НАЦІОНАЛЬНИЙ ЛІСОТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
(повне найменування вищого навчального закладу)
ІНСТИТУТ ДЕРЕВООБРОБНИХ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ І ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
ІНФОРМАЦІЙНИХ СИСТЕМ ТА КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

бакалавр

(освітньо-кваліфікаційний рівень)

на тему: Програмно-алгоритмічне забезпечення системи моделювання чисельності популяцій з урахування часової нелокальності процесу

Виконав: студент IV курсу, групи ІСТС-21
спеціальності
126 «Інформаційні системи та технології»

(шифр і назва напряму підготовки, спеціальності)

Боднар В.-Б.І.
(прізвище та ініціали)

Керівник доц. Шиманський В.М.
(прізвище та ініціали)

Рецензент Крольчик І.М.
(прізвище та ініціали)

Львів - 2023 року

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ *деревообробних та комп'ютерних технологій і дизайну*
Кафедра *інформаційних систем та комп'ютерного моделювання*
Освітньо-кваліфікаційний рівень *бакалавр*
Спеціальність *126 " Інформаційні системи та технології "*
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____ Сторожук О. Л.
" 21 " 11 2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Боднару Віталію-Богдану Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи " *Програмно-алгоритмічне забезпечення системи моделювання чисельності популяцій з урахування часової нелокальності процесу* "
керівник роботи *Шиманський В.М., к.т.н., доц.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "21" 11 2022 року
№ С-521
2. Термін подання студентом роботи 10.06.23
3. Вихідні дані до роботи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
Розділ 1. Стан проблемної області
Розділ 2. Інформаційне та математичне забезпечення.
Розділ 3. Програмне та технічне забезпечення.
Висновки.
Список використаних джерел.
Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 23.11.2022

КАЛЕНДАРНИЙ ПЛАН

№, п/п	Етапи виконання бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літературних джерел згідно досліджуваної теми.	15.03.2023 р.	виконано
2.	Системний аналіз об'єкту дослідження та стану проблемної області: 1. Аналіз досліджуваної проблеми. 2. Характеристика створюваної інформаційної системи. 3. Визначення вимог до системи.	29.03.2023 р.	виконано
3.	Вибір та обґрунтування методів і засобів створення інформаційної системи	12.04.2023 р.	виконано
4.	Програмна реалізація системи	03.05.2023 р.	виконано
6.	Відлагодження створеної програми	17.05.2023 р.	виконано
7.	Оформлення опису створеної програми	31.05.2023 р.	виконано
9.	Здача пояснювальної записки на перевірку та виправлення виявлених помилок	10.06.2023 р.	виконано

Студент _____ **Боднар В.-Б.І.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Шиманський В.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 41 сторінок пояснювальної записки, 14 рисунків, 1 додаток, 16 джерел.

Розглянено математичну модель чисельності популяцій з урахування часової нелокальності процесу. Розроблено інтуїтивно зрозумілий інтерфейс користувача, що дозволяє обчислити та прогнозувати чисельності популяцій з урахування часової нелокальності процесу. Проаналізовано отримані результати моделювання чисельності популяцій з урахування часової нелокальності процесу залежно від параметрів моделі, початкових розмірів популяцій та параметрів середовища.

Ключові слова: екологічна система, похідна дробового порядку, точки рівноваги, MatLab, метод декомпозиції домену.

ABSTRACT

Thesis contains 41 pages of explanatory note, 14 drawings, 1 appendix, 16 sources.

The mathematical model of the number of populations taking into account the temporal non-locality of the process is considered. An intuitive user interface has been developed that allows you to calculate and forecast population sizes taking into account the temporal non-locality of the process. The obtained results of modeling the number of populations, taking into account the temporal non-locality of the process, depending on the parameters of the model, the initial sizes of the populations and the parameters of the environment, were analyzed.

Keywords: ecological system, derivative of fractional order, equilibrium points, MatLab, domain decomposition method.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити програмно-алгоритмічне забезпечення для системи моделювання чисельності популяцій з урахування часової нелокальності процесу. Для опису чисельності популяцій з урахування часової нелокальності процесу використовувати систему звичайних диференціальних рівнянь з похідними дробового порядку та відповідними початковими умовами.

Використовуючи середовище MatLab розробити систему для обчислення та прогнозування чисельності популяцій з урахування часової нелокальності процесу. Інтерфейс програмного забезпечення має бути інтуїтивно зрозумілим та добре документованим.

Оформити виведення числових результатів і графічних залежностей у зручному для користувача форматі.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

GUIDE – Graphical User Interface Development Environment, середовище розробки графічних інтерфейсів користувача в MATLAB.

IDE – Integrated Development Environment, інтегроване середовище розробки програмного забезпечення.

ООП – об’єктно-орієнтоване програмування.

МДО – метод декомпозиції області, чисельний метод розв’язування диференціальних рівнянь.

ВСТУП

Моделювання чисельності популяцій з урахуванням часової нелокальності процесу є дуже актуальним і важливим у сучасній науці та прикладних дослідженнях.

Часова нелокальність відноситься до того факту, що зміни в популяції на певний момент часу можуть бути залежні від її стану в попередні часи. Це означає, що вплив на популяцію не обмежується лише миттєвими змінами у середовищі, а може включати історичні та затримані ефекти.

Одним з основних прикладів, де це має значення, є екологічне моделювання. При вивченні динаміки популяцій тварин або рослин, часова нелокальність може включати такі фактори, як час розмноження, розповсюдження, зміни середовища та взаємодії з іншими видами. Врахування цих нелокальних ефектів дозволяє отримати більш точне уявлення про те, як популяції розвиваються в часі та пристосовуються до змін.

Також, чисельні моделі з урахуванням часової нелокальності є корисними в інших галузях науки та технологій. Наприклад, в епідеміології можна використовувати такі моделі для прогнозування поширення захворювань, враховуючи часові затримки у передачі інфекції та реакції на щеплення. У фізиці такі моделі можуть застосовуватися для дослідження динаміки систем з запізненням, наприклад, в електроніці або в управлінні процесами з затримкою.

Актуальність теми полягає в тому, що моделювання чисельності популяцій з урахуванням часової нелокальності дозволяє краще розуміти та передбачати поведінку популяцій в часі, враховуючи історичні зміни та затримки в їх взаємодії з середовищем. Це є важливим інструментом для вивчення різних систем та вирішення реальних проблем у різних галузях науки й технологій.

Об’єкт дослідження – динаміка зміни чисельності популяції.

Предметом дослідження – математична модель чисельності популяцій з урахуванням часової нелокальності.

Метою є дослідити чисельність популяцій з урахуванням часової нелокальності.

Завданням являється розроблення програмно-алгоритмічного забезпечення для моделювання чисельності популяцій з урахуванням часової нелокальності.

Практичне значення даного дипломного проекту заключається у створенні системи моделювання чисельності популяцій з урахуванням часової нелокальності.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Визначення проблемної області: моделювання чисельності популяцій

Моделювання чисельності популяцій є важливим інструментом для вивчення та розуміння природних систем. Ця задача має значення в екологічних дослідженнях, біології, охороні природи та ряду інших галузей, де важливо аналізувати та передбачати зміни в популяціях живих організмів. Актуальність цієї задачі полягає в її потенціалі для збереження біорізноманіття, розуміння екологічних процесів та розробки науково обґрунтованих стратегій управління природними ресурсами.

Однією з найбільш актуальних проблем сучасного світу є збереження біорізноманіття та управління природними ресурсами. Моделювання чисельності популяцій допомагає вивчити взаємозв'язок між організмами та їх середовищем, виявити ключові екологічні фактори, що впливають на популяції, і визначити ефективні стратегії збереження видів.

Моделі чисельності популяцій дозволяють нам аналізувати та розуміти складні екологічні взаємодії між різними видами та організмами в екосистемах. Вони допомагають вивчити конкуренцію, хижацтво, симбіоз, міграції та інші взаємозв'язки, які впливають на структуру та динаміку популяцій. Це дозволяє нам розуміти природні процеси та передбачати наслідки змін в екосистемах.

Моделі чисельності популяцій дозволяють нам прогнозувати екологічні зміни та впливати на них. Застосування моделей дозволяє передбачати, як зміниться чисельність популяцій під впливом зміни середовища, забруднення, глобальних змін клімату та інших факторів. Це допомагає приймати обґрунтовані рішення щодо управління природними ресурсами та охорони довкілля.

Моделювання чисельності популяцій є важливим інструментом для раціонального управління природними ресурсами. Воно дозволяє оцінити

вплив вирубки лісу, промислового рибальства, землеробства та інших господарських діяльностей на популяції та екосистеми. Це допомагає розробляти стратегії сталого використання ресурсів та збереження природних систем.

Моделювання чисельності популяцій дозволяє нам аналізувати екологічні взаємодії між різними видами та оцінювати їх вплив на екосистеми. Наприклад, ми можемо вивчити конкуренцію між видами, хижацтво, симбіоз та інші взаємозв'язки, які впливають на розподіл та динаміку популяцій. Це допомагає зрозуміти природні процеси та розробляти стратегії управління екосистемами.

Моделі чисельності популяцій мають важливе значення в епідеміології та громадському здоров'ї. Вони допомагають прогнозувати поширення хвороб, оцінювати ефективність контролю та вакцинації, та розробляти стратегії запобігання епідеміям. Моделювання дозволяє оцінити вплив факторів, таких як імунітет населення, мобільність та інші чинники на поширення хвороб.

Моделі чисельності популяцій є важливим інструментом для розробки ефективних стратегій управління ресурсами та збереження видів. Вони допомагають оцінювати вплив різних варіантів управлінських рішень та прогнозувати їх наслідки на чисельність популяцій. Це дозволяє приймати обґрунтовані рішення щодо охорони природи, рибного господарства, лісового господарства та інших галузей.

У сучасному світі стикаємося з численними глобальними викликами, такими як зміна клімату, енергетичні кризи, виснаження природних ресурсів. Моделювання чисельності популяцій допомагає нам розуміти вплив цих викликів на природні системи та розробляти стратегії адаптації та відновлення. Воно є важливим інструментом для розуміння складності екологічних систем та пошуку раціональних рішень.

Актуальність задачі моделювання чисельності популяцій очевидна в контексті збереження біорізноманіття, прогнозування екологічних змін, вивчення екологічних взаємодій, прогнозування епідемій та розробки стратегій управління. Це важлива галузь досліджень, яка допомагає зрозуміти та

передбачати природні процеси, а також розробляти науково обґрунтовані підходи до управління природними ресурсами та охорони довкілля. Моделювання чисельності популяцій є цінним інструментом для наукової спільноти та прийняття рішень в галузі екології та природоохоронних заходів.

У сучасному світі, де збереження біорізноманіття та стале управління природними ресурсами стають все більш важливими завданнями. Цей підхід дозволяє нам зрозуміти екологічні взаємодії, прогнозувати зміни в екосистемах та розробляти науково обґрунтовані стратегії збереження природи та управління ресурсами. Дальший розвиток моделювання чисельності популяцій сприятиме створенню ефективних інструментів для збереження біорізноманіття, прогнозування екологічних змін та раціонального управління природними ресурсами, що є критичними для нашої майбутньої сталості та благополуччя.

1.2. Розв’язування задач за допомогою моделювання та симулювання

В сучасному світі зростає необхідність в розумінні та прогнозуванні динаміки популяційних процесів. Чи вирощуємо ми рослини, виводимо тварин чи аналізуємо розподіл населення, розуміння та прогнозування змін в чисельності популяцій має вирішальне значення для екології, сільського господарства, медицини та інших галузей. Одним з найефективніших інструментів для досягнення цих цілей є моделювання чисельності популяцій.

Моделювання чисельності популяцій - це процес створення математичних або комп'ютерних моделей, що дозволяють аналізувати та передбачати зміни в чисельності популяційних процесів з часом. Ці моделі базуються на вивченні взаємодії між різними факторами, що впливають на чисельність популяції, такими як народжуваність, смертність, міграція та взаємодія з оточуючим середовищем.

Одним з основних аспектів значення моделювання чисельності популяцій є розуміння популяційних процесів. Шляхом аналізу факторів, що впливають

на чисельність популяції, таких як зміни середовища, харчова база або кліматичні умови, моделі дозволяють встановити зв'язки та причинно-наслідкові залежності між цими факторами та чисельністю популяцій. Це допомагає науковцям отримати глибше розуміння та інсайти щодо впливу різних чинників на стабільність та здоров'я популяцій.

Крім того, моделювання чисельності популяцій має практичне значення для прогнозування майбутніх змін. За допомогою моделей можна оцінити, які наслідки можуть мати різні сценарії на популяційні процеси в майбутньому. Наприклад, модель може допомогти визначити, які заходи потрібно вжити для збереження зникаючого виду або як зміниться розподіл населення залежно від змін клімату. Такі прогнози дозволяють приймати обґрунтовані рішення та розробляти ефективні стратегії управління популяціями.

Додатково, моделювання чисельності популяцій є економічно ефективним підходом. Воно дозволяє проводити віртуальні експерименти та тестувати різні сценарії без необхідності витрат на реальні дослідження. Моделі можуть бути засновані на реальних даних, а також враховувати невизначеність та стохастичність популяційних процесів, що дозволяє проводити різноманітний аналіз та враховувати різні можливі варіанти розвитку подій.

Загалом, моделювання чисельності популяцій є важливим інструментом для розуміння та прогнозування динаміки популяційних процесів. Воно допомагає науковцям та приймачам рішень зберегти та управляти популяціями, сприяє розвитку екологічної стійкості, підтримує стійке виробництво сільськогосподарських культур та відкриває можливості для дослідження впливу людської діяльності на природні екосистеми.

1.3. Основні фактори, що впливають на чисельність популяцій

1.3.1. Екологічні фактори, що впливають на чисельність популяцій

Природа має свої закони, які визначають чисельність та динаміку популяцій різних видів. Чисельність популяцій залежить від багатьох факторів, зокрема від екологічних умов, які оточують популяцію. Розуміння та аналіз екологічних факторів, що впливають на чисельність популяцій, є важливим для збереження біорізноманіття та екосистемного рівноваги.

Один з ключових екологічних факторів, що впливають на чисельність популяцій, - доступність ресурсів. Ресурси, такі як їжа, вода, простір для життя та розмноження, є необхідними для виживання та розмноження популяцій. Наявність достатньої кількості ресурсів може сприяти збільшенню чисельності популяцій, тоді як обмежений доступ до ресурсів може впливати на зменшення чисельності та навіть на загибель популяцій.

Крім доступності ресурсів, інший важливий екологічний фактор - взаємодія з іншими видами. Взаємодія може бути як конкурентною, так і співіснуючою. Конкуренція за ресурси може впливати на зміни в чисельності популяцій, зокрема через зменшення доступності ресурсів для одного виду. У той же час, взаємодія може мати позитивний ефект на популяції, наприклад, у випадку симбіозу або взаємовигідних відносин.

Інші екологічні фактори, що впливають на чисельність популяцій, включають кліматичні умови, наявність хижаків та хвороб, а також вплив людської діяльності. Зміна клімату може призвести до зсуву в ареалі розповсюдження виду або до зміни в доступності ресурсів. Присутність хижаків та хвороб також може має значний вплив на чисельність популяцій, зменшуючи їх розмноження та виживання.

Останнім, але не менш важливим екологічним фактором є вплив людської діяльності. Зміна природного середовища через індустріальну діяльність, вирубку лісів, забруднення водних ресурсів та інші антропогенні фактори мають негативний вплив на популяції тварин і рослин. Це може призводити до зменшення чисельності популяцій, зсуву їх ареалів та викликати загрозу для виживання деяких видів.

Таким чином, екологічні фактори впливають на чисельність популяцій та їх динаміку. Розуміння цих факторів допомагає науковцям та управлінцям розробляти стратегії збереження біорізноманіття та управління популяціями. Враховуючи вплив ресурсів, взаємодії з іншими видами, кліматичних умов, присутності хижаків, хвороб та людської діяльності, можна виявити чинники, які впливають на стабільність популяцій та розробити ефективні стратегії для збереження та регулювання популяційних процесів.

1.3.2. Біологічні фактори, що впливають на чисельність популяцій

У природних екосистемах чисельність популяцій різних видів є результатом взаємодії між різними біологічними факторами. Ці фактори включають розмноження, смертність, конкуренцію, хижацтво, хвороби та багато інших аспектів життєдіяльності популяцій. Розуміння та аналіз цих біологічних факторів є важливим для вивчення та управління популяціями тварин і рослин.

Один з найважливіших біологічних факторів, що впливають на чисельність популяцій, - розмноження. Швидкість та ефективність розмноження визначають, наскільки швидко популяція збільшується. Чинники, які впливають на розмноження, включають народжуваність, розподіл статевих відносин, тривалість репродуктивного періоду та інші. Наприклад, висока народжуваність та короткий репродуктивний цикл можуть сприяти швидкому збільшенню чисельності популяції.

Смертність є ще одним важливим біологічним фактором, який впливає на чисельність популяцій. Смертність може бути спричинена різними причинами, такими як хижацтво, хвороби, конкуренція за ресурси або природні катастрофи. Висока смертність може знизити чисельність популяції, тоді як низький рівень смертності може сприяти збільшенню чисельності популяції.

Конкуренція є ще одним біологічним фактором, що впливає на чисельність популяцій. Конкуренція виникає, коли різні види або особини в

одній популяції змагаються за обмежені ресурси, такі як їжа, простір або партнери для розмноження. Конкуренція може призвести до зменшення чисельності популяції або до зсуву в просторовому розподілі видів.

Хижацтво є іншим важливим біологічним фактором, що впливає на чисельність популяцій. Присутність хижаків у популяції може зменшити чисельність жертв, що має наслідком зміну структури та динаміки популяції.

Хвороби також можуть мати значний вплив на чисельність популяцій. Хвороби можуть поширюватися серед особин у популяції, зменшуючи їх виживання та репродуктивний успіх. Відповідно, це може призвести до зменшення чисельності популяцій.

Біологічні фактори, такі як розмноження, смертність, конкуренція, хижацтво та хвороби, грають важливу роль у визначенні чисельності популяцій. Розуміння цих факторів допомагає вивчати та прогнозувати динаміку популяцій тварин і рослин. Також воно сприяє розробці стратегій для збереження біорізноманіття та управління популяціями, що є важливими для сталого розвитку природних екосистем.

1.3.3. Взаємодія з іншими видами: конкуренція, хижацтво, симбіоз

В природних екосистемах взаємодія між різними видами є ключовим аспектом життєдіяльності популяцій. Ця взаємодія може приймати різні форми, включаючи конкуренцію, хижацтво та симбіоз. Розуміння цих форм взаємодії допомагає нам краще усвідомити складність та різноманітність природних екосистем.

Конкуренція є одним з найважливіших видів взаємодії між видами. Вона виникає, коли два або більше види змагаються за обмежені ресурси, такі як їжа, простір або світло. Конкуренція може бути прямою, коли види взаємодіють безпосередньо, або опосередкованою, коли вони впливають на одні й ті ж ресурси. Конкуренція може мати різний результат: один вид може витіснити інший, або вони можуть знаходити способи співіснування та поділу ресурсів.

Хижацтво є іншим видом взаємодії між видами. Хижак полює на інші види, щоб забезпечити собі їжу та виживання. Ця взаємодія відіграє важливу роль у регулюванні чисельності популяцій, зменшуючи чисельність жертв і підтримуючи екологічний баланс. Хижацтво може впливати на поведінку та стратегії виживання видів, що полюють, а також видів-жертв.

Симбіоз є третьою формою взаємодії між видами, в якій обидва види отримують взаємну вигоду. Симбіоз може бути різним за своєю природою. Наприклад, мутуалізм є формою симбіозу, в якій обидва види отримують користь. Один вид може забезпечувати житло або захист іншому виду, водночас отримуючи поживу або інші переваги. Існують також інші форми симбіозу, такі як паразитизм і коменсалізм, де один вид отримує вигоду, а інший вид не має впливу або завдає йому шкоди.

Взаємодія з іншими видами включає конкуренцію, хижацтво та симбіоз. Ці форми взаємодії визначають структуру та функціонування екосистем. Розуміння цих взаємодій є важливим для збереження біологічного різноманіття та підтримання екологічної рівноваги. Вивчення цих взаємодій допомагає нам краще усвідомити природу та складність природних екосистем і розробляти стратегії для їх збереження та сталого управління.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Огляд середовища MATLAB

MATLAB (MATrix LABoratory) є одним з найпопулярніших і потужних середовищ програмування та чисельних обчислень, яке широко використовується в наукових, інженерних та фінансових галузях. Це інтерактивне середовище дозволяє розв'язувати складні математичні задачі, аналізувати дані, розробляти алгоритми, візуалізувати результати та моделювати різноманітні процеси.

Одним з найбільших переваг MATLAB є його простий та інтуїтивний синтаксис. Мова програмування MATLAB базується на математичних виразах та операціях з матрицями, що дозволяє розробникам швидко та зручно писати код. Синтаксис мови MATLAB дуже схожий на математичний запис, що дозволяє зменшити час, необхідний для перекладу математичних концепцій в програми.

Однією з найсильніших сторінок MATLAB є його потужність у сфері чисельних обчислень. MATLAB надає широкий набір математичних функцій та операцій, які дозволяють виконувати різноманітні операції над матрицями. Це включає розв'язування систем лінійних рівнянь, чисельну інтеграцію та диференціювання, апроксимацію функцій, розв'язування диференціальних рівнянь та багато іншого. Інтеграція цих функцій у середовище MATLAB робить його потужним інструментом для виконання складних обчислень.

MATLAB також забезпечує розширені можливості для аналізу та візуалізації даних. За допомогою вбудованих функцій, користувачі можуть виконувати статистичний аналіз, обробку сигналів, аналіз зображень, фільтрацію даних та багато іншого. Всі ці функції доповнюються інтуїтивним інтерфейсом користувача, що дозволяє швидко та ефективно аналізувати та візуалізувати дані.

Бібліотеки MATLAB також забезпечують широкі можливості для розробки алгоритмів та моделювання процесів. За допомогою спеціалізованих інструментів, користувачі можуть розробляти алгоритми для обробки сигналів, машинного навчання, оптимізації, керування системами та багато іншого. Це робить MATLAB популярним серед вчених та інженерів, що працюють над складними проектами та дослідженнями.

Незалежно від своїх чисельних та аналітичних можливостей, MATLAB також має широку підтримку спільноти користувачів. Велика кількість документації, форумів та онлайн-ресурсів доступна для отримання допомоги та спільного обміну знаннями. Крім того, MATLAB має зручне інтегроване середовище розробки (IDE), яке надає користувачам засоби для ефективної роботи зі своїм кодом та проектами.

MATLAB є потужним та універсальним середовищем програмування та чисельних обчислень. Він надає користувачам зручні інструменти для розв'язання складних математичних задач, аналізу даних, розробки алгоритмів та візуалізації результатів. Його простий синтаксис та потужність роблять його популярним в наукових та інженерних галузях, де вимагаються швидкі та точні обчислення.

У середовищі MATLAB m-файли є основним засобом організації та збереження коду, який використовується для виконання різноманітних завдань. М-файли - це текстові файли, які містять MATLAB-код та функції, які можуть бути викликані з командного рядка або іншого m-файлу.

Один з основних способів використання m-файлів - це визначення та виклик функцій. Можна створювати власні функції у окремих m-файлах, що дозволяє організовувати код у логічні блоки та повторно використовувати його. Функції можуть приймати аргументи, виконувати розрахунки та повертати результати. Це дозволяє створювати модульний та легкозмінний код.

Крім функцій, m-файли можуть містити інші конструкції мови MATLAB, такі як оператори умови, цикли та обробники помилок. Це дає можливість створювати скрипти, які автоматизують виконання послідовності команд або

розрахунків. Скрипти є зручним засобом для виконання операцій та аналізу даних без необхідності повторного введення команд.

Крім того, m-файли дозволяють створювати власні класи та об'єкти. Об'єктно-орієнтоване програмування (ООП) є важливою частиною MATLAB і дозволяє створювати структуровані та модульні програми. Класи можуть містити власні методи (функції, що викликаються на об'єкті) та властивості (змінні, які містять дані об'єкта). Це полегшує розробку та обслуговування складних програм, де об'єкти мають свою внутрішню структуру та взаємодіють між собою.

Однією з головних переваг використання m-файлів є їх легкість у використанні та розповсюдженні. Код, розміщений у вигляді m-файлів, може бути легко поділений між різними розробниками та виконувати команди або функції з довільного місця. Це сприяє спільній роботі та співпраці над проектами. Крім того, m-файли можна компілювати в виконувані файли, що дозволяє їх використання без необхідності встановлення MATLAB на машині користувача.

Оглядаючи всі можливості використання m-файлів у середовищі MATLAB, варто зазначити, що вони допомагають створювати структурований, модульний та повторно використовуваний код. Вони полегшують розробку програм, забезпечують легкість у використанні та сприяють спільній роботі над проектами. З моменту їх впровадження, m-файли стали невід'ємною частиною екосистеми MATLAB і використовуються мільйонами розробників по всьому світу.

2.2. Побудова GUIDE у середовищі MATLAB

GUIDE (Graphical User Interface Development Environment) є потужним інструментом у середовищі MATLAB, який дозволяє створювати інтерактивні графічні інтерфейси користувача для програм. Використання GUIDE спрощує розробку та взаємодію з програмами, оскільки воно надає гнучкість та

можливості для візуального проектування інтерфейсу без необхідності написання складного коду.

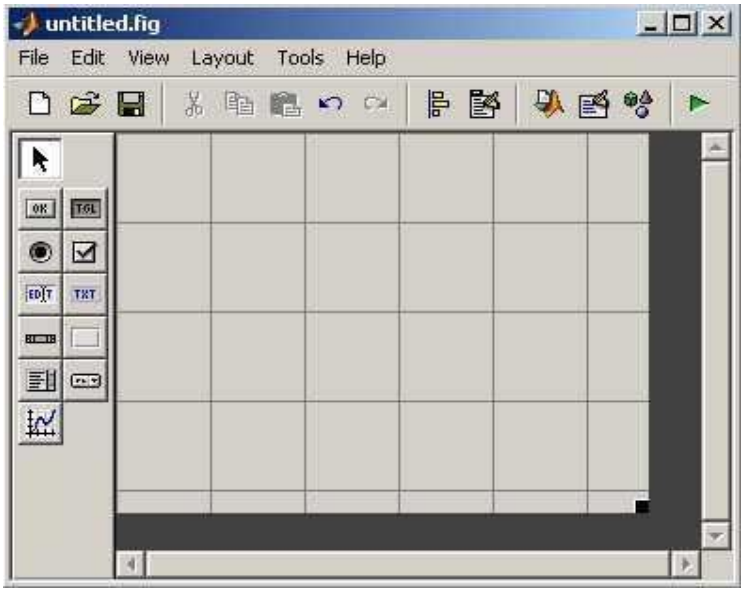

Рис. 2.1. Редактор додатки

Побудова GUIDE розпочинається з створення нового інтерфейсу у середовищі MATLAB. Візуальний редактор GUIDE надає користувачеві палітру елементів управління, таких як кнопки, тексти, поля вводу, списки, графіки тощо. Користувач може просто перетягнути ці елементи на форму та налаштувати їх властивості, такі як розміщення, розмір, колір, текст тощо. Це дозволяє швидко створювати вигляд інтерфейсу за допомогою простих дій мишею.

Після створення вигляду інтерфейсу, користувач може додати обробники подій до елементів управління. Обробники подій - це функції, які виконуються при спрацюванні певної події, наприклад, натисканні кнопки або введенні тексту. Користувач може вибрати елемент управління, вибрати відповідну подію та вибрати або створити функцію, яка буде виконуватися при спрацюванні цієї події. Це дозволяє реагувати на дії користувача та виконувати певні дії у відповідь.

Рис. 2.2. Панель інструментів для додавання елементів інтерфейсу

Однією з ключових переваг GUIDE є автоматичне генерування коду. Після побудови інтерфейсу та налаштування обробників подій, GUIDE може автоматично згенерувати код, який описує структуру інтерфейсу та функції обробки подій. Це дозволяє ефективно працювати з кодом, вносити зміни та розширювати функціональність інтерфейсу без необхідності написання всього коду вручну.

Побудова GUIDE значно спрощує розробку складних програм з інтерактивними елементами. Вона дозволяє розробникам швидко створювати професійно виглядаючі інтерфейси з мінімальними зусиллями. Більше того, GUIDE інтегровано з іншими функціональними можливостями MATLAB, такими як обробка даних, аналіз та візуалізація, що дозволяє легко поєднувати інтерфейси з розрахунками та аналізом.

У підсумку, GUIDE є потужним інструментом для побудови інтерактивних графічних інтерфейсів у середовищі MATLAB. Використання GUIDE дозволяє розробникам швидко створювати і налаштовувати інтерфейси, додавати функціональність та реагувати на дії користувача. Це робить MATLAB ще більш потужним і зручним інструментом для розробки та взаємодії з програмами.

2.3. Програмування подій у середовищі MATLAB

Програмування подій в середовищі MATLAB дозволяє розробникам створювати інтерактивні програми, які відповідають на різні події та дії користувача. Події можуть бути спрацьовані при натисканні кнопки, зміні значення поля вводу, переміщенні миші тощо. Програмування подій дозволяє забезпечити взаємодію користувача з програмою та керувати її поведінкою в реальному часі.

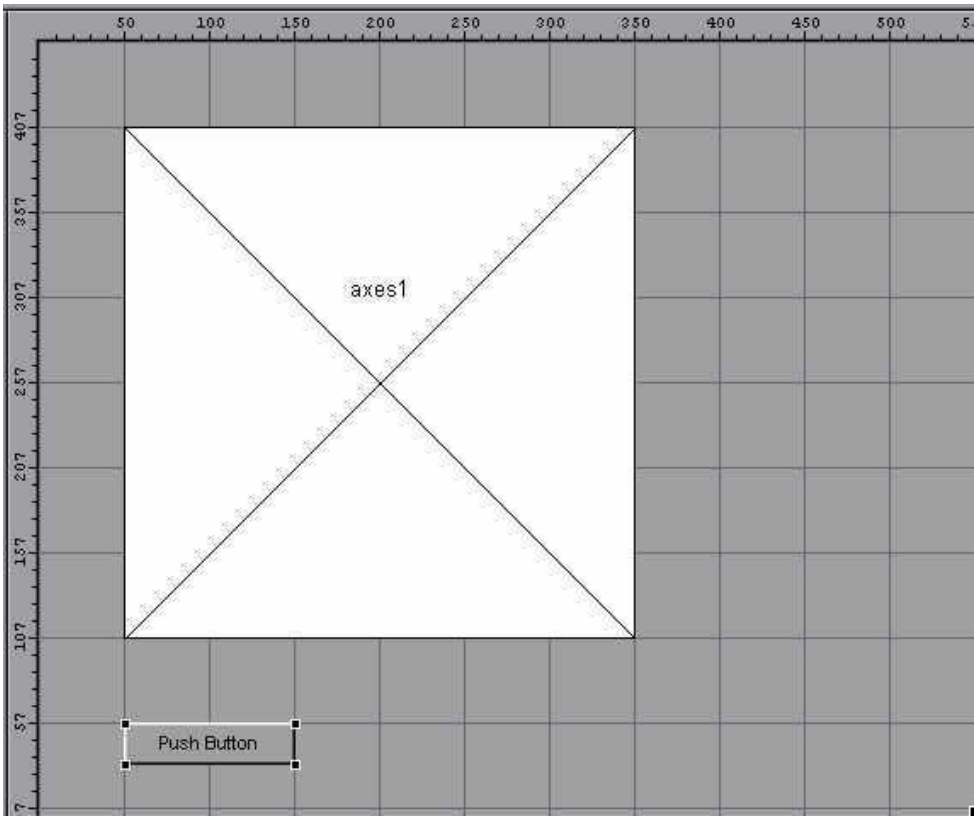

Рис. 2.3. Розташування кнопки і осей у вікні програми

Один з основних механізмів програмування подій у MATLAB - це використання обробників подій. Обробники подій - це функції, які виконуються автоматично при спрацюванні певної події. Розробники можуть визначати свої власні обробники подій для елементів управління, таких як кнопки, поля вводу, списки тощо. При спрацюванні події, викликається відповідний обробник, що дозволяє виконати певні дії, залежно від поточної ситуації.

Рис. 2.4. Вікно властивостей Property Inspector

Програмування подій у MATLAB також передбачає використання слухачів подій. Слухачі подій - це об'єкти, які відстежують виникнення певних подій і виконують відповідні дії при спрацюванні цих подій. Слухачі подій можуть бути пов'язані з різними об'єктами, які мають можливість генерувати події. Наприклад, слухач подій може відстежувати рух миші на графічному вікні та виконувати певні дії, коли миша рухається або клікає.

Події у MATLAB також можуть бути пов'язані з графічними об'єктами, такими як вікна, графіки, кнопки тощо. Наприклад, можна створити програму, яка реагує на подію натискання кнопки та зміну значення поля вводу. При спрацюванні події, програма може виконувати певні обчислення, візуалізацію даних або взаємодіяти з іншими частинами програми.

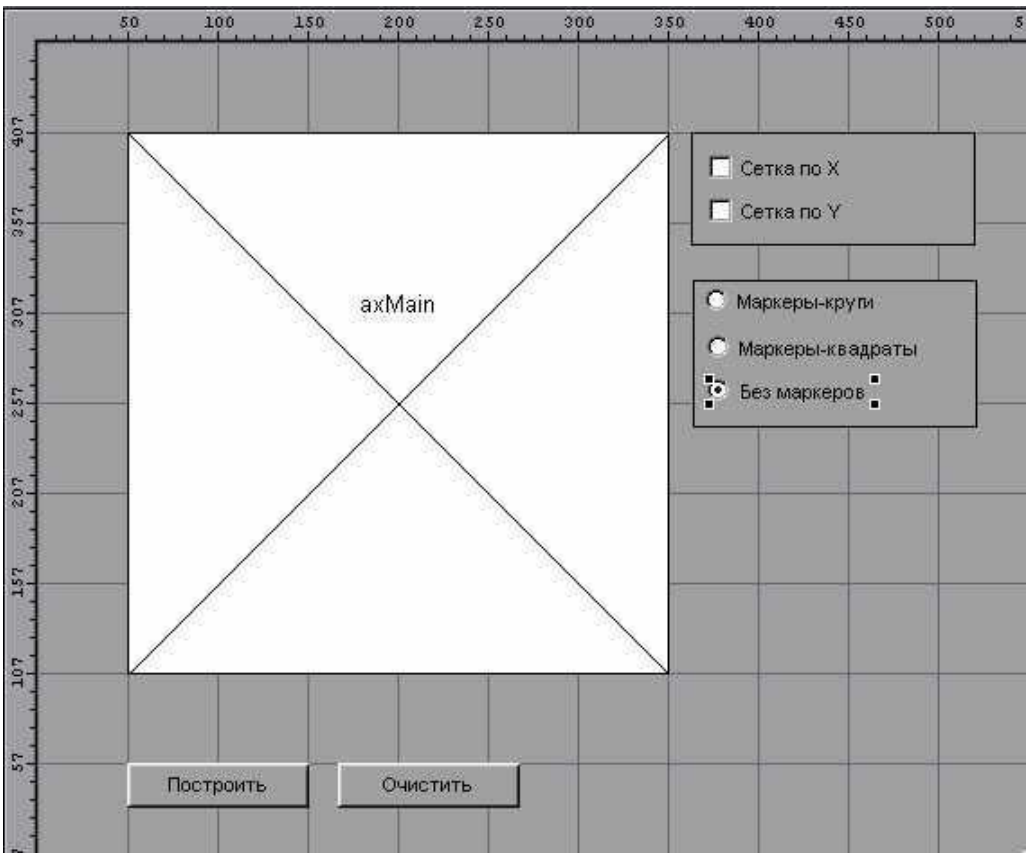

Рис. 2.5. Додавання групи перемикачів

Програмування подій в середовищі MATLAB дозволяє створювати інтерактивні та відзивчиві програми, які взаємодіють з користувачем у реальному часі. Використання обробників та слухачів подій дозволяє забезпечити логіку та функціональність, яка відповідає на різні дії користувача. Це робить MATLAB потужним інструментом для розробки інтерактивних програм, що використовуються в багатьох галузях, включаючи науку, інженерію, фінанси та багато інших.

2.4. Математичний апарат інтегро-диференціювання дробового порядку

Інтегро-диференціювання дробового порядку є важливим інструментом у сучасній математиці та прикладних науках, який дозволяє узагальнити класичне диференціювання та інтегрування на нецілих порядках. Цей

математичний апарат знайшов широке застосування в різних галузях, таких як фізика, інженерія, економіка, біологія та інші, де зустрічаються фрактальні, нестаціонарні та нелінійні явища.

Диференціювання та інтегрування дробового порядку використовуються для опису динаміки систем, які проявляють нелокальні, пам'ятливі та довготривалі ефекти. Вони дозволяють моделювати розповсюдження хвиль, розпад речовини, електричні коливання, рух частинок, фінансові ряди та багато інших явищ. Інтегро-диференціювання дробового порядку дозволяє враховувати нестаціонарність та довготривалі ефекти, що можуть бути несуттєвими для класичних диференціальних рівнянь та інтегралів.

Однією з основних переваг інтегро-диференціювання дробового порядку є його здатність моделювати явища зі зворотними зв'язками, нестаціонарність та нелінійність. Воно знайшло широке застосування у багатьох наукових галузях, включаючи фізику, інженерію, біологію та фінанси. Наприклад, у фізиці, інтегро-диференціювання дробового порядку використовується для моделювання поведінки матеріалів зі складною реологією, таких як в'язкі рідини або полімерні матеріали. У фінансовій математиці, дробове інтегрування допомагає управляти ризиками та прогнозувати цінові рухи на фінансових ринках.

Окрім основних понять дробового порядку, інтегро-диференціювання також включає в себе розширені техніки та методи для розв'язання диференціальних рівнянь дробового порядку, аналізу стійкості та властивостей систем дробового порядку, апроксимації та чисельного розв'язання дробових рівнянь та багато інших аспектів.

Для n-кратного інтеграла відома формула

$$\int_a^x dx \int_a^x dx \ \ldots \int_a^x \varphi(x)\,dx = \frac{1}{(n-1)!}\int_a^x (x-t)^{n-1}\varphi(t)\,dt, \tag{2.1}$$

Помітивши, що (n - 1)! = Γ(n), бачимо, що правій частині в (2.1) можна надати сенс і при нецілих значеннях n. Тому природно визначати інтегрування нецілого порядку таким чином.

Нехай φ(x) Є L_1(a, b). Інтеграли

$$(I_{a+}^{\alpha}\varphi)(x) \overset{\text{def}}{=} \frac{1}{\Gamma(\alpha)} \int_a^x \frac{\varphi(t)}{(x-t)^{1-\alpha}}\, dt, \quad x > a, \tag{2.2}$$

$$(I_{b-}^{\alpha}\varphi)(x) \overset{\text{def}}{=} \frac{1}{\Gamma(\alpha)} \int_x^b \frac{\varphi(t)\, dt}{(t-x)^{1-\alpha}}, \quad x < b, \tag{2.3}$$

де α > 0, називаються інтегралами дробового порядку α. Перший з них називають іноді лівостороннім, а другий - правостороннім. Оператори I^{α}_{a+} I^{α}_{b-} називають операторами дробового інтегрування. Інтеграли (2.2), (2.3) прийнято називати також дробовими інтегралами Рімана-Ліувілля.

Відзначимо просту зв'язок між операторами I^{α}_{a+} I^{α}_{b-}

$$QI_{a+}^{\alpha} = I_{b-}^{\alpha}Q, \quad QI_{b-}^{\alpha} = I_{a+}^{\alpha}Q, \tag{2.4}$$

Де Q – операторов ”відображення” : $(Q\varphi)(x) = \varphi(a+b-x)$.

$$\int_a^b \varphi(x)\,(I_{a+}^{\alpha}\psi)(x)\, dx = \int_a^b \psi(x)\,(I_{b-}^{\alpha}\varphi)(x)\, dx, \tag{2.5}$$

Дробове інтегрування має властивість

$$I_{a+}^{\alpha} I_{a+}^{\beta}\varphi = I_{a+}^{\alpha+\beta}\varphi, \quad I_{b-}^{\alpha} I_{b-}^{\beta}\varphi = I_{b-}^{\alpha+\beta}\varphi, \quad \alpha > 0, \quad \beta > 0. \tag{2.6}$$

Для функції f (x), заданої на відрізку [a, b] , кожне з виразів

$$(\mathcal{D}_{a+}^{\alpha} f)(x) = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dx} \int_a^x \frac{f(t)\,dt}{(x-t)^{\alpha}}, \tag{2.7}$$

$$(\mathcal{D}_{b-}^{\alpha} f)(x) = -\frac{1}{\Gamma(1-\alpha)} \frac{d}{dx} \int_x^b \frac{f(t)\,dt}{(t-x)^{\alpha}} \tag{2.8}$$

називається дробовою похідною порядку α, 0 <α <1, відповідно лівосторонньої і правобічної. Дробові похідні (2.7), (2.8) називають зазвичай похідними Рімана - Ліувілля.

2.5. Математична модель чисельності популяцій з урахування часової нелокальності процесу

Математичне моделювання чисельності популяцій є важливим інструментом для вивчення та розуміння динаміки та поведінки популяційних систем в природних та соціальних науках. Одним з ключових аспектів, який може бути врахований в таких моделях, є часова нелокальність процесу, що вказує на те, що зміни в чисельності популяції в певний момент часу можуть залежати від значень в минулому, а не тільки в поточний момент часу.

Математичні моделі, які враховують часову нелокальність процесу, зазвичай базуються на інтегро-диференціальних рівняннях з дробовим порядком або на диференціально-різницевих рівняннях. Вони використовуються для опису динаміки чисельності популяції на основі розподілу популяції в просторі та часі. Такі моделі можуть враховувати різні фактори, що впливають на чисельність популяції, такі як взаємодія між особинами, розмноження, смертність, міграція та інші.

Одна з широко використовуваних моделей з урахуванням часової нелокальності процесу - це модель інтегро-диференціального типу. В цій

моделі чисельність популяції в будь-який момент часу залежить від інтегрального внеску від минулих моментів часу, що відображає нелокальні ефекти. Це може бути використано, наприклад, для моделювання поширення хвороб, де зараження в даний момент часу залежить від контактів з інфікованими особинами у минулому.

Математична модель чисельності популяцій з урахування часової нелокальності процесу може бути записана наступним чином:

$$\frac{\partial^{\alpha} R}{\partial t^{\alpha}} = \alpha(R)R - V(R)N$$
$$\frac{\partial^{\alpha} N}{\partial t^{\alpha}} = N\left[\eta V(R) - m\right] + D\frac{\partial^{2} N}{\partial x^{2}} \qquad (2.9)$$

У середовищі MATLAB можна реалізувати такі математичні моделі чисельності популяцій з урахуванням часової нелокальності процесу. Застосовуючи інтегрування та чисельні методи, можна отримати чисельні розв'язки цих моделей для аналізу та прогнозування поведінки популяційних систем.

Математична модель чисельності популяцій з урахуванням часової нелокальності процесу є потужним інструментом для вивчення та аналізу динаміки популяційних систем. Вона дозволяє враховувати залежність чисельності від минулих значень та нелокальні ефекти, що може бути важливим для розуміння поведінки складних систем. MATLAB надає зручні інструменти для реалізації та чисельного аналізу таких моделей, що робить його корисним середовищем для дослідження популяційних процесів з урахуванням часової нелокальності.

2.3. Чисельні методи розв’язку систем диференціальних рівнянь.

Диференціальні рівняння з похідними дробового порядку знаходять широке застосування в різних галузях, таких як фізика, інженерія, біологія та економіка. Розв'язування таких рівнянь може бути викликано складністю їх аналітичного підходу. Однак, методи чисельного розв'язання, зокрема метод декомпозиції області (МДО), стали ефективним інструментом для розв'язання цих задач.

$$D_t^{\alpha} y = f(t, y); \tag{2.10}$$

МДО є чисельним методом, який базується на розбитті диференціального рівняння на менші підобласті, а потім розв'язує ці підобласті незалежно одна від одної. Кожна підобласть може бути розв'язана окремо з використанням стандартних чисельних методів, таких як метод скінченних різниць або метод скінченних елементів. Потім отримані розв'язки об'єднуються для отримання розв'язку задачі в цілому.

Основна ідея МДО полягає в тому, що диференціальне рівняння розбивається на підобласті таким чином, щоб границі між ними були вибрані так, щоб забезпечити належне зв'язку між розв'язками на суміжних підобластях. Цей зв'язок може бути забезпечений шляхом включення граничних умов або умов зв'язку між розв'язками на границях підобластей. Зазвичай розділення підобластей здійснюється таким чином, що розмір підобластей зменшується по мірі наближення до границі задачі.

Розв’язок (2.10) можна отримати у вигляді:

$$y(t) = \sum_{k=0}^{\lceil \alpha \rceil} y^{(k)}_0 \frac{t^k}{k!} + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(t, y(t)) d\tau; \tag{2.11}$$

Для апроксимації операторів інтегрування дробового порядку скористаємося формулою квадратурного трапецеїдального добутку

$$y(t_{n+1}) = \sum_{k=0}^{[\alpha]} y_0^{(k)} \frac{t^k}{k!} + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n+1} a_{j,n+1} f(t_j, y(t_j)); \tag{2.11}$$

$$a_{j,n+1} = \begin{cases} n^{\alpha+1} - (n-\alpha)(n+1)^\alpha, & j = 0, \\ (n-j+2)^{\alpha+1} + (n-j)^{\alpha+1} - 2(n-j+1)^{\alpha+1}, & 1 \le j \le n, \\ 1, & j = n+1. \end{cases} \tag{2.12}$$

Однією з переваг МДО є його здатність ефективно розв'язувати задачі зі складними геометричними областями або змінними коефіцієнтами в диференціальних рівняннях. Метод дозволяє використовувати різні чисельні методи для кожної підобласті, використовуючи їх переваги та адаптувати розбиття області відповідно до особливостей задачі.

МДО також може бути застосований до систем диференціальних рівнянь з похідними дробового порядку, що відображає більш складні моделі залежності. Цей метод дозволяє зручно моделювати та аналізувати різні фізичні, біологічні та інженерні процеси, де поява дробових похідних виправдана фізичними або експериментальними даними.

Метод декомпозиції області є потужним чисельним методом для розв'язання диференціальних рівнянь з похідними дробового порядку. Його основними перевагами є ефективність, здатність до моделювання складних геометричних областей та адаптабельність до різних чисельних методів. МДО дозволяє вирішувати складні задачі оптимізації та аналізу динамічних систем з дробовими похідними, що розширює область застосування чисельних методів у наукових дослідженнях та інженерних застосуваннях.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Інтерфейс розробленого прикладного програмного забезпечення

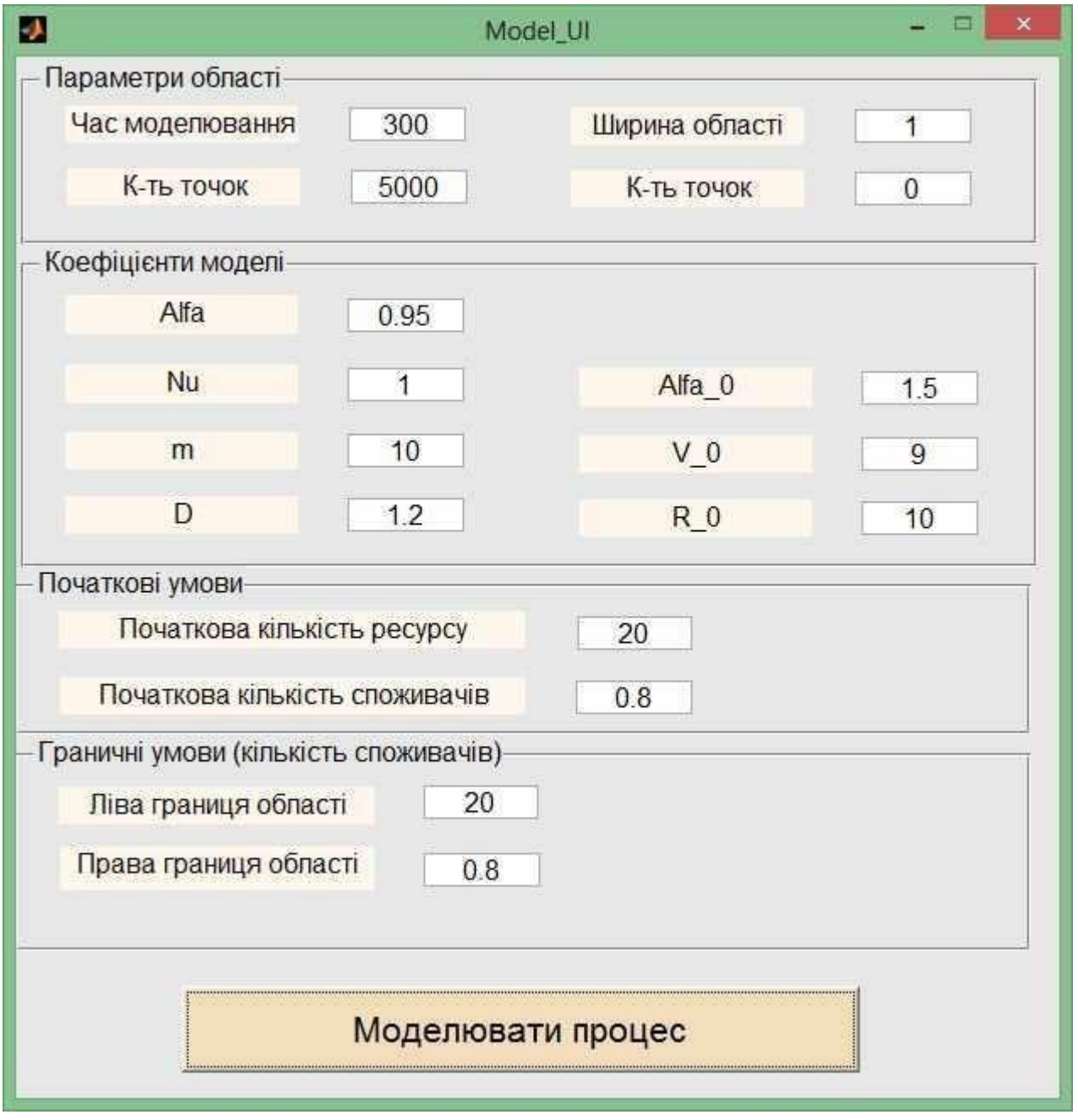

Рис. 3.1. Інтерфейс прикладного програмного забезпечення

На рисунку 3.1 показано інтерфейс прикладного програмного забезпечення, яке було розроблено для реалізації математичної моделі динаміки обсягу популяцій у екологічній системі "ресурс-споживач". Інтерфейс

складається з декількох областей, зокрема "Параметри області", "Коефіцієнти моделі", "Початкові умови", "Граничні умови (кількість споживачів)" та графічного відображення залежностей.

Область "Параметри області" дозволяє задати основні параметри процесу моделювання, такі як тривалість моделювання, ширина просторового розподілу популяцій та кількість точок дискретизації у просторовій та часовій координатах.

Область "Коефіцієнти моделі" дозволяє задати основні параметри математичної моделі, яка описує динаміку обсягу популяцій у екологічній системі "ресурс-споживач".

Область "Початкові умови" дозволяє задати початкові значення обсягу ресурсу та кількості споживачів.

Область "Граничні умови (кількість споживачів)" дозволяє задати значення кількості споживачів на границі області.

Цей інтерфейс допомагає користувачу зручно встановити необхідні параметри та початкові умови для моделювання динаміки популяцій у вказаній екологічній системі "ресурс-споживач".

3.2. Аналіз отриманих результатів

При розв’язувані даної моделі використовувався метод декомпозиції області. Для такого набору параметрів $\alpha_0 = 1.5, V_0 = 9, R_0 = 10, K = 5, \eta = 1, m = 10$ розв’яжемо систему на такій області $x \in [0,1]$, $t \in [0,300]$ застосувавши метод декомпозиції області для наступних початково-крайових умов: $R_0 = 20;\ N_0 = 2$

Розглянемо поведінку моделі при традиційному значення параметра дробової похідної $\alpha = 1$.

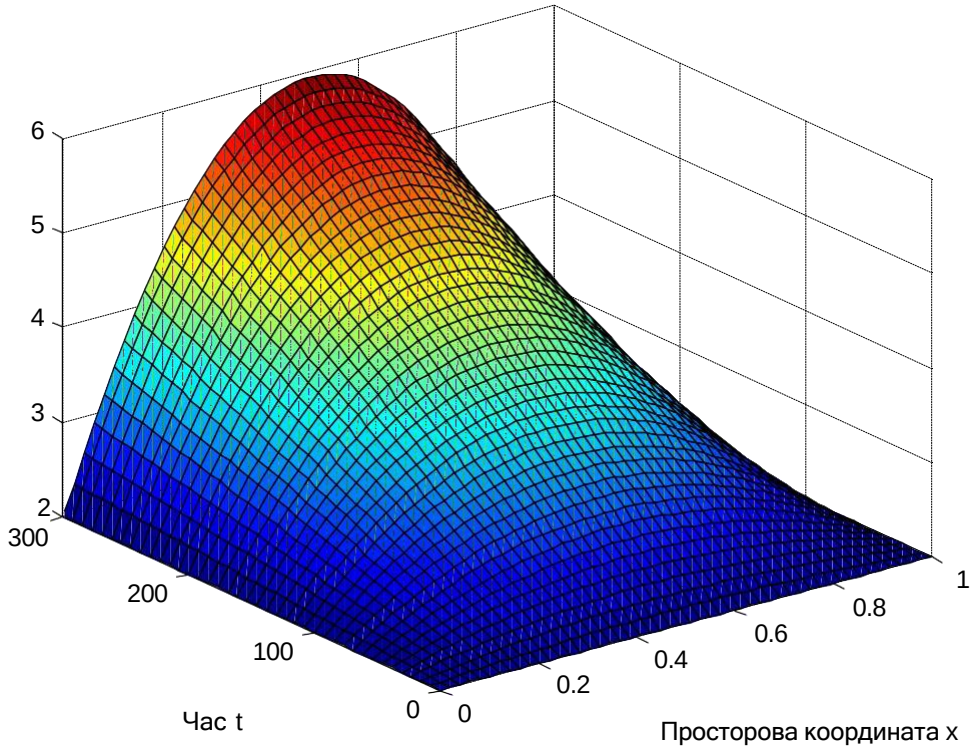

Рис. 3.2. Густина споживача залежно від простору та часу.

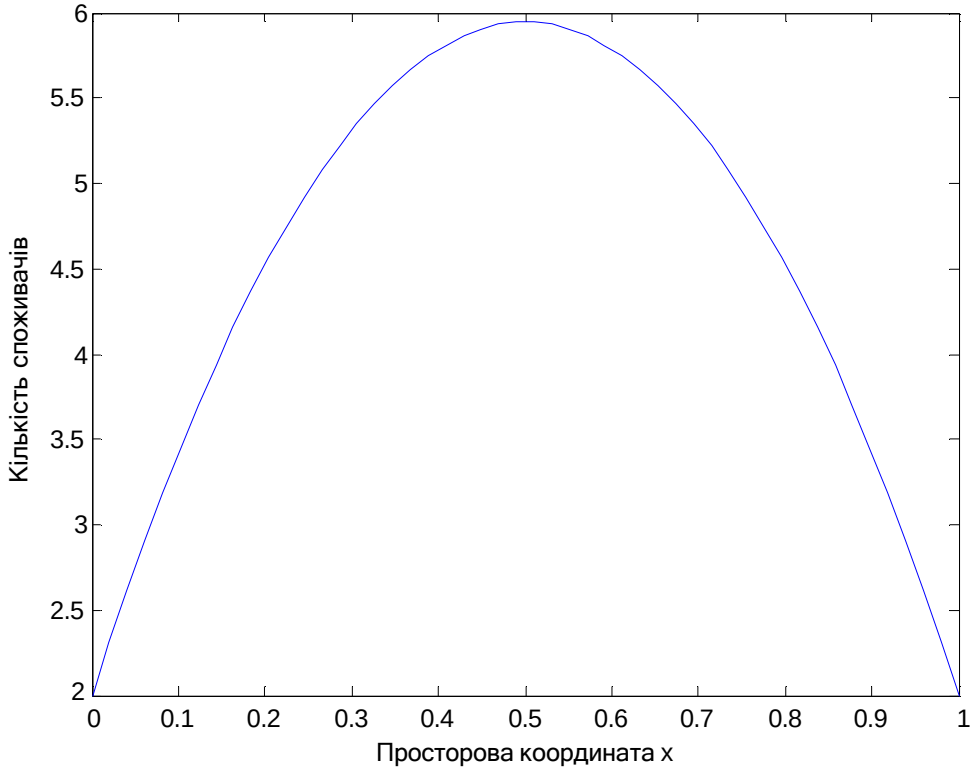

Рис. 3.3. Густина споживача в кінцевий момент часу.

Як видно на рис. 3.2 зображено динаміку розподілу споживачів по просторовій та часовій координаті. Можна бачити, що максимальна концентрація споживачів засереджена у центрі області. На рис. 3.3 забражено розподіл споживачів в кінцевий час моделювання. Цей графік підтверджує описану вище динаміку.

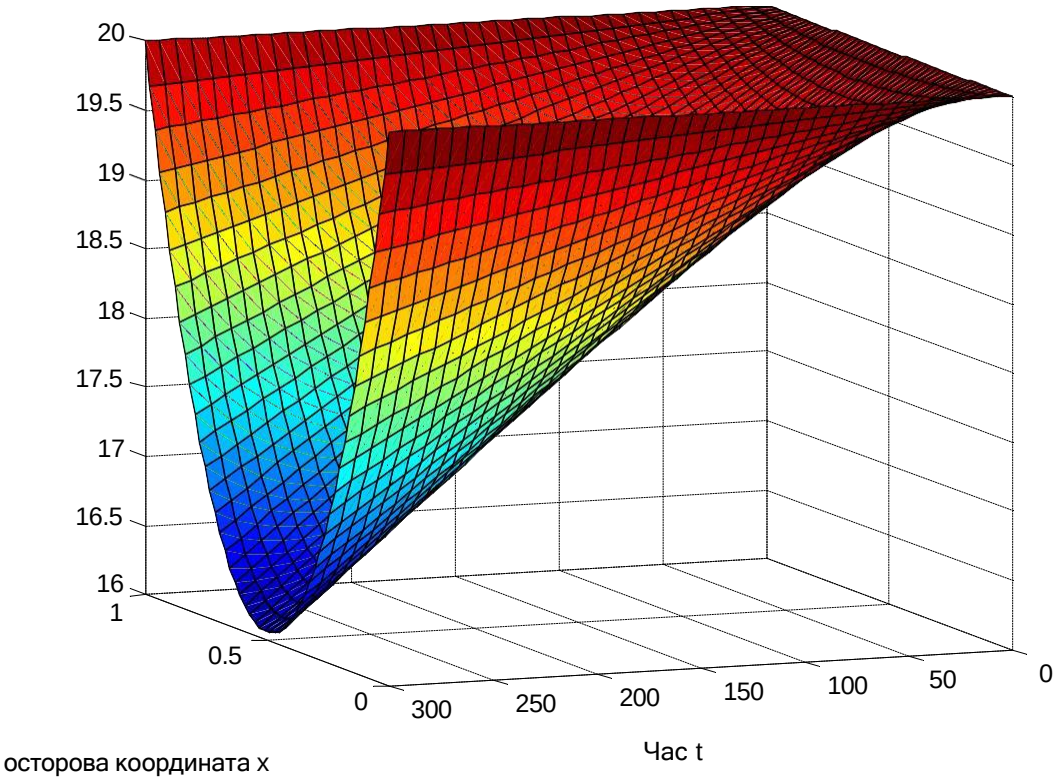

Рис. 3.4. Густина ресурсу залежно від простору та часу.

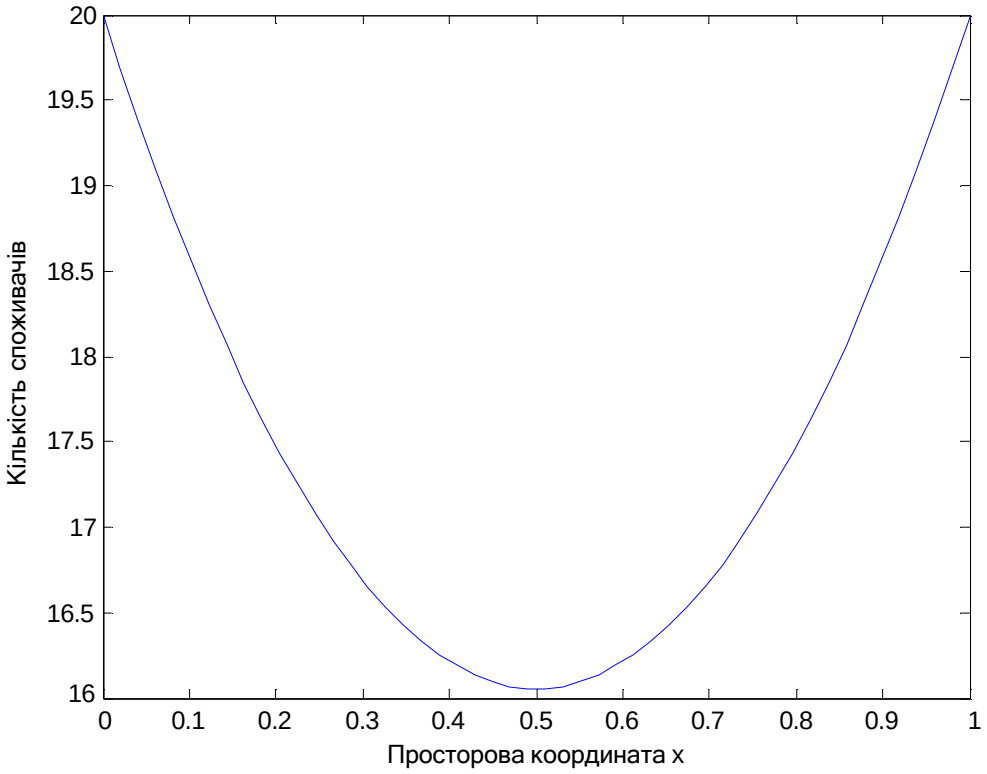

Рис. 3.5. Густина ресурсу в кінцевий момент часу.

Як видно на рис. 3.4 зображено динаміку розподілу ресурсів по просторовій та часовій координаті. Можна бачити, що максимальна концентрація споживачів засереджена по краях області, на максимальній відстані від найвищої концентрації споживачів. На рис. 3.5 забражено розподіл ресурсу в кінцевий час моделювання.

Розглянемо тепер поведінку системи з урахуванням часової нелокальності процесу. Це можна зробити шляхом задання параметра дробової похідної у значення із проміжку від 0 до 1. Виберемо $\alpha = 0{,}95$

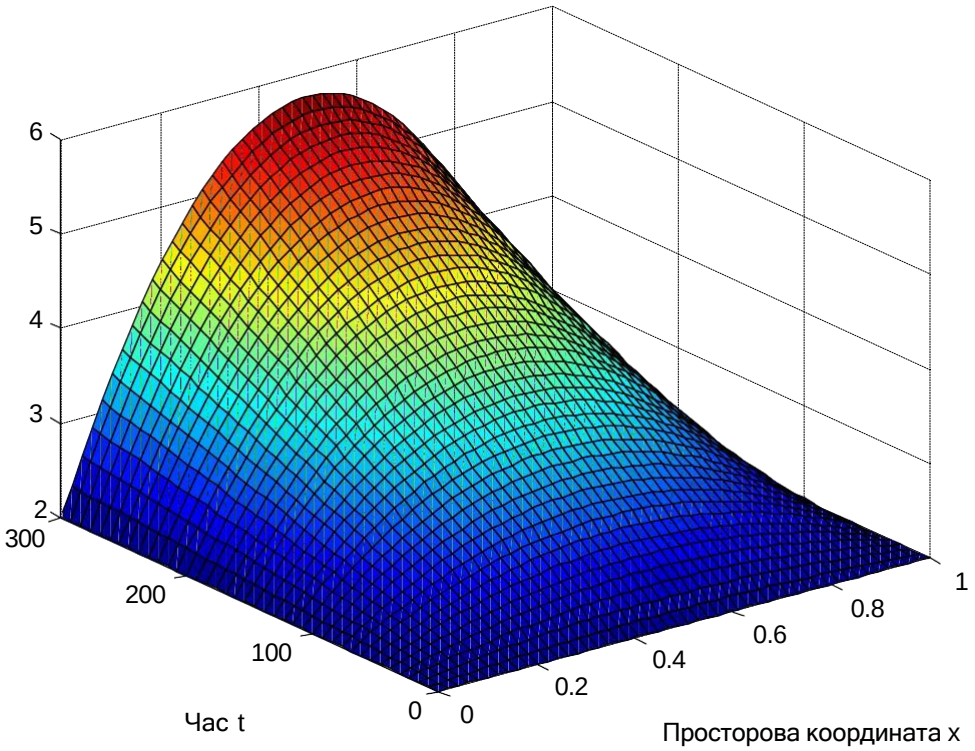

Рис. 3.6. Густина споживача залежно від простору та часу.

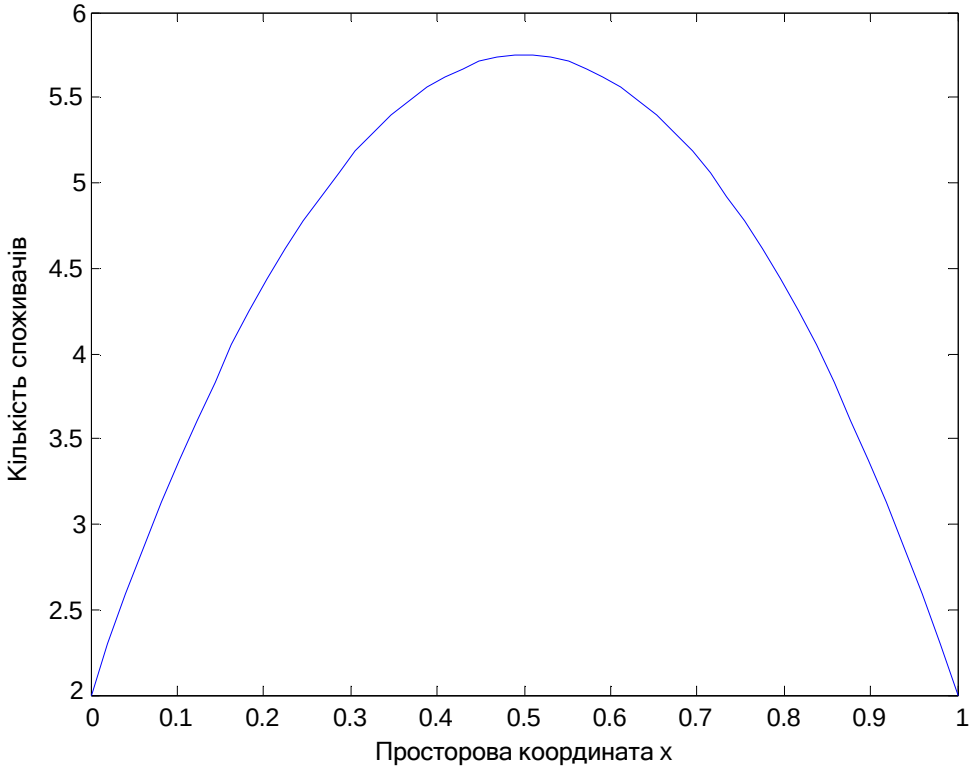

Рис. 3.7. Густина споживача в кінцевий момент часу.

На рис. 3.6 зображено динаміку розподілу споживачів по просторовій та часовій координаті з урахуванням часової нелокальності процесу. Можна бачити, що максимальна концентрація споживачів нижча у порівнянні із традиційним випадком моделі. На рис. 3.7 забражено розподіл споживачів в кінцевий час моделювання.

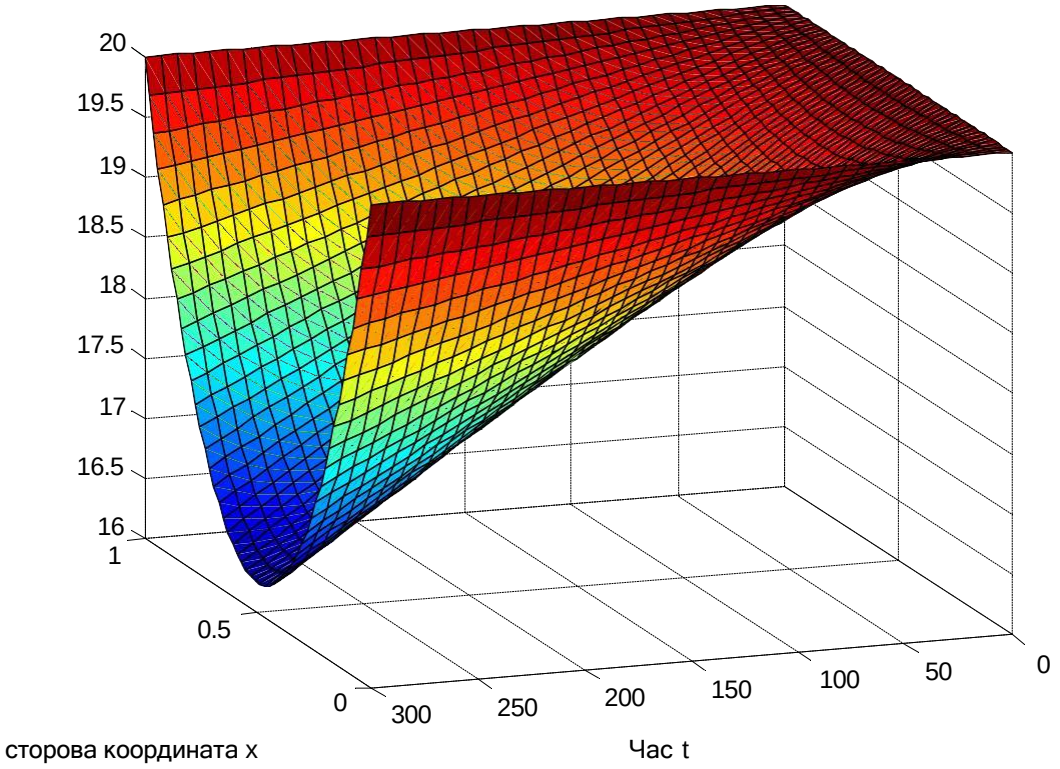

Рис. 3.8. Густина ресурсу залежно від простору та часу.

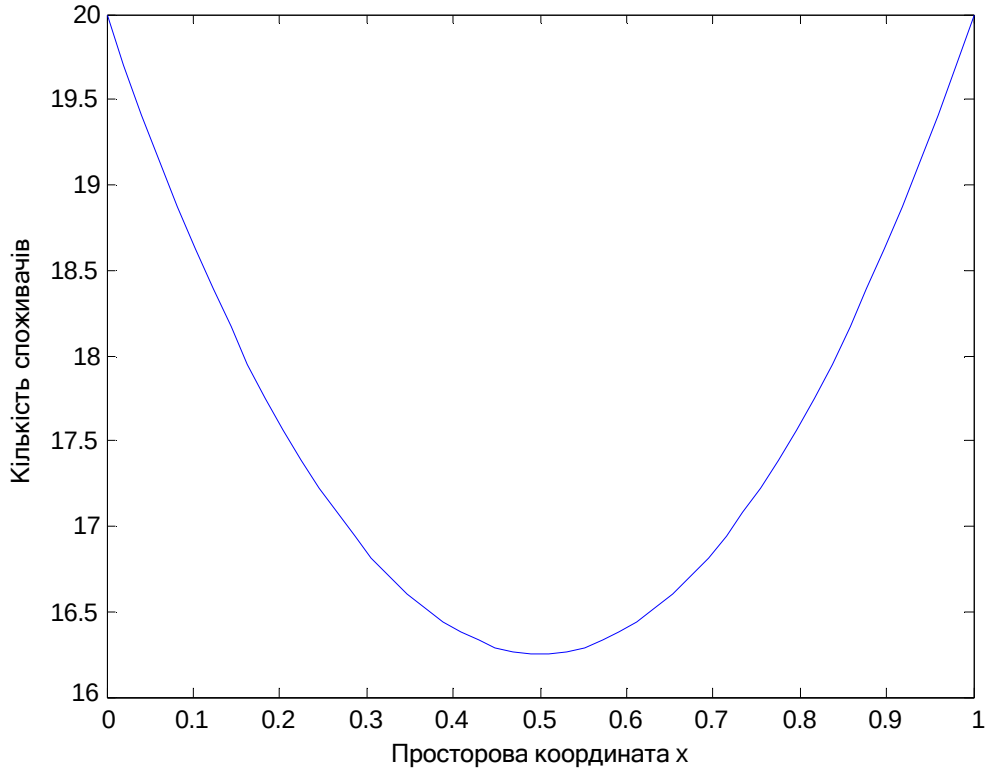

Рис. 3.9. Густина ресурсу в кінцевий момент часу.

На рис. 3.8 зображено динаміку розподілу ресурсу по просторовій та часовій координаті з урахуванням часової нелокальності процесу. Можна бачити, що максимальна концентрація ресурсу вища у порівнянні із традиційним випадком моделі. На рис. 3.9 забражено розподіл споживачів в кінцевий час моделювання.

3.3. Вимоги до програмного та апаратного забезпечення

Вимоги до програмного та апаратного забезпечення системи моделювання чисельності популяцій з урахуванням часової нелокальності процесу будуть залежати від рівня складності моделі та обсягу даних, які потрібно обробляти. Однак, основні вимоги можуть включати наступне:

Програмне забезпечення:

- Система моделювання: Потрібна програма або середовище, яке підтримує моделювання чисельності популяцій з урахуванням

часової нелокальності процесу. Наприклад, MATLAB, Python з використанням наукових бібліотек, таких як NumPy та SciPy.

- Обробка даних: Для роботи з великими обсягами даних може знадобитися база даних або інструменти для ефективної обробки та зберігання даних.
- Візуалізація: Можливість графічного відображення результатів моделювання для аналізу та візуалізації чисельності популяцій.
- Оптимізація: Можливість використання оптимізаційних алгоритмів для підбору параметрів моделі та пошуку оптимального розв'язку.

Апаратне забезпечення:

- Обчислювальна потужність: Моделювання чисельності популяцій може бути обчислювально витратним процесом, особливо якщо задіяні складні моделі або великі обсяги даних. Вимоги до потужності обчислень залежатимуть від обсягу та складності задачі.
- Пам'ять: Необхідна достатня кількість оперативної пам'яті для ефективної обробки та зберігання даних моделі.
- Зберігання даних: Можливість зберігання та доступу до великих обсягів даних, які можуть виникати під

Таким чином дане завдання було фізично реалізовано на системі, яка має наступні характеристики:

- Процесор: Intel Core i3-380m
- Відеокарта: NVIDIA GT540m
- Кількість оперативної пам’яті: 4GB.
- Жорсткий диск: 500GB

ВИСНОВКИ

Було розроблено програмне забезпечення за допомогою середовища Matlab, яке дозволяє обчислювати обсяг популяцій у екологічній системі "ресурс-споживач" з урахуванням часової нелокальності процесу. Інтерфейс програмного забезпечення є інтуїтивно зрозумілим та зручним для користувача. Результати обчислень та графічні залежності можуть бути виведені у зручному форматі для подальшого аналізу. Користувач може вводити вхідні дані безпосередньо у систему. Загалом, можна зробити наступні висновки.

Незважаючи на аналогію між біологічними та економічними системами, які відображають автоколивальні процеси, застосування математичних моделей типу "хижак-жертва" для пояснення та прогнозування їх поведінки є актуальним з кількох причин:

а) Прості постулати класичної моделі Лоткі-Вольтерра надають можливість точно описати та прогнозувати реальну поведінку біологічних систем.

б) Існує велика розмаїтість моделей, що дозволяє обрати певну модифіковану модель для опису та прогнозування конкретної ситуації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лоценко О. І., Гурін Ю. Г. Моделювання та дослідження динаміки чисельності популяцій. Київ: Видавничий дім "Слово", 2010.
2. Халіна Ю. Моделі та методи дослідження чисельності популяцій. Київ: Наукова думка, 2015.
3. Голубєва Л. С., Голубєва І. О., Терехов В. В. Моделювання та аналіз чисельності популяцій. Київ: Видавництво "Майстер-клас", 2018.
4. Черкасова В. А., Кислюк О. Л., Стеблюк В. М. Математичне моделювання біологічних систем. Київ: Видавництво "Либідь", 2013.
5. Кузьмін О. В., Капустян О. В., Сидорова Т. П. Моделі та алгоритми динаміки популяцій. Київ: Видавництво Національного авіаційного університету, 2018.
6. Яценко В. М., Кривошея Л. В., Рибін О. А. Моделі та методи дослідження біологічних систем. Київ: Видавничий центр "Київський університет", 2011.
7. Soetaert K., Petzoldt T., Setzer R. W. Solving Differential Equations in R: Package deSolve. Journal of Statistical Software, 2010, 33(9), 1-25.
8. Edelstein-Keshet L. Mathematical Models in Biology. Society for Industrial and Applied Mathematics, 2005.
9. Murray J. D. Mathematical Biology: I. An Introduction. Springer, 2002.
10. Brauer F., Castillo-Chávez C. Mathematical Models in Population Biology and Epidemiology. Springer, 2000.
11. Ludwig D., Jones D. D. Qualitative Analysis of Biological Populations. Princeton University Press, 1974.
12. Rinaldi S., Muratori S., Kuznetsov V. A. Numerical Methods for Delay Differential Equations. Oxford University Press, 2003.
13. Banks H. T., Tran H. T., Hu S., Castillo-Chávez C. Mathematical and Statistical Estimation Approaches in Epidemiology. Springer, 2009.

14. Ma J., Takeuchi Y. Permanence and Global Attractivity for Competitive Lotka-Volterra Systems with Delay. Nonlinear Analysis: Real World Applications, 2004, 5(4), 673-682.
15. Gourley S. A., Kuang Y. Dynamics of a Delayed Predator-Prey Model with Mixed Functional Responses. Nonlinear Analysis: Real World Applications, 2006, 7(1), 16-35.
16. Gopalsamy K. Stability and Oscillations in Delay Differential Equations of Population Dynamics. Springer, 1992.

ДОДАТКИ

ДОДАТОК А.

```
function varargout = Model_UI(varargin)
% MODEL_UI MATLAB code for Model_UI.fig
%      MODEL_UI, by itself, creates a new MODEL_UI or raises the
existing
%      singleton*.
%
%      H = MODEL_UI returns the handle to a new MODEL_UI or the
handle to
%      the existing singleton*.
%
%      MODEL_UI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MODEL_UI.M with the given input
arguments.
%
%      MODEL_UI('Property','Value',...) creates a new MODEL_UI or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before Model_UI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Model_UI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Model_UI

% Last Modified by GUIDE v2.5 20-Jun-2023 19:37:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',        mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Model_UI_OpeningFcn, ...
                   'gui_OutputFcn',  @Model_UI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Model_UI is made visible.
function Model_UI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Model_UI (see VARARGIN)

% Choose default command line output for Model_UI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Model_UI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Model_UI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

alfa = str2double(get(handles.edit21,'String'));

Nu = str2double(get(handles.edit1,'String'));
p_m = str2double(get(handles.edit2,'String'));
D = str2double(get(handles.edit17,'String'));
alfa_0 = str2double(get(handles.edit18,'String'));
V_0 = str2double(get(handles.edit19,'String'));
```

```
R_0 = str2double(get(handles.edit20,'String'));

R0 = str2double(get(handles.edit7,'String'));
N0 = str2double(get(handles.edit14,'String'));

V_L = str2double(get(handles.edit15,'String'));
V_R = str2double(get(handles.edit16,'String'));

t_end = str2double(get(handles.edit3,'String'));
n = str2double(get(handles.edit4,'String'));
h = str2double(get(handles.edit5,'String'));
m = str2double(get(handles.edit6,'String'));
Main(alfa, t_end, n, h, m, Nu, p_m, D, alfa_0, V_0, R_0, R0, N0,
V_L, V_R);



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double


% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double


% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of
edit7 as a double


% --- Executes during object creation, after setting all
properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of
edit8 as a double


% --- Executes during object creation, after setting all
properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of
edit9 as a double


% --- Executes during object creation, after setting all
properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of
edit3 as a double


% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of
edit4 as a double


% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of
edit5 as a double


% --- Executes during object creation, after setting all
properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of
edit6 as a double


% --- Executes during object creation, after setting all
properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of
edit11 as a double


% --- Executes during object creation, after setting all
properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of
edit12 as a double
```

```
% --- Executes during object creation, after setting all
properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of
edit13 as a double


% --- Executes during object creation, after setting all
properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of
edit14 as a double


% --- Executes during object creation, after setting all
properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of
edit15 as a double


% --- Executes during object creation, after setting all
properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of
edit16 as a double


% --- Executes during object creation, after setting all
properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of
edit17 as a double


% --- Executes during object creation, after setting all
properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of
edit18 as a double


% --- Executes during object creation, after setting all
properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of
edit19 as a double


% --- Executes during object creation, after setting all
properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of
edit20 as a double


% --- Executes during object creation, after setting all
properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of
edit21 as a double


% --- Executes during object creation, after setting all
properties.
function edit21_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```