 SRFLIPFLOP	RS тригер. Вихід Q залежить від стану входів S і R. Вихід !Q – інверсія Q. R=S=0 – вихід не змінюється; R=S=1 – недозволений стан

	Логічна операція (установлення виду операції в параметрах) Блок логічного оператора виконує вказану логічну операцію над його входами

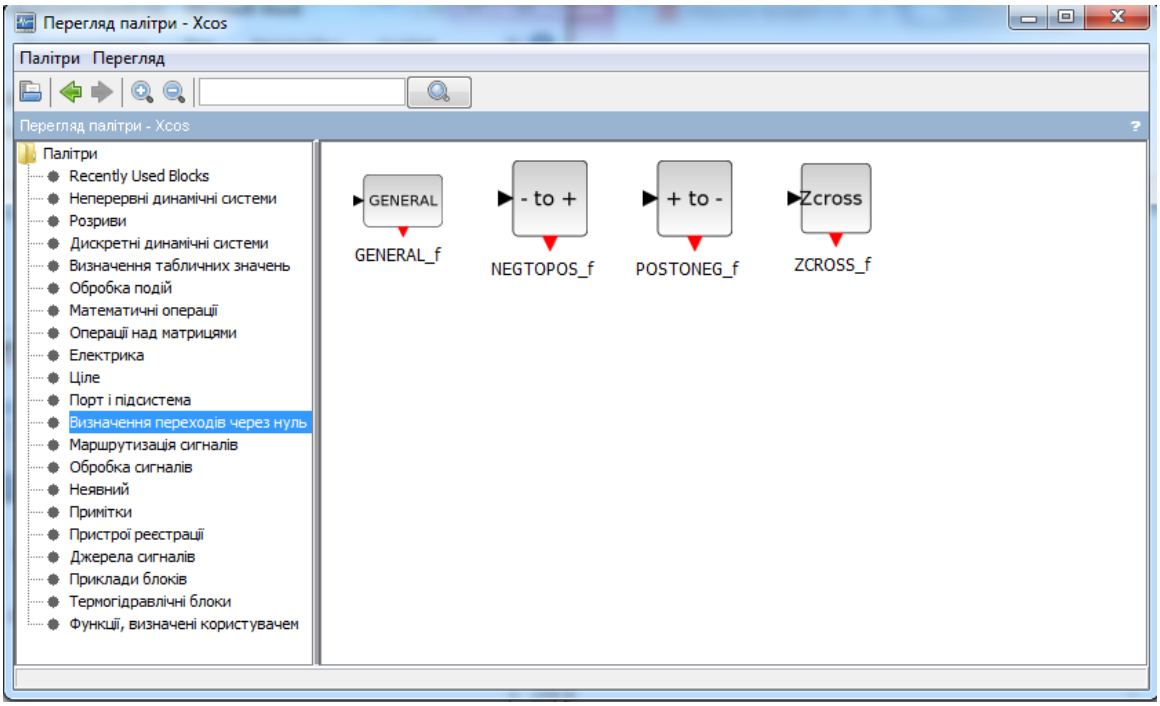
5.10 Port & Subsystem (Порт і підсистема)

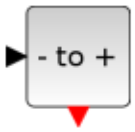


	Вхідний порт активації. Він має використовуватися тільки усередині суперблоків. Якщо цих входів кілька, вони нумеруються підряд, починаючи з одиниці
	Вихідний порт активації Його потрібно використовувати лише усередині суперблоків
	Вхідний порт Він має використовуватися лише усередині суперблоків

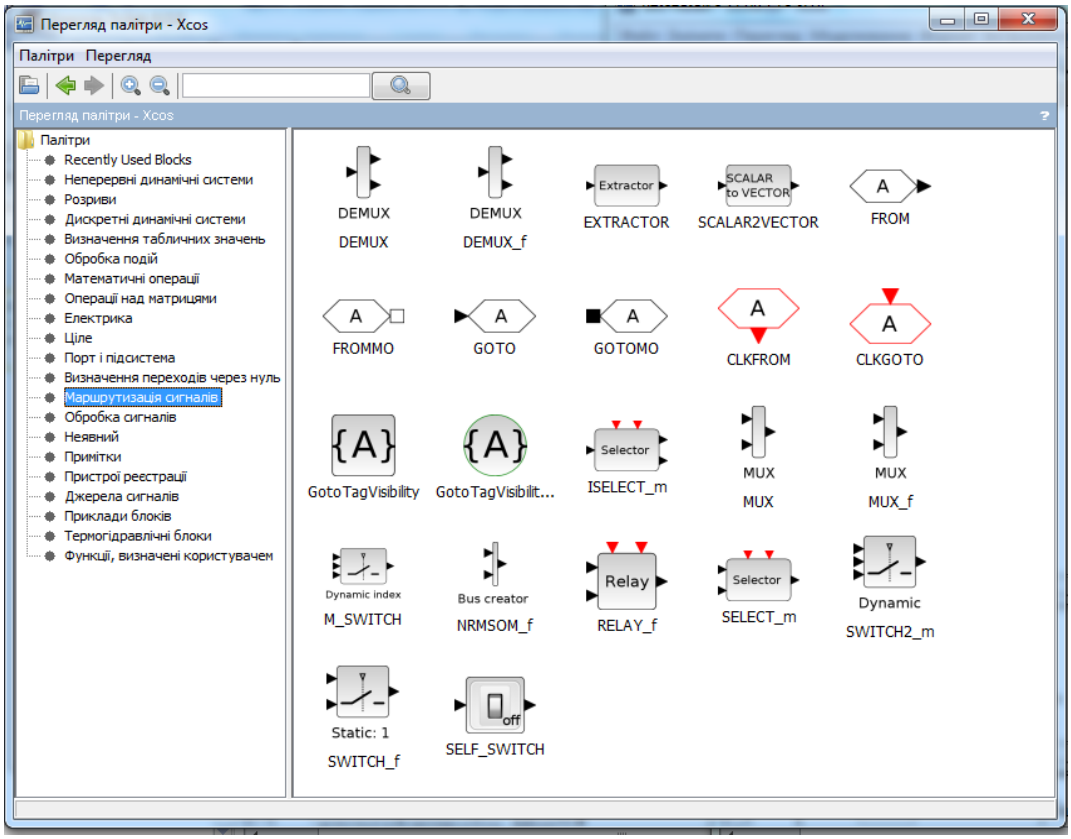
	Вхідний порт, вхід в систему ззовні Блок зображає неявний вхідний порт. Використовується разом з моделями електричних, гідравлічних та подібних блоків. Він має використовуватися лише усередині суперблоків
	Вихідний порт Використовується в разом з моделями електричних, гідравлічних та подібних блоків. Він має використовуватися лише усередині суперблоків
	Вихідний порт Його потрібно використовувати лише усередині суперблоків
	Створення суперблока Блок відкриває нове вікно з шаблоном суперблока

5.11 Zero crossing detection (Визначення переходів через нуль)



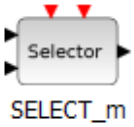
	Блок видає задану кількість сигналів активації (подій)
	Перетин нуля від мінуса до плюса Вихідна подія генерується, коли вхід перетинає нуль у позитивному напрямку
	Поріг перетину нуля від плюса до мінуса Вихідна подія генерується, коли вхід перетинає нуль з негативному напрямку
	Виявлення перетину нуля Вихідна «подія» генерується, коли всі входи (якщо їх більше ніж один) перетинають нуль одночасно у будь-якому напрямі

5.12 Signal Routing (Маршрутизація сигналів)

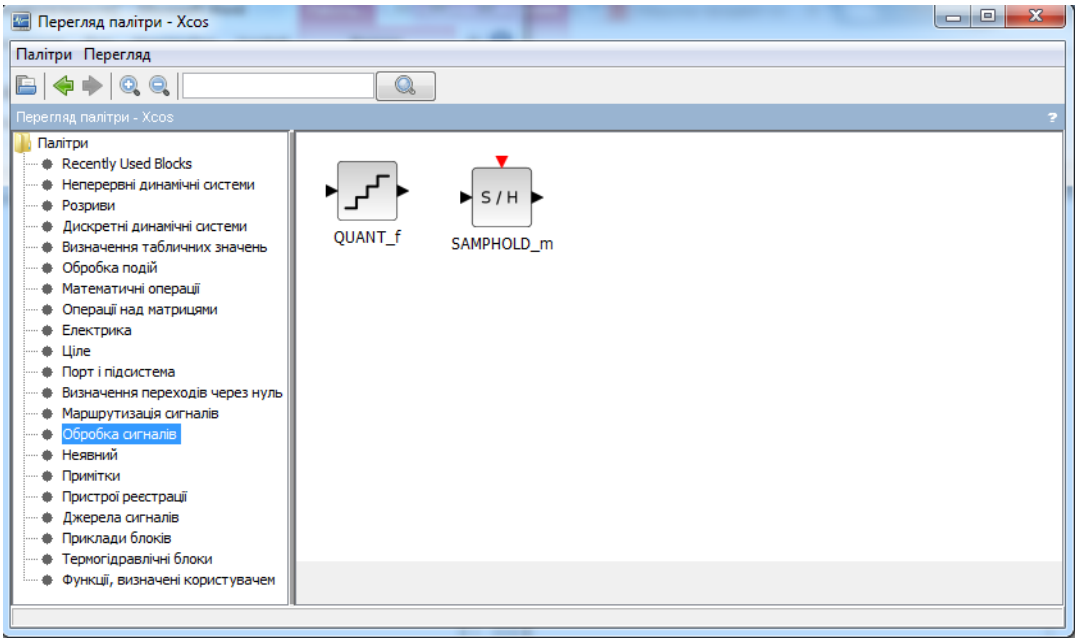


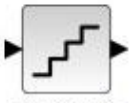
 DEMUX  DEMUX DEMUX_f	Демультиплексор Блок розподіляє вхідний вектор на кілька векторних виходів. Кількість виходів задається параметром. Розмір вхідного та вихідного порту визначається контекстом
 EXTRACTOR	Виділяє один сигнал з векторного вхідного сигналу
 SCALAR2VECTOR	Перетворення скаляра на вектор
 FROM	Прийом даних від відповідного GOTO Основна роль блоків GOTO / FROM полягає в передачі сигналів від блока до іншого блока без фізичного підключення. Використовується для спрощення великих моделей
 FROMMO	Прийом даних від відповідного GOTOMO Використовується для спрощення великих моделей електричних, гідравлічних та інших подібних систем
 GOTO	Передача даних блоку FROM Основна роль блоків GOTO/FROM полягає в передачі сигналів від блока до іншого блока без фізичного підключення. Використовується для спрощення великих моделей
 GOTOMO	Передача даних блоку FROMMO Використовується для спрощення великих моделей електричних, гідравлічних та інших подібних систем
 CLKFROM	Прийом даних від відповідного CLKGOTO Використовується для з'єднання блоків «подій». В параметрах задається «область видимості»
 CLKGOTO	Передача даних відповідному блоку CLKFROM Використовується для з'єднання блоків «подій»

	Визначають область видимості етикетки CLKGOTO Блок визначає доступність блока GOTO, коли він налаштований як «обладнаний». Блок FROM, що відповідає цьому GOTO, має знаходитись в тій самій підсистемі GotoTagVISibility або в підсистемах, що знаходяться нижче в ієрархії моделі
	Блок використовується у випадку з'єднання з системою Modelica
	Вибирає і передає на вихід значення вхідного сигналу. Перший вхід «подій» керує першим виходом, другий – другим виходом
 	Мультиплексор Блок об'єднує входи в єдиний вихідний вектор. Розмір вхідного та вихідного портів визначається контекстом
 	Блок пропускає сигнал через перший (верхній) вхід або через третій (нижній), залежно від величини сигналу на середньому (другому) керівному вході. Можна вибрати критерій (поріг) підключення до першого входу
	Злиття даних Блок об'єднує свої входи в єдину лінію виходу, значення якого в будь-який момент часу дорівнює значенню входу, який надійшов останнім
	Реле Блок спрямовує один з входів на один вихід. Вибір того, який вхід має бути, спочатку виконується за допомогою параметра «початковий під'єднаний вхід». Потім, кожного разу, коли вхідний потік надходить до і-го порту «події» вводу, і-й черговий вхідний порт перенаправляється до виходу

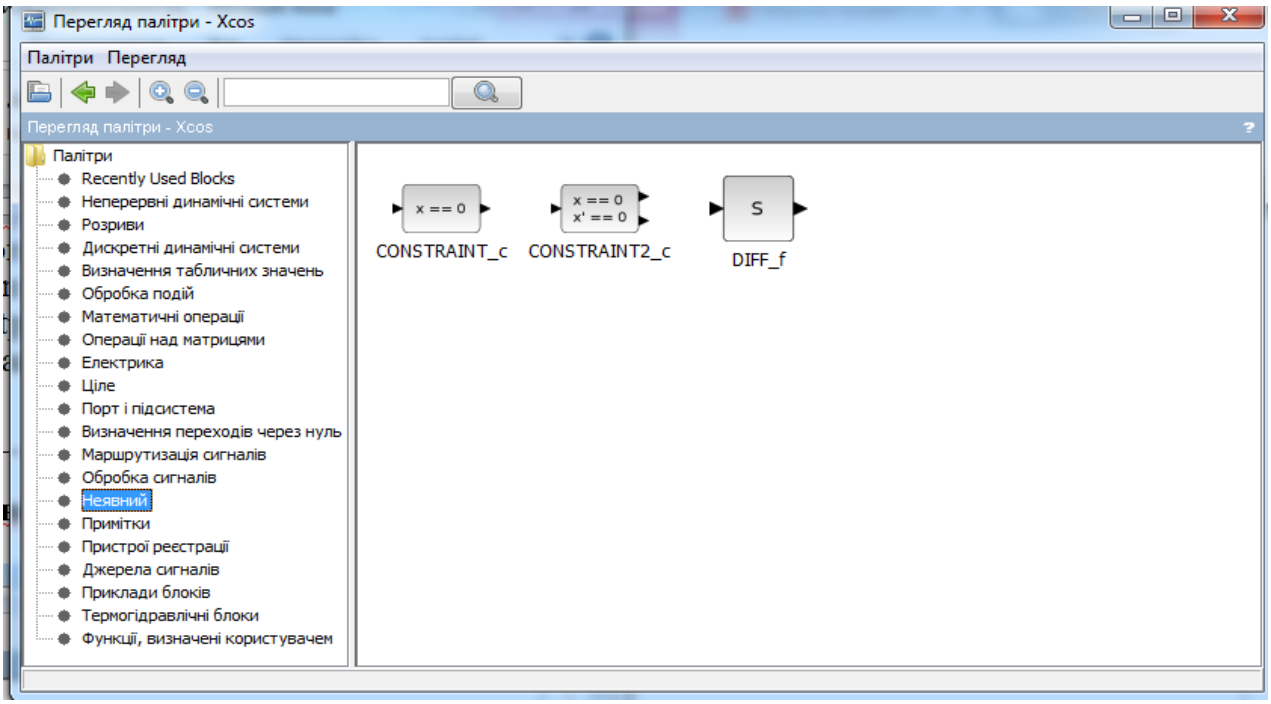
 SELECT_m	Блок приймає векторні або матричні сигнали і передає на вихід той, номер якого дорівнює номеру входу «події», яка надійшла
 SWITCH_f	Блок ручного перемикання. В параметрах задається кількістю положень перемикача, і в якому становищі перемикач знаходиться
 SELF_SWITCH	Блок ручного перемикання. Двічі клацніть по ньому, щоб переключити його стан (вкл / викл)

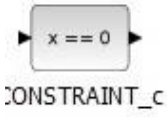
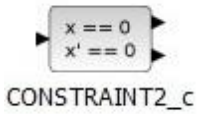
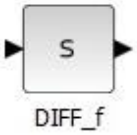
5.13 Signal Processing (Обробка сигналів)



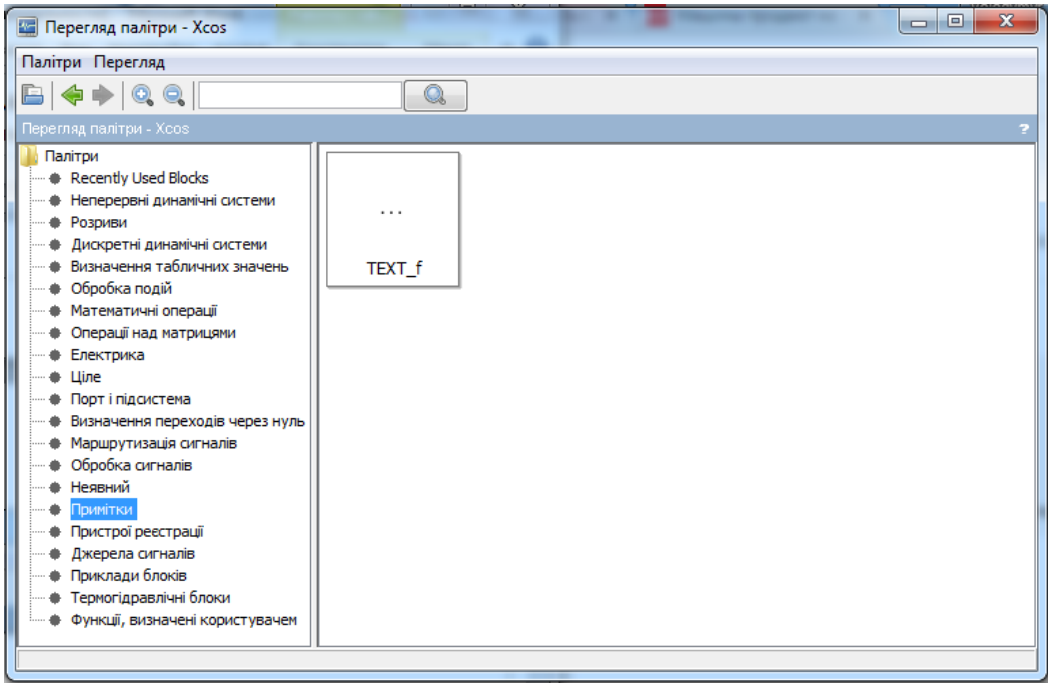
	Квантування Блок виводить квантування вводу відповідно до вибору методу (типу квантування): • 1: округлення; • 2: за верхньою границею кванта; • 3: за нижньою границею кванта
 SAMPHOLD_m	Дискретизація (запам’ятовування значення) Відтворення на виході вхідного сигналу і зчитування нового вхідного за сигналом активації

5.14 Impicit (Неявний)



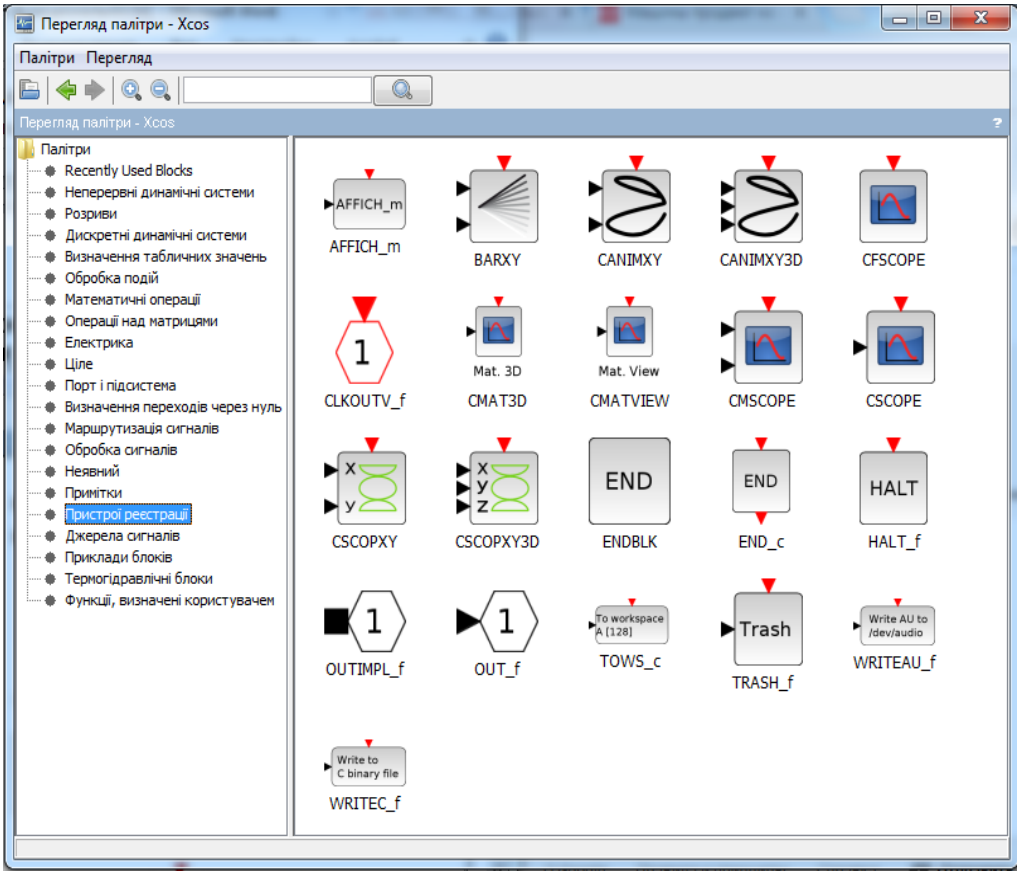
	Визначення неявних алгебраїчних виразів (Детальна документація відсутня)
	С-макроси доступні шляхом включення С-файлу (Детальна документація відсутня)
	Обчислення похідної

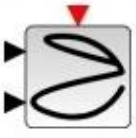
5.15 Annotation (Примітки)

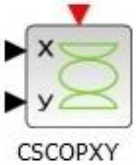
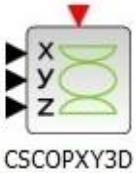


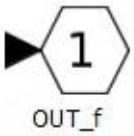
TEXT_f	Текстовий блок Використовується лише для додавання тексту в будь-яке місце діаграми

5.16 Sinks (Пристрої реєстрації)

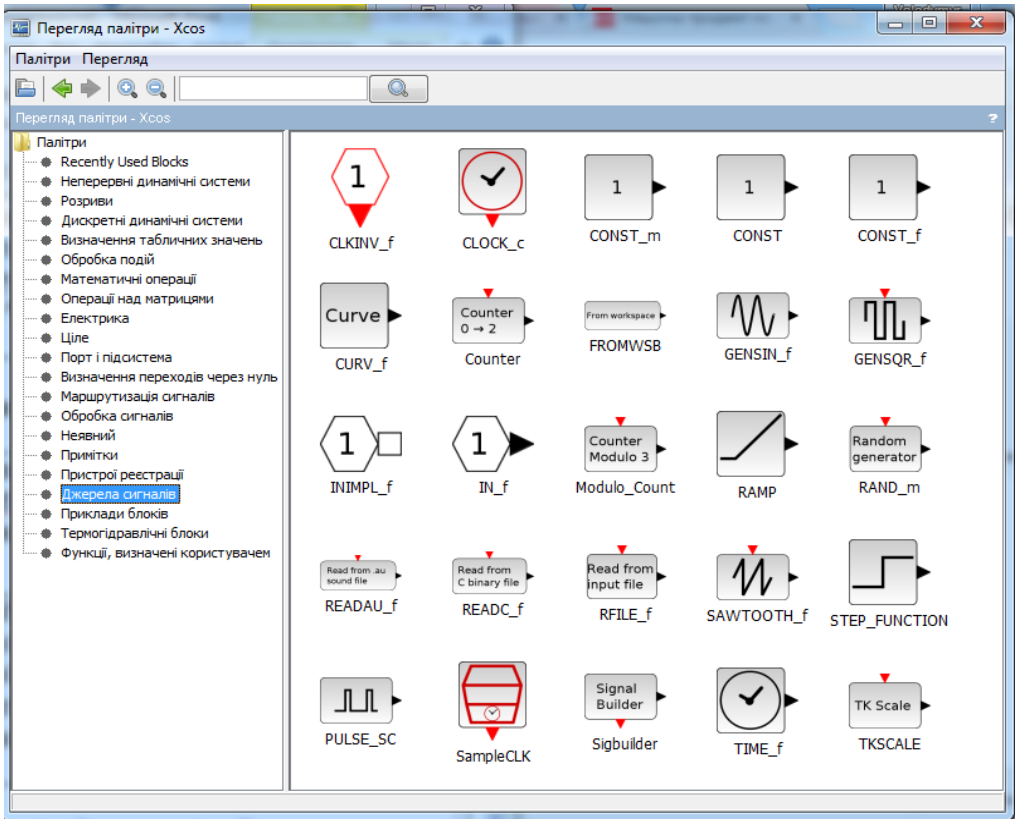


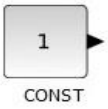
	Дисплей Блок відображає цифрове значення входу під час моделювання
	Блок будує вектор від початку координат до точки, яка визначається двома входами. За надходження нових даних попередній вектор зникає. Використовується для анімації
	Блок будує точку y(x), де x і y – входи. За надходження нових даних попередня точка зникає. Використовується для анімації
	Будує графік в 3D координатах

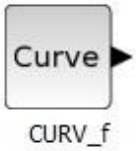
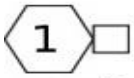
	Осцилограф з плаваючою точкою підключення. Показує дані, що виводяться у блок Trash. Може в одних осях показувати графіки декількох вихідних потоків. Output window number – номер графічного вікна, використовуваного для показу. Його потрібно ставити великим, щоб не переплутати з іншими вікнами
	Див. «Порти і підсистеми»
	Матричний 3D осцилограф CMAT3D – це область, яка показує значення матриць у вигляді z-значень в сітці xy
	Матричний осцилограф з кольоровим зображенням CMATVIEW – це область, яка показує значення матриці в кольоровій сітці
	Багатовіконний дисплей Блок виводить декілька графіків в окремих осях
	Простий осцилограф
	Відображає залежність y(x). Змінні x і y подаються на два входи
	Відображає залежність z = f(x,y). Змінні x, y і z подаються на три входи. Блок будує поверхню – візуалізацію еволюції трьох вхідних сигналів, рисуючи третій вхід як функцію двох інших в моменти «подій» на вхідному порту «події»

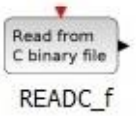
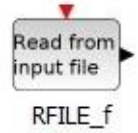
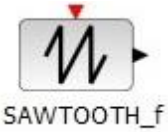
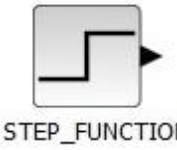
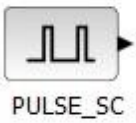
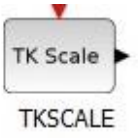
	Див. «Обробка подій»
	Див. «Обробка подій»
	Див. «Обробка подій»
	Див. «Порти і підсистеми»
	Див. «Порти і підсистеми»
	Блок використовується для додавання імітаційних даних в робочу область Scilab
	Блок є кінцевим блоком, в якому накопичуються дані
	Запис звукового файлу AU Блок записує звуковий файл, заданий файлом string*.au. Дані мають бути організовані одним каналом для кожного стовпця. Значення амплітуди за межами діапазону [-1, + 1] обрізаються перед записуванням
	Запис двійкових даних Блок дозволяє користувачеві записувати дані у бінарному файлі C з ім'ям, визначеним ім'ям вихідного файлу рядка. Файл являє собою послідовність записів.

5.17 Sources (Джерела сигналів)

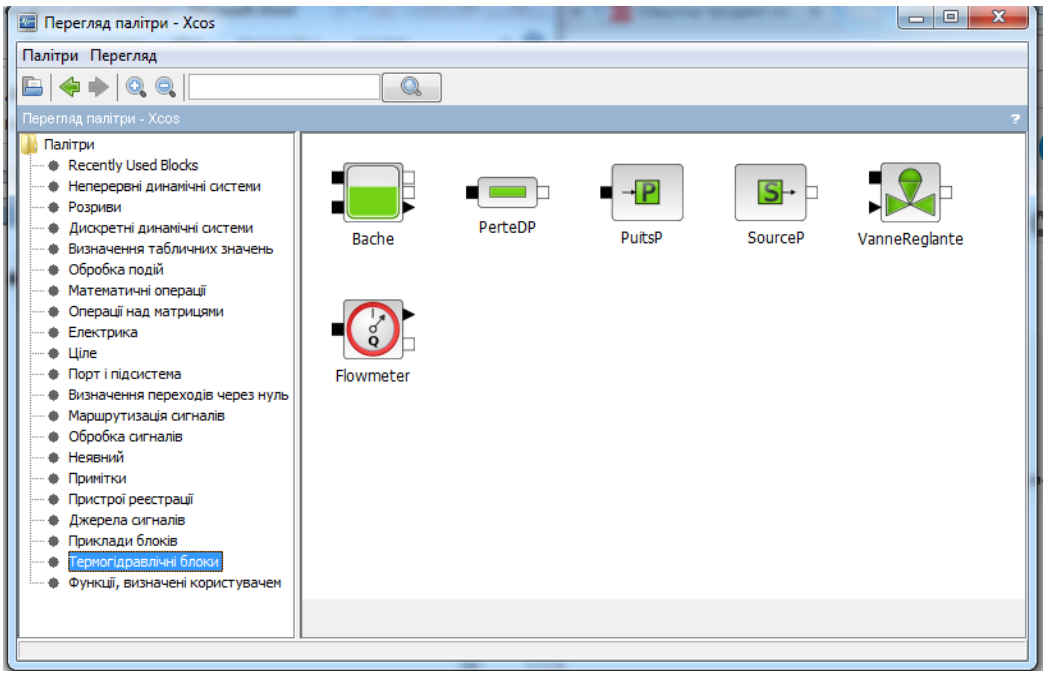


 CLKINV_f	Див. «Порти і підсистеми»
 CLOCK_c	Годинник активації Генерує регулярний потік «подій» з параметром Period в секундах. Початковий час генерації «подій» можна встановити в секундах за допомогою параметра «Час ініціалізації»
 CONST	Константа
 CONST_m	
 CONST_f	

 CURV_f	Формує табульовану функцію часу. Функція задається вбудованим графічним редактором. Між точками використовується лінійна інтерполяція
 Counter	Лічильник. Встановлюється максимальна величина рахування і спосіб рахування: на збільшення або на зменшення
 FROMWSB	Передача даних з робочого простору Scilab в Xcos Потрібно, щоб дані мали поля «час» та «значення»
 GENSIN_f	Генератор синусоїдального сигналу Ви можете налаштувати: • величину амплітуди задають параметром Magnitud; • частоту F в радіан/с задають параметром Frequency; • початкову фазу P в радіанах
 GENSQR_f	Генератор прямокутних імпульсів. Необхідна дискретна активація. Блок CLOCK_c для цього не підходить
 INIMPL_f	Див. «Порти і підсистеми»
 IN_f	Див. «Порти і підсистеми»
 Modulo_Count	Лічильник за модулем. Параметр – величина модуля Блок виводить періодичний скалярний кусочно-постійний сигнал. Сигнал починається зі значення Initial State. Вихід збільшується, коли блок отримує «події» на вхід активації, доки вихід не досягне значення параметра верхнього межі. У цей момент стан змінюється на 0. Виходом є квантований пилкоподібний сигнал
 RAMP	Блок видає сигнал у встановлений час, величина і швидкість наростання якого задаються
 RAND_m	Генератор рівномірно або нормально розподілених випадкових чисел. Параметри розподілу задаються

	Зчитування звукових файлів AU Завантажує звуковий файл, вказаний файлом string*.au. Додається розширення .au, якщо не вказано розширення. Значення амплітуди знаходяться в діапазоні [-1, + 1]
	Зчитування двійкових даних Блок дозволяє читати дані в файлі С. Вибір часу запису дозволяє вибирати дані з файлу. Кожен виклик блока запускає один запис у файлі
	Зчитування з файлу Блок дозволяє користувачеві читати дані у файлі з ім'ям, визначеним параметром Input File Name, у текстовому форматі або в бінарному режимі
	Генератор пилкоподібного сигналу в діапазоні 0...1
	Генератор ступінчастого сигналу
	Імпульсний генератор. Він генерує імпульс зі скважністю 1/P, де P – період сигналу. Другий параметр блока дає ширину імпульсу, а четвертий параметр дає амплітуду імпульсу
	Годинник активації Різниця між SampleCLK і CLOCK_c полягає в тому, що всі блоки SampleCLK в схемі є синхронними
	Блок Builder сигналу – це суперблок, що містить блок, порт вихідної «події» якого підключений до порту «події» вхідних даних
	Час. Установлень немає. Лінійно наростаючий сигнал. Крутизна дорівнює одиниці
	Блок видає постійний сигнал, величина якого може бути змінена в установленому діапазоні безпосередньо під час роботи моделі за допомогою повзункового регулятора з урахуванням фактора нормалізації (для збільшення точності).

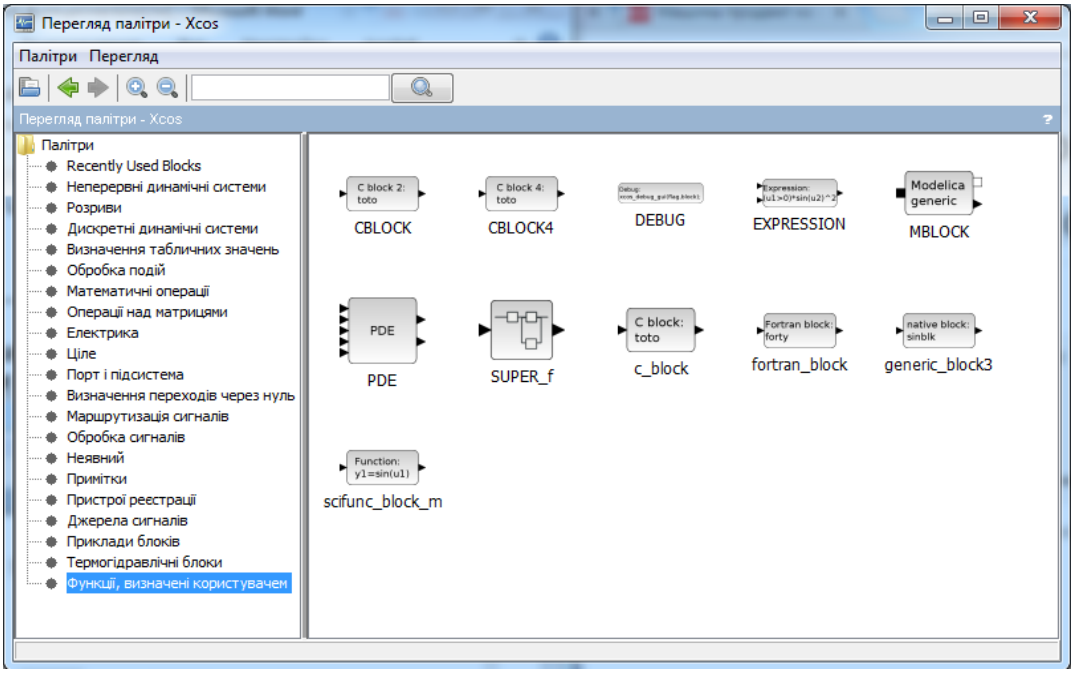
5.18 Thermo-Hydraulics (Термогідравлічні блоки)



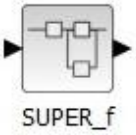
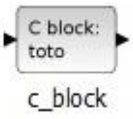
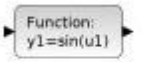
 Bache	Бейч-блок являє собою резервуар. Блок має два впускні отвори і два випускні, висоту яких може змінити користувач. Для вхідних портів (чорних портів) напрям потоку є позитивним, коли рідина входить в резервуар. Для вихідних портів (білих портів) напрямок потоку позитивний, коли рідина витікає з резервуара. Користувач може встановити площу поверхні резервуара, початкову температуру та початковий рівень рідини в резервуарі. Якщо вхідний або вихідний порт залишився невикористовуваним, він має бути заблокований блоком зупинення
 PerteDP	Труба (гідравлічний опір) Цей компонент являє собою тиски, прямо пропорційні витраті. Традиційно напрям потоку є позитивним, коли рідина тече від чорного порту до білого порту
 PuitsP	Термогідравлічний витік Цей термогідравлічний компонент являє собою термогідравлічний конденсатор. Блок визначений його тиском і температурою
 SourceP	Цей термогідравлічний компонент являє собою термічногідравлічне живлення постійного тиску (свердловину). Блок вказаний з вихідним тиском і температурою. Традиційно напрям потоку є позитивним, коли рідина витікає з блока

	Блок VanneReglante являє собою регульований клапан з отвором. Швидкість потоку через клапан пропорційна відкриттю клапана
	Цей компонент використовується для вимірювання об'ємної витрати

5.19 User-Defined Functions (Функції, визначені користувачем)



	Блок створює каркас функції C-обчислень. Він також створює бібліотечні файли та об'єктні файли
	Блок створює каркас функції C-обчислень. Він також створює бібліотечні файли та об'єктні файли
	Використовується для налагодження сценарію Scilab. Після виклику debug () ви входите в режим налагодження. Цей режим дозволяє управляти точками зупинення, запускати виконання зі stop on error, виконувати сценарій крок за кроком

	Блок Expression застосовує вказані вирази Scilab до його вводу
	Блок «MBlock» забезпечує простий спосіб побудови блока xcos, поведінка якого визначається програмою Modelica. Використовуючи цей блок, користувач зможе писати та складати програми Modelica в xcos без створення будь-якої функції інтерфейсу
	Блок являє собою реалізацію кількох числових схем (кінцевих елементів(1-го та 2-го порядку, кінцевих приростів 1-го та 2-го порядку, кінцевих розрізів 1-го порядку) для вирішення диференціального рівняння у Xcos
	Блок відкриває нове вікно Xcos для редагування нової блок-схеми. Ця діаграма описує внутрішні функції суперблока. Входи і виходи суперблока (звичайний або «подія») позначаються спеціальними блоками (вхідними або вихідними портами)
	Блок створює каркас функції C обчислень. Також створює файли бібліотеки та об'єктні файли
	Блок створює скелет функції обчислень FORTRAN. Також створює файли бібліотеки та об'єктні файли.
	Блок забезпечує загальну функцію інтерфейсу, однак обчислювальна функція має бути визначена окремо як функція Scilab, Fortran або C
	Блок може реалізувати будь-який тип блока Xcos. Функція блока визначається інтерактивно за допомогою діалогових вікон та мовою Scilab

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Володимир Михайлович Дубовой
Марія Сергіївна Юхимчук
Юлія Ярославівна Лещенко

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ В СИСТЕМІ SCILAB/XCOS

Навчальний посібник

Рукопис оформила *Ю. Лещенко*
Редактор *Т. Старічек*
Оригінал-макет виготовлено *Т. Старічек*

Підписано до видання 15.01.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-018.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.