МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

В.О.Любчак, Л.Г.Острівна

КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ

НАВЧАЛЬНИЙ ПОСІБНИК

для студентів та аспірантів фізико-математичних, інженерних та економічних спеціальностей

Затверджено
редакційно-видавничою
радою університету.
Протокол № 3 від 04.09.2001 р.

Суми
Видавництво Сумського державного університету
2002

ПЕРЕДМОВА

Сьогодні для фахівця - інженера, економіста знання методів оптимізації настільки ж необхідне, як і знання основ математики та інших дисциплін, що стали традиційними.

У практичній діяльності часто з багатьох можливих способів розв'язання задачі необхідно вибрати оптимальний. Наприклад, із декількох варіантів перевезення сировини споживачам потрібно вибрати найбільш дешевий, але такий, що враховує обмеження на допустимі терміни постачань; із можливих планів розкрою матеріалу вибрати такий, що дозволяє виконати план при найменшій кількості відходів і т.д. Складність задач оптимізації істотно залежить від їх розмірності, тобто від кількості аргументів цільової функції.

У багатьох випадках задача пошуку оптимальних розв'язків може бути формалізована і розв'язана точно або приблизно відомими методами.

Сьогодні існує багато різної методичної і навчальної літератури з методів оптимізації. Нашим завданням було зібрати в єдиному посібнику опис можливостей розроблених пакетів, подати інформативний перелік матеріалів присвячених оптимізації, а також створити дистанційний курс з питань комп'ютерної реалізації методів оптимізації.

Посібник має на меті:

- *надати інформацію про алгоритми та чисельну реалізацію методів оптимізації;*
- *описати електронний варіант курсу;*
- *надати користувачу можливість працювати з пакетами програм;*
- *розповісти про практичне застосування оптимізаційних методів.*

Працювати з посібником рекомендуємо одночасно з електронним варіантом курсу *http://dl.sumdu.edu.ua/mo/*

Посібник не має на меті повне, докладне викладення методів оптимізації. Більше того, теоретичні основи оптимізаційних методів продані стисло, без доведень теорем та детальних описів алгоритмів.

Автори рекомендують звернутися до світових розробок з теорії методів оптимізації, а саме до електронної версії лекцій “Lecture Notes on Optimization” професора Правіна Вараї (University of California, Berkeley) [35] та до матеріалів “Optimization for Engineering Systems and Solutions Manual” професора Ральфа Пайка (Louisiana State University) [36].

Посібник дає опис програмних пакетів для знаходження оптимуму, а також рекомендації щодо використання можливостей курсу та розроблених програм. З можливостями програмних пакетів можна ознайомитися більш докладно, якщо звернутися до електронного курсу, що знаходиться за адресою *http://dl.sumdu.edu.ua/mo/*

Вважаємо, що посібник буде корисним не тільки для студентів та аспірантів фізико-математичних та інженерних спеціальностей, але й для інженерів, економістів та наукових працівників.

Автори висловлюють подяку В.А.Хажанцю, О.І.Божко, Н.В.Тиркусовій, В.В.Авраменко.

ЧАСТИНА І ДИСТАНЦІЙНИЙ КУРС "КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ"

ТЕМА 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО КУРС

Курс “Комп’ютерна реалізація методів оптимізації” оформлено в електронному вигляді і він доступний дистанційно за допомогою мережі Internet.

Дистанційний курс організований так, що не потребує встановлення, настроювання та супроводу з боку учасника курсів, і стає доступним відразу після безпосереднього відвідування відповідного розділу Web-сервера СумДУ. Для використання можливостей системи потрібні мінімальні навички роботи як із комп'ютером, так і з мережею.

1.1 СТРУКТУРА ДИСТАНЦІЙНОГО КУРСУ

Дистанційний курс з питань комп’ютерної реалізації методів оптимізації” знаходиться за адресою *http://dl.sumdu.edu.ua/mo/*, або ж його можна знайти відвідавши офіційний сайт Сумського державного університету *http://www.sumdu.edu.ua/*

Дистанційний курс розрахований для роботи як в мережі Інтернет, так і Інтранет. Отже, відповідно до нових інформаційних технологій викладення матеріалу курсу передбачає певну методику навчання, тестування та контролю.

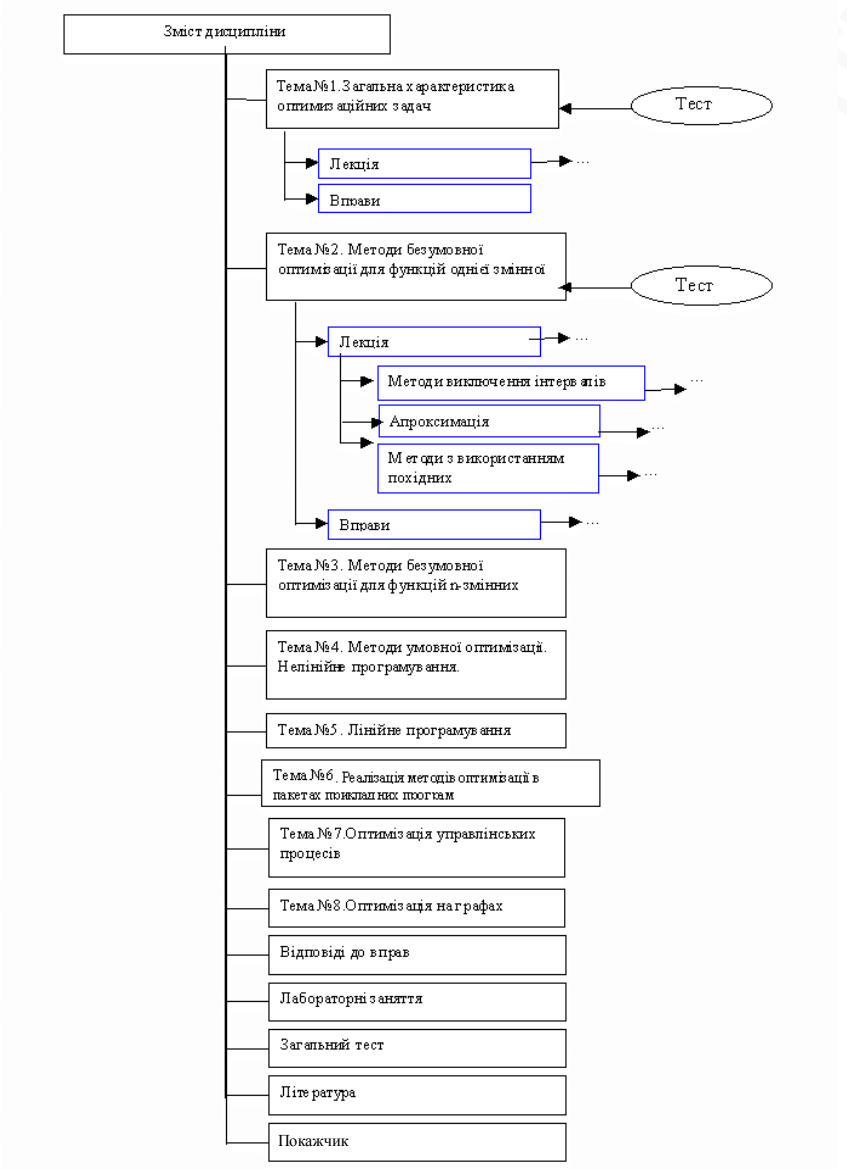


Рисунок 1.1 – Схема дистанційного курсу з дисципліни "Комп'ютерна реалізація методів оптимізації"

Даний дистанційний курс навчання складений у вигляді розгалуженої структури текстових документів:

- інформація про курс;
- зміст курсу;
- теми;
- відповіді до вправ;
- список рекомендованої літератури, а також інформація про ресурси в мережі Internet;
- покажчик на всі методи, що наявні в тексті;
- загальний тест.

Курс складається з восьми тем, кожна із яких, у свою чергу, поділена на підрозділи. Це зроблено для зручності читання та швидкої передачі по мережі (рис.1.1).

Кожна тема містить у собі лекційний матеріал, прикладні приклади, комп'ютерну реалізацію того або іншого методу (залежно від того який метод був вибраний для вивчення), контрольні питання і тест.

Лекційний матеріал підручника поданий у вигляді послідовно завантажуваних файлів. Кожну лекцію можна подати у вигляді схеми (рис.1.2).

Вправи і контрольні питання являють собою один документ - текст із завданнями, що пропонуються до розв’язання. На деякі завдання є правильні відповіді для того, щоб дати можливість тому, хто навчається, перевірити себе.

Тест - це система контролю знань, що допомагає закріпити і перевірити знання, отримані при вивченні матеріалу теми. Тест - автономна програма, розроблена для тестування різних навчальних дисциплін. По закінченні тестування пропонується переглянути результати відповідей, де червоним позначені неправильні відповіді на поставлене запитання, а зеленим – правильні, а також позначена вибрана відповідь. Пропонується користуватися цим тестом для самоконтролю набутих знань після вивчення лекції. Складені тестові питання дозволяють проходити

тест з кожної теми, а також так званий загальний тест, що містить в собі питання всього курсу.

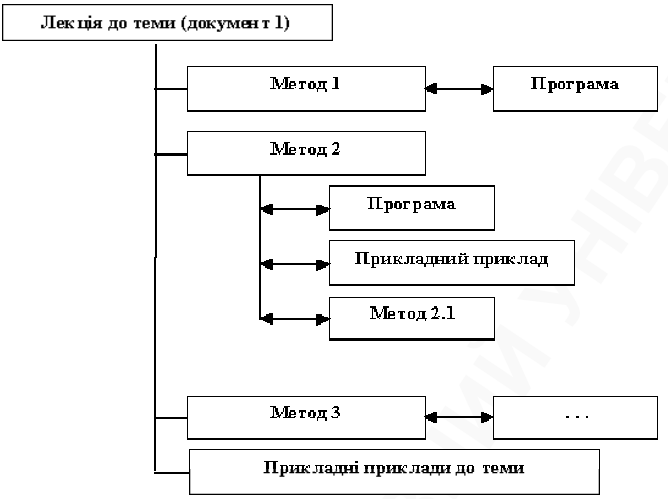


Рисунок 1.2 - Схема лекції

При розроблюванні курсу особлива увага була приділена інтерфейсу користувача і стриманості в графічному оформленні. Навігаційна оболонка містить посилання, за якими можна швидко перейти в необхідний розділ. З огляду на сьогоднішню ситуацію з технічними можливостями при доступі в Інтернет (низька швидкість передачі даних), використання графіки в курсі зведено до необхідного мінімуму.

1.2 РЕКОМЕНДАЦІЇ ЩОДО РОБОТИ З ДИСТАНЦІЙНИМ КУРСОМ

Після того як було натиснуто посилання *"Дистанційний курс "Методи оптимізації"* на екрані з'явиться початкова сторінка системи. На ній подана інформація про розробників

курсу, електронна адреса Web-майстра (тобто посилання із зазначеним *e-mail* - якщо на ній клацнути лівою кнопкою миші, то з'явиться вікно відправляння електронної пошти), а також посилання безпосередньо на матеріали курсу (їх 2, тому що текстовий матеріал курсу поданий двома мовами – українською та російською).

Після вибору мови (натиснути одне з посилань *Rus* або *Ukr*) – на екрані з'явиться перша сторінка ДКН* (рис.1.3)

У верхній частині по всій ширині екрана є ділянка (фрейм), що містить заголовок теми чи підрозділу курсу "Методи оптимізації" (у даному випадку це "Інформація про дистанційний курс"). При виборі посилання на тему ця інформація змінюється, тобто користувач майже завжди знає, в якій частині курсу він перебуває.

У лівому куті екрана розміщений малюнок університету. Це також посилання, при натисканні на яке виконується перехід до основної сторінки офіційного сайту Сумського державного університету (далі СумДУ), що знаходиться за адресою *http://www.sumdu.edu.ua/*

Зліва міститься фрейм із гіперзв'язком і схожими на меню посиланнями (далі ми називатимемо цю частину ***"Основним меню"***). Основне меню являє собою незмінну частину курсу, тобто, в якому б підрозділі не працював користувач, у нього завжди на екрані основне меню.

Основну частину екрана займає вікно для виведення даних. Воно відрізняється від усіх інших кольором фону, він - білий. Інформація в цій частині змінюється у міру того, як користувач натискає посилання (це пов'язане з особливостями гіпертекстових технологій, тобто з тим, що за кожним посиланням можна перейти до нової інформації, яка з'являється на екрані тільки після звертання до неї**).

---

* ДКН - дистанційний курс навчання

** Рекомендуємо ознайомитися з [1], [2, 28-29]

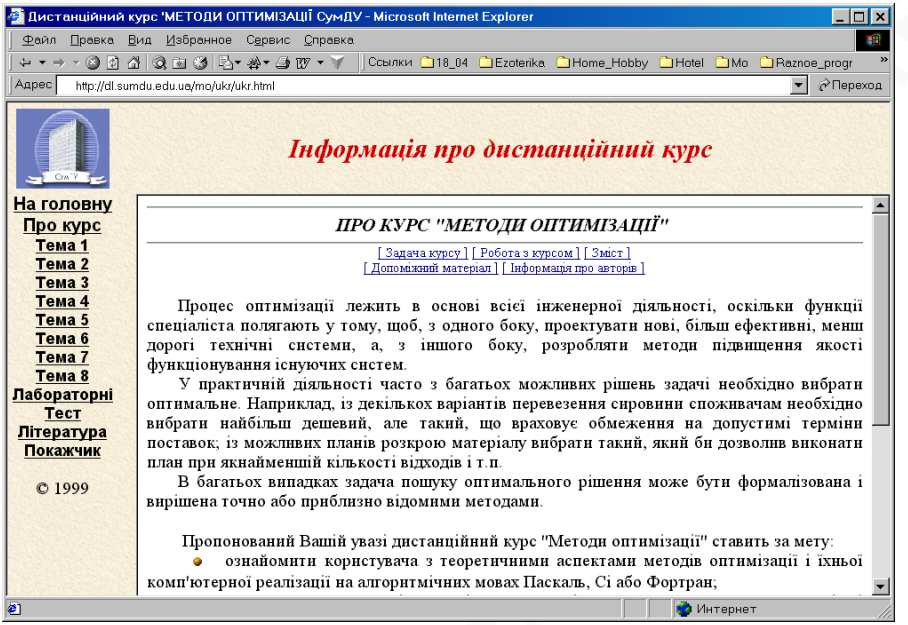


Рисунок 1.3 – Вигляд першої сторінки ДКН

Для переходу до потрібної теми мишкою виберіть відповідне їй посилання в основному меню. Вибрана тема і її лекційне наповнення з'явиться у вікні даних. У тексті також є посилання для переходу до конкретного методу. Іноді вони також подані у вигляді меню. Для перегляду використовуються смуги прокручування.

Кожна тема має короткий зміст основних її частин, він також поданий посиланнями. Це було зроблено для того, щоб користувач міг перейти до розділу, що цікавить його безпосередньо, без зайвих витрат часу на пошук посилань в тексті теми.

Для переходу між частинами підручника досить клацнути лівою кнопкою миші на потрібному Вам розділі теми – чи то «*Лекція*», чи «*Тест*», чи «*Метод "золотого" перетину*», чи будь-який інший підрозділ теми (рис. 1.4),

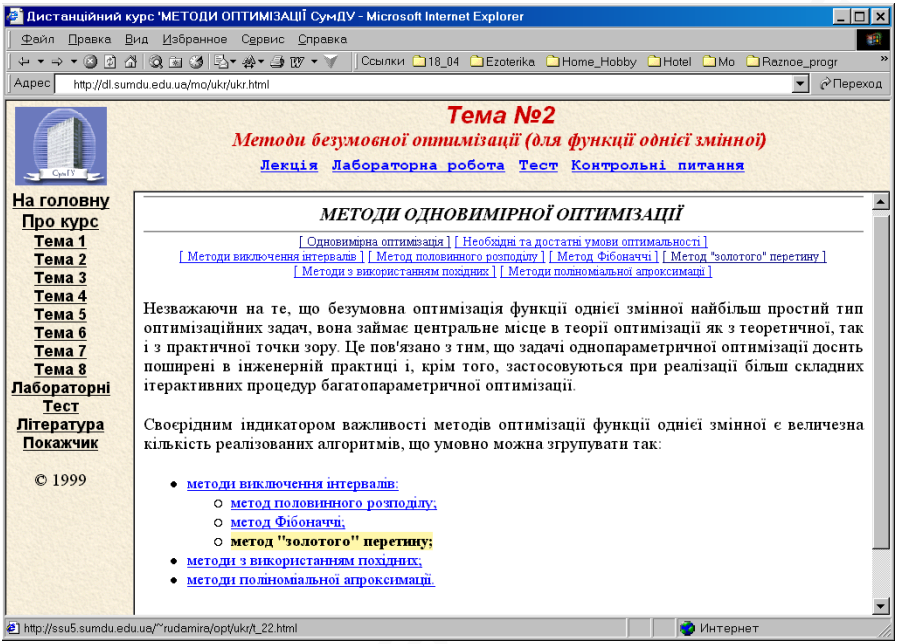


Рисунок 1.4 - Вибір розділу у темі 2

і текст з'явиться на екрані комп'ютера.

Для переходів по лекційному матеріалу передбачено посилання в тексті курсу. Вони мають такий вигляд:

метод найшвидшого спуску

і при наїзді на них мишкою - стрілка перетвориться у стиснуту руку.

Посилання, якими ще не користувалися, в тексті в основному мають яскравий синій колір. А ті, що користувач переглянув, - фіолетовий.

Як зазначалося раніше, лекційний матеріал тем курсу поділений на окремі частини (текстові файли), щоб користувач швидше отримував інформацію*.

Перехід від однієї частини лекції до іншої здійснюється за посиланням «*Далі*» або за посиланням в тексті (але в цьому випадку не гарантується послідовність викладення матеріалу). З основної частини теми (тобто з першої сторінки теми, що завантажується за посиланням «*Тема ...*») можна потрапити практично в будь-яку частину лекційного матеріалу**. Тому в багатьох частинах посилання «*Назад*» означає повернення до цієї сторінки (на схемі лекції (рис. 1.2) до документа 1).

У зв'язку із вищезгаданим необхідно відзначити, що якщо, наприклад, Ви по ланцюжку потрапили в текст опису методу з блока вправ, то, натиснувши на кнопку «*Назад*», Ви потрапляєте не на ту сторінку, що дійсно була у Вас на екрані, а зовсім на іншу. У цьому випадку ми рекомендуємо скористатися кнопкою «*Back*» на панелі вашого броузера (рис. 1.5) або вибрати один із розділів, натиснувши на посилання в основному меню курсу. Також рекомендуємо користуватися «*Покажчиком*», тому що в ньому зібрані посилання на всі наявні у даному курсі методи (додаток А).

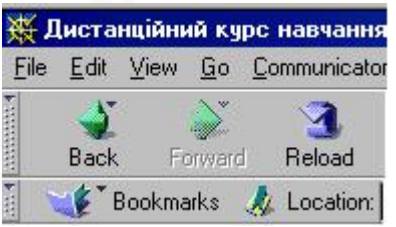


Рисунок 1.5 - Кнопка «Back» на панелі броузера Netscape Navigator

---

* Рекомендуємо ознайомитись з [1], [2, 28-29с.].

** Згадайте про короткий зміст, який є в кожній темі!

Однією із особливостей ДКН є програмний комплекс ***Met_Opt***, за допомогою якого можна розв'язати будь-яку задачу оптимізації (звичайно ж, якщо правильно задати її в математичному вигляді). Комплекс являє собою сукупність програмних продуктів, кожен із яких може бути автономною програмою. Основні складові комплексу будуть розглянуті в темі 2.

ДКН має сторінку «*Допоміжний матеріал*». Даний підрозділ містить посилання на матеріали, що можуть Вам допомогти при роботі з ДКН з дисципліни "Комп'ютерна реалізація методів оптимізації". Його склад:

- таблиця похідних;
- перелік програм;
- тестові функції;
- архіватор;
- компілятор (Фортран).

ТЕМА 2 ПРОГРАМНИЙ КОМПЛЕКС MET_OPT

2.1 СТРУКТУРА КОМПЛЕКСУ

Програмний комплекс поділений на складові частини, кожна з яких має свої підрозділи. Ці підрозділи також мають розгалуження. В ДКН подано як цілісний комплекс програм, так і окремі його складові частини (що в основному являють собою програму, яка реалізує той чи інший метод для конкретної функції).

Цей комплекс складається з таких частин:

1 Бібліотека програм, що реалізовані на Паскалі.
  1.1 Модулі методів оптимізації (*.tpu).
  1.2 Модулі допоміжних процедур (*.tpu).
  1.3 Приклади використання процедур (*.pas, *.exe).
2 Бібліотека програм, що реалізовані на Сі.
  2.1 Бібліотека методів одновимірної оптимізації.
  2.2 Бібліотека методів багатовимірної оптимізації без обмежень.

2.3 Бібліотека методів багатовимірної оптимізації з обмеженнями.

3 Бібліотека програм, створених в середовищі Delphi 5 з використанням мови Object Pascal.

4 Бібліотека програм, що реалізовані на Фортрані.

5 Програмний пакет LP-88.

Кожна частина комплексу має свої підрозділи та може використовуватися як автономна програма. В подальших розділах посібника деякі з них будуть розглянуті більш докладно.

2.2 ДОСТУП ДО КОМПЛЕКСУ

Програмний комплекс ***Met_Opt*** та його основні складові розміщені на офіційному сервері СумДУ. Більшість матеріалів зберігаєься у вигляді архівів. Матеріали, що виносяться на розгляд широкої аудиторії, являють собою готові програми та модулі. Їх можна використовувати в тому вигляді, в якому вони подані. Також є ціла низка прикладів приєднання та використання процедур, що реалізують методи оптимізації та запрограмовані в модулях. Вони мають вигляд текстових файлів з розширенням ****.pas*** чи ****.c (*.cpp)***. Питання щодо можливості перегляду основного коду написання складових програмного комплексу вирішуються через особистий контакт з його розробниками.

Усі складові частини та комплекс в цілому можна побачити (а також скористатися ними), якщо з меню «*Покажчика*» чи «*Допоміжного матеріалу*» вийти у «*Перелік програм*», де зібрані посилання на всі програми, що зустрічаються в ДКН* (додаток В).

Існує ще один спосіб переходу до потрібних програм. Переглядаючи матеріал курсу, можна побачити приклади написання програм, що використовують можливості

---

* Дистанційний курс з дисципліни “Методи оптимізації” знаходиться за адресою *http://www.sumdu.edu.ua/cources/mo/*

розробленого комплексу. У тексті опису програм, що реалізують методи оптимізації, може бути посилання «*Програма*», кнопка з написом «*Скачати*» або «*Текст програми*».

Щоб отримати потрібну програму, натисніть на будь-якому посиланню ліву клавішу миші, і у Вас на екрані повинно з'явитися вікно повідомлень, подібне до того, що зображено на рисунках 2.1 та 2.2.

Залежно від того, що висвітилось на екрані, Ви вибираєте опцію «*Save it to disk*» або «*Save File…*», тому що під таким посиланням звичайно ховається архів (файл із розширенням *rar* чи *zip*), який вимагає окремої програми для його розпакування* (порада: якщо Ви ще не стикалися з цим – поцікавтеся інформацією про архіватори rar, zip, arj, winrar, вона Вам буде корисна не тільки для роботи з курсом).

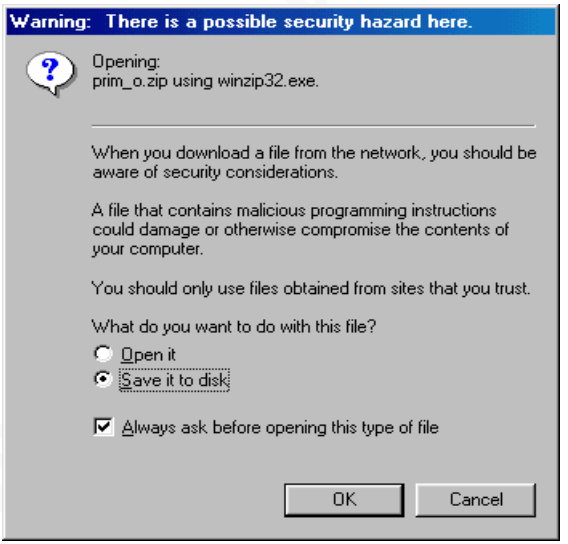


Рисунок 2.1 - Вигляд вікна повідомлень Netscape-навігатору

* У ДКН є посилання на архіватор WinRar (див. «*Покажчик*» чи «*Допоміжний матеріал*»).

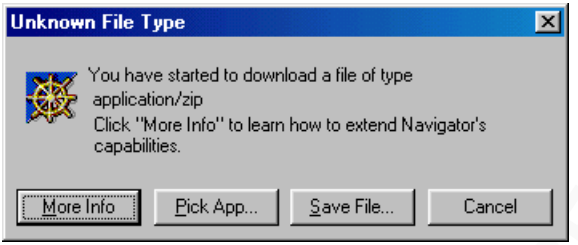


Рисунок 2.2 - Вигляд вікна повідомлень Netscape-навігатору (файл - *.zip)

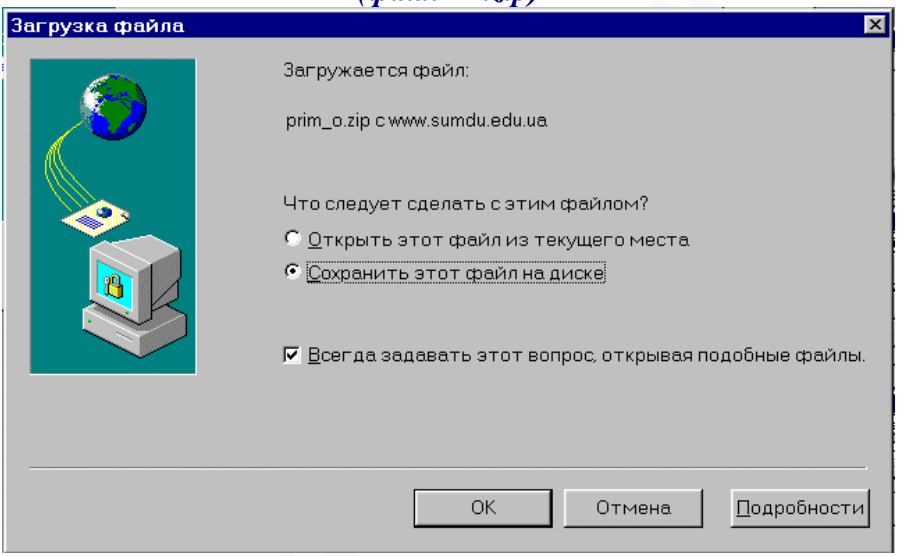
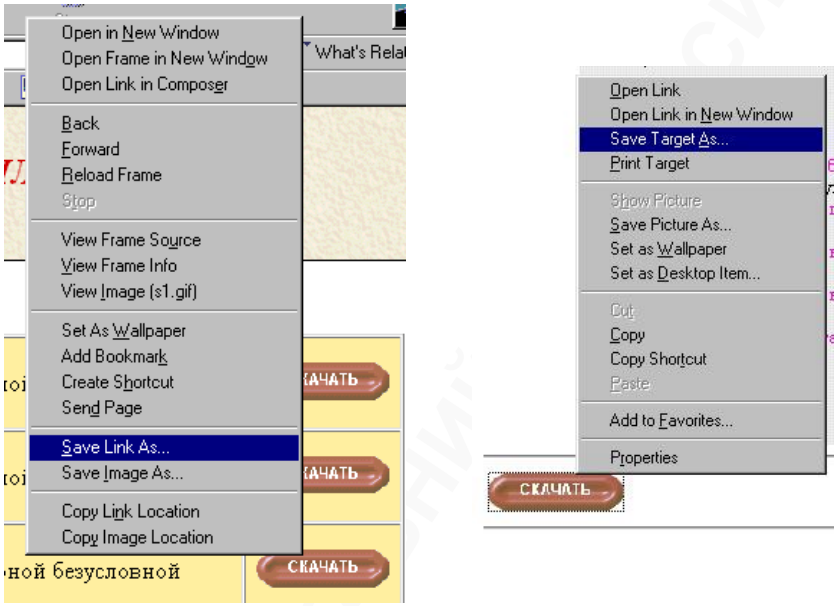


Рисунок 2.3 - Вигляд вікна повідомлень Internet Explorer

Ми пропонуємо ще один спосіб запису таких файлів на диск: клацніть правою кнопкою миші по посиланню «*Програма*» і виберіть у меню, що випадає і яке з'явилося на екрані, «*Save Link As:*» (рис. 2.4), далі Вам запропонують вибрати, куди записати цей файл під тим ім'ям, яким він названий розробником (Ви звичайно можете його змінити, але ми не радимо це робити, тому що намагалися, щоб ім'я файлу відбивало

інформацію, що зберігається у ньому. Наприклад, kompl.rar містить комплексний метод знаходження оптимуму (рис. 2.5)).



А) При роботі з Netscape *б) При роботі з Internet Explorer*

Рисунок 2.4 - Вигляд меню, що випадає при натисканні правої кнопки миші на посиланні

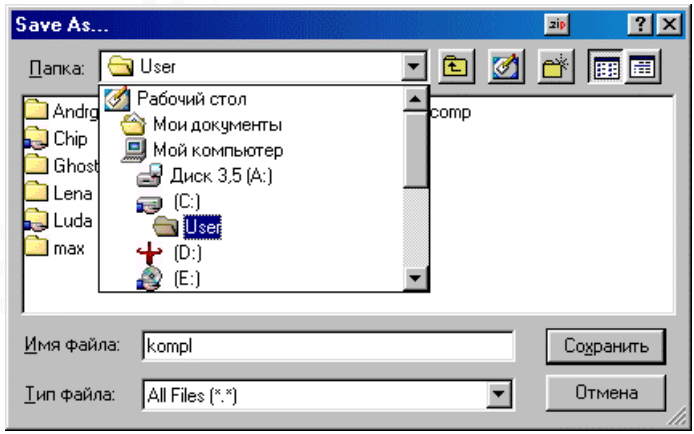


Рисунок 2.5 - Вигляд вікна запису

Щоб почати користуватися записаним файлом, його потрібно розпакувати. Усі складові курсу запаковані архіватором WinRar версії 2.50. Щоб розпакувати програму, можна скористатися одним із декількох способів. Ми пропонуємо роботу виконати у такій послідовності:

1 Запустити програму *WinRar* (Якщо її немає на комп'ютері, то можна її встановити, у курсі є відповідне посилання на).

2 Знайти потрібний файл і натиснути кнопку «*Extract to*» (рис. 2.6).

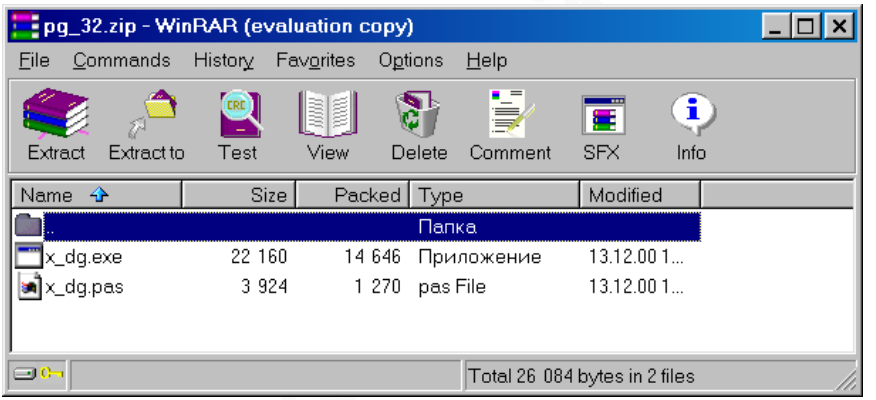


Рисунок 2.6 – Вікно архіватора

3 У вікні, яке з'явиться на екрані (рис.2.7), вибрати шлях для запису розпакованих файлів та натиснути кнопку «*Ок*». На екрані з'явиться вікно (рис.2.8), яке відображатиме процес розпакування архіву. Цей процес можна призупинити, якщо в цьому виникне необхідність. Після того як це вікно зникне з екрана, в тій директорії, яку вибрали для розпакування, повинні з'явитися розпаковані файли. Тепер вони готові до подальшої роботи.

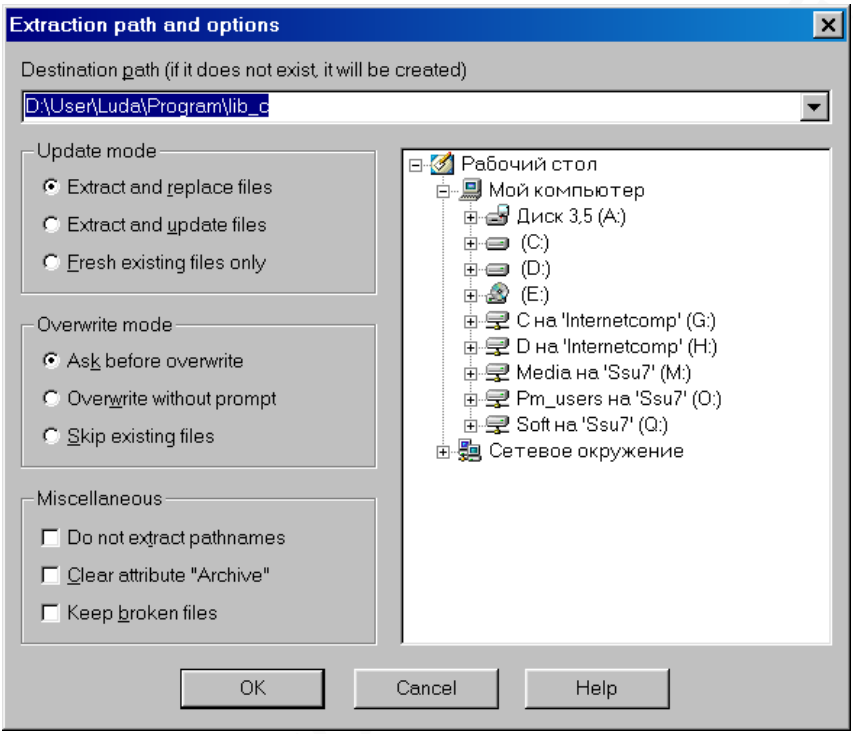


Рисунок 2.7 – Вікно для вибору шляху розпакування архівних файлів

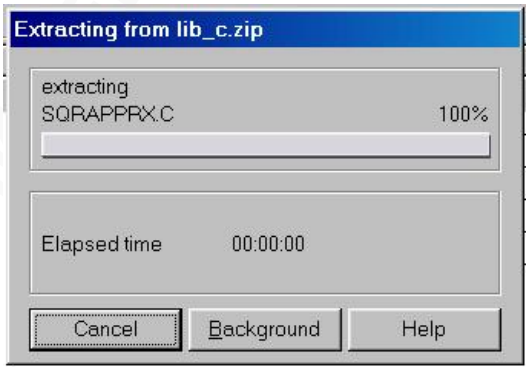


Рисунок 2.8 – Вікно, що відображає процес розпакування

2.3 РЕКОМЕНДАЦІЇ ЩОДО КОРИСТУВАННЯ МОЖЛИВОСТЯМИ ПРОГРАМНОГО КОМПЛЕКСУ

Використання бібліотеки програм, що реалізовані на Паскалі

Усі методи реалізовані на алгоритмічній мові Паскаль. Для компіляції функцій бібліотеки були використані компілятори Turbo Pascal 7.0 та Borland Pascal 7.0*.

При використанні оболонки Turbo Pascal 7.0 треба виконати:

1 Після запуску ***turbo.exe*** у головному меню вибрати пункт *Options\Directories…* і в рядку введення *Unit Directories* додати шлях до модульних файлів після "**;**", наприклад, *D:\BP\UNITS\MO* (рис. 2.9).

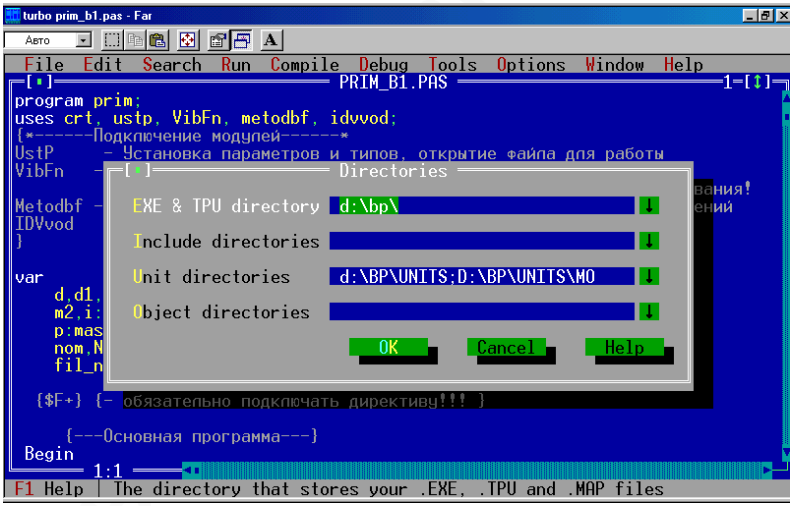


Рисунок 2.9 – Вигляд вікна після виклику Options\Directories

* Рекомендуємо користуватись компілятором Borland Pascal 7.0.

2 У розділі *USES* необхідно підключити один чи декілька модулів методів оптимізації*, наприклад,**

```
uses crt, ustp, metodo, idvvod;
```

3 У тексті програми викликати функцію з підключених модулів, що реалізує необхідний метод пошуку, наприклад,

```
VvodIsx_GS(a,b,e,fil);
GoldSech(q,a,b,e,fil);
```

4 Запускаємо написану програму на виконання за допомогою сполучення клавіш ***"Ctrl+F9"*** чи по ***F10*** виходимо в меню, вибираємо ***Run*** і потім у підменю, яке з'явиться на екрані, ще раз ***Run***.

При використанні компілятора Borland Pascal 7.0 послідовність дій така сама, як і в попередньому випадку з тією різницею, що запускаємо на початку файл ***bp.exe***.

Приклад

Необхідно знайти мінімум функції $f = x^4-14x^3+60x^2-70x$. Проведемо обчислення методом «золотого» перетину на інтервалі (1;2) з похибкою 1e-5.

Програма буде мати вигляд:

```
program prim;
uses crt, ustp, metodo, idvvod;
{*------Підключення модулів ------*
Ust - Установлення параметрів і типів, відкриття файлу
для роботи
Metod - Методи знаходження оптимуму функції 1-змінної
без обмежень
IDVvod - Процедури введення вихідних даних для методів
оптимізації
}
```

* Залежно від вибраного методу пошуку - metodo, metodp, metodb, metodg, metodu.

** Так виглядає рядок підключення модулів.

```
var
a,b,e:real;
nom:integer;
fil:text;
{$F+} {- обов'язково підключати директиву!!! }

{--Описуємо функцію, оптимум якої необхідно знайти--}
function q(x:real):real;
begin
q:=x*x*x*x-14*x*x*x+60*x*x-70*x;
end;

{---Основна програма ---}
Begin
Clrscr;
{--Виводимо рядок для зручності користувача, тобто
уточнюємо метод--}
writeln('Знаходження оптимуму за методом "золотого"
перетину');
{--Підключаємо процедуру одержання імені файлу для
введення результатів --}
GetInputName(fil);
{--Підключаємо процедуру введення вихідних даних --}
VvodIsx_GS(a,b,e,fil);
{--Підключаємо процедуру обчислення оптимуму --}
GoldSech(q,a,b,e,fil);
{--Закриваємо файл результатів--}
close(fil);
end.
```

Використання бібліотеки програм, що реалізовані на Сі

Усі методи реалізовані на алгоритмічній мові С. Для компіляції функцій бібліотеки були використані компілятори Borland C++ 3.1 і Watcom C/C++ 11.0.

Для використання бібліотеки до програми необхідно підключити один із заголовних файлів (*.h), що містять

прототипи функцій. Для використання функцій одновимірної оптимізації потрібна директива компіляції `#include "monod.h";` для функцій багатовимірної оптимізації – директива `#include "multid.h";` для функцій багатовимірної оптимізації з обмеженнями – директива `#include "multidr.h".`

При компіляції програми необхідно зазначити шляхи до заголовних файлів і бібліотек та бібліотеки, які потрібно підключати при компонуванні. У результаті компіляції і компонування одержуємо файл із розширенням *.exe, що і використовується для знаходження оптимуму.

При використанні компілятора Borland, що запускається з командного рядка, треба використовувати таку команду:

bcc –I<includepath> -L<librarypath> <c_program> <library> ,

де *includepath* – шлях до заголовних файлів; *librarypath* – шлях до бібліотек; *c_program* – програма на мові С; *library* – бібліотека, що підключається.

При використанні оболонки Borland треба виконати:

1 У головному меню вибрати пункт *Options\Directories…* і в рядку введення *Include Directories* додати шлях до заголовних файлів бібліотеки після символа '**;**'.
2 Створити проект, для чого в головному меню вибрати пункт *Project\Open project...* і в рядку *Open Project File* ввести назву проекту. В результаті з'явиться вікно *Project*.
3 У головному меню вибрати пункт *Project\Add item...* і в діалоговому вікні, що з'явилося, зазначити файл, що містить програму, яка використовує бібліотеку оптимізації.
4 За допомогою пункту *Project\Add item…* зазначити всі необхідні для програми бібліотеки.

При використанні компілятора Watcom треба використовувати таку команду:

wcl386 –i=<includepath> -"libp <librarypath>" <c_program> <library>

де *includepath* – шлях до заголовних файлів; *librarypath* – шлях до бібліотек; *c_program* – програма на мові С; *library* – бібліотека, що підключається.

Приклад

Необхідно знайти мінімум функції

$$f(x)= 2 \cdot x^2 + \frac{16}{x}.$$

Проведемо обчислення методом «золотого» перетину на інтервалі (1;2) з похибкою 0.001.

Програма буде мати вигляд:

```
/* Програма, що використовує функцію обчислення
оптимуму методом "золотого" перетину */
#include <stdio.h>
#include "monod.h" // підключення бібліотеки
double w(double);

void main()
{
printf("x* = %20.11f\n", GoldCut( w, 1, 2, 1e-3) );
}

// обчислення функції
double w(double x)
{
return 2*x*x + 16/x;
}
```

У даному випадку найменуванням функції є **w** типу ***double f(double)***. Цей параметр ми і передаємо в бібліотечний модуль.

Якщо програма має назву ***main.c***, заголовний файл знаходиться в каталозі ***include*** , а бібліотека ***monod.lib*** - у каталозі ***library***, то відкомпілювати програму можна командою

```
bcc -Iinclude\ -Llibrary\ main.c monod.lib
```

У результаті компіляції одержимо файл main.exe. Надалі можна використовувати цей файл для знаходження оптимуму даної функції вибраним методом.

ЧАСТИНА II ПРОГРАМНИЙ КОМПЛЕКС "МЕТ_ОРТ"

ТЕМА 3 ТЕОРЕТИЧНІ АСПЕКТИ

Опис методів оптимізації в повному обсязі наведено на сторінках ДКН та в літературі [4-20]. У даному посібнику подані лише стислі відомості.

3.1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ОПТИМІЗАЦІЮ

Незважаючи на те, що прикладні задачі належать до цілком різних галузей, вони мають загальну форму. Всі ці задачі можна класифікувати як задачі мінімізації дійснозначної функції *f(x)* N-вимірного векторного аргументу $x=(x_1, x_2,..., x_n)$, компоненти якого задовольняють систему рівнянь $h_k(x)=0$, набір нерівностей $g_i(x)\geq 0$, а також обмежені зверху і знизу, тобто $x_i^{(u)} \geq x_i \geq x_i^{(l)}$. У подальшому викладі функцію *f(x)* будемо називати *цільовою функцією,* рівняння $h_k(x)=0$ - *обмеженнями у вигляді рівностей,* а нерівності $g_i(x)\geq 0$ - *обмеженнями у вигляді нерівностей.* При цьому передбачається, що усі функції, які фігурують у задачі, є дійснозначними, а число обмежень - скінченна величина.

Задача загального вигляду: мінімізувати *f(x)* при обмеженнях

$$h_k(x)=0, \qquad k=1, ..., K,$$
$$g_j(x) \geq 0 \qquad j=1, ..., J,$$
$$x_i^{(u)} \geq x_i \geq x_i^{(l)}, \quad i=1, ..., N$$

називається *задачею оптимізації з обмеженнями* або *задачею умовної оптимізації.* Задачі, у яких немає обмежень, тобто

$$J=K=0; \qquad x_i^{(u)=} - x_i^{(l)} = \infty \; ; \qquad i=1, ..., N,$$

називаються *оптимізаційними задачами без обмежень* або *задачами безумовної оптимізації.*

Класифікація задач

Задачі оптимізації можна класифікувати відповідно до вигляду функцій f, h_k, g_i і розмірності вектора x. Задачі без обмежень, у яких x являє собою одновимірний вектор, називаються задачами з однією змінною і складають найпростіший, але водночас дуже важливий підклас оптимізаційних задач. Завдання умовної оптимізації, у яких функції h_k, і g_i є лінійними, мають назву *задач із лінійними обмеженнями.*

У таких задачах цільові функції можуть бути або лінійними, або нелінійними. Задачі, що мають тільки лінійні функції вектора неперервних змінних х, називаються *задачами лінійного програмування;* у *задачах цілочислового програмування* компоненти вектора x повинні набувати тільки цілі значення.

Задачі з нелінійною цільовою функцією і лінійними обмеженнями іноді називають *задачами нелінійного програмування з лінійними обмеженнями.* Оптимізаційні задачі такого роду можна класифікувати на основі структурних особливостей нелінійних цільових функцій. Якщо $f(x)$ - квадратична функція, то ми маємо справу з *задачами квадратичного програмування;* якщо $f(x)$ є відношення лінійних функцій, то відповідні задачі мають назву *задачі дробово-лінійного програмування* і т.д. Розподіл оптимізаційних задач на ці класи становить значний інтерес, оскільки специфічні особливості тих або інших задач відіграють важливу роль при розробленні методів їх розв′язання.

Необхідні і достатні умови оптимальності

Функція $f(x)$ має локальний мінімум у точці x_0, якщо існує деяка позитивна величина δ, така, що якщо $|x - x_0| < \delta$, то $f(x) >= f(x_0)$, якщо існує окіл точки x_0, такий, що для всіх значень x у цьому околі $f(x) > f(x_0)$. Функція $f(x)$ має глобальний мінімум у точці x^*, якщо для всіх x справедлива нерівність $f(x) >= f(x^*)$.

На рис.3.1 показано графічне зображення функції f(x), що має локальний мінімум у точці x_0 і глобальний мінімум у точці x^*.

$$f`'(x)=0. \qquad (3.1)$$

Рівняння (3.1) є *необхідною* умовою.

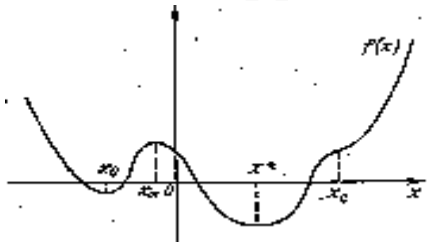


Рисунок 3.1 - Графічне зображення функції

$$f(x_0 + h) - f(x_0) = hf'(x_0) + (h^2/2!) f''(x_0) + \ldots \qquad (3.2)$$

Якщо в точці x_0 досягається мінімум, то ліва частина (3.2) буде невід`ємною для будь-якого досить малого h ($|h| < \delta$). Отже, перша похідна $f'(x_0)$ повинна дорівнювати нулю, і це і є достатня умова (див. рівняння 3.1).

Оскільки в наступному члені (3.2) завжди $h^2 > 0$, то, якщо

$$f''(x^*) > 0, \qquad (3.3)$$

в точці x_0 досягається мінімум. Якщо $f'(x_m) = 0$ і $f''(x_m) < 0$, то з аналогічних міркувань у точці x_m досягається максимум. Для визначення різниці між локальним і глобальним мінімумами необхідно порівняти значення функцій $f(x_0)$ і $f(x^*)$.

Необхідною умовою мінімуму функції n дійсних змінних в точці x_0 є рівняння

$$\nabla f(x_0) = 0, \qquad (3.4)$$

тобто

$$\partial f(x_0) / \partial x_i = 0, \ (i=1,\ldots n). \qquad (3.5)$$

Тоді знак різниці $f(x_0 + h) – f(x_0)$ визначається членом

$$1/2\ h^T G^*(x_0)h. \qquad (3.6)$$

Якщо матриця $G(x_0)$ додатно визначена, то це член додатний для всіх h.

Отже, *необхідними і достатніми умовами мінімуму* є

$$\nabla f(x_0) = 0,\ G(x_0) \text{ додатно визначена.} \qquad (3.7)$$

Необхідними і достатніми умовами максимуму є

$$\nabla f(x_m) = 0,\ G(x_m) \text{ невід`ємно визначена.} \qquad (3.8)$$

3.2 МЕТОДИ ОПТИМІЗАЦІЇ

У багатьох випадках задача пошуку оптимальних розв'язків може бути формалізована і розв'язана точно або приблизно відомими методами. Методи оптимізації поділяються на:

- *методи оптимізації для функції однієї змінної;*
- *методи оптимізації для функції n змінних без обмежень;*
- *методи оптимізації для функції n змінних з обмеженнями.*

Теоретичні основи оптимізаційних методів подані стисло, без доведень теорем та детальних описів алгоритмів. Посібник має на меті дати пояснення до процедур програмного комплексу Met_Opt та опис параметрів, що потрібні для правильної роботи програми.

---

* Матриця Гессе (гессіан) функції $f(x)$ позначається як G(x) і є симетричною матрицею $n*n$ елементів вигляду $G_{ij} = 2\ \partial^2 f / \partial x_i \partial x_j$.

3.2.1 МЕТОДИ ОПТИМІЗАЦІЇ ДЛЯ ФУНКЦІЇ ОДНІЄЇ ЗМІННОЇ

Методи виключення інтервалів

Методи пошуку, що дозволяють визначити оптимум функції однієї змінної шляхом зменшення інтервалу пошуку, називаються *методами виключення інтервалів*.

Усі методи одновимірної оптимізації засновані на припущенні, що досліджувана цільова функція в допустимій області принаймні має властивість унімодальності, тому що для унімодальної функції W(x) порівняння значень W(t) у двох різних точках інтервалу пошуку дозволяє визначити, в якому із заданих двома зазначеними точками підінтервалів точки оптимуму відсутні.

Правило виключення інтервалів. Нехай W(x) унімодальна на відрізку $[a,b]$, а її мінімум досягається в точці x^*. Розглянемо x_1 і x_2, розміщені $a<x_1<x_2<b$.

- Якщо $W(x_1)>W(x_2)$, то точка мінімуму W(x) не лежить в інтервалі (a,x_1), тобто $x^*\in(x_1,b)$.
- Якщо $W(x_1)<W(x_2)$, то точка мінімуму W(x) не лежить в інтервалі (x_2,b), тобто $x^*\in(a,x_2)$.

Це правило дозволяє реалізувати процедуру пошуку шляхом послідовного виключення частин вихідного обмеженого інтервалу. Пошук завершується тоді, коли підінтервал, що залишився, зменшується до досить малих розмірів.

Головна позитивна властивість пошукових методів полягає в тому, що вони засновані на обчисленні тільки значень функції і, отже, не вимагають виконання умови диференційованості і запису в аналітичному вигляді. Останнє особливо цінне при імітаційному моделюванні.

Методи виключення інтервалів:

- метод ділення навпіл [11];
- метод “золотого” перетину [10, 11];
- метод Фібоначчі [10, 11].

При застосуванні методу “золотого” перетину інтервал пошуку ділиться на дві частини так, щоб відношення довжини великого відрізка до довжини всього інтервалу дорівнювало відношенню $\frac{z_1}{z} = \frac{z_2}{z_1}$. З огляду на те, що $z_1+z_2=z$, маємо

$$z_1^2 = z\, z_2 = (z_1+z_2)z_2 = z_1 z_2 + z_2^2, \qquad z_1 z_2 + z_2^2 - z_1^2 = 0,$$

звідки $\frac{z_2}{z_1} = 0{,}618$.

Таким чином, застосування методів виключення інтервалів накладає єдине обмеження на досліджувану цільову функцію - унімодальність. Отже, розглянуті методи можна використовувати для аналізу як неперервних, так і розривних і дискретних функцій. Логічна структура пошуку заснована на простому порівнянні значень функції в двох пробних точках.

Методи поліноміальної апроксимації

Сутність методу. Відповідно до теореми Вейєрштраса про апроксимацію, неперервну функцію в деякому інтервалі можна апроксимувати поліномом досить високого порядку. Отже, якщо функція унімодальна і можна знайти поліном, що досить точно її апроксимує, то координати точки оптимуму функції можна оцінити шляхом обчислення координати точки оптимуму полінома.

Використовується декілька значень функції у визначених точках для апроксимації функції звичайним поліномом принаймні в невеликій області значень. Потім положення мінімуму функції апроксимується положенням мінімуму полінома, оскільки останній простіше обчислити.

Найпростіший випадок заснований на тому факті, що функція, що набуває мінімального значення у внутрішній точці інтервалу, повинна бути принаймні квадратичною.

Якщо цільова функція W(x) у точках x_1, x_2, x_3 набуває відповідних значень W_1, W_2, W_3, то можна визначити коефіцієнти a_o, a_1, a_2 таким чином, що значення квадратичної функції

$$q(x) = a_o + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$$

збігаються зі значенням W(x) у трьох зазначених точках.

Властивості апроксимації використовуються при квадратичній інтерполяції, яку називають також методом Пауелла.

Методи з використанням похідних

Доцільно припустити, що ефективність пошукових процедур істотно підвищиться, якщо на додаток до умови безперервності ввести вимогу диференційованості цільової функції. Необхідна умова існування оптимуму цільової функції в точці x*

$$W'(x^*) = d/dx \mid_{x=x^*} = 0.$$

У тому випадку, якщо цільова функція має члени, що містять *x* у третій і більш високих степенях, то одержання аналітичного розв'язання рівняння W'(x) складне. В цих випадках доцільно використовувати чисельні методи знаходження коренів нелінійних рівнянь:

- метод середньої точки [11];
- метод хорд [11];
- метод дотичних (Ньютона-Раффсона) [11].

Сутність методу дотичних. Орієнтований на знаходження кореня рівняння *W'(x)* в інтервалі *[a,b]*, у якому є дві точки *N* і *P*, в яких знаки похідних різні. Робота алгоритму починається з точки x_0, що являє собою початкове наближення кореня рівняння *W'(x)=0*. Далі будується лінійна апроксимація функції *W'(x)* у точці x_1, і точка, у якій апроксимуюча лінійна функція перетворюється на нуль, приймається як наступне наближення. Якщо точка x_k прийнята як поточне наближення до

оптимальної точки, то лінійна функція, що апроксимує функцію $W'(x)$ у точці x_k, записується у вигляді

$$W'(x,x_k) = W'(x_k) + W''(x_k)(x-x_k).$$

Прирівнявши праву частину рівняння до нуля, одержимо наступне наближення до точки, що шукаємо.

Алгоритм методу

Крок 1 Наступне наближення до стаціонарної точки x* визначається за формулою $x_{k+1} = x_k - [W'(x_k)/W''(x_k)]$.

Крок 2 Обчислити $W'(x_{k+1}), W''(x_{k+1})$.

Крок 3 Якщо $|W'(x_{k+1})|<e$, то слід закінчити пошук. У противному разі необхідно повернутися до кроку 1.

Із алгоритму випливає, що цільова функція W(x) повинна бути двічі диференційована.

3.2.2 МЕТОДИ ОПТИМІЗАЦІЇ ДЛЯ ФУНКЦІЇ N ЗМІННИХ

Методи прямого пошуку

Одними із методів знаходження мінімуму функції n змінних є методи прямого пошуку. Методи прямого пошуку – це методи, які використовують тільки значення функції.

До методів прямого пошуку належать: метод покоординатного спуску, метод Хука-Дживса, метод Нелдера-Міда [10, 11]. Найпростішим із них є *метод покоординатного спуску.*

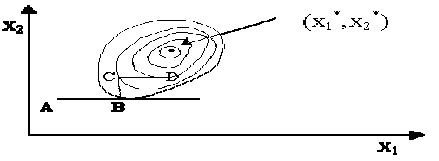


Рисунок 3.2 - Метод прямого пошуку

З точки А зробимо пошук мінімуму уздовж напрямку осі x_1 і, таким чином, знаходимо точку В, у якої дотична до лінії постійного рівня рівнобіжна осі x_1. Потім, проводячи пошук із

точки B в напрямку осі x_2, одержуємо точку C, проводячи пошук паралельно осі x_2, одержуємо точку D, і так далі. Таким чином, ми приходимо до оптимальної точки. Цю ідею можна застосувати для функції n змінних.

Теоретично даний метод ефективний у випадку єдиного мінімуму функції. Але на практиці він виявляється занадто повільним. Тому були розроблені більш складні методи, що використовують більше інформації на підставі вже отриманих значень функції.

Градієнтні методи

При використанні навіть найефективніших прямих методів для одержання розв'язку іноді потрібна дуже велика кількість обчислень значень функції. Розглянуті методи засновані на використанні градієнта цільової функції*. Ці методи мають ітераційний характер. Процедура реалізується відповідно до формули

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} S(x^{(k)}), \qquad (3.9)$$

---

* *Зауваження*. Напрямок градієнта перпендикулярний у будь-якій точці лінії постійного рівня, оскільки уздовж цієї лінії функція не змінна. Таким чином, якщо $(d_1,d_2,..,d_n)$ - малий крок уздовж лінії рівня, то

$f(x_1+d_1,x_2+d_2,...,x_n+d_n) = f(x_1,x_2,..,x_n)$

і, отже,

$$df = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} d_j = [\nabla f(x)]^T d = 0, \quad (3.10)$$

(див. рис.3.3).

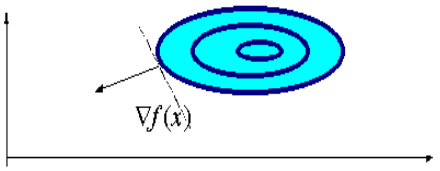


Рисунок 3.3 - Метод градієнтного спуску

де $x^{(k)}$ – поточне наближення до розв'язку x^*;
$\alpha^{(k)}$ – параметр, що характеризує довжину кроку;
$S(x^{(k)}) = S^{(k)}$ – напрямок пошуку в N-вимірному просторі керованих змінних.

Спосіб вибору $S(x)$ і α залежить від вибору методу.

3.2.3 МЕТОДИ УМОВНОЇ ОПТИМІЗАЦІЇ. НЕЛІНІЙНЕ ПРОГРАМУВАННЯ

Методи прямого пошуку

У методах прямого пошуку обмеження враховуються в явному вигляді. Необхідність розробки цих методів пов'язана з тим, що в інженерних додатках часто доводиться стикатися з випадками, коли цільові функції не задані в явному вигляді. Ці методи будуються на інтуїтивних міркуваннях, не підкріплені суворою теорією і, отже, не гарантується їх збіжність. Незважаючи на це, внаслідок своєї логічної простоти ці методи легко реалізуються.

Перед безпосереднім застосуванням методів прямого пошуку необхідно провести ряд заходів щодо підготовки задачі до розв'язання, а саме:

- виключити обмеження у вигляді рівнянь;
- визначити початкову допустиму точку.

Після проведення процедури підготовки задачі до розв'язання слід вибрати один із методів умовної оптимізації:

- модифікований метод Хука-Дживса [10];
- метод комплексів [10, 11];
- метод випадкового пошуку [10, 11, 18].

При використанні вищезгаданих методів потрібно перевірити, чи кожна точка, отримана в процесі пошуку, належить до області обмежень. Якщо кожна, то цільова функція обчислюється звичайним шляхом. Якщо ні, то цільовій функції

надається дуже велике значення. Таким чином, пошук буде здійснюватися знову в допустимій області у напрямку до мінімальної точки всередині цієї області. При реалізації цієї процедури може виникнути проблема – не завжди можна отримати правильні результати. За допомогою даного методу неможливо рухатися уздовж межі, де міститься розв'язок. Загальне завдання оптимізації за наявності обмежень дуже складне і для одержання практичного методу розв'язання вимагаються більш точні процедури, ніж наведена вище.

Методи штрафних функцій

За допомогою штрафних функцій

$$P(x,R) = W(x) + \Omega(R,g(x),h(x)),$$

де R - набір штрафних параметрів;

Ω - штраф,

вихідна задача умовної оптимізації перетвориться в послідовність задач безумовної оптимізації. Штраф Ω визначається так, щоб допустимі точки задачі мали перевагу перед недопустимими відносно безумовної оптимізації штрафної функції. Тут штраф немовби створює уздовж межі допустимої області бар'єр із нескінченно великих значень функції P.

До штрафу ставляться такі вимоги:

- розв'язки підзадач повинні прямувати до розв'язку вихідної задачі нелінійного програмування $\lim_{t \to T < \infty} x^k = x^*$;
- складність оптимізації P(x, R) повинна бути такого самого порядку, що і W(x).

Методи штрафних функцій класифікуються у відповідності до способів обліку обмежень - нерівностей g(x), тому що обмеження - рівняння h(x) враховуються у всіх методах однаково за допомогою квадратичного штрафу

$$\Omega = R\{h(x)\}^2.$$

При розгляданні будь-якій штрафній функції потрібно вибрати початкове значення R і змінити його після розв'язування кожної підзадачі безумовної оптимізації для того, щоб забезпечити збіжність. Для квадратичного штрафу, що враховує обмеження-рівняння, є доцільним починати з $R=0$, а потім послідовно збільшувати R на деяке ΔR або використовувати зростаючі степені якогось числа, наприклад 10. У результаті одержувані точки будуть усе точніше й точніше задовольняти обмеження.

ТЕМА 4 БІБЛІОТЕКА ПРОГРАМ, ЩО РЕАЛІЗОВАНІ НА ПАСКАЛІ

Комп'ютерна реалізація методів оптимізації на алгоритмічній мові Паскаль оформлена у вигляді трьох блоків: безпосередньо процедури, що реалізують методи, допоміжні процедури та приклади із використання вищезазначених процедур.

Модулі методів оптимізації (*.tpu)

Модулі одновимірної оптимізації. Цей підрозділ складається з таких модулів:

- модуль методів одновимірної оптимізації із використанням тільки значення функції *metodo*;
- модуль методів одновимірної оптимізації із використанням тільки значення функції *metodof*[*];
- модуль методів одновимірної оптимізації із використанням похідних *metodp*;
- модуль методів одновимірної оптимізації із використанням похідних *metodpf*.

[*] Буква "**f**" у кінці назви модуля означає, що в даному модулі використовуються процедури із модуля функцій.

Модулі багатовимірної оптимізації. Цей підрозділ складається з таких модулів:

- модуль методів багатовимірної безумовної оптимізації *metodb*;
- модуль методів багатовимірної безумовної оптимізації *metodbf*;
- модуль методів багатовимірної оптимізації із використанням властивостей градієнта *metodg*;
- модуль методів багатовимірної оптимізації з використанням властивостей градієнта *metodgf*;
- модуль методів багатовимірної умовної оптимізації *metodu*;
- модуль методів багатовимірної умовної оптимізації *metoduf*.

Модулі допоміжних процедур (*.tpu)

- модуль встановлення параметрів *ustp;*
- модуль процедур вводу вхідних даних для методів оптимізації *idvvod*;
- модуль функцій однієї змінної *vibfo*;
- модуль функцій декількох змінних *vibfn.*

Приклади використання процедур (*.pas, *.exe)

- приклади використання процедур одновимірної оптимізації *prim_oN*[*], *prim_pN*[†];
- приклади використання процедур багатовимірної безумовної оптимізації *prim_bN*[‡];

---

[*] N – може набувати значення від 1 до 8 залежно від методу та вибору функції. Тобто під непарними номерами мається на увазі, що функцію користувач записує безпосередньо в програмі і при обчисленні може її змінити тільки тоді, коли змінить код програми (якщо це *.exe файл – то це практично неможливо для звичайного користувача), а під парними - функцію користувач вибирає під час роботи програми із запропонованого списку. На кожний метод припадає по 2 програми.

[†] N Є (1; 6).

[‡] N Є (1; 6).

- приклади використання процедур багатовимірної безумовної оптимізації, що базуються на властивостях градієнта, *prim_gN*[*];
- приклади використання процедур багатовимірної умовної оптимізації *prim_uN*[†].

4.1 МОДУЛІ МЕТОДІВ ОПТИМІЗАЦІЇ (.TPU)*

У розробленому нами комплексі комп'ютерних програм, що реалізують методи оптимізації, є окремий підрозділ присвячений одновимірній оптимізації. Він складається з двох бібліотек (тобто з їх підрозділів, див. рис.4.1)

1 Бібліотека програм, що реалізовані на Паскалі
Модулі методів оптимізації (*.tpu)
Модулі одновимірної оптимізації.
Цей підрозділ складається з таких модулів:
- *модуль методів одновимірної оптимізації metodo;*
- *модуль методів одновимірної оптимізації metodof;*
- *модуль методів одновимірної оптимізації metodp;*
- *модуль методів одновимірної оптимізації metodpf.*

2 Бібліотека програм, що реалізовані на Сі
Бібліотека методів одновимірної оптимізації monod

Рисунок 4.1 – Структура підрозділу комплексу, присвяченого одновимірній оптимізації

Опис модуля методів одновимірної оптимізації metodo

Використовує при роботі параметри, встановлені в модулі *ustp*.

---

[*] N Є (1; 6).
[†] N Є (1; 6).

Призначення: даний модуль застосовується для зручності використання запрограмованих методів оптимізації функцій однієї змінної.

Даний модуль містить процедури, які реалізують методи оптимізації для функцій однієї змінної, що використовують тільки значення функції:

1) процедура знаходження оптимуму методом половинного ділення *PolDelen(f, a, b, e, fil);*

2) процедура знаходження оптимуму методом "золотого" перетину *GoldSech(ff, a, b, eps, fil);*

3) процедура знаходження оптимуму методом Фібоначчі *Fibonach(z, f, n, a, b, e, fil);*

4) процедура знаходження оптимуму методом квадратичної інтерполяції *KvadratInterp(z, a, h, e, fil).*

Опис параметрів процедур модуля

1 Процедура знаходження оптимуму методом половинного ділення

procedure PolDelen(f:funo; a,b,e:real; var fil:text);

де f - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

a,b - межі інтервалу пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, у який будуть занесені результати пошуку.

2 Процедура знаходження оптимуму методом "золотого" перетину

procedure GoldSech(ff:funo; a,b,eps:real; var fil:text);

де ff - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

a,b - межі інтервалу пошуку оптимуму;

eps - точність (похибка) пошуку;

fil - змінна імені файлу, у який будуть занесені результати пошуку.

3 Процедура знаходження оптимуму методом Фібоначчі

procedure Fibonach(z:funo; f:mas; n:integer; a,b,e:real; var fil:text);

де z - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

f - числа Фібоначчі;

n - кількість розбивок;

a,b - межі інтервалу пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, у який будуть занесені результати пошуку.

4 Процедура знаходження оптимуму методом квадратичної інтерполяції

procedure KvadratInterp(z:funo; a,h,e:real; var fil:text);

де z - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

a,b - межі інтервалу пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

Опис модуля методів одновимірної оптимізації metodp

Використовує при роботі параметри, встановлені в модулі ustp.

Призначення: даний модуль застосовується для зручності використання запрограмованих методів оптимізації функцій однієї змінної.

Даний модуль містить процедури, що реалізують методи оптимізації для функцій однієї змінної, що використовують значення похідних функції:

1) процедура знаходження оптимуму методом хорд *Hords(f, d, a, b, e, fil);*

2) процедура знаходження оптимуму методом Ньютона-Раффсона *NewtonRaffson(f,df,ddf, x, e, fil);*

3) процедура знаходження оптимуму методом середньої точки *MiddlePoint(f,df, a, b, e, fil).*

Опис параметрів процедур модуля

1 Процедура знаходження оптимуму методом хорд

procedure Hords(f,df:funo; a,b,e:real; var fil:text);

де f - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

df - найменування похідної функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

a,b - межі інтервалу пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

2 Процедура знаходження оптимуму методом Ньютона-Раффсона

procedure NewtonRaffson(f,df,ddf:funo; x,e:real; var fil:text);

де f - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

df, ddf- найменування 1 і 2 похідних функції, оптимум якої необхідно знайти (тип функцій повинен відповідати типу, встановленому в модулі ustp);

x - початкова точка пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

3 Процедура знаходження оптимуму методом середньої точки

procedure MiddlePoint(f,df:funo; a,b,e:real; var fil:text);

де f - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

df - найменування похідної функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

a,b - межі інтервалу пошуку оптимуму;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

У розробленому нами комплексі комп'ютерних програм, що реалізують методи оптимізації, є окремий підрозділ, присвячений багатовимірній оптимізації. Він складається із двох бібліотек (тобто з їх підрозділів, див. рис.4.2)

1 Бібліотека програм, що реалізовані на Паскалі
Модулі методів оптимізації (*.tpu)
Модулі багатовимірної оптимізації без обмежень.
Цей підрозділ складається з таких модулів:
- *модуль методів багатовимірної оптимізації metodb;*
- *модуль методів багатовимірної оптимізації metodbf;*
- *модуль методів багатовимірної оптимізації metodg;*
- *модуль методів багатовимірної оптимізації metodgf.*

2 Бібліотека програм, що реалізовані на Сі
Бібліотека методів багатовимірної оптимізації multid

Рисунок 4.2 – Структура підрозділу комплексу, присвяченого багатовимірній оптимізації

Опис модуля методів багатомірної оптимізації metodb

Використовує при роботі параметри, встановлені в модулі ustp.

Призначення: даний модуль застосовується для зручності використання запрограмованих методів оптимізації функцій декількох змінних.

Даний модуль містить процедури, що реалізують методи оптимізації для функцій декількох змінних, що використовують тільки значення функції:

1) процедура знаходження оптимуму методом Хука-Дживса *XykJivs(ff, x, n, h, e, fil);*

2) процедура знаходження оптимуму методом покоординатного спуску *PkoordSp(ff, p, n, m2, d, d1, r, fil);*

3) процедура знаходження оптимуму симплексним методом Нелдера-Міда *NeldMid(z, s, n, k, al, be, ga, eps, fil).*

Опис параметрів процедур модуля

1 Процедура знаходження оптимуму методом Хука-Дживса

procedure XykJivs(ff:funn; x:mas; n:integer; h,e:real; var fil:text);

де ff - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

x - координати початкової точки пошуку оптимуму;

n - кількість змінних функції;

h - довжина кроку;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

2 Процедура знаходження оптимуму методом покоординатного спуску

procedure PkoordSp(ff:funn; p:mas; n,m2:integer; d,d1,r:real;var fil:text);

де ff - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

pp - координати початкової точки пошуку оптимуму;

n - кількість змінних функції;

m2 - максимальна кількість обчислень функції;

d - крок;

r - коефіцієнт зміни кроку;

d1 - мінімальний розмір кроку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

3 Процедура знаходження оптимуму симплексним методом Нелдера-Міда

procedure NeldMid(z:funn; s:mass; n:integer; k,al,be,ga,eps:real; var fil:text);

де z - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

s - початкове наближення;

n - кількість змінних функції;

k - довжина кроку;

al,be,ga - значення коефіцієнтів альфа, бета і гамма;

eps - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

Опис модуля методів багатовимірної оптимізації metodg

Використовує при роботі параметри, встановлені в модулі ustp.

Призначення: даний модуль застосовується для зручності використання запрограмованих методів оптимізації функцій декількох змінних.

Даний модуль містить процедури, що реалізують методи оптимізації для функцій декількох змінних, що використовують властивості градієнта:

1) процедура знаходження оптимуму методом найшвидшого спуску *NSpusk(zf, pr, x, n, h, eps, fil);*

2) процедура знаходження оптимуму методом ДФП *Metod_DFP(f, g_r, x, n, eps, fil);*

Опис параметрів процедур модуля

1 Процедура знаходження оптимуму методом найшвидшого спуску

procedure NSpusk(zf:funn; pr:funpr; x:mas; n:integer; h,eps:real; var fil:text);

де zf - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

pr - найменування функції обчислення градієнта (тип функцій повинен відповідати типу, встановленому в модулі ustp);

x - координати початкової точки пошуку оптимуму;

n - кількість змінних функції;

h - довжина кроку;

e - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

2 Процедура знаходження оптимуму методом ДФП

procedure Metod_DFP(f:funn; g_r:funpr; x:mas; n:integer;eps:real;var fil:text);

де f - найменування функції, оптимум якої необхідно знайти (тип функції повинен відповідати типу, встановленому в модулі ustp);

g_r - найменування функції обчислення градієнта (тип функцій повинен відповідати типу, встановленому в модулі ustp);

x - координати початкової точки пошуку оптимуму;

n - кількість змінних функції;

eps - точність (похибка) пошуку;

fil - змінна імені файлу, в який будуть занесені результати пошуку.

4.2 МОДУЛІ ДОПОМІЖНИХ ПРОЦЕДУР (.TPU)*

У модулі *ustp* задані загальні параметри для процедур методів оптимізації (найбільша кількість змінних задачі, типи масивів, типи функцій), розміщені процедури допоміжного характеру. А саме:

- процедура *GetInputName(fil)* - відкриття текстового файлу для запису інформації (як нового, так і для доповнення вже існуючого), де *fil* – змінна імені файлу, яке вводиться при запуску процедури в роботу;
- процедура *Proverka(x,p,fil)* – перевірка відповідності значення, де x – дійсне число, яке необхідно перевірити; p – ознака перевірки (якщо 0 – значення не вище допустимого); *fil* – змінна імені файлу (вхідний параметр);
- процедура *Sgn(c,b)* – визначення знаку різниці двох чисел *(c-b)*, де c , b – дійсні числа;
- процедура *Stp(y,k)* – піднесення числа (виразу) до степеня, де y – дійсне число, що підносять до степеня, k – степінь, до якого підносять (обов'язково ціле!).

Модуль *idvvod* містить процедури введення вхідних даних для методів оптимізації (додаток Б).

Модуль функцій однієї змінної vibfo містить процедури:

- *PrintSpFunO(nf)* – процедура, що виводить список функцій з однією змінною на екран та дозволяє вибрати функцію за номером;
- *FO(x, nf, fil)* – процедура, що дозволяє зробити безпосереднє обчислення значення вибраної функції (за номером);
- *PrFO(x, nf, fil)* - обчислення перших похідних вибраної функції;
- *ddFO(x, nf, fil)* - обчислення других похідних вибраної функції.

Модуль функцій декількох змінних vibfn містить процедури:

- *v_fun(nf, nn)* – виведення списку функцій n – змінних на екран та вибирання номера функції;
- *f(x, nf, fil)* – безпосереднє обчислення значення вибраної функції;
- *gradient(x, nf, g, fil)* – обчислення складових градієнта (часткових похідних);
- *pr(x, nf, n, g, fil)* – обчислення значення градієнта.

4.3 ПРИКЛАДИ ВИКОРИСТАННЯ ПРОЦЕДУР

Даний підрозділ комплексу складається із текстових файлів **.pas*, які підлягають подальшій обробці (компіляція та компонування), та **.exe-файлів*, що не потребують ніяких додаткових зусиль.

Приклад написання програми

Необхідно знайти мінімум декількох функцій на інтервалах (1;2), (0;-3) і (-1;1), а також порівняти отримані результати. Проведемо обчислення методом “золотого” перетину з похибкою 1e-5.

Програма буде мати вигляд:

```
program prim;
uses crt, UstP, VibFo, MetodOf, IDVvod;
{*------Підключення модулів ------*
UstP      - Установлення параметрів і типів, відкриття
файлу для роботи.
VibFo - Допоміжні процедури (виведення списку функцій на
екран, обчислення значення функцій та їхніх похідних) –
рекомендується для тестування!
MetodOf - Методи знаходження оптимуму функції 1-
змінної без обмежень.
IDVvod - Процедури введення вихідних даних для методів
оптимізації
}
var
  a,b,e:real;
  nom:integer;
  fil:text;
 {$F+} {- обов'язково підключати директиву!!! }
   {--- Основна програма ---}
Begin
Clrscr;
  writeln('Знаходження оптимуму методом "золотого"
перетину');
  {--Одержання імені файлу для введення результатів --}
  GetInputName(fil);
  {--Підключення процедури вибору функції--}
   PrintSpFun(nom);
    writeln(fil,'Функція (номер) - ',nom);
  {--Введення вихідних даних --}
  VvodIsx_GS(a,b,e,fil);
  {--Обчислення оптимуму --}
    GoldSech(FO,nom,a,b,e,fil);
  close(fil);
end.
```

Приклад написання програми

Необхідно знайти мінімум функції

$$f=(x[1]-1)^2+(x[2]-3)^2+4(x[3]+5)^2.$$

Проведемо обчислення методом найшвидшого спуску.

Програма буде мати вигляд:

```
program prim; {градієнтний, найшвидшого спуску}
uses crt,ustp,metodg,idvvod;
var x:mas;
   h,eps:real;
   n:integer;
   fil:text;
  {$F+} {- обов'язково підключати директиву!!! }
{--Обчислення значення функції --}
function zf(x:mas):real;
 var z:real;
 begin
    z:=0;
    z:=sqr(x[1]-1)+sqr(x[2]-3)+4*sqr(x[3]+5);
    zf:=z;
 end;
{--Обчислення градієнта --}
function zgr(x:mas; n:integer; var g:mas):real;
 var vr,go:real;
    i:integer;
 begin
     go:=0;
     g[1]:=2*(x[1]-1);
     g[2]:=2*(x[2]-3);
     g[3]:=8*(x[3]+5);
     for i:=1 to n do go:=go+g[i]*g[i];
     vr:=sqrt(go);
     zgr:=vr;
 end;
{--Основна програма --}
begin
clrscr;
```

```
  writeln('Знаходження оптимуму методом найшвидшого
спуску');
  {--Одержання імені файлу для занесення результатів --}
  GetInputName(fil);
  {--Введення вихідних даних --}
  n:=3;
  VvodIsx_NS(n,x,h,eps,fil);
  {--Обчислення оптимуму --}
  NSpusk(zf,zgr,x,n,h,eps,fil);
  close(fil);
end.
```

ТЕМА 5 БІБЛІОТЕКА ПРОГРАМ, ЩО РЕАЛІЗОВАНІ НА СІ

Програми на алгоритмічній мові Сі оформлено у вигляді трьох бібліотек: бібліотека методів одновимірної оптимізації, бібліотека методів багатовимірної оптимізації без обмежень та бібліотека багатовимірної оптимізації з обмеженнями.

5.1 БІБЛІОТЕКА МЕТОДІВ ОДНОВИМІРНОЇ ОПТИМІЗАЦІЇ MONOD

Бібліотека пошуку мінімуму функцій однієї змінної *monod* містить функції, що реалізують такі методи:

- метод рівномірного розбиття;
- метод випадкового пошуку;
- методи виключення інтервалів:
  - *метод половинного ділення;*
  - *метод "золотого" перетину;*
  - *метод Фібоначчі;*
- методи поліноміальної апроксимації:
  - *метод квадратичної апроксимації;*
  - *метод кубічної апроксимації;*

- методи з використанням похідних:
  - *метод Ньютона-Раффсона;*
  - *метод середньої точки;*
  - *метод січних.*

Опис функцій бібліотеки

1 Метод рівномірного розбиття

```
#include "monod.h"

double EquDiv(double f(double), double a, double
b, double e);
```

Функція **EquDiv** знаходить мінімум функції **f** на інтервалі (**a**;**b**) із похибкою **e** методом рівномірного розбиття.

Функція повертає мінімум функції **f**.

2 Метод випадкового пошуку

```
#include "monod.h"

double RandFind(double f(double), double a,
double b, long n);
```

Функція **RandFind** знаходить мінімум функції **f** на інтервалі (**a**;**b**) методом випадкового пошуку, виконавши n обчислень цільової функції.

Функція повертає мінімум функції **f**.

3 Метод половинного ділення

```
#include "monod.h"

double HalfDivision(double f(double), double a,
double b, double e);
```

Функція **HalfDivision** знаходить мінімум функції **f** на інтервалі (**a**;**b**) із похибкою **e** методом половинного ділення.

Функція повертає мінімум функції **f**.

4 Метод “золотого” перетину

```
#include "monod.h"
double GoldCut(double f(double),double a, double b, double e);
```

Функція **GoldCut** знаходить мінімум функції **f** на інтервалі (**a**;**b**) із похибкою **e** методом “золотого” перетину.

Функція повертає мінімум функції **f**.

5 Метод Фібоначчі

```
#include "monod.h"
double Fibonacci(double f(double), double a, double b,double e);
```

Функція **Fibonacci** знаходить мінімум функції **f** на інтервалі (**a**;**b**) із похибкою **e** методом Фібоначчі.

Функція повертає мінімум функції **f**.

6 Метод квадратичної апроксимації

```
#include "monod.h"
double SqrApprox(double f(double), double a, double h,double e);
```

Функція **SqrApprox** знаходить мінімум функції **f** із похибкою **e** методом квадратичної апроксимації. Пошук починається у точці **a** і проводиться із початковим кроком **h**.

Функція повертає мінімум функції **f**.

7 Метод кубічної апроксимації

```
#include "monod.h"
double CubApprox(double f(double), double df(double), double a, double h, double e);
```

Функція **CubApprox** знаходить мінімум функції **f** , що має похідну **df**, методом кубічної апроксимації. Обчислення проводиться із похибкою **e**. Пошук починається в точці **a** і проводиться із початковим кроком **h**.

Функція повертає мінімум функції **f**.

8 Метод Ньютона-Раффсона

```
#include "monod.h"

double NewtonRafson(double df(double), double
ddf(double), double x0, double e);
```

Функція **NewtonRafson** знаходить мінімум функції, що має першу похідну **df** і другу похідну **ddf**, методом Ньютона-Раффсона. Обчислення проводиться із похибкою **e**. Пошук починається у точці **x0**.

Функція повертає мінімум функції **f**.

9 Метод середньої точки

```
#include "monod.h"

double MiddlePoint(double df(double), double a,
double b, double e);
```

Функція **MiddlePoint** знаходить мінімум функції із похідною **df** на інтервалі (**a**;**b**) методом середньої точки. Обчислення проводиться із похибкою **e**.

Функція повертає мінімум функції **f**.

10 Метод січних

```
#include "monod.h"

double Chords(double df(double), double a,
double b, double e);
```

Функція **Chords** знаходить мінімум функції з похідною **df** на інтервалі (**a**;**b**) методом середньої точки. Обчислення проводиться із похибкою **e**.

Функція повертає мінімум функції **f**.

Приклад написання програми

Необхідно знайти мінімум функції $f(x) = 2 \cdot x^2 + \frac{16}{x}$. Проведемо обчислення методом "золотого" перетину на інтервалі (1;2) із похибкою 0.001.

Програма буде мати вигляд:

```
/* Програма, що використовує функцію обчислення
оптимуму методом "золотого" перетину */

#include <stdio.h>
#include "monod.h"   // підключення
бібліотеки
double w(double);

void main()
{
printf("x* = %20.11f\n", GoldCut( w, 1, 2,
1e-3) );
}

// обчислення функції
double w(double x)
{
return 2*x*x + 16/x;
}
```

У даному випадку найменуванням функції є **w** типу `double f(double)`. Цей параметр ми і передаємо в бібліотечний модуль.

Якщо програма має назву `main.c`, заголовний файл знаходиться в каталозі `include\`, а бібліотека `monod.lib` у каталозі `library\`, то відкомпілювати програму можна командою

bcc –Iinclude\ -Llibrary\ main.c monod.lib

У результаті компіляції одержимо файл main.exe. Надалі можна використовувати цей файл для знаходження оптимуму даної функції вибраним методом.

5.2 БІБЛІОТЕКА МЕТОДІВ БАГАТОВИМІРНОЇ ОПТИМІЗАЦІЇ БЕЗ ОБМЕЖЕНЬ MULTID

Бібліотека пошуку мінімуму функцій декількох змінних *multid* містить функції, що реалізують такі методи:

- метод покоординатного спуску;
- метод Хука-Дживса;
- метод пошуку за симплексом;
- симплексний метод Нелдера-Міда;
- градієнтний метод Коші.

Опис функцій бібліотеки

1 Метод покоординатного спуску

```
#include "multid.h"
int CoordDescent( double f(double*), const double *x0, double *xr, int n, double dx, double e, long ns, long *ks);
```

Функція **CoordDescent** робить пошук мінімуму функції **f** декількох змінних методом покоординатного спуску. У функцію передаються покажчики на масиви координат початкової точки **x0** і точки мінімуму **xr**, кількість змінних **n**, початковий крок пошуку **dx**, допустима похибка пошуку **e**, максимально

допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, або **0** у протилежному разі.

2 Метод Хука-Джівса

```
#include "multid.h"

int HookJeeves(double f(double*), const double
*x0, double *xr, int n, double dx, double e, double
fe, long ns, long *ks);
```

Функція **HookJeeves** робить пошук мінімуму функції **f** декількох змінних методом Хука-Джівса. У функцію передаються покажчики на масиви координат початкової точки **x0** і точки мінімуму **xr**, кількість змінних **n**, початковий крок пошуку **dx**, припустима похибка пошуку **e**, максимально допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

3 Метод пошуку за симплексом

```
#include "multid.h"

int Simplex( double f(double*), const double
*x0, double *xr, int n, double alpha, double e,
long ns, long *ks);
```

Функція **Simplex** робить пошук мінімуму функції **f** декількох змінних методом пошуку за симплексом. У функцію передаються покажчики на масиви координат початкової точки

x0 і точки мінімуму **xr**, кількість змінних **n**, початковий розмір симплексу **alpha**, допустима похибка пошуку **e**, максимально допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

4 Симплексний метод Нелдера-Міда

```
#include "multid.h"
int NelderMeade( double f(double*), const double *x0, double *xr, int n, double dx, double e, long ns, long *ks);
```

Функція *NelderMeade* робить пошук мінімуму функції **f** декількох змінних методом пошуку за симплексом. У функцію передаються покажчики на масиви координат початкової точки **x0** і точки мінімуму **xr**, кількість змінних **n**, початковий розмір симплекса **dx**, припустима похибка пошуку **e**, максимально допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

5 Градієнтний метод Коші

```
#include "multid.h"
int GradCauchy(double (*f)(double*),double (*df[])(double*), const double *xs,double *xr, int n, double ealpha, double e, long ns, long *ks);
```

Функція **GradCauchy** робить пошук мінімуму функції **f** декількох змінних градієнтним методом Коші. У функцію передаються покажчик на цільову функцію **f**, масив покажчиків на часткові похідні **df**, покажчики на масиви координат початкової точки **xs** і точки мінімуму **xr**, кількість змінних **n**, початковий крок пошуку **ealpha**, припустима похибка пошуку **e**, максимально допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

Приклад написання програми

Необхідно знайти мінімум функції $f(x, y) = 8 \cdot x^2 + 4 \cdot x \cdot y + 5 \cdot y^2$. Проведемо обчислення методом Коші.

Програма буде мати вигляд:

```
/* Програма, що використовує функцію обчислення
оптимуму методом Коші      */

#include <stdio.h>
#include "multid.h"        //підключення бібліотеки (опис
прототипів функцій)
double f(double *);                    // цільова функція
double df1(double *);      // перша частинна похідна за x
double df2(double *);      // перша частинна похідна за y

double (*df[2])(double *) = { df1, df2};
double xs[2] = { 10., 10.};          //початкова точка пошуку
double xr[2];                        // результат обчислень
long ks;
/* Основна програма */
void main()
```

```
{
      if( GradCauchy( f, df, xs, xr, 2, 0.1, 1e-5, 10000, &ks) )
           printf("f(%g,%g)=%g %ld кроків \n", xr[0], xr[1],
f(xr), ks);
       else
           printf("Мінімум не знайдений за %ld кроків \n", ks);
       printf("\n");
}
/*Обчислення функції */
double f(double *x)
{
   return 8*x[0]*x[0] + 4*x[0]*x[1] + 5*x[1]*x[1];
}

/*Обчислення першої частинної похідної */
double df1(double *x)
{
    return 16*x[0] + 4*x[1];
}
/*Обчислення другої частинної похідної */
double df2(double *x)
{
     return 4*x[0] + 10*x[1];
}
```

Якщо програма має назву main.c, заголовний файл зберігається в каталозі `include\` , а бібліотеки `monod.lib` і `multid.lib` у каталозі `library\`, то відкомпілювати програму можна командою

bcc –Iinclude\ -Llibrary\ main.c monod.lib multid.lib

У результаті компіляції одержимо файл main.exe. Надалі можна використовувати цей файл для знаходження оптимуму даної функції обраним методом.

5.3 БІБЛІОТЕКА МЕТОДІВ БАГАТОВИМІРНОЇ ОПТИМІЗАЦІЇ З ОБМЕЖЕННЯМИ MULTIDR

Бібліотека пошуку мінімуму функцій декількох змінних із обмеженнями *multidr* містить функції, що реалізують такі методи:

- метод штрафних функцій;
- метод комплексів.

1 Метод штрафних функцій

```
#include "multidr.h"
int Penal(double f(double*), double x0[], double x[], int n, double (*g[])(double*), int J, double (*h[])(double*), int K, double dx, double e, double fe, long ns, long *ks);
```

Функція реалізує метод пошуку мінімуму функції декількох змінних із обмеженнями – метод штрафних функцій. У функцію передаються покажчик на цільову функцію **f**, покажчики на масиви координат початкової точки **x0** і точки мінімуму **xr**, кількість змінних **n**, масив покажчиків на функції обмежень-нерівностей **g** і їхня кількість **J**, масив покажчиків на функції обмежень-рівностей **h** та їхня кількість **K**, початковий крок пошуку **dx**, припустима похибка пошуку координат **e**, припустима похибка цільової функції **fe**, максимально допустима кількість обчислень функції **ns**, покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

2 Метод комплексів

```
#include "multidr.h"
int Complexs(double f(double*), double *x, const double *xl, const double *xu, int n, double (*g[])(double*), int J, double e, double fe, long ns, long *ks);
```

Функція реалізує метод пошуку мінімуму функції декількох змінних із обмеженнями – метод комплексів. У функцію передаються покажчик на цільову функцію **f**; покажчики на масиви координат точки мінімуму **x**, верхніх і нижніх границь **xl** і **xu**; кількість змінних **n**; масив покажчиків на функції обмежень-нерівностей **g** і їхня кількість **J**; припустима похибка пошуку координат **e**; припустима похибка цільової функції **fe**; максимально припустима кількість обчислень функції **ns**; покажчик на кількість обчислень функції **ks**.

Якщо кількість обчислень функції **f** досягає значення **ns**, вважається, що мінімум не знайдений.

Функція повертає **1**, якщо знайдений мінімум функції **f**, і **0** у протилежному разі.

Приклад написання програми

Необхідно знайти мінімум функції $f(x, y) = 30x_1x_2+48*10^4/x_1+96*10^4/x_2$. Проведемо обчислення методом комплексів.

Програма буде мати вигляд:

```
/* Програма, що використовує функцію обчислення
оптимуму методом комплексів      */
```

```
#include <stdio.h>
#include "multidr.h"        //підключення бібліотеки (опис
прототипів функцій)
double f(double *x);
double g1(double *x);
double g2(double *x);
double g3(double *x);

double (*g[])(double*) = { g1, g2, g3 };
double xl[] = { 0, 0 };
double xu[] = { 60, 70 };

double x[2];
long ns = 1000;
long ks;

void main()
{
    if( Complexs( f, x, xl, xu, 2, g, 3, 0.001, 0.001, ns, &ks) )
        printf("%lf %lf", x[0], x[1]);
     else
        printf("Розв'язок не знайдено за %ld кроків \n", ks);
}

double f(double *x)
{return 30*x[0]*x[1] + 48e4/x[0] + 96e4/x[1];}

double g1(double *x)
{return 110 - x[0] - x[1];}

double g2(double *x)
{return 3*x[0] - x[1];}

double g3(double *x)
{return 2/3.*x[1] - 16e3/x[1]/x[0];}
```

Якщо програма має назву main.c, заголовний файл зберігається в каталозі `include\`, а бібліотека `multidr.lib` у каталозі `library\`, то відкомпілювати програму можна командою

bcc –Iinclude\ -Llibrary\ main.c multidr.lib

У результаті компіляції одержимо файл main.exe. Надалі можна використовувати цей файл для обчислення оптимуму даної функції вибраним методом.

ТЕМА 6 БІБЛІОТЕКА ПРОГРАМ, СТВОРЕНИХ У СЕРЕДОВИЩІ DELPHI 5

У програмний комплекс “Met_Opt” входять також програми, створені в середовищі Delphi 5 з використанням мови Object Pascal. Це:

- програма, що реалізує метод Давідона-Флетчера-Пауелла;
- програма, що реалізує комплексний метод;
- програма, що реалізує градієнтний метод Ньютона.

Кожна з програм використовує у своїй роботі модуль аналізатора математичних виразів. Це дає можливість звичайному користувачеві не звертатися до складного коду програми, а працювати тільки через зовнішній інтерфейс. Тобто, щоб змінити умову задачі, потрібно лише у рядку для введення змінити функцію, оптимум якої потрібно знайти.

6.1 ПРОГРАМА, ЩО РЕАЛІЗУЄ МЕТОД ДАВІДОНА-ФЛЕТЧЕРА-ПАУЕЛЛА

Дана програма демонструє метод пошуку оптимуму (мінімуму) функції від n змінних Давідона-Флетчера-Пауелла (ДФП). Програма є цілком самостійною і не потребує

перекомпіляції для різних функцій. Така перевага досягнута за рахунок впровадження аналізатора математичних виразів. Для проведення пошуку достатньо ввести функцію, задати початкову точку і натиснути кнопку "GO". Через те, що метод ДФП використовує градієнт, програма автоматично визначає градієнт функції (в аналітичному вигляді).

У випадку, якщо введена функція, що не збігається, або неправильно задана початкова точка, то незабаром після запуску з'явиться вікно про помилку. Це може бути, наприклад, переповнення або повідомлення про зростання значення функції порівняно із попередньою ітерацією.

Зовнішній вигляд головного вікна програми із пронумерованими червоним кольором елементами керування наводиться на рис.6.1, а нижче наводиться таблиця (табл.6.1) із поясненнями.

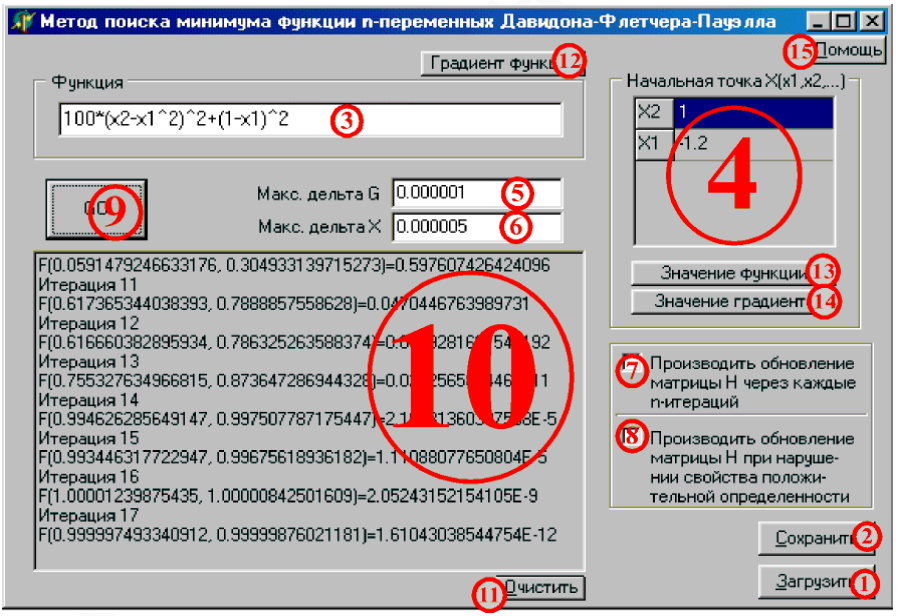


Рисунок 6.1 - Зовнішній вигляд програми

Таблиця 6.1 - Призначення елементів керування

Номер елемента	Призначення елемента керування
1	Кнопка "Загрузить". За допомогою цієї кнопки відбувається завантаження збереженого прикладу
2	Кнопка "Сохранить". За допомогою цієї кнопки проводиться зберігання прикладу
3	Вікно редагування функції, яку потрібно мінімізувати. У цьому вікні відбувається введення і редагування функції, яку необхідно мінімізувати. Передбачені такі операції: додавання (знак "+"), віднімання (знак "-"), множення (знак "*"), ділення (знак "/"), піднесення до степеня (знак "^"). Реалізовано такі функції: cos(), arccos(), sin(), arcsin(), tan(), arctan(), exp(), ln(), || (модуль). У виразах можна також використовувати дужки (круглі)
4	Вікно введення початкової точки пошуку. Список змінних заповнюється автоматично після введення функції і натискання клавіші "ENTER" (або при виході фокуса з вікна редагування функції)
5	Одна з умов припинення пошуку. Максимальне значення градієнта, при якому ще продовжується пошук
6	Одна з умов припинення пошуку. Максимальна відстань між попередньою точкою пошуку і поточною, при якій ще продовжується пошук
7	Якщо даний прапорець встановлений, то через n ітерацій матриця H буде оновлюватися (ставатиме одиничною)

8	Якщо даний прапорець встановлений, то на кожній ітерації буде відбуватися перевірка на позитивну визначеність матриці H, і при невиконанні цієї умови матриця H оновлюється
9	Кнопка запуску пошуку. При натисканні на дану кнопку відбувається пошук мінімуму зазначеної функції із зазначеною початковою точкою
10	Вікно виведення інформації про процес пошуку. Виводиться результат виконання кожної ітерації у вигляді: "Ітерація # F(x1, x2,...,xn)=a" де # - номер ітерації; $x_1,x_2,...,x_n$ – координати точки пошуку на даній ітерації; a – значення функції у даній точці пошуку. Останній запис і є результатом пошуку
11	Кнопка "Очистить" служить для очищення вікна виведення інформації про процес пошуку
12	Кнопка "Градиент функции" служить для перегляду градієнта функції
13	Кнопка "Значение функции" служить для перегляду значення функції у заданій початковій точці
14	Кнопка "Значение градиента" служить для перегляду значення градієнта функції у заданій початковій точці
15	Кнопка виклику файлу допомоги (у якому описано все вищезгадане)

6.2 ПРОГРАМА, ЩО РЕАЛІЗУЄ КОМПЛЕКСНИЙ МЕТОД

Розв'язувана задача полягає в мінімізації функції $f(x_1, x_2, ..., x_n)$, де x визначається явними обмеженнями

$$l_j \le x_j \le u_j \text{ при } j=1,2, \ldots,n, \qquad (6.1)$$

а також неявними обмеженнями

$$g_i \le b_i \text{ при } i=1,2, \ldots,m. \qquad (6.2)$$

Якщо цільова функція $f(x)$ опукла і функції $g_i(x)$ теж опуклі, то задача буде мати єдиний розв'язок. Значення l_j і u_j є нижньою і верхньою границями змінних. Якщо в конкретній задачі задані змінні теоретично не мають обмежень, то припущення про наявність у них "безпечних меж" тобто меж, що включають оптимум, дозволить застосувати комплексний метод.

Зовнішній вигляд головного вікна програми з пронумерованими елементами керування поданий на рис.6.2, нижче наводиться таблиця (табл.6.2) з поясненнями.

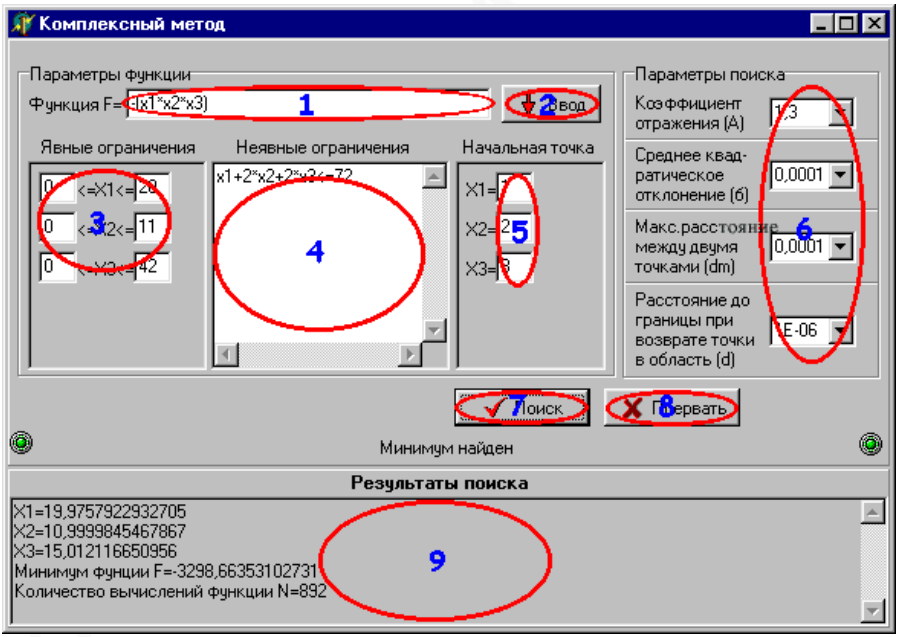


Рисунок 6.2 - Зовнішній вигляд головного вікна програми

Таблиця 6.2 - Елементи інтерфейсу програми, що реалізує комплексний метод

Номер елемента	Призначення елемента керування
1	Поле введення функції для мінімізації. У цьому вікні робиться введення і редагування функції, яку необхідно мінімізувати. Допускаються такі операції:
	додавання (знак "+"), віднімання (знак "-"), множення (знак "*"), ділення (знак "/"), піднесення до степеня (знак "^"). Передбачено такі функції: 'ARCTAN', 'COS', 'SIN', 'TAN', 'ABS', 'EXP', 'LN', 'LOG','SQRT', 'SQR', 'INT', 'FRAC','TRUNC', 'ROUND', 'ARCSIN', 'ARCCOS', 'SIGN' (за функціональним наповненням аналогічні відповідним функціям мови Pascal)
2	Кнопка введення функції. Приводить до активізації елементів 2,3,4
3	Елементи для задання явних обмежень для кожної змінної
4	Елементи для задання неявних обмежень. Вводяться порядково. Допускається використання тих самих операцій і функцій, що й у пункті 1. Додатково необхідне використання одного із таких знаків: "<" (менше), ">" (більше), "<=" (менше чи дорівнює), ">=" (більше чи дорівнює)
5	Елементи для задання початкової точки. З цієї точки починається пошук мінімуму. Точка повинна задовольняти явні та неявні обмеження
6	Параметри пошуку. Щоб зрозуміти значення даних параметрів, необхідно ознайомитися з алгоритмом методу

Номер елемента	Призначення елемента керування
1	Поле введення функції для мінімізації. У цьому вікні робиться введення і редагування функції, яку необхідно мінімізувати. Допускаються такі операції:
7	Кнопка, при натисканні на яку стартує процедура пошуку
8	Кнопка скасування пошуку. При скасуванні, як наслідок, видається та точка, до якої дійшов алгоритм на момент скасування. Її використання необхідно при тривалій відсутності результатів
9	Вікно для видачі результатів пошуку. У ньому відображається точка мінімуму, значення функції у цій точці і кількість обчислень функції

6.3 ПРОГРАМА, ЩО РЕАЛІЗУЄ ГРАДІЄНТНИЙ МЕТОД НЬЮТОНА

Дана програма демонструє градієнтний метод Ньютона. Програма є цілком самостійною і не потребує перекомпіляції для різних функцій. Це досягнуто за рахунок підключення аналізатора математичних виразів. Для проведення пошуку достатньо ввести функцію, задати початкову точку і натиснути кнопку "Расчет".

Зовнішній вигляд головного вікна програми з пронумерованими червоним кольором елементами керування наводиться на рис.6.3, а нижче наводиться таблиця (табл.6.3) із роз'ясненнями.

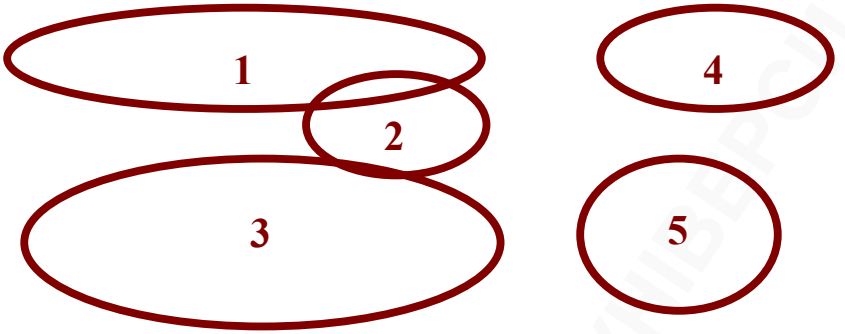


Рисунок 6.3 - Зовнішній вигляд головного вікна програми, що реалізує градієнтний метод Ньютона

Таблиця 6.3 - Елементи інтерфейсу програми, що реалізує градієнтний метод Ньютона

Номер елементу	Призначення елемента керування
1	Вікно редагування функції, яку потрібно мінімізувати. У цьому вікні відбувається введення і редагування функції, яку необхідно мінімізувати. Передбачені такі операції: додавання (знак "+"), віднімання (знак "-"), множення (знак "*"), ділення (знак "/"), піднесення до степеня (знак "^"). Реалізовано такі функції: cos(), arccos(), sin(), arcsin(), tan(), arctan(), exp(), ln(), || (модуль). У виразах можна також використовувати дужки (круглі)
2	Кнопка “Parse (разбор функции)”. Натискаємо цю кнопку після того як у вікні введення функції записали вираз, оптимум якого потрібно знайти. У результаті у вікні введення початкової точки пошуку

	з’явиться список змінних
3	Вікно введення початкової точки пошуку. Список змінних заповнюється автоматично після натискання кнопки “Parse”. Користувачеві пропонується ввести значення початкової точки відповідно до поданого списку змінних. З цієї точки починається пошук мінімуму. Точка повинна відповідати області визначення функції
4	Кнопка “Расчет”, при натисканні на яку стартує процедура пошуку
5	Вікно для видачі результатів пошуку. У ньому відображається точка мінімуму, тобто підсумковий вектор

ТЕМА 7 БІБЛІОТЕКА ПРОГРАМ, ЩО РЕАЛІЗОВАНІ НА ФОРТРАНІ

Бібліотека на Фортрані подана в цьому курсі для ознайомлення з світовими розробками. Тут наведено тільки кілька підпрограм, з іншими Ви зможете ознайомитися за адресою *http://www.netlib.org/*

Усі алгоритми методів оптимізації мовою FORTRAN реалізовані у вигляді підпрограм SUBROUTINE чи функцій FUNCTION, що підключаються до основної програми. За допомогою компілятора з мови FORTRAN gcc version 2.95.2 це можна зробити, наприклад, так:

```
g77 -o exmpl1.exe exmpl1.f dfmin.f
```

де `exmpl1.exe` - файл-результат дії компілятора, а `exmpl1.f` - основна програма, `dfmin.f` - підпрограма, що підключається. Можна досягти того самого результату, якщо цілком скопіювати підпрограму в основну програму, у цьому випадку командний рядок буде мати вигляд:

```
g77 -o exmpl1.exe exmpl1
```

До пакета входять:

dfmin.f - комбінація алгоритму “золотого” перетину і квадратичної інтерполяції;

goldsear.f - алгоритм “золотого” перетину;

gradmeth.f - алгоритм найшвидшого спуску;

neldmead.f - алгоритм Нелдера-Міда;

qmin.f - квадратична інтерполяція.

Директорія EXAMPLE містить приклад застосування цих методів для найпростіших функцій. Кожен приклад зберігається у файлі з таким же ім'ям як і відповідного алгоритму *+'_pr.f'*. У кожному прикладі (крім *DFMIN_PR.F*) містяться підпрограми введення *iputs* і виведення результату *result*. Введення організоване із клавіатури, а виведення у файл. При бажанні пристрій вводу і виводу можна замінити (для цього необхідно зробити заміну в операторах *WRITE* і *READ*).

Точність, з якою необхідно знайти мінімум, задана константою в основній програмі за допомогою оператора *PARAMETER*.

У прикладах на багатовимірну оптимізацію необхідно зазначити кількість змінних, це зроблено в підпрограмах типу

SUBROUTINE PRINTFUN(N), зазначене N=... У програмі *gradmeth_pr.f* використовується формула для похідної функції, вона зазначається у підпрограмі S*UBROUTINE XGRADVECT(P,S,N).*

Ці приклади можна легко модифікувати для інших функцій, зазначивши в підпрограмах типу *REAL FUNCTION F(P)* необхідну функцію й у підпрограмах типу *SUBROUTINE PRINTFUN(N)* для друку нової функції, а також, зробивши необхідні зміни в інших частинах програми, використовуючи вищезазначені зауваження.

Опис процедур та їхніх параметрів

Метод "золотого" перетину

Підпрограма

```
SUBROUTINE GOLDSEARCH(F,A,B,P,K,Delta,Epsilon)
```

реалізує алгоритм методу "золотого" перетину, де F- цільова функція, яку в основній програмі необхідно оголосити як EXTERNAL F. Цільова функція в підпрограмі буде мати вигляд:

```
REAL FUNCTION F(X)
REAL X
F=X*X-SIN(X)
RETURN

END
```

де A, B - (A, B) інтервал, на якому відбувається пошук;

P- значення абсциси, у якій досягається мінімум;

K- лічильник ітерацій;

Delta, Epsilon - точність, з якою необхідно знайти мінімум функції F (Delta - точність для абсциси, Epsilon - для ординати).

Квадратична апроксимація

Підпрограма, що реалізує алгоритм квадратичної апроксимації, має такий заголовок:

```
SUBROUTINE QMINIM(F, F1, P0, Delta, Epsilon,
Jmax, Kmax, Pmin, H, Ymin, Error, Cond, Count)
```

де F - цільова функція; F1 – похідна від цільової функції;

P0- початкова точка;

Delta, Epsilon - точність для абсциси й ординати точки мінімуму;

(Pmin, Ymin) - точка мінімуму;

Count - лічильник ітерацій;

Cond - код закінчення виходу із підпрограми.

Комбінація методу "золотого" перетину і послідовної параболічної інтерполяції реалізована у функції

```
DOUBLE PRECISION FUNCTION DFMIN(AX,BX,F,TOL)
```

де AX (типу DOUBLE PRECISION) - лівий кінець початкового інтервалу;

BX (типу DOUBLE PRECISION) - правий кінець початкового інтервалу;

F - дійсна функція типу DOUBLE PRECISION, що обчислює F (X) для будь-якого X з інтервалу (AX, BX);

TOL (типу DOUBLE PRECISION) бажана довжина інтервалу невизначеності кінцевого результату (>= 0.0);

DFMIN - наближена абсциса мінімуму функції F.

Метод найшвидшого спуску

У цьому методі поряд зі значенням функції використовується і її градієнт.

Заголовок підпрограми, що реалізує цей метод, має вигляд:

```
SUBROUTINE XGRADSR (F, Y0, P0, P1, P2, Pmin, S,
N, Jmax, Max, Delta, Epsilon, H, Ymin, Error, Cond,
Count)
```

де F- цільова функція, яку в основній програмі необхідно оголосити як EXTERNAL F. Цільова функція в підпрограмі буде мати вигляд:

```
REAL FUNCTION F(P,N)
.................
F=
RENURN
END
```

де N- кількість змінних;

Delta, Epsilon - точності, з якими необхідно знайти мінімум;

P0, Y0 - початкова точка і її значення;

Jmax - максимальна кількість ітерацій;

Count - лічильник ітерацій;

Pmin - точка, у якій досягається мінімум;

Ymin - мінімальне значення функції F.

Метод Нелдера-Міда

Симплексний метод Нелдера і Міда реалізований у підпрограмі

```
SUBROUTINE XNELDER(F, V, Y, N, Epsilon, Norm,
Lo, Hi, Count)
```

де F- цільова функція;

V - координати вершин симплексу;

Y - значення функції у вершинах симплексу;

Epsilon - необхідна точність;

Norm - розмір симплексу;

Hi (Lo) - номер вершини симплексу, у якій значення функції найбільше (найменше);

Count - лічильник ітерацій.

ТЕМА 8 ЛІНІЙНЕ ПРОГРАМУВАННЯ

8.1 СИМПЛЕКС-МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Задача лінійного програмування (ЗЛП) - це задача пошуку найбільшого (або найменшого) значення лінійної функції

$$z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \qquad (8.1)$$

за умов, коли змінні $x_1, x_2, \ldots, x_n$ мають задовольняти систему лінійних обмежень типу рівностей або нерівностей:

$$\begin{aligned} &a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i, && i=1,2,\ldots,m_1,\\ &a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i, && i=m_1+1,\ldots,m_2,\\ &a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i, && i=m_2+1,\ldots,m_3. \end{aligned} \qquad (8.2)$$

Вектор змінних є допустимим планом ЗЛП, якщо його компоненти задовольняють кожне з обмежень (8.2) цієї задачі. Оптимальним планом ЗЛП є такий її допустимий план, при якому цільова функція (8.1) набуває найбільшого (найменшого) значення порівняно з її значеннями за будь-якого іншого допустимого плану.

Розв'язати ЗЛП - означає знайти один (кілька або всі) з оптимальних планів і обчислити відповідне - оптимальне - значення цільової функції або встановити, що задача не є розв'язаною.

Розв'язок ЗЛП з двома змінними легко знайти графічним методом.

Графічне розв'язування ЗЛП з трьома змінними стає менш наочним, а в разі більшої кількості змінних навіть неможливим.

Загальна постановка ЗЛП охоплює велику кількість різноманітних конкретних варіантів, які різняться між собою як вимогами до цільової функції (знайти її максимум чи мінімум),

так і структурою систем і обмежень (самі лише нерівності або рівності чи поєднання рівностей і нерівностей у будь-яких кількісних співвідношеннях). Один із варіантів ЗЛП узято за стандарт. Задачу, сформульовану у відповідній формі, називають канонічною ЗЛП. Вона має такий вигляд:

$$z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max,$$
$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i \quad \text{i=1,2,...,m,}$$
$$x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0.$$

Будь-яка ЗЛП, якщо вона не канонічна, може буде зведена до рівнозначної їй канонічної ЗЛП.

Існують три правила переходу від загальної ЗЛП до канонічної.

1 Перехід від мінімізації до максимізації

Задача пошуку мінімуму деякої функції рівнозначна задачі пошуку максимум протилежної функції:

$$z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \min,$$

$$z_1 = -z = -c_1x_1 - c_2x_2 - \ldots - c_nx_n \to \max,$$

де z та z_1 рівнозначні.

2 Перехід від нерівності до рівності

Будь-яке обмеження, що має вигляд нерівності, можна звести до обмеження-рівності, ввівши додаткову невід'ємну змінну.

Так, нерівність вигляду

$$\alpha_1x_1 + \alpha_2x_2 + \ldots + \alpha_nx_n \leq \beta$$

рівнозначна вимогам

$$\alpha_1x_1 + \alpha_2x_2 + \ldots + \alpha_nx_n + \mu = \beta, \qquad \mu \geq 0$$

а нерівність вигляду

$$\nu_1 x_1 + \nu_2 x_2 + \ldots + \nu_n x_n \geq \sigma$$

рівнозначна вимогам

$$\nu_1 x_1 + \nu_2 x_2 + \ldots + \nu_n x_n - \eta = \sigma, \qquad \pi \geq 0.$$

3 Перехід від змінних, які не мають обмежень щодо знака до невід'ємних

Нехай x_j не має обмежень щодо знака. Тоді таку змінну можна замінити різницею двох невід'ємних змінних x_j' і x_j'':

$$x_j = x_j' - x_j'' \geq 0, \qquad x_j' \geq 0, \qquad x_j'' \geq 0.$$

Для розв'язання задач лінійного програмування розроблено багато різних методів. Одним із основних є симплекс-метод (Дж.Данциг), що базується на спеціальному алгоритмі. Цей метод полягає у спрямованому перебиранні вершин допустимої багатогранної множини задачі лінійного програмування за допомогою симплекс-алгоритму, за яким значення цільової функції зростає чи спадає від вершини до вершини залежно від того, максимізується чи мінімізується цільова функція задачі. Початкова вершина симплекс-процесу теж визначається за симплекс-алгоритмом для допоміжної задачі лінійного програмування, яка будується за певними правилами з основної задачі. Простота симплекс-алгоритму робить симплекс-метод досить зручним.

Алгоритм симплекс-методу та його реалізація за допомогою симплекс-таблиць

Виберемо допустимий базис $B=\{A_{i1},A_{i2},...,A_{im}\}$ канонічної ЗЛП і побудуємо симплекс таблицю, в якій використовуються такі позначення:

C_B - вектор коефіцієнтів при базисних змінних;

$\Delta_0 = C_B X_B = c_{i1} b_{i1} + \ldots + c_{im} b_{im}$,

$\Delta_j = C_P * P_J - c_j$,

P_j - 'j'-стовпець канонічної системи,

Δ_j - відносна оцінка небазисної змінної X_j.

Таблиця 8.1 – Симплекс-таблиця

X_B	C_B	B	c_i	...	c_j	...	c_{is}	...	c_n
			x_1	...	...	...	x_{is}	...	x_n
X_{i1}	C_{1i}	β_i	a_{11}	...	a_{1j}	...	0	...	a_{in}
...	...	...	...	...	...	...	...	...	...
X_{is}	C_{is}	β_s	a_{s1}	...	a_{sj}	...	1	...	a_{sn}
...	...	...	...	...	...	...	...	...	...
X_{is*}	C_{is*}	β_{s*}	a_{s*1}	...	a_{s*j}	...	0	...	a_{s*n}
...	...	...	...	...	...	...	...	...	...
X_{im}	C_{im}	β_m	$a_{1ь}$	....	a_{mj}	...	0	...	a_{mn}
	Δ	Δ	Δ	....	Δ	...	0	...	Δ

1 Аналізуємо 'Δ'-рядок і перевіряємо, чи виконується ознака оптимальності: $\Delta_j \geq 0$ для всіх j=1,2,...,n. Якщо виконується, то поточний базис оптимальний. Оптимальне значення цільової функції дорівнює Δ_0.

Якщо ознака оптимальності не виконується, переходимо до п.2.

2 Перевіряємо, чи немає серед стовпців X_j такого, в якому $\Delta_j < 0$ і $a_{sj} \leq 0$ для всіх s = 1,2,...,m.

Якщо є, робимо висновок про необмеженість зверху значень цільової функції на множині допустимих планів.

Інакше - п.3.

3 Вибираємо такий стовпець X_{j*}, якому відповідає найменша від'ємна оцінка Δ_{j*} (якщо таких стовпців кілька, то вибираємо перший з них).

Далі такий стовпець називатимемо провідним стовпцем.

4 Проглядаємо згори вниз стовпець b і провідний стовпець X_{j*}. Якщо елемент a_{s*j} провідного стовпця додатний, знаходимо відношення $\beta_s / a_{s*j} = \theta$. Вибираємо найменше з цих відношень, відповідний рядок - провідний.

5 Встановлюємо новий допустимий базис, який утворюється з поточного базису вилученням з нього вектора X_{js} і введенням вектора X_{j*}.

6 Обчислюємо елементи нової симплекс-таблиці, яка відповідає новому базису, після чого повертаємось до п.1, маючи на увазі новий базис і нову симплекс-таблицю.

A		b
C		d

Рисунок 8.1 – Елементи нової таблиці

$a^1 = (ad-bc)/d;$
$a^{(1)}_{ks} = 1/a_{ks}$;
$a^{(1)}_{kj} = a_{kj}/a_{ks};$
$a^{(1)}_{is} = -a_{is}/a_{ks}.$

Приклад

Знайти максимум функції

$$Z = 5x_1 + 3x_2 + 2x_3 + 6x_4 \rightarrow max$$

при обмеженнях

$$3x_1+x_2+2x_4+x_5=700,$$
$$2x_1+2x_2+x_3+x_6=800,$$
$$2x_2+5x_3+4x_4+x_7=500,$$
$$x_j \geq 0 \quad j=1,2,...,7.$$

Побудуємо симплекс-таблицю $X_B=\{X_5X_6X_7\}$.

Таблиця 8.2 – Симплекс-таблиця 1 прикладу 1

X_B	C_B	b	5	3	2	6	0	0	0
			X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_5	0	700	3	1	0	2	1	0	0

X_6	0	800	2	2	1	0	0	1	0
X_7	0	500	0	2	5	4	0	0	1
			-5	-3	-2	-6	0	0	0

Обчисливши елементи 'Δ'-рядка, помітимо, що серед чисел $\Delta_1,...,\Delta_7$ є від'ємні. Це означає, що поточний базис потрібно буде покращити.

Найменшою з від'ємних оцінок векторів умов є оцінка вектора X_4. Тому, щоб покращити базис, введемо до нього вектор X_4. Знайдемо θ:

$$\theta_1=700/2=350, \quad \theta_3=500/4=125.$$

Мінімальному з чисел θ_1 та θ_3 відповідає третій рядок, який і буде провідним. Цьому рядку відповідає базисний вектор X_8. Його потрібно з базису вилучити. Новий базис $X_B=\{X_5X_6X_4\}$.

Таблиця 8.3 – Симплекс-таблиця 2 прикладу 1

X_B	C_B	b	5	3	2	6	0	0	0
			X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_5	0	450	3	0	-5/2	0	1	0	-1/2
X_6	0	800	2	2	1	0	0	1	8
X_4	6	125	0	1/2	5/4	1	0	0	1/4
			-5	0	11/2	0	0	0	3/2

Оскільки Δ_1=-5, то поточний базис підлягає поліпшенню. Вилучити з базису потрібно вектор X_5: $X_B=\{X_1X_6X_4\}$.

Таблиця 8.4 – Симплекс-таблиця 3 прикладу 1

X_B	C_B	b	5	3	2	6	0	0	0
			X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_1	5	150	1	0	-5/6	0	1/3	0	-1/6
X_6	0	500	0	2	2/3	0	-2/3	1	1/3
X_4	6	120	0	1/2	5/4	1	0	0	1/4

		1500	0	0	4/3	0	5/3	0	2/3

Всі оцінки векторів умов додатні. Отже, знайдений базис оптимальний. Оптимальне значення цільової функції дорівнює 1500.

Докладніше симплекс-метод описаний в [6, 13].

Сьогодні існує декілька пакетів, що реалізують симплекс-метод. Ми подаємо на розгляд один із них – пакет LP88 [3], в якому реалізовані особливості розв'язання задач лінійного програмування.

8.2 ПАКЕТ LP88

LP88 - це програма, що працює на персональному комп'ютері фірми IBM (PC, XT або PC/AT) під керуванням дискової операційної системи ПВМ (PC DOS) версії 2.00 і вище.

LP88 - дозволяє формулювати і розв'язувати задачі ЛП, що містять до 255 обмежень і 2255 змінних. LPX88 призначений для розв'язання розріджених задач (менше десяти відсотків коефіцієнтів ненульові), що містять до 510 обмежень і 2510 перемінних. Розріджені задачі ЛП великої розмірності звичайно можна розв'язати в кілька раз швидше за допомогою LPX88, ніж із використанням LP88. Припустимо, що ви в цілому знайомі з поняттями, методами і термінологією лінійного програмування. Більшість змістовних навчальних посібників з дослідження операцій, керування або дискретної математики містять досить матеріалу з лінійного програмування, щоб зрозуміти даний посібник. Додатки лінійного програмування містять завдання виробництва, номенклатури, укладання розкладу, керування запасами, готівкою, транспортні задачі, мережні задачі.

LP88 - це автономна програма на машинній мові, що не вимагає якогось спеціального програмного забезпечення, крім операційної системи DOS.

У LP88 застосовується модифікований симплекс-алгоритм для ефективного розв'язання задач лінійного програмування, при цьому використовуються центральний процесор, пам'ять і дисковод IBM PC. Крім того, LP88 містить потужну систему команд для установки і збереження в пам'яті задач, керування ходом обчислень і одержання роздрукованих вихідних даних.

LP88 можна динамічно конфігурувати під час рахунку, щоб щонайкраще використовувати наявну масову пам'ять, включаючи дискети, жорсткі диски, емулятори електронних дисків (RAM) і пам'ять.

Вихідні дані програми можуть бути відразу ж надруковані або ж передані на вихідний файл. Великі таблиці можна сформувати з метою використання швидких режимів друку і широких кареток для друку.

Чотири меню дозволяють Вам легко підбирати варіанти для:

- розв'язання і повторного розв'язання задач;
- формулювання, запису і пошуку задач ЛП;
- перерви і зміни ходу обчислень;
- одержання роздруку і вихідних файлів.

Можливий як інтерактивний, так і пакетний режим роботи.

В інтерактивному режимі керування процесом виконання програми здійснюється за допомогою функціональних клавіш IBM PC. У пакетному режимі керування здійснюється через послідовний файл, який містить список простих команд, що відіграють ту саму роль, що й функціональні клавіші в інтерактивному режимі.

За допомогою екранного редактора LP88 задачі ЛП можна виводити на екран і подавати у вигляді електронних таблиць, використовуючи монітор і клавіатуру.

Задачі ЛП можуть вводитися в такому вигляді, як вони сформульовані, приводити до стандартного вигляду не потрібно. Можна розв'язувати як задачу максимізації, так і мінімізації.

Обмеження можуть мати будь-яку комбінацію співвідношень: >=, = або <=.

Змінним привласнюються імена за замовчуванням, які користувач може замінити. Аналогічним способом імена за замовчуванням привласнюються обмеженням, їх також можна замінити. Ці імена використовують для того, щоб точно позначити усі вихідні дані на екрані дисплея та в роздруку.

Розмірність задач можна збільшити або зменшити під час їхнього введення або редагування.

Коефіцієнти задачі ЛП можуть вводитися з клавіатури за допомогою екранного редактора або ж із попередньо записаного послідовного файлу або ж із командного файлу в пакетному режимі.

Задачі ЛП можна зберігати на дискетах. Задачі записуються в компактному упорядкованому вигляді, що забезпечує їхній швидкий пошук і використання. Базиси розв'язків також можна зберігати, відшукувати і потім використовувати для ефективного повторного розв'язання задачі ЛП.

Існує кілька варіантів запуску симплекс-алгоритму. Алгоритм можна запустити з вихідного базису, що містить одиничні елементи і штучні змінні. Якщо збережений базис доступний із попереднього розв'язку, то його також можна використовувати як відправну точку. Нарешті, алгоритм можна повторно запустити з кінцевої точки попереднього етапу обчислень.

Значення за замовчуванням різних обчислювальних меж, допусків і програмних керувань можуть бути змінені користувачем, поки LP88 здійснює виконання програми.

Ви можете розпочати діалог із LP88, поки він виконує симплекс-алгоритм, щоб здійснити такі операції:

- зробити паузу після наступної ітерації;
- зробити паузу після кожної ітерації;
- знову інвертувати базисну матрицю;

- змінити обчислювальні межі, допуски і програмне керування;
- ввести в базис змінну;
- вивести на екран наведену вартість/корисність небазисної змінної;
- видрукувати запис опорних точок у міру їх створення;
- видрукувати розв'язок поточного базису;
- завершити симплекс-алгоритм.

У ході розв'язання задачі поточний запис статусу обчислень зберігається, виводиться на екран, включаючи, якщо дозволяє простір, поточний базис.

Всі арифметичні операції з подвійною точністю. Додатковий контроль за помилками за необхідності забезпечує повторне інвертування матриці базисних стовпців.

Пакет має систему генерування професійних звітів (див. додаток Г). Список наявних звітних таблиць містить:

- повний масив задачі;
- результат прямого/двоїстого розв'язання;
- розв'язання прямої задачі;
- розв'язання двоїстої задачі;
- аналіз чутливості елемента вартості/ корисності;
- аналіз чутливості елементів правої частини;
- базис зворотної матриці;
- базис зворотної матриці * небазисні стовпці.

Основний і двоїстий розв'язки можуть бути записані у послідовному файлі для зчитування й обробки іншими програмами.

Досконала система контролю в LP88 дозволяє користувачам змінювати формулювання задач, їх розв'язання й одержання результатів. Таким чином, задачу ЛП можна постійно змінювати, швидко розв'язувати повторно, використовуючи попередній базис або знову запускати зворотний базис із попереднього розв'язку.

Усі помилки виконання реєструються в LP88. Помилки викликають повідомлення із зазначенням номера головного коду і місця розташування помилки в рядку програми. Потім LP88 повернеться в інтерактивний режим операцій, що дає користувачу можливість відновити всі дані без втрат або виводу.

Рекомендації щодо використання вищеописаного пакета подані в додатку Д.

ЧАСТИНА ІІІ ПРАКТИЧНЕ ЗАСТОСУВАННЯ МЕТОДІВ ОПТИМІЗАЦІЇ

ТЕМА 9 РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ В ПАКЕТАХ ПРИКЛАДНИХ ПРОГРАМ

9.1 ПАКЕТ EXCEL

Excel - одна із найпотужніших програм для створення електронних таблиць і роботи з ними. Переваги Excel не тільки в її здатності виконувати різні обчислення, це можна зробити і за допомогою калькулятора, а й те, що Excel дозволяє проводити глибокий аналіз даних і одержувати в результаті нову корисну інформацію. У кожній новій версії компанія Microsoft пропонує додаткові можливості й удосконалює старі, щоб полегшити роботу користувачів із Excel. Версії Excel 97 і вище дозволяють розв’язувати задачі оптимізації як з обмеженнями, так і без них. Наприклад, пошук кількох параметрів, що забезпечують деякий наперед заданий результат.

Крім того, частіше за все цікавить не конкретний результат, а мінімально чи максимально можливий. Наприклад, як мінімізувати витрати на перевезення продукції або максимізувати прибуток від реалізації товарів і послуг. Такі задачі в Excel розв’язують за допомогою опції "Пошук розв’язку".

Процедуру “Пошук розв’язку” можна використовувати для визначення значення клітини, що впливає, яке відповідає екстремуму залежної клітини (наприклад, витрати на рекламу, що забезпечують максимальний прибуток). Клітини, що впливає, і цільова повинні бути зв'язані формулою листа, інакше при зміні значення однієї не буде змінюватися інша.

Приклад 9.1 Знайти мінімальне значення функції

$$F(x)=f(x_1,x_2,x_3)=(x_1-1)^2+(x_2-3)^2+4(x_3+5)^2$$

із заданою точністю ε>1e-5. Знайти точку мінімуму функції F(x), використовуючи початкову точку (4,-1,2).

Для того щоб розв’язати дану задачу, скористаємося пакетом Excel і його процедурою "Пошук розв’язку". Запишемо умову задачі відповідно до Excel. Числова й аналітична умови задачі подані в таблицях 9.1 і 9.2 відповідно.

Таблиця 9.1 - Числова умова

	A	B	C
1	4	3	9
2	-1	-4	16
3	2	7	196
4			221

Таблиця 9.2 - Аналітична умова

	A	B	C
1	4	=A1-1	=B1*B1
2	-1	=A2-3	=B2*B2
3	2	=A3+5	=4*B3*B3
4			=СУММ(C1:C3)

Вибираємо опцію ***Сервіс => Пошук розв’язку.*** У вікні "Пошук розв’язку", що з'явилося, встановлюємо цільову клітину

С4, яка буде дорівнювати мінімальному значенню, зазначаємо клітини А1:А3, що будуть змінюватися.

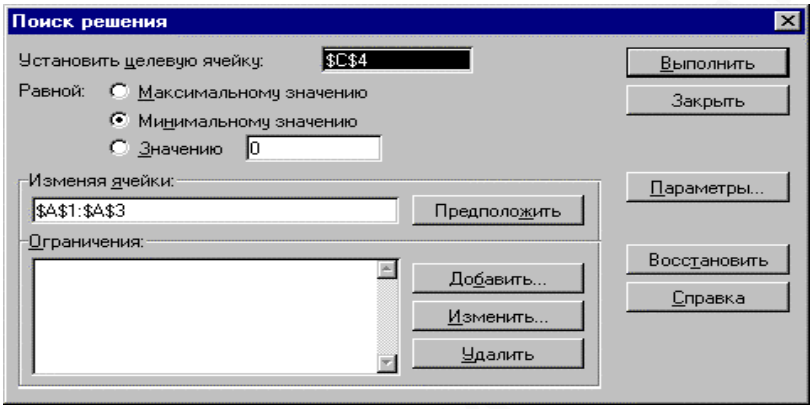


Рисунок 9.1 - Вигляд вікна "Пошук розв'язку"

Натискаємо кнопку "Параметри". У вікні, що з'явилося, встановлюємо параметри пошуку розв'язку (час пошуку (додаток Є), кількість ітерацій, точність, припустиме відхилення, вказівка на метод оцінки, на метод числового диференціювання і на алгоритм оптимізації).

На рис. 9.2 подані встановлені нами параметри для розв'язання даного прикладу.

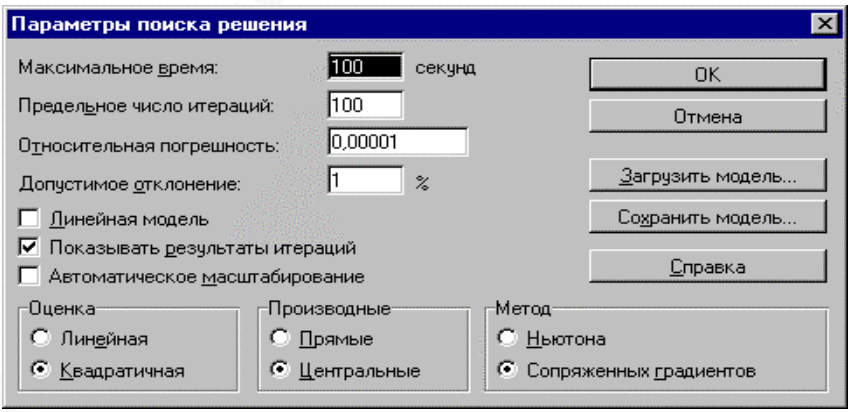


Рисунок 9.2 - Вигляд вікна "Параметри пошуку розв'язку"

Натискаємо кнопку "ОК" і переходимо назад у вікно "Пошук розв'язку". Коли всі необхідні для розв'язання вікна діалогу заповнені, натискаємо на кнопку "Виконати". Програма робить обчислення. Через те що ми установили опцію "Показувати результати ітерацій" , то після виконання першої з'являться вікно "Поточний стан пошуку розв'язку" (рис.9.3) і результати обчислень (таблиця 9.3). Натиснувши кнопку "Зберегти сценарій", ми можемо зберегти під унікальними іменами результати кожної ітерації пошуку і потім у будь-який час звертатися до них і аналізувати. На цьому етапі ми можемо як призупинити процес знаходження оптимального значення, так і продовжити його, а також одержати додаткову інформацію. За умови, що вибрано "Продовжити", після виконання кожної ітерації на екрані будуть з'являтися це вікно і нові результати.

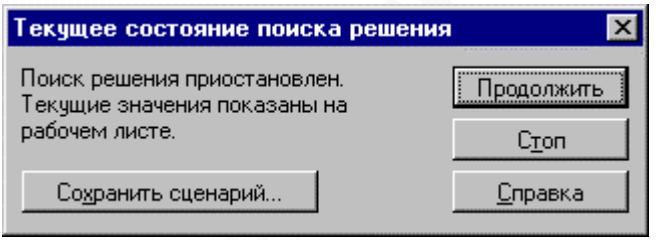


Рисунок 9.3 - Вікно "Поточний стан пошуку розв'язку"

Таблиця 9.3 - Результати 1-ї ітерації пошуку розв'язку

	A	B	C
1	4	3	9
2	-1	-4	16
3	2,000002	7,000002	196,0001
4			221,0001

Результати подальших ітерацій наведені в таблицях 9.4 - 9.9.

Таблиця 9.4 - Результати 2-ї ітерації пошуку розв'язку

	A	B	З
1	*3,232205*	*2,232205*	*4,982737*
2	*0,023726*	*-2,97627*	*8,858204*
3	*-5,16609*	*-0,16609*	*0,110339*
4			*13,95128*

Таблиця 9.5 - Результати 3-ї ітерації пошуку розв'язку

	A	B	З
1	*3,232205*	*2,232205*	*4,982737*
2	*0,023726*	*-2,97627*	*8,858204*
3	*-5,16609*	*-0,16609*	*0,110339*
4			*13,95128*

Таблиця 9.6 - Результати 4-ї ітерації пошуку розв'язку

	A	B	C
1	*1,189377*	*0,189377*	*0,035863*
2	*2,747496*	*-0,2525*	*0,063758*
3	*-4,55811*	*0,441888*	*0,78106*
4			*0,880682*

Таблиця 9.7 - Результати 5-ї ітерації пошуку розв'язку

	A	B	C
1	*1,140909*	*0,140909*	*0,019855*
2	*2,812119*	*-0,18788*	*0,035299*
3	*-5,01048*	*-0,01048*	*0,00044*
4			*0,055594*

Таблиця 9.8 - Результати 6-ї ітерації пошуку розв'язку

	A	B	C
1	*1,140909*	*0,140909*	*0,019855*
2	*2,812119*	*-0,18788*	*0,035299*
3	*-5,01048*	*-0,01048*	*0,00044*
4			*0,055594*

Таблиця 9.9 - Результати 7-ї ітерації пошуку розв'язку

	A	B	C
1	*0,9999991683751*	*-0,0000008316249*	*0,0000000000007*
2	*2,9999989506224*	*-0,0000010493776*	*0,0000000000011*
3	*-5,0000000607915*	*-0,0000000607915*	*0,0000000000000*
4			*0,0000000000018*

Коли процес пошуку закінчений, на екрані з'явиться вікно "Результати пошуку розв'язку" (рис.9.4). Вам пропонують вибрати які дані залишити і тип звіту, а також Ви можете зберегти сценарій отриманого розв'язку. Встановивши опції, натискаємо "ОК" і одержуємо звіт зазначеного типу (рис.9.5 - 9.7).

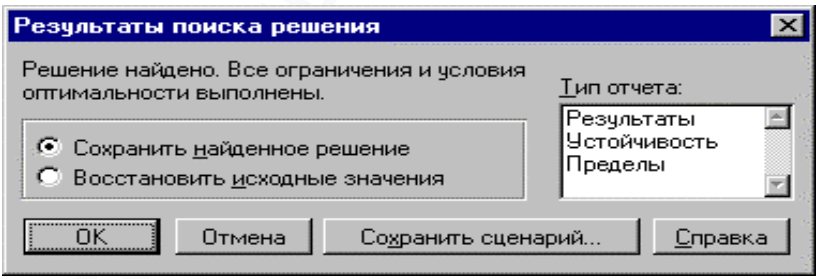
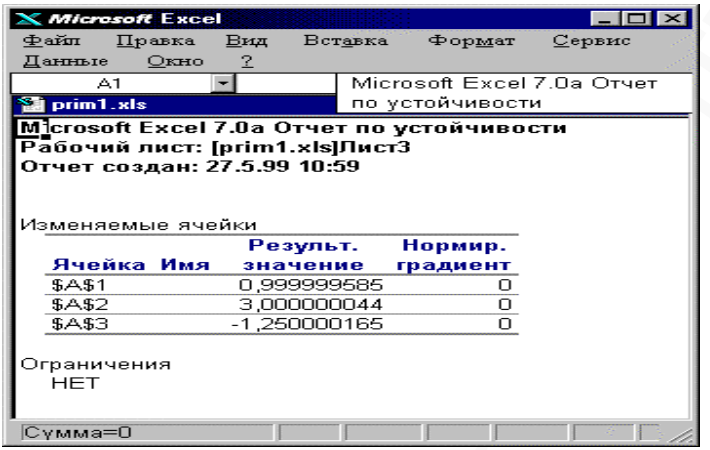


Рисунок 9.4 - Вікно "Результати пошуку розв'язку"



Microsoft Excel

Файл Правка Вид Вставка Формат Сервис
Данные Окно ?

A1 | Microsoft Excel 7.0a Отчет по устойчивости

prim1.xls

Microsoft Excel 7.0a Отчет по устойчивости
Рабочий лист: [prim1.xls]Лист3
Отчет создан: 27.5.99 10:59

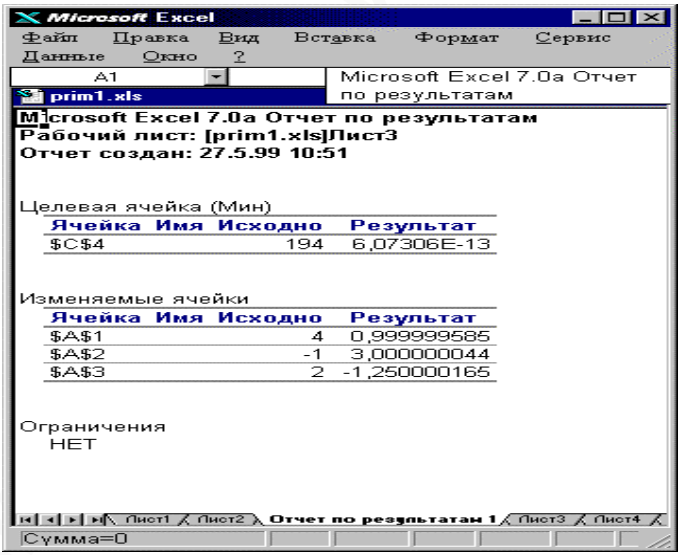
Изменяемые ячейки

Ячейка	Имя	Результ. значение	Нормир. градиент
A1		0,999999585	0
A2		3,000000044	0
A3		-1,250000165	0

Ограничения
НЕТ

Сумма=0

Рисунок 9.5 - Приклад звіту за усталеністю



Microsoft Excel

Файл Правка Вид Вставка Формат Сервис
Данные Окно ?

A1 | Microsoft Excel 7.0a Отчет по результатам

prim1.xls

Microsoft Excel 7.0a Отчет по результатам
Рабочий лист: [prim1.xls]Лист3
Отчет создан: 27.5.99 10:51

Целевая ячейка (Мин)

Ячейка	Имя	Исходно	Результат
C4		194	6,07306E-13

Изменяемые ячейки

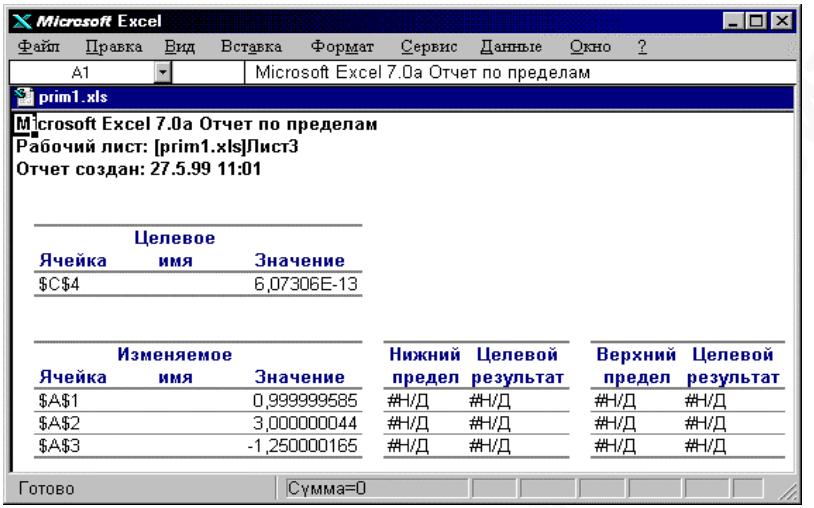
Ячейка	Имя	Исходно	Результат
A1		4	0,999999585
A2		-1	3,000000044
A3		2	-1,250000165

Ограничения
НЕТ

Лист1 Лист2 Отчет по результатам 1 Лист3 Лист4

Сумма=0

Рисунок 9.6 - Приклад звіту за результатами



Ячейка	Целевое имя	Значение
C4		6,07306E-13

Ячейка	Изменяемое имя	Значение	Нижний предел	Целевой результат	Верхний предел	Целевой результат
A1		0,999999585	#Н/Д	#Н/Д	#Н/Д	#Н/Д
A2		3,000000044	#Н/Д	#Н/Д	#Н/Д	#Н/Д
A3		-1,250000165	#Н/Д	#Н/Д	#Н/Д	#Н/Д

Рисунок 9.7 - Приклад звіту за границями

У складі Microsoft Excel у папці Examples\Solver міститься книга з прикладами використання процедури пошуку розв'язку (Solver.xls). У книзі Solverex.xls можна довідатися про процедури максимізації або мінімізації цільової функції, а також про накладення і зміну обмежень і збереження моделі оптимізації.

Листи з прикладами розрахунків із книги Solvsamp.xls можна використовувати як основу для постановки прикладних задач оптимізації.

9.2 ПАКЕТ MAPLE

Maple - пакет програм, призначений для математичних обчислень. Дозволяє знаходити аналітичний розв`язок, числовий розв`язок, будувати графіки як на площині, так і в просторі.

Maple має бібліотеки:

- *numaprox - числові наближення;*
- *combinat - функція для роботи з комбінаторикою;*

- *extrema - екстремуми функцій;*
- *linalg - лінійна алгебра (робота з матрицями);*
- *logic - робота з булевою алгеброю;*
- *networks - робота з графами;*
- *numteory - теорія чисел;*
- *plot - графічний пакет;*
- *simplex - задачі лінійної оптимізації;*
- *statss - статистичні обчислення;*
- *student - для студентських обчислень.*

Підключення бібліотеки дозволяє більш оптимально розв`язувати поставлену задачу, наприклад, підключення бібліотеки *linalg* дає можливість користувачу проводити обчислення з векторами та матрицями.

Засоби оптимізації

За допомогою бібліотеки **extrema** знаходять екстремуми функції, її найбільше та найменше значення.

Функція **extrema (expr, vars)** знаходить екстремуми функції за заданою змінною. **expr -** функція, екстремум якої потрібно знайти; **vars -** змінна, за якою ведеться пошук.

Перед тим, як розпочати роботу з цією функцією, потрібен виклик команди **readlib** (**extrema**).

Приклад:

```
readlib(extrema):
extrema(abs(x^2-2),x);
{2}
```

minimize(expr, vars, ranges), **maximize**(expr, vars, ranges)- обчислюють найменше та найбільше значення функції відповідно. Пошук може проводитися як на всій числовій прямій, так і на заданому відрізку (*ranges* - інтервал, у межах якого

змінюється *vars*). Для функцій декількох змінних обчислюються максимуми і мінімуми за окремими змінними.

Приклад:

```
minimize(x^2+y^2+3);
                3
minimize(x^2+y^2,{x});
                y2
minimize(x^2+y^2,{x,y},{x=-10..10,y=10..20});
                0
```

Симплекс - засіб

Окремою задачею знаходження коренів є засоби оптимізації. Один із засобів лінійної оптимізації, симплексний, реалізовано в Maple у вигляді пакета **simplex.**

Функції бібліотеки simplex

maximize (f, c, vartype) - обчислює максимальне значення при заданих умовах. f - лінійна функція, яку потрібно максимізувати; c - список лінійних обмежень. Функція повертає множину значень, систему лінійних рівнянь, пусту множину або **NUL**, якщо кількість розв'язків необмежена.

minimize (f, c, vartype) - обчислює мінімальне значення; використання аналогічно до попереднього випадку.

Приклад:

```
with(simplex):
minimize(x-y,{4x+3y=5,3x+4y<=4});
        {x=8/7, y=1/7}
```

display(c) відображає набір рівнянь або нерівностей в матричній формі.

```
display({x+3y+z<=0, w-2y-z<=2});
```

$$\begin{bmatrix} 1 & 3 & 1 & 0 \\ 0 & -2 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \le \begin{bmatrix} 0 \\ 2 \end{bmatrix};$$

standartize(c) - перекладає набір лінійних рівнянь у лінійні нерівності.

```
standartize({3x+4y<=4,4x+3y=5});
    {3x+4y<=4, 4x+3y<=5, -4x-3y<=-5}
```

Приклад. Розв'язання задач лінійного програмування.

На меблевій фабриці із стандартних листів фанери необхідно вирізати заготовки трьох видів у кількості 24, 31 і 18 штук відповідно. Кожний лист фанери може бути розрізаний на заготовку двома способами. Кількість одержуваних заготовок при даному способі розкрою наведена в таблиці 9.10. У ній же зазначені розміри відходів, що будуть отримані при даному способі розкрою одного листа фанери.

Таблиця 9.10 - Кількість заготовок

Вид заготовки	***Кількість заготовок, шт., при розкрою способом***	
	1	***2***
1	***2***	***6***
2	***5***	***4***
3	***2***	***3***
Величина відходів ($см^3$)	***12***	***16***

Визначити, скільки листів фанери і яким способом слід розкроїти так, щоб було отримано не менше від потрібної кількості заготовок при мінімальних відходах.

У математичній формі наша задача має вигляд:

цільова функція $f=12x+16y \rightarrow \min$,

де f - відходи, $см^3$;

x - кількість листів фанери, які потрібно розкроїти першим способом;

y - кількість листів фанери, які потрібно розкроїти другим способом;

обмеження:

$2x+6y=24$,
$5x+4y=31$,
$2x+3y=19$.

Розв'язати дану задачу можна за допомогою функції *minimize*. Присвоюємо змінній f вираз нашої функції $f=12x+16y$, а змінній c - систему обмежень $\{2x+6y=24, 5x+4y=31, 2x+3y=18\}$. Підключаємо бібліотеку simplex і викликаємо функцію minimize(f, c). У результаті отримали розв'язок $x=3$, тобто три листи треба розкроїти першим способом, $y=4$ - чотири листи треба розкроїти другим способом, $f=100$- мінімальне значення відходів. Результати роботи пакета Maple показані на рисунку 9.13.

Значна кількість алгоритмів оптимізації на сьогодні реалізована програмно за допомогою різних засобів. Більшість математичних пакетів програмного забезпечення містить процедури, що реалізують той чи інший оптимізаційний алгоритм. У даному виданні ми розглянули тільки деякі з них. Оглядово з певними розробками, поданими в Інтернет, можна ознайомитися в додатку Ж.

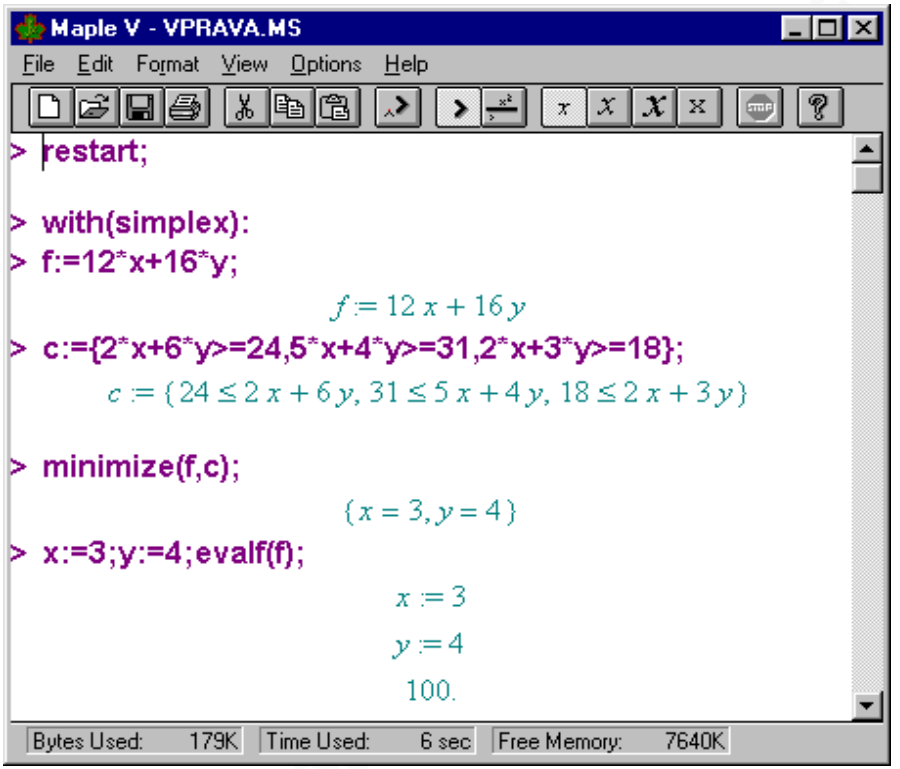


Рисунок 9.13 - Результати роботи пакета Maple

ТЕМА 10 ЗАДАЧІ ТА ЇХ РОЗВ'ЯЗАННЯ ЗА ДОПОМОГОЮ РІЗНИХ ПІДХОДІВ

10.1 ЗАДАЧА ПРО ЛІСОВУ ПОЖЕЖУ

Лісова пожежа поширюється у вузькій долині шириною 2 км зі швидкістю 32 м/хв (рис.10.1). Затримати поширення вогню можна шляхом побудови загороджувальної перегородки, що перетинає ліс по всій ширині долини. Один робітник може

побудувати 2 м перегородки в хвилину. Витрати на транспортування кожного робітника до місця подій і назад складають 20 грн., оплата праці кожного робітника складає 6 грн. за годину. Вартість 1 км квадратного лісу дорівнює 2000 грн. Скільки робітників варто послати на боротьбу з вогнем, щоб повні витрати були мінімальні?

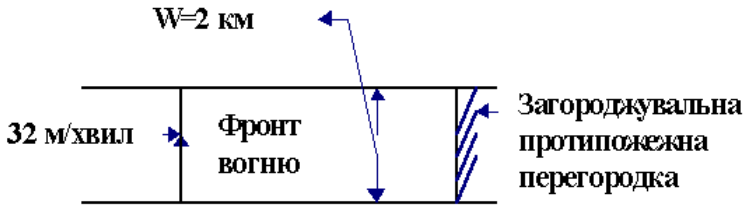


Рисунок 10.1 - Схема до прикладу

Складемо математичну модель даної задачі. Для цього виразимо повні витрати через складові

* вартість перевезення робітників - $C_п = 20x$, грн.;
* вартість будівлі перегородки - $C_{сп} = 6xt/60$, грн.,

де t - час будівлі перегородки, $t=2000/2x$, м;

* площа фронту пожежі - $S = 32*2000t$, м^2;
* вартість 1 м^2 лісу - $C_л = 2*10^{-3}$ грн/м^2.

Повні витрати (загальна вартість): $C_{пв} = C_п+C_{сп}+C_л*S$;

$$C_{пв} = 20x + 0{,}1x * 1000/x + 32*2000*2*10^{-3} *1000/x;$$

$$C_{пв} = 20x + 100 + 128000/x \rightarrow \min.$$

Таблиця 10.1 – Результати розв'язання задачі методом "золотого" перетину

Номер досліду	Інтервал	Точність	x_{min}	$F(x_{min})$	Кількість обчислень
1	[1;100]	1e-5	79,9999	3300,00	34
2	[1;50]	1e-5	49,9999	3660,00	33

3	[1;1000]	1e-5	79,9999	3300,00	39

Таблиця 10.2 – Результати розв'язання задачі різними методами

Метод	Інтервал	Точність	x_{min}	$F(x_{min})$	Кількість обчислень
"Золотого" перетину	[1;50]	1e-5	49,9999	3660,00	33
Половинного ділення	[1;50]	1e-5	25,5000	5629,61	22
Квадратична інтерполяція	1*;1†	1e-5	79,9999	3300,00	1201
Хорд	[1;50]	1e-5	79,9999	3300,00	23
Ньютона-Раффсона	1‡	1e-5	79,9999	3300,00	15
Середньої точки	[1;100]	1e-5	80,0000	3300,00	21

10.2 ЗАДАЧА ПРО ЗНАХОДЖЕННЯ ОПТИМАЛЬНОЇ ВІДСТАНІ ДЛЯ ВЛУЧЕННЯ В ЦІЛЬ

Радар запеленгував ворожий літак, що летить на висоті 20,000 м зі швидкістю 500 м/с. Літак знаходиться якраз над мінометною установкою. Швидкість викиду міни дорівнює приблизно 300 м/с. Міни устатковані часовим механізмом, і якщо міна не влучила в літак, вона вибухає сама через установлений проміжок часу. Міна повинна вибухнути на якомога ближчій відстані від літака (рис.10.2). Визначити кут запуску та час

* Початкова точка.
† Значення кроку зміни *x*.
‡ Для цього методу потрібна лише одна точка – початкова.

вибуху міни, при яких вона завдає літаку найбільших ушкоджень. Чи влучить міна в літак?

Примітка. - Реальні характеристики військового обладнання відрізняються від наведених у задачі. Метою задачі є лише демонстрація роботи програми. Також для спрощення задачі міна вважається матеріальною точкою.

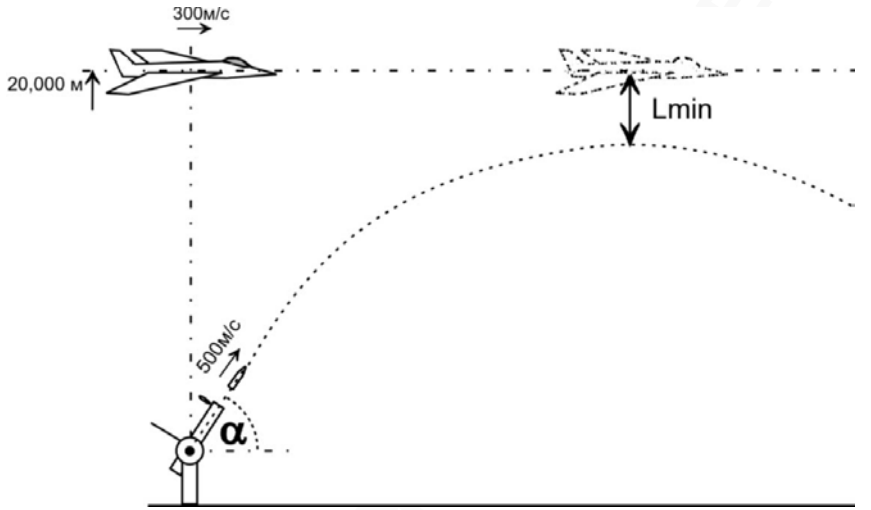


Рисунок 10.2 - Зображення траєкторії польоту міни

Розв'язання:

Мінометну установку візьмемо за початок системи відліку.

Координати міни залежно від часу будуть

$$x_м = v_м \cos \alpha \cdot t \ ,$$

$$y_м = v_м \sin \alpha \cdot t - \frac{gt^2}{2} \ .$$

Координати літака залежно від часу будуть

$$x_s = v_s t \,,$$

$$y_s = const.$$

Мінімізувати будемо функцію відстані між міною та літаком залежно від часу та кута запуску міни:

$$f(\alpha, t) = \sqrt{(x_m - x_s)^2 + (y_m - y_s)^2} =$$

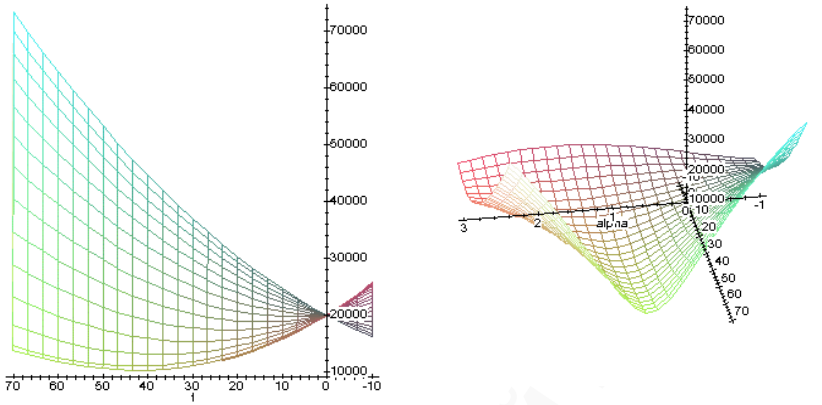
$$\sqrt{(v_m \cos\alpha \cdot t - v_s t)^2 + (v_m \sin\alpha \cdot t - \frac{gt^2}{2} - y_s)^2} = L \,.$$

Підставивши дані, отримаємо

$$f(\alpha, t) = \sqrt{(500\cos(\alpha)t - 300t)^2 + (500\sin(\alpha)t - \frac{9.8t^2}{2} - 20000)^2} \,. \qquad (*)$$

Остання функція і є функцією мінімізації. На рис.10.3 наведені графіки цієї функції в двох різних ракурсах, з яких можна зробити припущення про такі її властивості:

- функція має мінімум на інтервалі $1<a<2$ та $30<t<50$;
- значення функції в мінімумі знаходиться близько $1 \cdot 10^4$;
- у точці $\alpha=0,\ t=0$ функція має точку перегину.



а) б)

Рисунок 10.3 - Графіки функції (*), збудовані за допомогою програми Maple

Знаходження мінімуму функції за допомогою методу ДФП

Для мінімізації функції за допомогою програми з використанням методу ДФП візьмемо такі початкові дані:

а) функцію мінімізації (*);

б) за початкову точку візьмемо α=0,1; t=0 ;

в) максимальне значення градієнта 0.00000001 та максимальне значення відстані між суміжними точками приближення 0,00000005.

Функція мінімізується за 7 ітерацій. Остання ітерація виводить такі дані:

```
"Ітерація 7
```

F(1,15548007807752; 42.2617216740335)=10292,104485981"

Тобто оптимальний кут запуску міни дорівнює 1,15548007807752 ($\approx 66{,}2^0$), а час, за який вона наближається до літака на найменшу відстань, дорівнює 42,2617216740335 с.

Найменша відстань дорівнює 10292,104485981 м, що є досить великою відстанню, щоб як-небудь пошкодити літак.

Влучати в ціль міна почне, якщо літак летить на висоті до ≈8160 м (знаходиться легко за декілька спроб). Підставивши в формулу значення висоти польоту літака 8160 м замість 20000 м та виконавши розрахунок, одержимо останню (25) ітерацію:

```
"Ітерація 25
```

F(0,927295218001539; 39,9999999458919)=4,3374950304E-7"

Тобто оптимальний кут дорівнює 0,927295218001539 (≈53,13°), а час, за який міна влучить в літак, буде 39,9999999458919 с.

10.3 ЗАДАЧА НЕЛІНІЙНОГО ПРОГРАМУВАННЯ "ПЛАН ВИПУСКУ МЕБЛІВ"

Підприємство може випускати два види корпусних меблів. На їх виготовлення використовується деревина трьох видів. Запаси деревини на підприємстві, норми їхньої витрати a_{ij} (i=1,2,3; j=1,2), собівартість c_j і оптові ціни зазначені в таблиці 10.3. Через брак у процесі виробництва витрачання деревини залежить від об'єму x_j виробництва виробів і в першому наближенні виражається лінійною функцією $a_{ij} + x_j$, а собівартість продукції - функцією $c_j + 0{,}1x_j$. Вироби можуть випускатися в будь-яких співвідношеннях, тому що збут забезпечений. Підприємство зобов'язано з метою вивчення попиту населення випустити не менше двох комплектів кожного виду меблів. Скласти план випуску виробів, що забезпечує одержання максимального прибутку.

Таблиця 10.3 – Вхідні дані задачі

Порода	Запас сировини, м$_3$	Норми витрат на виріб виду	
		1	2
Сосна	100	10	20
Береза	120	20	20
Дуб	150	20	20
Собівартість, млн. грн.		5	10
Ціна, млн. грн.		7	13

Постановка задачі

1 За показник ефективності доцільно взяти прибуток підприємства за операцію (у млн. грн.).

2 За керовані змінні задачі слід взяти:

x_1 - кількість комплектів корпусних меблів 1;

x_2 - кількість комплектів корпусних меблів 2.

3 Цільова функція

$$W = [7 - (5+0,1x_1)]x_1 + [13 - (10+0,1x_2)]\, x_2 \rightarrow max,$$

або

$$W = 2x_1 - 0,1x_1^2 + 3x_2 - 0,1x_2^2 \rightarrow max.$$

4 Обмеження:

4.1 Щодо використання сосни, м^3:

$$(10+x_1)x_1 + (20+x_2)x_2 \leq 100.$$

4.2 Щодо використання берези, м^3:

$$(20+x_1)x_1 + (20+x_2)x_2 \leq 120.$$

4.3 Щодо використання дуба, м^3:

$$(20+x_1)x_1 + (20+x_2)x_2 \leq 150.$$

4.4 Зобов'язання за контрактом, шт.:

$$x_1 \geq 2.$$

4.5. Обласні обмеження:

$$x_2 \geq 2.$$

Задача зводиться до знаходження невід'ємних x_1^* і x_2^*, що задовольняють нелінійні обмеження і дозволяють знайти максимум нелінійної цільової функції.

10.4 ЗАДАЧА ОПТИМАЛЬНОГО ПРОЕКТУВАННЯ ДИСКА ТУРБІНИ

Необхідно спроектувати диск турбіни з такими заданими властивостями:

- маса диска W повинна бути мінімальна;
- повинна задовольняти вимоги щодо міцності

$$\max\{|\sigma_r-\sigma_\theta|,|\sigma_r|,|\sigma_\theta|\} <= 2\sigma_0,$$

де σ_0 - гранично допустимий тиск; σ_t і σ_θ - радіальний і тангенціальний тиск відповідно.

Задано обмеження на межі зміни товщини диска. Для її розв'язання використовувати метод випадкового пошуку і метод штрафних функцій.

Постановка задачі оптимального проектування

Диск турбіни можна розглядати як обертовий круглий диск змінної товщини (рис.10.4). Маса такого диска виражається за допомогою формули

$$W(r) = \int_{a1}^{am} 2\pi\rho\, rh(r)\,dr \qquad , \qquad (10.4.1)$$

де a_1 і a_m - внутрішній та зовнішній радіуси відповідно; p - щільність; h(r) - товщина диска на відстані r від осі обертання.

Мінімум досягається добором функції h (r). Однак цю функцію не можна змінювати довільно, тому що повинні бути

виконані вимоги до міцності диска при впливі сил, що виникають під час його обертання.

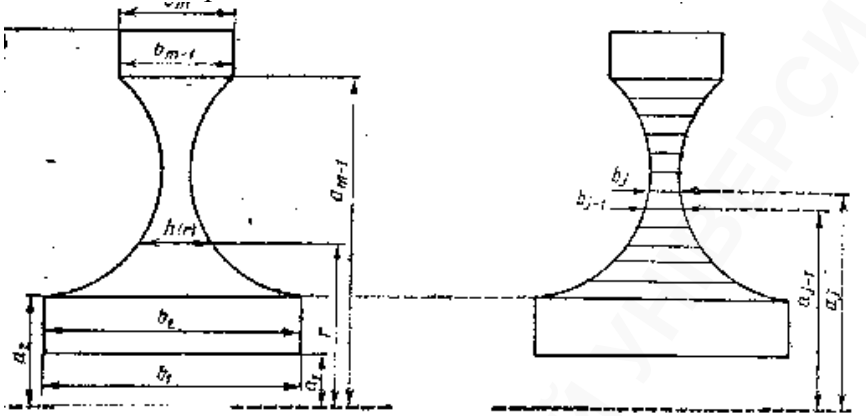


Рисунок 10.4- Поперечний переріз диска парової турбіни і його кусково-лінійна апроксимація

Рівняння рівноваги сил для обертового диска має вигляд

$$\frac{d}{dr}(h\sigma_r)+\frac{h}{r}(\sigma_r-\sigma_\theta)+\rho\omega^2 rh=0 \quad , \qquad (10.4.2)$$

де σ_t і σ_θ - радіальний і тангенціальний тиск відповідно, а ω - частота обертання диска.

Це рівняння справедливе в припущенні радіальної симетрії сил у площині, ортогональній до осі обертання.

Сили можуть бути виражені через розмір радіального зміщення *u(r)* за допомогою таких співвідношень:

$$\sigma_r=\frac{E}{1-\nu^2}(e_r+\nu e_\theta);\sigma_\theta=\frac{E}{1-\nu^2}(e_\theta+\nu e_r) \quad , \qquad (10.4.3)$$

$$e_r=du/dr;\ e_\theta=u/r, \qquad (10.4.4)$$

де e_r і e_θ - радіальна і тангенціальна деформації; E- модуль Юнга і ν - відношення Пуассона.

Підставляючи рівняння (10.4.3) і (10.4.4) у (10.4.2), одержуємо

$$\frac{d^2u}{dr^2}+(\frac{1}{r}+\frac{1}{h}\frac{dh}{dr})\frac{du}{dr}-(\frac{1}{r}-\frac{\nu}{h}\frac{dh}{dr})\frac{u}{r}+\frac{\rho\omega^2(1-\nu^2)}{E}r=0 \quad . \quad (10.4.5)$$

Звідси випливає, що для визначення *u(r)* необхідно знати геометрію диска, тобто *h=h(r)*, $a_1<=r<=a_m$. У цьому випадку *u(r)* визначиться як розв'язок крайової задачі для рівняння (10.4.5) із граничними умовами

$$\sigma_r|_{r=a1}=s_1 \text{ и } \sigma_r|_{r=am}=s_m \quad . \quad (10.4.6)$$

У результаті розв'язання крайової задачі і подальшого застосування формул (10.4.3) і (10.4.4) можна визначити тиски σ_t і σ_θ.

Крім умови рівноваги сил при обертанні диска, повинні задовольнятися вимоги щодо міцності. Відповідне обмеження має вигляд

$$\max\{|\sigma_r-\sigma_\theta|,|\sigma_r|,|\sigma_\theta|\}<=2\sigma_0, \quad (10.4.7)$$

де σ_0 - гранично допустимий тиск.

Нарешті, обмеження на межі зміни товщини диска задані таким способом:

$$h(r)\begin{cases}=b_1 & a_1\le r\le a_2\ ,\\ \ge\varepsilon \ \text{ при } & a_2\le r\le a_{m-1}\ ,\\ =b_m & a_{m-1}\le r\le a_m\ ,\end{cases} \quad (10.4.8)$$

де ε - нижня межа товщини, задана із технологічних розумінь; a_m-a_{m-1}=const.

Отримана в результаті задача мінімізації виразу (10.4.1) при обмеженнях (10.4.5)-(10.4.8) може бути розв'язана методами оптимального проектування.

Опис алгоритму

Значення функцій обмежень і критерію оптимальності цілком визначені, якщо задана залежність *h(r)*. Щоб подати задачу оптимального проектування як задачу варіювання

багатовимірного вектора конструктивних параметрів, проводиться апроксимація залежності *h(r)* деякою кусково-лінійною функцією. З цією метою здійснюється розбиття радіуса диска на *m* частин:

$$a_1 < a_2 < \ldots < a_{m-1} < a_m. \qquad (10.4.9)$$

На кожному відрізку будується лінійна функція, що апроксимує відповідну ділянку зміни *h(r)* (див. рис. 10.4).

Якщо товщина диска задана в точках розбиття розмірами

$$h(a_j)=b_j,\ j=1,2,\ldots,m, \qquad (10.4.10)$$

то кусково-лінійна апроксимація цілком визначена формулою

$$h(r) \approx \begin{cases} b_1 \quad при\ a_1 \le r \le a_2, \\ b_{j-1} + \left(\dfrac{b_j - b_{j-1}}{a_j - a_{j-1}}\right)(r - a_{j-1}) \quad при\ a_{j-1} \le r \le a_j, \\ b_m \quad при\ a_{m-1} \le r \le a_m. \end{cases} \quad j=3,\ldots m-1, \qquad (10.4.11)$$

Підставляючи формулу (10.4.11) у критерій оптимальності (10.4.1), одержуємо такий вираз:

$$W = \frac{1}{3}\pi\rho\left\{b_1\left(-3a_1^2 + a_2^2 + a_3^2 + a_2a_3\right) + \right.$$

$$+ \sum_{j=3}^{m-2} b_j\left(a_{j+1} - a_{j-1}\right)\left(a_{j+1} + a_j + a_{j-1}\right) + \qquad (10.4.12)$$

$$\left. + b_m\left(3a_m^2 - a_{m-1}^2 - a_{m-2}^2 - a_{m-1}a_{m-2}\right)\right\}.$$

Таким чином, критерій оптимальності виражається як нелінійна функція кінцевого числа змінних, що визначають геометрію диска.

Задача розбиття диска фіксованими точками a_1, a_3, a_4,..., a_m і змінною a_2 дає можливість перейти до такої постановки задачі оптимального проектування диска.

Припустимо, що значення $b_1 = b_2$ і $b_m = b_{m-1}$ фіксовані. Тоді інші проектовані параметри складають вектор $x = (b_3, b_4,..., b_{m-2}, a_2)$ розмірності $m-3$.

Обмеження на геометрію диска є наслідком (10.4.8) і записуються у вигляді

$$b_j \geq \varepsilon_1,\ j = 3...m-2,\ a_1 + \varepsilon_3 \leq a_2 \leq a_3 - \varepsilon_2, \qquad (10.4.13)$$

де ε_1, ε_2, ε_3 - задані розміри. У векторній формі ці обмеження можуть бути подані нерівністю

$$l \leq x \leq u, \qquad (10.4.14)$$

де $l^t = (\varepsilon_1,..., \varepsilon_1, a_1 + \varepsilon_3)$ та $u^t = (\infty,..., \infty, a_3 - \varepsilon_2)$.

Щоб виразити обмеження (10.4.5) відповідно до прийнятого розбиття, запишемо його окремо для кожної ділянки. При цьому будемо брати розбиття таким, що зміною $h(r)$ усередині кожної ділянки можна знехтувати.

Рівняння (10.4.5) для кожної ділянки набуває більш простої форми:

$$\frac{d^2u}{dr^2} + \frac{1}{r}\frac{du}{dr} - \frac{u}{r^2} + \frac{\rho\omega^2(1-\nu^2)}{E}r = 0 \quad ,$$

що дозволяє одержати загальний розв'язок у вигляді

$$u = c_1 r + \frac{c_2}{r} - \frac{\rho\omega^2(1-\nu^2)}{8E} \quad ,$$

де c_1 і c_2 - сталі, що одержуються таким методом:

1 У початковій точці

$$e_{r\,\text{поч}} = e_{\theta\,\text{поч}} = \frac{\tau_0}{E}(1-\nu),$$

$$e_{r\,\text{поч}} = \left.\frac{du}{dr}\right|_{r=r\ \text{поч}} = c_1 - \frac{c_2}{r^2_{\text{поч}}} = \frac{2*10^7}{E}(1-\nu),$$

$$e_{\theta\,\text{поч}} = \frac{U}{r} = c_1 + \frac{c_2}{r^2_{\text{поч}}} - \frac{\rho\omega^2(1-\nu^2)}{r_{\text{поч}}\ 8E} = \frac{2*10^7}{E}(1-\nu).$$

2 Нехай $r_{поч}$ = a[1]; E= $2*10^{10}$; ν=0,3.

3 Тоді прирівнявши $e_{\theta поч}$ та $e_{r\ поч}$ і підставивши числові значення, одержимо

$$\begin{cases} c_1 - \dfrac{c_2}{0{,}12} = 0{,}7*10^{-3}, \\ c_1 + \dfrac{c_2}{0{,}12} - 1{,}4*10^{-3} = 0{,}7*10^{-3}, \end{cases}$$

отже, c_1=1,4*10^{-3} і c_2=0,084*10^{-3}.

Звідси за допомогою (10.4.3) і (10.4.4) одержуємо вираз для сил обертання

$$\sigma_r = \frac{E}{1-\nu^2}\left(c_1(1+\nu) - \frac{c_2}{r_2}(1-\nu) - \frac{\nu\rho\omega^2(1-\nu^2)}{r8E}\right),$$

$$\sigma_\theta = \frac{E}{1-\nu^2}\left(c_1(1+\nu) + \frac{c_2}{r_2}(1-\nu) - \frac{\rho\omega^2(1-\nu^2)}{r8E}\right). \qquad (10.4.15)$$

Для проектованої конструкції визначаються також сили σ_r і σ_θ у точках $r_1,r_2,...,r_n$. Обмеження (10.4.7) записується окремо для кожного інтервалу $[r_{j-1},r_j]$. Маємо

$$\tau_1 = \frac{1}{2}|\sigma_r - \sigma_\theta|, \quad \tau_2 = \frac{1}{2}|\sigma_r|, \quad \tau_3 = \frac{1}{2}|\sigma_\theta|,$$

звідки через (10.4.7) одержуємо

$$\max\{\tau_1, \tau_2, \tau_3\} \le \tau_0 \qquad (10.4.16)$$

Вектор функцій обмежень може бути записаний у вигляді

$$y^t = \left(\tau_{r_1}, \tau_{r_2}, ..., \tau_{r_n}\right)$$

Відповідно до умови (10.4.16) межі зміни цього вектора записуються за допомогою нерівності

$$L \le y(x) \le U, \qquad (10.4.17)$$

де L=(0, 0,..., 0) і U=(τ_0, τ_0, τ_0).

Таким чином, завдання оптимального проектування диска парової турбіни зводиться до пошуку мінімуму критерію оптимальності (10.4.12) при обмеженнях (10.4.16) і (10.4.17) шляхом варіювання вектора $x=(b_3,..., b_{n-2}, a_2)$.

У курсі наведена програма, що дозволяє знайти оптимальні значення параметрів профілю диска турбіни.

Для перевірки працездатності програми взяті такі вихідні дані:

ω = 150 (число оборотів диска в секунду) рад/с;

ρ = 7800 (щільність металу) кг/м^3;

σ_0 = 20*10^6(гранична напруга) кг/м^2;

E = 20,6*10^9 (модуль Юнга) кг/м^2.

Отримані такі результати:

1 При обчисленні оптимального профілю диска методом ***випадкового пошуку*** (рис.10.4.1):

Число ітерацій 500 000.

Час обчислення 12 хв.

Вага диска W = 789,5 кг.

2 При обчисленні оптимального профілю диска методом ***штрафних функцій*** (рис.10.4.2):

Час обчислення 8 хв.
Вага диска W = 772,2 кг.

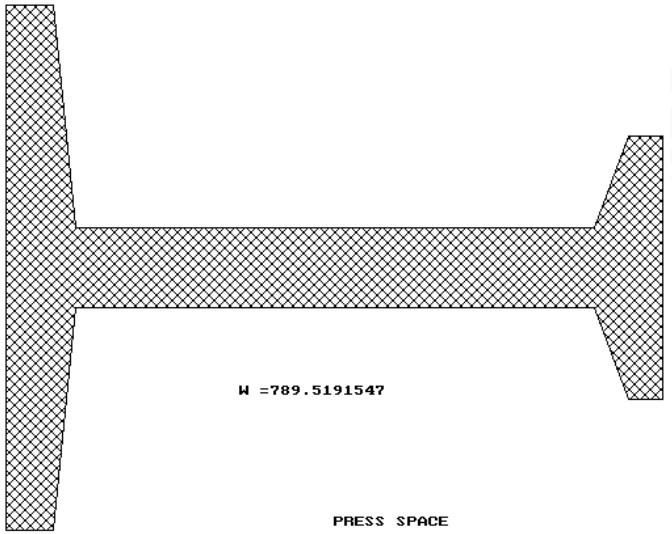


Рисунок 10.4.1 - Профіль диска, отриманий методом випадкового пошуку

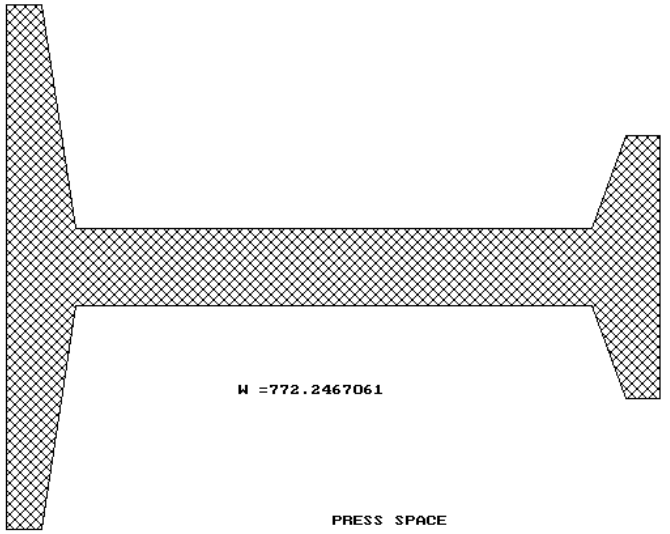


Рисунок 10.4.2 - Профіль диска, отриманий методом штрафних функцій

3 При обчисленні оптимального профілю диска методом ***випадкового пошуку з подальшим застосуванням методу штрафних функцій*** (рис.10.4.3):

Число ітерацій 5 000.

Час обчислення 3 хв.

Вага диска W = 787,0 кг.

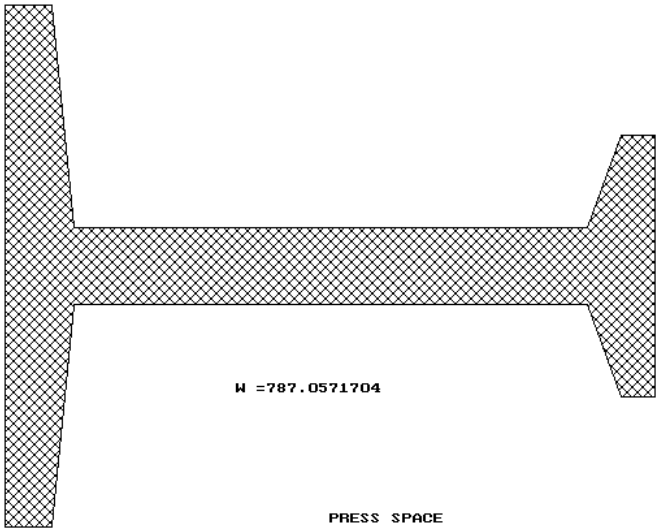


Рисунок 10.4.3 - Профіль диска, отриманий методом випадкового пошуку з подальшим застосуванням методу штрафних функцій

З отриманих результатів простежується, що метод випадкового пошуку цілком можна використати для розв'язання поставленої задачі багатопараметричної оптимізації. Він дає досить точний результат, хоча при цьому витрачається багато часу (це може істотно позначитися при розв'язанні дійсно складних задач подібного класу). Також бачимо, що якщо

враховувати витрати часу, то найбільш ефективним методом розв'язання даної задачі при прийнятній точності отриманого результату є метод випадкового пошуку з невеликою кількістю ітерацій для входження в область оптимальних значень цільової функції та подальше застосування методу штрафних функцій для уточнення результату.

10.5 ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

10.5.1 ЗАДАЧА ПРО ДОБОВИЙ ПЛАН ВИРОБНИЦТВА

Цех випускає три види деталей, що виготовляються на трьох верстатах. На рис.10.5.1 показана технологічна схема виготовлення деталі кожного виду із зазначенням часу, потрібного для її оброблення на верстатах. Добовий ресурс робочого часу верстатів 1, 2, 3 складає відповідно 890, 920 і 840 хвилин. Вартість однієї деталі вигляду 1, 2 і 3 дорівнює відповідно 3, 1 і 2 грн.

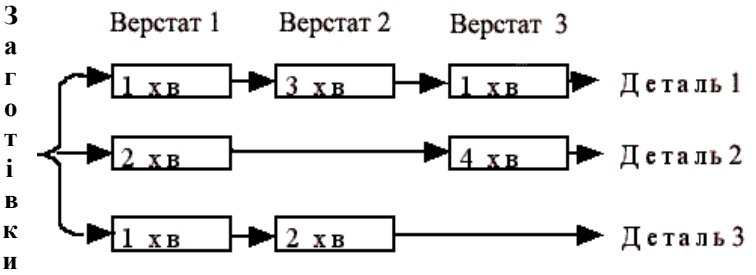


Рисунок 10.5.1 - Технологічна схема виготовлення деталей

Потрібно скласти добовий план виробництва з метою максимізації вартості випущеної продукції.

Розв'язання

Виразимо через x_{ij} об'єм випуску j-ї деталі. Тоді одержуємо задачу

$$3x_1+x_2+2x_3 \to max,$$
$$x_1+2x_2+x_3 \le 890,$$
$$3x_1+2x_3 \le 920,$$
$$x_1+4x_2 \le 840,$$
$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0.$$

Її розв'язок: $x_1^* = 0$, $x_2^* = 210$, $x_3^* = 460$.

В таблицях 10.5.1 – 10.5.2 наведений один із способів запису умов задачі для подальшої роботи з нею за допомогою Excel.

Таблиця 10.5.1 – Умова задачі в аналітичному вигляді

	A	B	C	D	E	F
1	=0,1	=3*A1		=A1	=3*A1	=A1
2	=211	=A2		=2*A2	=2*A3	=4*A2
3	=462	=2*A3		=A3	=СУММ(E1:E2)	=СУММ(F1:F2)
4		=СУММ(B1:B3)		=СУММ(D1:D3)		

Таблиця 10.5.2 – Умова задачі в числовому вигляді

	A	B	C	D	E	F
1	0,1	0,3		0,1	0,3	0,1
2	211	211		422	924	844
3	462	924		462	924,3	844,1
4		1135,3 {цільова функція}		884,1		

Таблиця 10.5.3 – Результати виконання процедури "Пошук розв'язку"

	A	B	C	D	E	F
1	0	0		0	0	0
2	210	210		420	920	840
3	460	920		460	920	840
4		1130		880		

Отже, отримано такі результати: $x_1=0$, $x_2=210$, $x_3=460$, f=1130.

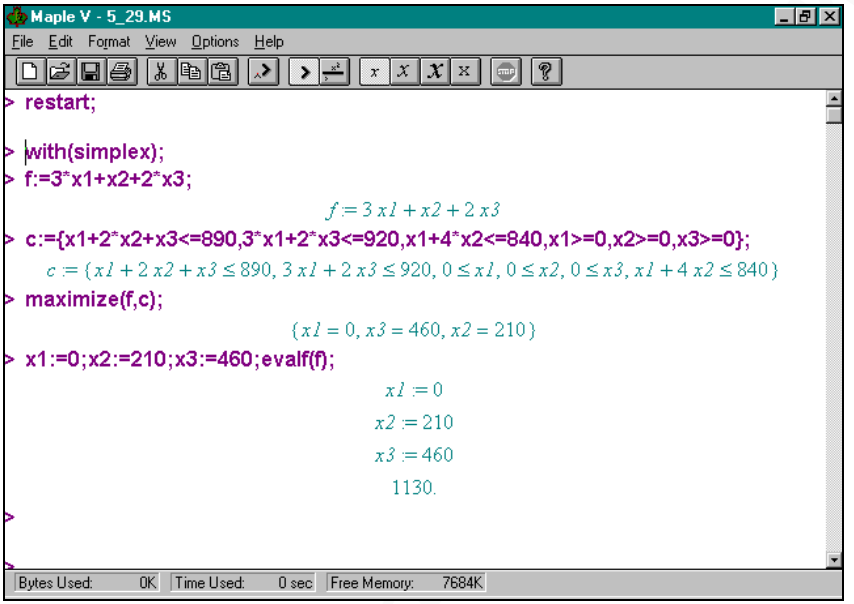


Рисунок 10.5.2 – Розв'язання задачі за допомогою Maple

10.5.2 ЗАДАЧА ПРО ОДЕРЖАННЯ ДЕШЕВОГО СПЛАВУ МЕТАЛІВ

У металургійний цех подається латунь (сплав міді з цинком) чотирьох типів із вмістом цинку 10, 20, 25 і 40% за ціною 10, 30, 40 і 60 грн. за 1 кг відповідно. У яких пропорціях потрібно переплавляти цю сировину в цеху, щоб одержати сплав (латунь), що містить 30% цинку і при цьому є найдешевшим?

Розв'язання

Виразимо через x_{ij} масу, кг, j-го типу сировини, що використовується для одержання 1 кг необхідного сплаву. Тоді поставлене завдання можна формалізувати у такий спосіб:

$$10x_1+30x_2+40x_3+60x_4 \to min,$$
$$10x_1+20x_2+25x_3+40x_4= 30,$$
$$x_1+x_2+x_3+x_4= 1,$$
$$x_j\ 0,\ j = 1,...,4.$$

Її розв'язок x*=(1/3,0, 0, 2/3). Таким чином, найдешевшим є сплав першого і четвертого типів сировини у відношенні 1:2.

При розв'язанні симплекс-методом одержуємо: x_1=0,33; x_2=0; x_3=0; x_4=0,67; f=43,33.

Запишемо умову задачі стосовно Excel.

	A	B	C	D	E
1	=1/3	=10*A1	Ограничения:	=10*A1	=A1
2	=0	=30*A2		=20*A2	=A2
3	=0	=40*A3		=25*A3	=A3
4	=2/3	=60*A4		=40*A4	=A4
5	*ЦФ*	=СУММ(B1:B4)		=СУММ(D1:D4)	=СУММ(E1:E4)

Рисунок 10.5.3 – Умова задачі в аналітичному вигляді

	A	B	C	D	E
1	0.322581	3.225806	Ограничения:	3.225806	0.322581
2	0	0		0	0
3	0	0		0	0
4	0.645161	38.70968		25.80645	0.645161
5	*ЦФ*	41.93548		29.03226	0.967742

Рисунок 10.5.4 – Умова задачі в числовому вигляді

При використанні пакету Excel і його процедури ***"Пошук розв'язку"*** отримано

$$x_1=0,3333;\ x_2=0;\ x_3=0;\ x_4=0,6667;\ f=43,333333$$

	A	B	C	D	E
1	0.333333	3.333333		0.333333	3.333333
2	0	0		0	0
3	0	0		0	0
4	0.666667	40		0.666667	26.66667
5	*ЦФ*	43.33333		1	30

Рисунок 10.5.5 – Результати пошуку розв'язків

10.5.3 ЗАДАЧА ПРО РОЗРАХУНОК ЕЛЕКТРИЧНОЇ СИСТЕМИ

Потрібно забезпечити стабілізацію напруги у межах від $U_{н1}$=36 В до $U_{н2}$=40 В при навантаженні $R_н$=400 Ом на схемі стабілізатора (рис.10.5.6), якщо вхідна напруга $U_{вх}$ змінюється від $U_{вх1}$=40 В до $U_{вх2}$=48 В. Передбачається використовувати стабілізатор із параметрами U_0=39 У, $r_{ст}$=18 Ом, i_0=0,01 А, $i_м$=0,15 А.

Потрібно розрахувати опір резисторів R_0 і R_1, якщо за критерій ефективності роботи схеми прийнята мінімальна потужність, що розсіюється на цих резисторах.

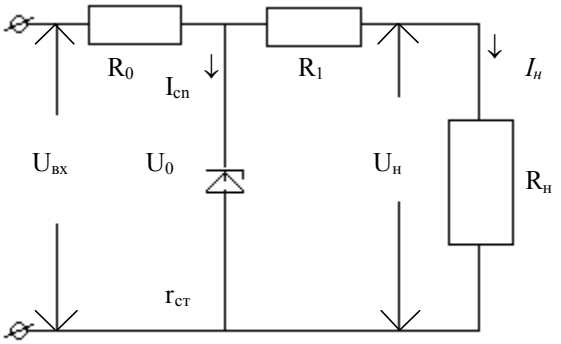


Рисунок 10.5.6 - Схема стабілізатора

Напишемо рівняння Кірхгофа для контурів:

$$(U_0 \to r_{ст}) \to R_1 \to R_н \;\; і \;\; U_{вх} \to R_0 \to R_1 \to R_н,$$

$$U_0 + i_0 r_{ст} - I_н(R_0 + R_1) = 0,$$

$$U_{вх} - R_0(i_{ст} + I_н) - I_н(R_1 + R_н) = 0.$$

Розв'яжемо рівняння відносно R_1 і R_0:

$$R_1 = (U_0 + i_{ст} r_{ст} - I_н R_н) / I_н = (U_0 + i_{ст} r_{ст} - U_н) R_н / U_{н1},$$

$$R_0 = [U_{вх} - I_н(R_1 + R_н)] / (i_{ст} + I_н) = [U_{вх} - U_н(R_1 / R_н + 1)] / (i_{ст} + U_н / R_н).$$

Запишемо отримані рівняння для двох граничних значень живильної напруги $U_{вх}$:

$$R_{1\min} = (U_0 + i_0 r_{ст} - U_{н1}) R_н / U_{н1},$$
$$R_{1\max} = (U_0 + i_m r_{ст} - U_{н2}) R_н / U_{н2},$$
$$R_{0\min} = [U_{вх1} - U_{н1}(R_1 / R_н + 1)] / (i_0 + U_{н1} / R_н),$$
$$R_{0\max} = [U_{вх2} - U_{н2}(R_1 / R_н + 1)] / (i_m + U_{н2} / R_н).$$

Підставимо в рівняння числові значення параметрів:

R_{1min}=(39+0,01*18-36)400/36=35,3 Ом,
R_{1max}=(39+0,15*18-40)400/40=17 Ом,
R_{0min}=[40-36(R_1/400 +1)]/(0,01+36/400) = 40-0,9R_1,
R_{0max}=[48-40(R_1/400+1)]/(0,15+40/400)= 32-0,4R_1

Тепер можна скласти обмеження для R_0 і R_1:

$R_1 \le 35,3$,
$R_1 \ge 17$,
$R_0 \ge 40-0,9R_1$,
$R_0 \le 32-0,4R_1$,
$R_1 \ge 0$, $R_0 \ge 0$.

Таким чином, якщо будуть виконуватися отримані нерівності, то схема буде забезпечувати стабілізацію напруги в заданих межах.

Обмеження пропонують деяку свободу вибору номінальних значень резисторів R_1 і R_0.

Відповідно до умови задачі критерієм якості роботи системи є мінімальна потужність P, що втрачається на резисторах R_1 і R_0:

$$P = (i_{\_\square} + I_{\cdot})^2 R_0 + I^{2\cdot} R_1 \to \min.$$

Приймемо для розрахунку середні значення струмів $i_{ст}$=0,15 А, I_H=0,1 А. Тоді рівняння цільової функції набуде вигляду

$$P = 0{,}0625R_0 + 0{,}01R_1 \to \min.$$

Таким чином, дана задача зведена до задачі лінійного програмування.

При застосуванні симплекс-методу для даної задачі одержимо:

Таблиця 10.5.4 – Симплекс-таблиця 1 прикладу 10.6

P_B	C_B	b	P_1	P_2	P_3	P_4	P_5	P_6
P_3	0	-40	-1	-0,9	1	0	0	0
← P_4	0	32	1	0,4	0	1	0	0
P_5	0	35,3	0	1	0	0	1	0
P_6	0	-17	0	-1	0	0	0	1
			0,0625	0,01	0	0	0	0

Таблиця 10.5.5 – Симплекс-таблиця 2 прикладу 10.6

P_B	C_B	b	P_1	P_2	P_3	P_4	P_5	P_6
P_3	0	-8	0	-0,5	1	1	0	0
P_1	0,0625	32	1	0,4	0	1	0	0
← P_5	0	35,3	0	1	0	0	1	0
P_6	0	-17	0	-1	0	0	0	1
			0	-0,015	0	-0,0625	0	0

Таблиця 10.5.6 – Симплекс-таблиця 3 прикладу 10.6

P_B	C_B	b	P_1	P_2	P_3	P_4	P_5	P_6
←P_3	0	965	0	0	1	1	0,5	0
P_1	-0,0625	1788	1	0	0	1	-0,4	0
P_2	-0,01	35,3	0	1	0	0	1	0
P_6	0	18,3	0	0	0	0	1	1
			0	0	0	-0,0625	0,025	0

Таблиця 10.5.7 – Симплекс-таблиця 4 прикладу 10.6

P_B	C_B	b	P_1	P_2	P_3	P_4	P_5	P_6
P_4	0	9,65	0	0	1	1	0,5	0
P_1	-0,0625	8,23	1	0	-1	0	-0,9	0
P_2	-0,01	35,3	0	1	0	0	1	0
P_6	0	8,3	0	0	0	0	1	1
			0	0	0,0625	0	0,046	0

Застосувавши пакет LP88, одержимо такий оптимальний розв'язок:

R_0= 8,23; R_1= 35,3; P=0,867375.

ВИСНОВКИ

Таким чином, зібрано воєдино програми для функції однієї змінної і декількох змінних. Ці програми пройшли тестування й усі вони зберігаються в електронному підручнику, що допоможе студентам опановувати знання з курсу «Комп'ютерна реалізація методів оптимізації» як на звичайних практичних заняттях, так і під час дистанційного навчання.

Одержавши навчальний матеріал в електронному вигляді з використанням телекомунікаційних мереж, студент може опановувати знання вдома, на робочому місці чи в спеціальному комп'ютерному класі в будь-якій точці України і зарубіжжя.

Звичайно, будучи одним з перших, цей курс не позбавлений деяких недоліків як з методичної точки зору, так і з технічної, але він демонструє явні переваги використання гіпертекстових технологій для навчальних систем.

Об'єднання методів та рекомендацій щодо використання розробленого програмного забезпечення, в якому реалізовані оптимізаційні алгоритми, в єдиний методичний посібник буде корисним широкому колу читачів: підприємцям, інженерам, економістам, фахівцям з математичного забезпечення ЕОМ, викладачам, аспірантам і студентам вищих навчальних закладів.

ДОДАТОК А

(довідковий)

ПЕРЕЛІК МЕТОДІВ, ОПИС ЯКИХ ПОДАНО У ДАНОМУ КУРСІ

1 Методи безумовної оптимізації

1.1 Методи одновимірної оптимізації:

- а) методи виключення інтервалів:
  - *половинного ділення;*
  - *"золотого" перетину;*
  - *Фібоначчі;*
- б) методи поліноміальної апроксимації;
- в) методи з використанням похідних:
  - *хорд;*
  - *дотичних;*
  - *середньої точки.*

1.2 Методи багатовимірної оптимізації:

а) методи нульового порядку:

- *покоординатного спуску;*
- *Хука-Дживса;*
- *симплексний метод Нелдера-Міда.*

б) методи першого порядку:

- *градієнтний покроковий;*
- *найшвидшого спуску (метод Коші);*
- *Ньютона;*
- *модифікований Ньютона;*
- *сполучених градієнтів:*
  - ⇒ *Давідона-Флетчера-Пауелла;*
  - ⇒ *Флетчера-Рівса.*

2 Методи умовної оптимізації:

- а) методи прямого пошуку:
  - *модифікований Хука-Дживса;*
  - *комплексів;*
  - *випадкового пошуку.*

- б) методи штрафних функцій:
  - *внутрішніх штрафних функцій;*
  - *зовнішніх штрафних функцій;*
  - *комбіновані штрафних функцій;*
  - *Фіако-Маккорміка;*
- в) методи лінеаризації:
  - *алгоритм Франка-Вульфа.*

3 Методи розв'язання задач лінійного програмування:

3.1 Графічний метод розв'язання задач лінійного програмування.

3.2 Симплекс-метод розв'язання задач лінійного програмування.

4 Методи оптимізації в пакетах прикладних програм:

4.1 Excel.

4.2 Maple.

4.3 LP88.

4.4 Met_Opt.

5 Методи оптимізації управлінських процесів:

- *принцип максимуму Понтрягіна;*
- *метод Рітца.*

6 Методи оптимізації на графах.

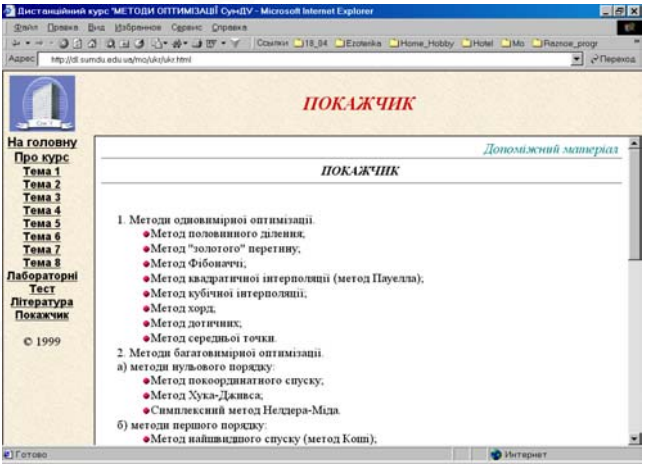


Рисунок А.1 – Вигляд "Покажчика" у вікні броузера

ДОДАТОК Б

(довідковий)

ПРОЦЕДУРИ МОДУЛЯ IDVVOD

Даний модуль містить процедури введення вихідних даних для методів оптимізації

```
{--Для методів одновимірної оптиміації--}

1.1 Для методу половинного ділення
procedure VvodIsx_PD(var a,b,e:real; var fil:text);
1.2 Для методу "золотого" перетину
procedure VvodIsx_GS(var a,b,e:real; var fil:text);
1.3 Для методу Фібоначчі
procedure VvodIsx_FC( var f:mas; var n:integer; var a,b,e:real; var fil:text);
1.4 Для квадратичної інтерполяції
procedure VvodIsx_Kv(var a,h,e:real; var fil:text);

{--Для методів із використанням похідних--}

1.5 Для методу хорд
procedure VvodIsx_Hord(var a,b,e:real; var fil:text);
1.6 Для методу середньої точки
procedure VvodIsx_MP(var a,b,e:real; var fil:text);
1.7 Для методу Ньютона-Раффсона
procedure VvodIsx_NR(var x,e:real; var fil:text);

{--Для методів прямого пошуку--}

2.1 Для методу Хука-Дживса
procedure VvodIsx_XD(n:integer; var x:mas; var h,e:real; var fil:text);
2.2 Для методу  покоординатного спуску
procedure VvodIsx_PS(n:integer; var x:mas; var d,d1,r:real; var m2:integer;
                     var fil_name:text);
2.3 Для симплексного методу Нелдера-Міда
procedure VvodIsx_NM(n:integer; var s:mass; var k,al,be,ga,eps:real; var fil:text);

{--Для градієнтних методів--}
2.4 Для методу найшвидшого спуску
procedure VvodIsx_NS(n:integer; var x:mas; var h,eps:real; var fil:text);
2.5 Для методу Давідона-Флетчера-Пауелла
procedure VvodIsx_DFP(n:integer; var x:mas; var eps:real; var fil:text);
```

Опис параметрів процедур введення вхідних даних для методів оптимізації

{--Для методів одновимірної оптимізації--}

1.1 Для методу половинного ділення
procedure VvodIsx_PD(var a,b,e:real; var fil:text);

a,b - межі інтервалу пошуку оптимуму;
e - точність (похибка) пошуку;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

1.2 Для методу "золотого" перетину
procedure VvodIsx_GS(var a,b,e:real; var fil:text);

a,b - межі інтервалу пошуку оптимуму;
e - точність (похибка) пошуку;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

1.3 Для методу Фібоначчі
procedure VvodIsx_FC(var f:mas; var n:integer; var a,b,e:real; var fil:text);

f - числа Фібоначчі;
n - кількість розбивок;
e - epsilon (точність);
a,b - інтервал (a,b);
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

1.4 Для квадратичної інтерполяції
procedure VvodIsx_Kv(var a,h,e:real; var fil:text);

a - початкове значення;
h - довжина кроку;
e - точність;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

{--Для методів із використанням похідних--}

1.5 Для методу хорд
procedure VvodIsx_Hord(var a,b,e:real; var fil:text);

a,b - межі інтервалу пошуку оптимуму;
e - точність (похибка) пошуку;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

1.6 Для методу середньої точки
procedure VvodIsx_MP(var a,b,e:real; var fil:text);

a,b - межі інтервалу пошуку оптимуму;
e - точність (похибка) пошуку;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

1.7 Для методу Ньютона-Раффсона
procedure VvodIsx_NR(var x,e:real; var fil:text);

x - початкове значення пошуку оптимуму;
e - точність (похибка) пошуку;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

{--Для методів прямого пошуку--}

2.1 Для методу Хука-Дживса
procedure VvodIsx_XD(n:integer; var x:mas; var h,e:real; var fil:text);

n - кількість змінних функції;
x - координати початкової точки;
h - довжина кроку h;
e - похибка;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

2.2 Для методу покоординатного спуску
procedure VvodIsx_PS(n:integer; var x:mas; var d,d1,r:real; var m2:integer;
var fil_name:text);
n - кількість змінних функції;
x - координати початкової точки;
d - довжина кроку;
r - коефіцієнт зміни кроку;
d1 - мінімальний розмір кроку;
m2 - максимальна кількість обчислень функції;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

2.3 Для симплексного методу Нелдера-Міда
procedure VvodIsx_NM(n:integer; var s:mass; var k,al,be,ga,eps:real; var fil:text);

n - кількість змінних функції;
s - початкове наближення;
k - довжина кроку;
al,be,ga - значення коефіцієнтів альфа, бета і гамма;
eps - точність;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

{--Для градієнтних методів--}
2.4 Для методу найшвидшого спуску
procedure VvodIsx_NS(n:integer; var x:mas; var h,eps:real; var fil:text);

n - кількість змінних функції;
x - координати початкової точки;
h - довжина кроку h;
e - похибка;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

2.5 Для методу Давідона-Флетчера-Пауелла
procedure VvodIsx_DFP(n:integer; var x:mas; var eps:real; var fil:text);

n - кількість змінних функції;
x - координати початкової точки;
e - похибка;
fil - змінна імені файлу, у який будуть виводитися результати пошуку.

ДОДАТОК В

(довідковий)

СПИСОК ПРОГРАМ

1 Програми, що реалізують методи одновимірної оптимізації:

- *програма, яка реалізує метод половинного ділення;*
- *програма, яка реалізує метод "золотого" перетину;*
- *програма, яка реалізує метод Фібоначчі;*
- *програма, яка реалізує метод квадратичної інтерполяції (метод Пауелла);*
- *програма, яка реалізує метод кубічної інтерполяції;*
- *програма, яка реалізує метод хорд;*
- *програма, яка реалізує метод дотичних;*
- *програма, яка реалізує метод середньої точки.*

2 Програми, що реалізують методи багатовимірної оптимізації:

а) програми, що реалізують методи нульового порядку:

- *програма, яка реалізує метод покоординатного спуску;*
- *програма, яка реалізує метод Хука-Дживса;*
- *програма, яка реалізує симплексний метод Нелдера-Міда;*

б) програми, що реалізують методи першого порядку:

- *програма, яка реалізує метод найшвидшого спуску (метод Коші);*
- *програма, яка реалізує метод Ньютона;*
- *програма, яка реалізує модифікований метод Ньютона;*
- *програма, яка реалізує метод Давідона-Флетчера-Пауелла;*
- *програма, яка реалізує метод Флетчера-Рівса.*

3 Програми, що реалізують методи умовної оптимізації;

- *програма, яка реалізує модифікований метод Хука-Дживса;*

- *програма, яка реалізує метод комплексів;*
- *програма, яка реалізує метод випадкового пошуку;*

-
- *програма, яка реалізує метод зовнішніх штрафних функцій;*
- *програма, яка реалізує комбінований метод штрафних функцій (метод Фіако-Маккорміка);*
- *програма, яка реалізує алгоритм Франка-Вульфа.*

4 Програми, що реалізують методи розв'язання задач лінійного програмування:

- *програма, яка реалізує графічний метод розв'язання задач лінійного програмування;*
- *програма, яка реалізує симплекс-метод розв'язання задач лінійного програмування;*
- *прикладний пакет LP88, що реалізує симплекс-метод розв'язання задач лінійного програмування.*

5 Програми, що реалізують методи оптимізації управлінських процесів:

- *програма, яка реалізує метод Рітца.*

6 Програми, що реалізують методи оптимізації на графах:

- *програма, яка дозволяє знайти найкоротший шлях у графі;*
- *програма, що дозволяє розв'язати задачу Прима-Краскала з різними вхідними даними.*

7 Допоміжні програми:

- *програма, яка дозволяє побудувати графік функції з однією змінною;*
- *програма, яка реалізує лабораторне заняття.*

ДОДАТОК Г

(довідковий)

ЗВІТИ LP 88

Одержання звітів - Generating Reports

OUTPUT MENU керує генеруванням звітів і аналізом чутливості. Натисніть функціональну клавішу F8 OUTPUT MENU (меню виводу). У рядку функціональної клавіші тепер з'явиться вибірка OUTPUT MENU (меню виводу).

Натисніть функціональну клавішу F1 (Primal Values) (Прямі значення). LP88 створить і роздрукує таблицю, яка являє собою основний розв'язок завдання-зразка. Ви можете одержати до 6 різних таблиць, натискаючи функціональні клавіші в будь-якому порядку, поки на екрані висвічується ім'я задачі. Коли закінчите, повертайтеся до MASTER MENU, натиснувши функціональну клавішу F9 (MASTER MENU).

Робота симплекс-алгоритму - To See How the Simplex Algorithm Works

LP88 дозволяє побачити, як симплекс-алгоритм додає і вилучає змінні доти, поки не знайдено оптимальний розв'язок або встановлено, що задача ЛП не може бути розв'язана. При бажанні Ви можете записати цей процес.

Щоб побачити, як працює алгоритм, Ви повинні знову розв'язати SAMPLE, тому натисніть клавішу F2 Solve Problem (Розв'язати задачу). Коли на екрані з'явиться список допусків і програмних запитів, дайте відповідь на запитання в 23-му рядку:

```
**************************************************
      * Input Control Number (<Enter> to Run)? 14 *
      * Input New Value? Yes *
      *=========================================*
      * Введене контрольне число (<Enter>, щоб виконати)? 14 *
      * Введення нового значення? Так *
**************************************************
```

LP88 видрукує рядок для обчислення кожної опорної точки, показуючи вхідні і вихідні змінні та іншу інформацію. Тепер натисніть клавішу <Enter>, щоб почати процес розв'язання SAMPLE. Ви можете "провести" LP88 через симплекс-алгоритм за допомогою функціональних клавіш, поки розв'язується SAMPLE. Для цього натисніть функціональну клавішу F2 Step/Continue (Крок/Продовження) відразу ж після того як LP88

почне розв'язання SAMPLE. LP88 зробить паузу після обчислення кожної опорної точки. Під час паузи ім'я задачі з'являється у верхньому лівому куті екрана в реверсивному кольорі. Для виконання обчислень тільки по одній опорній точці натисніть функціональну клавішу F1 Pause/Continue (Пауза/Продовження). Після того як LP88 виконає обчислення опорної точки, він знову зробить паузу. Ви повинні натискати функціональну клавішу F1 для кожної відправної точки доти, поки знову не натиснете F2. Коли Ви натиснете F2 вдруге, паузи припиняються і відновлюється безупинне виконання програми.

Закінчення сеансу - To End the Session

Натисніть функціональну клавішу F9 End Session (Кінець сеансу) у MASTER MENU. Вам запропонують верифікувати кінець сеансу:

```
**********************************************
    * Do You Want to End the Session (Y/N)? Y*
    *=========================================*
    * Ви хочете закінчити сеанс (Так/ні)? Так *
**********************************************
```

Керування повертається до DOS.

Робота в пакетному режимі - Do It All in Batch Mode

Всі операції можуть бути виконані автоматично в пакетному режимі. Файл BATCH.SAM на дистрибутивній дискеті включає набір команд, що дублюють послідовність інструкцій, які Ви тільки що дали LP88 в інтерактивному режимі. Для виконання усіх команд з цього командного файлу введіть таку команду DOS:

A>LP88 BATCH.SAM

або

C>LP88 BATCH.SAM, якщо файл на диску С.

Команди з'являються на екрані в 23-му рядку під час зчитування з BATCH.SAM та його виконання.

ДОДАТОК Д

(довідковий)

РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ LP 88

Даний додаток являє собою послідовний опис використання LP88 для постановки задачі ЛП, її збереження і пошуку, а також розв'язання та одержання друкованого звіту. Ми не намагаємося показати усе, що можна робити з LP88, а описуємо тільки основні кроки.

Для навчання Вам потрібно мати розмічену резервну дискету. Крім того, потрібен друкувальний пристрій, зв'язаний із комп'ютером; він повинен бути включеним і готовим до друку.

Для прикладу будемо використовувати таку задачу максимізації з трьома обмеженнями і шістьма змінними:

Знайти: X_1, X_2, X_3, X_4, X_5 і X_6,
MAX: $2*X_1 + X_2 + 4*X_3 - 2*X_4 + X_5 - X_6$
за умови:

$X_1 + X_2 + X_3 + X_4 + X_5 - X_6 <= 100$,
$X_1 - 2*X_3 + X_4 + X_6 = 25$,
$5*X_2 + X_3 + X_5 = 40$.

Усі змінні невід`ємні.

Нефіктивні змінні позначені X_1, X_2 і т.д. Цільова функція - це лінійний вираз, тобто її потрібно максимізувати. Обмеження містить лінійну рівність і дві лінійних нерівності. Нерівності можна перетворити в рівності, додавши фіктивну змінну S_1 у ліву частину першого і помноживши фіктивну змінну S_2 із другої нерівності. Усі змінні, враховуючи фіктивні, можуть бути більшими або дорівнювати нулю.

Запуск програми

Виберіть на панелі або надрукуйте у командному рядку LP88.EXE. Після завантаження даного файлу Ви повинні бачити на дисплеї повідомлення про авторські права на LP88, потім лістинг варіантів меню, доступних для користувача, під кожним із чотирьох меню програм (рис.Д.1).

Після лістингу меню на екрані з'явиться серія запитань.

Ваші відповіді на ці запитання нададуть LP88 дані про конфігурацію системи на час сеансу.

Linear Programming for the IBM PC LPX8

	MASTER MENU	SETUP MENU	EXECUTION MENU	OUTPU
F1	SETUP MENU	FILES LIST	PAUSE/CONTINUE	PRIMA
F2	SOLVE PROBLEM	NEW PROBLEM	STEP/CONTINUE	DUAL
F3	RERUN w BASIS	DISPLAY EDITOR	INVERT MATRIX	COST
F4	RESTART w INV	SAVE PROBLEM	CHANGE LIMITS	RHS R
F5	GET SOLUTION	DELETE FILE	PIVOT ON X	INVER
F6	OLD PROBLEM	OLD PROBLEM	DELTA X	INV*A
F7	BATCH MODE	INPUT FILE	SHORT PRINTOUT	SAVE
F8	OUTPUT MENU	RECORD PIVOTS	SAVE BASIS	SAVE
F9	END SESSION	MASTER MENU	MASTER MENU	MASTE
F10	HELP	HELP	HELP	HELP

8 - Version 4.11 (C)Copyright Eastern Software Products, Inc. 1983, 1984
19750 Bytes Free
Input Storage File for Printed Output (<Enter for Printer)?

Рисунок Д.1 – Вигляд екрана після запуску lp88.exe

Щоб почати працювати з друкувальним пристроєм із 80-символьним рядком (наприклад, з матричним принтером Epson LX1050+), треба відповісти на такі запитання:

```
**************************************************
    * Input Storage File for Printed Output * * (<Enter for Printer>)? CON *
    * Input Line Width for Printed Output? 80 or <Enter> *
    * Input Storage File for Coefficient Array * * (<Enter for Memory>)? <Enter> *
    * Input Storage File for Inverse Array * * (<Enter for Memory>)? <Enter> *
    * WARNING: Default Drive Must NOT be Write Protected. *
*================================================*
    * Вхідний файл збереження для друкування* * (<Введення для принтера>)? CON*
    * Вхідна ширина рядка для друкування? 80 або <введення> *
    * Вхідний файл збереження масиву коефіцієнтів (<Введення для пам'яті>)? <введення> *
    * Вхідний файл збереження масиву зворотної матриці (<Введення для пам'яті>)?
<введення>*
    * ПОПЕРЕДЖЕННЯ: Дисковод за замовчуванням не повинен бути захищений від запису.)
**************************************************
```

За відсутності принтера Ви може передати вихідні дані на екран дисплея, відповівши на перше запитання ім'ям файлу "CON". Коли Ви закінчите відповідати на запитання, LP88 посилає порожній рядок на той файл/пристрій, що Ви визначили для друку. Якщо рядок не можна записати, то на екрані з'явиться ОСНОВНИЙ (BASIC) код помилки і LP88 завершить роботу.

Зараз Ви повинні бачити в 23-му рядку інструкцію:

```
**************************************************
    * Press Function Key for MASTER Menu Selection *
    * 1SETUP *
    * 2SOLVE *
    * 3RERUN *
    * 4RESTAR *
    * 5GET SO *
    * 6OLD PR *
    * 7BATCH *
    * 8OUTPUT *
    * 9END SE *
    * 10HELP *
*================================================*
    * Натисніть функціональну клавішу для виклику головного меню *
**************************************************
```

Вибірка головного меню (MASTER MENU) з'явиться в рядку функціональної клавіші внизу екрана. LP88 перебуває в інтерактивному режимі і готовий виконувати команди, як тільки Ви натиснете функціональні клавіші.

Допомога - F10 - To Get Help

Остання функціональна клавіша F10 у кожному меню - це клавіша допомоги (HELP). Як тільки Ви натиснете клавішу F10(HELP), на екрані відтворюється повідомлення про авторські права на LP88 і меню програм, як і зараз. Активне меню завжди ідентифіковано в 23-му рядку, а меню програм завжди з'являється в рядку функціональних клавіш внизу екрана.

Постановка задачі ЛП - F1 - Setting Up a Linear Program

Для введення нової задачі ЛП LP88 повинен перебувати під керуванням SETUP MENU (меню встановлення параметрів). Це здійснюється натисканням функціональної клавіші F1 (SETUP MENU). Екран вивільниться, і вгорі в 2 рядках з'явиться заголовок задачі. Вибірка MASTER MENU (головного меню) у рядку функціональних клавіш буде замінена на вибірку SETUP MENU (меню встановлення параметрів). Якщо Ви ще цього не зробили, то натисніть функціональну клавішу F1 (SETUP MODE).

Тепер натисніть функціональну клавішу F2 (нова задача). Ви починаєте вводити завдання-зразок, відповідаючи на наступні запитання, що з'являються по черзі у верхній частині екрана:

```
**************************************************
      * Name of New Problem? SAMPLE *
      * Objective (MAX or MIN)? MAX *
      * Number of Constraints? 3 *
      * Number of Nonslack Variables? 6 *
*================================================*
      * Ім'я нової задачі? SAMPLE *
      * Цільова функція (MIN або MAX)? MAX *
      * Число обмежень ? 3 *
      * Число нефіктивних змінних? 6 *
**************************************************
```

Відновлюється заголовок задачі, і Ви готові почати введення коефіцієнтів завдання-зразка ЛП. LP88 вважає SAMPLE поточною задачею. Вибірки усіх меню застосовуються до SAMPLE, поки вона виступає поточною задачею лінійного програмування. Зазначимо, що ім'я поточної задачі з'являється в реверсивному кольорі в заголовку задачі. Це означає, що LP88 очікує від Вас натискання однієї з функціональних клавіш.

LP88 автоматично надає імена обмеженням і змінним нової задачі за замовчуванням. Імена обмежень - Y_1, Y_2, Y_3 та змінних X_1, X_2, X_3, X_4, X_5 і X_6. Для обмежень можуть бути також створені фіктивні змінні. Відповідно до обмежень фіктивним змінним надаються імена S_1, S_2 і S_3.

Використання екранного редактора - F3 - Using the Display Editor

Найбільш легкий спосіб введення коефіцієнтів задачі невеликої розмірності - використання екранного редактора LP88. Натисніть функціональну клавішу F3 (Екранний редактор). У 23-му рядку Вас запитають імена рядка і стовпця у верхньому лівому куті секції масиву коефіцієнтів розмірністю 10 рядків на 15 стовпців. Ця секція масиву разом із відповідними елементами рядка цільової функції (RETURN) і стовпця правої частини (RHS) виводиться на екран для редагування. Ім'я рядка Y_1 і ім'я стовпця X_1 вказують для екранного редактора верхній лівий кут задачі SAMPLE. Ви можете показати на екрані всю задачу, відповідаючи на такі запитання:

```
**************************************************
      * Upper Left-Hand Corner Row ID? Y1 *
      * Upper Left-Hand Corner Column ID? X1 *
*================================================*
      * Верхній лівий кут рядка ID? Y1 *
      * Верхній лівий кут стовпця ID? X1 *
**************************************************
```

Після введення імен рядків і стовпців виникне коротка пауза, доки LP88 встановлює зображення на екрані. Коли на екрані з'явиться зображення, відзначте, що число обмежень тимчасово зросло до 15, а число змінних - до 10. Ці доповнення дозволять Вам додати в поточну задачу нові обмеження і цілочислові змінні. Якщо Ви коли-небудь користувалися якоюсь популярною програмою оброблення електронних таблиць, то робота з екранним редактором не викликає ускладнень. Майже все, що Ви бачите на екрані між заголовком і рядком функціональних клавіш, можна модифікувати за допомогою екранного редактора, враховуючи цільову функцію (MAX), ім'я параметрів функції мети (RETURN), імена обмежень (Y_1 і т.д.), вигляд співвідношення в обмеженнях (<, =, > і т.д.).

Усі елементи масиву коефіцієнтів, рядки цільової функції і стовпця правої частини спочатку мають значення 0. Відповідні елементи не відображені на екрані, за винятком елементів правої частини, де на екрані з'являється 0. Усі обмеження подані спочатку у вигляді рівностей. Екранний редактор має курсор, що містить рядок із шести від'ємних символів. Ви можете знайти курсор на початку 4-го рядка екрана відразу за "MAX" (позиція Home). Позиція курсора показує компонент поточної задачі, що буде замінений на дані, введені з клавіатури. Для керування позицією курсора використовується клавіша <Enter> і клавіші цифрової клавіатури (клавіші цифрової клавіатури розміщені в правій частині клавіатури ПК), клавіші, позначені стрілочками і словами "Home" і "End". Ці клавіші функціонують

відповідно до нанесених на них символів переміщення курсора від елемента до елемента.

Введемо тепер коефіцієнти задачі-зразка. Щоб ввести коефіцієнт "3" для змінної X_1, натисніть цифрові клавіші "6", а потім "2". Курсор тепер повинен знаходитися під X_1 у рядку з міткою RETURN. Видрукуйте "3" і перемістіть курсор на наступну позицію. Тепер коефіцієнт "3" з'являється на екрані в потрібному місці. Повторюйте процес для інших коефіцієнтів задачі-зразка. Не забувайте ввести <= і = для обмежень Y_1, Y_2 і Y_3.

Запис лінійної програми - F4 - To Save Linear Program

Замініть робочу дискету в дисководі A на відформатовану порожню дискету. Для запису поточної задачі перевірте, по-перше, рядок функціональної клавіші, щоб переконатися, що система під керуванням SETUP MENU, потім натисніть функціональних клавіш F4 SAVE PROBLEM (запис задачі).

LP88 підказує Вам ім'я файлу для задачі , що записується. Ім'я файлу і команду SAVE перевіряють до того, як запишуть на дискету задачу. Для запису SAMPLE з ім'ям файлу SAMPLE треба відповісти на запитання, що з'являються під заголовком задачі:

```
*****************************************************
      * Name for Saved Problem? SAMPLE *
      * SAMPLE to be Saved as SAMPLE (Y/N)? Y *
*=================================================*
      * Ім'я задачі, що записується? SAMPLE *
      * SAMPLE повинна бути записана як SAMPLE (ТАК/НІ)? ТАК *
*****************************************************
```

Послідовний файл SAMPLE.LP буде записаний на вибраний вами диск (за замовчуванням). Щоб перевірити, чи записаний файл, потрібно дати відповідь на запитання в 4-му рядку:

```
*************************************
      * Name of Drive? A or <Enter> *
*===================================*
      * Ім'я дисковода? A або <Enter>) *
*************************************
```

Усі без винятку файли на диску виводяться на екран. SAMPLE.LP повинен бути серед них.

Пошук лінійної програми - To Retrieve a Linear Program

Переконайтеся, що SETUP MENU все ще в рядку функціональних клавіш. Тепер натисніть функціональну клавішу F6 OLD PROBLEM (Стара

задача). Введіть ім'я записаної задачі, відповівши на запитання, що з'являється в 4-му рядку:

```
**********************************
      * Old Problem Name? SAMPLE *
*================================*
      * Ім'я старої задачі? SAMPLE *
**********************************
```

Не додавайте до імені задачі розширення LP.

Задача-зразок ЛП знайдена в тому вигляді, як була записана, і тепер це поточна задача. Ви можете переконатися, що запис і пошук SAMPLE здійснені правильно, вивівши задачу на екран за допомогою екранного редактора. Просто натисніть функціональну клавішу F3 (Екранний редактор) і введіть, як і раніше, верхній лівий кут.

Розв'язання задачі ЛП - F2 - To Solve a Linear Program

У Вас на екрані в рядку функціональних клавіш усе ще SETUP MENU? Якщо так, то натисніть функціональну клавішу F9 (MASTER MENU), щоб повернутися до вибірки головного меню (MASTER MENU). Коли на екрані в рядку функціональних клавіш з'явиться головне меню (MASTER MENU), натисніть функціональну клавішу F2 SOLVE PROBLEM (розв'язати задачу).

Доки LP88 розв'язує задачу ЛП, на екрані зберігається поточний опис статусу обчислень. Заголовок на екрані розділений на дві частини. Тепер Ви повинні бачити вихідний стан обчислень, потім список значень за замовчуванням для різних допусків і програмних керувань. LP88 не вимагає змін у цих параметрах при розв'язанні SAMPLE, тому Ви можете відповісти на запитання в 23-му рядку так:

```
*************************************************
    * Input Control Number (<Enter> to Run)? <Enter> *
*================================================*
    * Введіть контрольне число (<Enter>, щоб виконувати програму)? <Enter> *
*************************************************
```

LP88 потрібно 1-2 хвилини на пошук оптимального розв'язання SAMPLE при використанні симплекс-методу. Коли LP88 закінчить розв'язання SAMPLE, інформація на Вашому дисплеї повинна виглядати так:

```
*************************************************
      SAMPLE SOLUTION IS OPTIMAL DATE XX-XX-XXXX TIME XX:XX:XX
      MAXIMUM ENTERS: X5 BASIS X: 3 VARIABLES: 6
      PIVOTS: 3    LEAVES: BASIS  S: 0      SLACKS: 2
      LAST INV: 0  DELTA -1.6667    RETURN 212.5   CONSTRAINTS: 3
      BASIS X3 X1 X5
      PRIMAL 17.5 60 22.5
      DUAL 3.5 -1.5 -2.5
```

****

Цей короткий виклад розв'язання задачі ЛП також посилається на друкувальний пристрій або у файл, якщо це було передбачено при введенні початкових параметрів. Якщо дані на екрані інші, то Ви некоректно задали задачу-зразок. Поверніться до того моменту, коли Ви шукали файл SAMPLE.LP.

Базис розв'язання SAMPLE складається із змінних, поданих на екрані в рядку BASIS. Значення їх розв'язків з'являться нижче в рядку PRIMAL. Усі інші змінні задачі-зразка дорівнюють 0, включаючи фіктивні змінні, додані LP88.

Подвійні, або тіньові, значення обмежень з'являються одне за одним на екрані в рядку DUAL. Значення розв'язку подвійної змінної для обмеження $У_1$=3,5, для $У_2$=-1,5 і для $У_3$= -2,5.

Поки LP88 розв'язував SAMPLE, Ви могли помітити, що в рядку функціональних клавіш MASTER MENU було замінено на EXECUTION MENU. Однак, коли задача розв'язана, знову з'являється MASTER MENU.

Ви також могли помітити ім'я "****" серед списку базисних змінних. Цим способом LP88 ідентифікує штучну змінну в базисі під час фази I у симплекс-алгоритмі.

Запис і багаторазове використання базису
F8 - To Save and Reuse a Basis

LP88 дозволяє записувати і багаторазово використовувати базис розв'язання. Це дуже зручно при розв'язанні задач ЛП великої розмірності, що можуть включати багато опорних точок для розв'язання зі штучного базису, який складається з окремих елементів і штучних змінних.

Натисніть у головному меню (MASTER MENU) клавішу F8 (OUTPUT MENU) для переходу в меню виводу. Тепер натисніть функціональну клавішу F8 Save Basis (Запис базису) у меню виводу (OUTPUT MENU). LP88 запитає ім'я файла базису і потім верифікує файл на предмет того, чи записаний він. Ви можете записати базис розв'язання задачі-зразка у файлі BASIS.LPB, відповівши на такі питання:

```
**************************************************
      * Name for Saved Basis File? BASIS *
      * SAMPLE Basis is to be Saved as BASIS (Y/N)? Y *
*================================================*
      * Номер файлу базису, що записується? BASIS *
      * Базис SAMPLE повинен бути записаний як BASIS (ТАК/Hi)? ТАК *
**************************************************
```

Не додавайте модифікатор ".LP" до імені файлу базису. LP88 зробить це автоматично.

Щоб використовувати записаний базис як відправну точку в симплекс-алгоритмі, натисніть функціональну клавішу F3 (Rerun w Basis – повернутися до базису) замість F2 (Solve Problem - розв’язати задачу), коли Ви готові розв’язувати поточну задачу. Потім LP88 запитає у Вас ім'я збереженого файлу базису.

Щоб знову розв’язати SAMPLE, використовуючи BASIS, натисніть функціональну клавішу F3 (Rerun w Basis) і ім'я файлу BASIS на диску:

```
**************************
      * Name of Basis? BASIS *
*=======================*
      * Ім'я базису? BASIS *
**************************
```

Тепер, щоб реконструювати розв’язання, LP88 потрібно тільки повторно інвертувати матрицю базисних стовпців. Коли LP88 зробить це, на екрані дисплея знову з'явиться розв’язок.

ДОДАТОК Е

(довідковий)

ПАРАМЕТРИ “ПОШУКУ РОЗВ’ЯЗКУ”

Максимальний час

Застосовується для обмеження часу, що відводиться на пошук розв’язання задачі. У поле можна ввести час (у секундах), числове значення якого не повинно перевищувати 32767; значення 100, використовуване за замовчуванням, може застосовуватися для розв’язання більшості простих задач.

Ітерації

Опція використовується для керування часом розв’язання задачі, шляхом обмеження кількості проміжних обчислень. У поле можна ввести кількість, яка не перевищує 32767; значення 100, використовуване за замовчюванням, застосовується для розв’язання більшості простих задач.

Точність

Використовується для задання точності, з якою визначається відповідність клітини цільовому значенню або наближення до зазначених меж. Поле повинне містити число з інтервалу від 0 (нуля) до 1. Низька точність відповідає введеному числу, що містить меншу кількість десяткових знаків, ніж число, використовуване за замовчуванням (наприклад, 0,0001). Висока точність збільшить час, який потрібен для того, щоб зійшовся процес оптимізації. Можна істотно прискорити процес пошуку розв’язання, якщо задати вихідні значення клітин моделі, які впливають, близькими до шуканих результатів.

Допустиме відхилення

Використовується для задання допуску на відхилення від оптимального розв’язання, якщо безліч значень клітини, яка впливає, обмежено безліччю цілих чисел. При зазначенні більшого допуску пошук розв’язку закінчується швидше.

Лінійна модель

Використовується для прискорення пошуку розв’язку лінійної задачі оптимізації або лінійної апроксимації нелінійної задачі.

Показувати результати ітерацій

Використовується для припинення пошуку розв'язку, щоб надати можливість переглядати результатів окремих ітерацій.

Автоматичне масштабування

Використовується для запуску автоматичної нормалізації вхідних і вихідних значень, які якісно розрізняються за розмірами, наприклад, максимізація прибутку у відсотках стосовно вкладень, що обчислюються у мільйонах гривень.

Оцінка

Використовується для зазначення методу екстраполяції (лінійна або квадратична), який буде використовуватися для одержання вихідних оцінок значень змінних у кожному одновимірному пошуку.

Лінійна

Застосовується для використання лінійної екстраполяції вздовж дотичного вектора.

Квадратична

Застосовується для використання квадратичної екстраполяції, що дає кращі результати при розв'язанні нелінійних задач.

Похідні

Ця опція необхідна для зазначення методу чисельного диференціювання (прямі або центральні похідні), який використовується для обчислення частинних похідних цільових та обмежувальних функцій.

Прямі

Опція використовується для гладких безперервних функцій.

Центральні

Опція використовується для функцій, що мають розривну похідну. Незважаючи на те, що даний спосіб вимагає більшої кількості обчислень, він може допомогти при одержанні підсумкового повідомлення про те, що процедура пошуку розв'язку не може поліпшити поточний набір клітин, які впливають.

Метод

Опція використовується для вибору алгоритму оптимізації (метод Ньютона або сполучених градієнтів) для зазначення напрямку пошуку.

Ньютона

Використовується для реалізації квазіньютонівського методу, у якому запитується більше пам'яті, але виконується менше ітерацій, ніж у методі сполучених градієнтів.

Сполучених градієнтів

Використовується для реалізації методу сполучених градієнтів, у якому запитується менше пам'яті, але виконується більше ітерацій, ніж у методі Ньютона. Даний метод слід використовувати, якщо задача досить велика і необхідно заощаджувати пам'ять, а також якщо ітерації дають занадто малу відмінність у послідовних наближеннях.

Завантажити модель[*]

Використовується для відображення на екрані вікна діалогу "Завантажити модель", у якому можна задати посилання на область клітин, що містять модель, яка завантажується.

Зберегти модель

Використовується для відображення на екрані вікна діалогу "Зберегти модель", у якому можна задати посилання на область клітин, призначену для збереження моделі оптимізації. Даний варіант передбачений для збереження на листі більш однієї моделі оптимізації (перша модель зберігається автоматично).[†]

---

[*] Вікно діалогу "Завантажити модель"

Область моделі

Використовується для задання посилання на область моделі оптимізації, що завантажується. Посилання повинне адресувати область моделі цілком (недостатньо зазначити тільки першу клітину).

[†] Вікно діалогу "Зберегти модель"

Область моделі

За замовчуванням пропонується область моделі, в якій поточна клітина листа є першою. Кількість клітин області моделі дорівнює сумі трьох додаткових клітин та кількості обмежень. Інакше, за область збереження моделі можна прийняти на листі верхню клітину вільного стовпця.

ДОДАТОК Ж

(довідковий)

ОГЛЯД РОЗРОБОК, ЩО ПОДАНІ В ІНТЕРНЕТ

Існуючі види програмного забезпечення для розв'язку задач нелінійного програмування

Можна виділити декілька критеріїв, за якими розділяють в Інтернет програмне забезпечення для розв'язання задач нелінійного програмування. Найбільш поширений критерій напряму пов'язаний із окремими видами задач нелінійного програмування. Причиною такого розподілу стало те, що програмне забезпечення подібного роду розроблюється в основному під конкретне завдання, яке, у свою чергу, намагаються звести до найбільш вивчених задач нелінійного програмування. Це пов’язане з тим, що такі задачі мають перевірені і чіткі алгоритми розв'язання. Однак такий напрямок розвитку даної області привів до того, що в Інтернет досить важко знайти універсальні програмні продукти, придатні для розв'язання як найбільш складних задач "глобальної оптимізації", так і найбільш простих задач "лінійного програмування". Таким чином, якщо виникає проблема розв'язання тієї чи іншої задачі нелінійного програмування і є плани скористатися ресурсами Інтернет для її розв'язку, треба проаналізувати свою задачу, визначити, з яким випадком нелінійного програмування маємо справу, переконатися в тому, що задача поставлена коректно і не може бути зведена до більш простої форми. Тільки після цього починаємо пошук існуючого програмного забезпечення, яке було створено для подібної задачі. Практично кожен сайт, присвячений проблемам нелінійного програмування та такий, що надає доступ до бібліотек ПЗ, намагається дотримуватися вищезазначеного критерію при збереженні і наданні інформації про програмні пакети з нелінійного програмування.

Поряд із цим критерієм використовують також поділ на платне і безкоштовне програмне забезпечення. У першому випадку користувачу зазвичай висилають інструкцію щодо використання і інсталяційну програму, яка встановлює програмний пакет на комп'ютер користувача. При цьому побачити вихідні тексти можливості немає. До того ж це також неможливо і після того, як отримана ліцензія на використання. Наприклад, Центр оптимізаційних технологій (Optimization Technology Center [41] of Northwestern University and Argonne National Laboratory) пропонує платне програмне забезпечення для задач квадратичного програмування [42] з додатковими модулями для лінійної та квадратичної апроксимацій цільових функцій. Це, звичайно, дуже зручно. Можна, наприклад, перетворити цільову функцію для розв’язання і загальної задачі нелінійного програмування, але за визначену суму. Виходячи зі списку продажів даного програмного забезпечення, опублікованого на цьому сайті, подібними послугами користується чимала кількість різних компаній і приватних осіб. Аналогічних прикладів можна навести дуже багато. При цьому варто сказати, що в середовищі платного програмного забезпечення тенденція до створення універсальних програм виявляється набагато сильнішою. (Оскільки чим програма універсальніша, тим більше типів задач нелінійного

програмування можна розв’язати за її допомогою, тим дорожче її вартість. І як наслідок, більше можливостей для створення її авторами нових програмних продуктів, які у міру накопичення зазвичай об'єднують у великі універсальні програмні пакети).

Найбільший інтерес для нас становить безкоштовне програмне забезпечення. Як вже зазначалося, це в основному програми, які написані під конкретне завдання або ті, що реалізують окремий метод розв'язання задач нелінійного програмування. На відміну від платного програмного забезпечення розповсюджувачі пересилають не тільки опис і інсталяційну програму, але і код програми (здебільшого це тільки опис і код, що можна внести у свої розробки). Безкоштовно надають свої програми, як правило, окремі розробники, математичні асоціації і лабораторії оптимізації при різних НДІ. Автори безкоштовного програмного забезпечення мають свої невеликі сторінки в Інтернет, де подається інформація про їх діяльність. Існують сайти, що містять цілі каталоги посилань на подібні сторінки. Також безкоштовне програмне забезпечення, в якому реалізовані методи нелінійного програмування, пропонують і великі фірми. В основному це старі розробки, які вже дали достатній прибуток компанії, поруч із безкоштовним пакетом зазвичай знаходиться пропозиція одержати і більш нову, універсальну, зручну, але платну версію. У такий спосіб безкоштовне програмне забезпечення, як правило, призначене для реклами і тому досить часто являє собою частину якогось проекту, бібліотеку або просто вирізаний шматок коду. Щоб використовувати таке програмне забезпечення, треба або дуже довго розбиратися в програмі, або бути її розроблювачем.

Ще одним критерієм, за яким розділяють програмне забезпечення для розв'язання задач нелінійного програмування в Інтернет, є авторство. В даному контексті можна виділити три основні групи: окремі розроблювачі та їх невеликі групи, різні математичні організації середніх розмірів і великі компанії. Щодо невеликих груп окремих розроблювачів можна сказати тільки одне - їх дуже багато. Тому сформулювати якусь загальну думку про цю групу досить складно. В основному надаються безкоштовні програмні коди, в яких реалізовано який-небудь метод нелінійного програмування. Опис їх програм має вкрай скорочений або заплутаний характер. Тому точно зрозуміти, що робить програма, найчастіше можна після аналізу коду або після консультації з виконавцем. Посилання на сторінки цієї групи зустрічаються не тільки на спеціалізованих сайтах з методів оптимізації, а й в будь-якому місці, де була можливість їх розмістити. Серед подібних програм зустрічаються як на 100% робочі, так і зовсім непрацюючі. Це вже як пощастить користувачу.

Найбільш цікавою є друга категорія розробників програмного забезпечення. До неї відносять математичні асоціації і товариства, центри і лабораторії оптимізації при різних вузах і НДІ. Це вже більш серйозні розробники. Їх програмне забезпечення містить в основному працездатні і перевірені програмні пакети з детальною документацією, яка додається, до того ж більшість із них безкоштовні. Крім того, сайти подібних організацій містять велику кількість додаткової інформації про проблеми нелінійного програмування. Це каталоги різного програмного забезпечення, яке існує в даній області, результати тестів цього програмного забезпечення, відповіді на запитання з даної теми, які найбільш часто ставляться (FAQ), списки "перевірених" розробників ПЗ, які співпрацювали з даною організацією та багато іншої корисної інформації.

Третя категорія складається із великих компаній, які розробляють потужні програмні продукти. На відміну від перших двох категорій тут домінує платне програмне забезпечення, крім того, дані розробки одержують потужний супровід. До стандартного набору методів програмного пакета можна додати за окрему плату величезну кількість розширених функцій, новітніх методів і т.п. Ця категорія розробників дає також і безкоштовне ПЗ, але ці програми схожі з тими, що пропонуються розробниками першої категорії, тобто це в основному частини коду яких-небудь великих проектів, бетта-версії, застаріле ПЗ.

Щодо таких критеріїв поділу ПЗ як мова програмного коду і мова документації про нього можна сказати, що основними мовами, які використовуються для написання програм, є Fortran і C++. Програми, написані на інших мовах програмування, найчастіше просто "перекладені" (трансльовані) з основних. Основною мовою документації усе ще залишається англійська, тому загалом будь-який "неангломовний" розробник намагається дати докуметацію мінімум на двох мовах (рідною та англійською, іноді навіть тільки англійською). Проблема з російськомовними сайтами (не кажучи вже про україномовні) залишається невирішеною: практично всі сайти або перебувають у стадії розробки, або не супроводжуються після свого створення.

Моделі тестування програмних продуктів для розв'язання задач нелінійного програмування

З проблемою тестування будь-якого програмного продукту стикаються щораз при завершенні етапу кодування. Для програм, що розв’язують задачі нелінійного програмування, етап тестування є принциповим. Від того, як він буде пройдений, залежить подальша доля проекту. Тому так важливо вибрати модель тестування не тільки коректну, але також максимально наближену до реальних практичних задач. В Інтернет існують декілька великих колекцій, які містять різнопланові моделі тестування, а також має місце значна кількість малих колекцій моделей тестування для окремих задач нелінійного програмування. Нижче наведено список найбільш відомих збірок подібних моделей:

- Handbook of Test Problems in Local and Global Optimization - збірник моделей для програмних продуктів, котрі реалізують як методи локальної, так і глобальної оптимізації [43].
- COPS (Constrained Optimization ProblemS, розробники Джорж Море (Jorge More) і його колеги) - збірник великомасштабних нелінійно обмежених задач оптимізації. У поточній версії (2.0) збірника подані моделі тестування за такими темами: динаміка рідини, динаміка населення, оптимальне керування та інші. Кожна модель супроводжується описом і формулюванням, а також результатами тестування з використанням найбільш відомих програмних продуктів [44].
- CUTE (Constrained and Unconstrained Testing Environment) - збірник процедур, докладно описаних методів і моделей тестування для задач лінійної і нелінійної оптимізації. Містить більше 800 різних типів задач. На основі даного збірника проводиться порівняння вже існуючих пакетів, а також нових розробок [45].
- Ганс Д.Міттельман і П.Спеллуккі (Hans D.Mittelmann, P.Spellucci) створили середовище для тестування, яке містить понад 400 задач нелінійної оптимізації з

обмеженнями та без них, а також програмний код, що дозволяє "підключити" Вашу програму до цього середовища [46]. На їх сайті також розрозміщений збірник стандартів програмного забезпечення [47].

- Клаусс Шіттковські (Klaus Schittkowski) пропонує понад 600 найменувань моделей тестування, заснованих на квадратичній оптимізації оцінок параметрів динамічних систем [48].

Нелінійна оптимізація в on-line

Крім можливості придбати ПЗ з нелінійного програмування і систем для його тестування, в Internet пропонується також безліч додаткових, хоча і не часто використовуваних послуг. Наприклад, за допомогою таких сайтів Ви можете одержати розв'язок проблеми, яка пов'язана з постановкою і розв'язанням задачі нелінійної оптимізації. Для цього потрібно відіслати формулювання проблеми на сайт, що пропонує подібну послугу, зазначити свої дані для контакту і чекати на відповідь. (Подібні сайти досить часто модифікуються і тому кожного разу при їх використанні необхідно звертати увагу на такі деталі, як формат вхідних і вихідних даних, метод розв'язку задач і таке інше):

- AMPL (http://www.ampl.com/ampl/TRYAMPL) - сайт, що підтримує власний формат даних, заснований на AMPL-мові моделювання, дозволяє розв'язувати задачі нелінійного програмування, які мають до 300 обмежень, поданих у вигляді рівностей і 300 обмежень - у вигляді нерівностей, містить 8 пристроїв для розв'язку і величезну бібліотеку прикладів. Користувач передає умову задачі тільки в форматі AMPL.
- BARON (http://archimedes.scs.uiuc.edu/cgi/run.pl). Дозволяє розв'язувати різні задачі нелінійного програмування (навіть глобального характеру). Користувач може передати умову своєї задачі у вигляді файлу або заповнивши форму. Для використання можливостей сайту потрібен пароль.
- HIRON (http://vasbo.comtel.ru/hiron.htm). Дозволяє розв'язувати задачі нелінійного програмування з 2-5 обмеженнями.
- Network-Enabled Optimization System (NEOS) Server (http://www-neos.mcs.anl.gov/). Містить більше дюжини надійних пристроїв для розв'язання (лінійне і нелінійне програмування, мережеве і стохастичне лінійне програмування, безумовна й умовна оптимізація нелінійних функцій). Особливості:
  - функції для оптимізації беруться на C, Fortran, а також у форматі мов моделювання AMPL і GAMS;
  - розрахунки виконуються сервером автоматично, для трансляції використовують такі системи: ADIOR для Fortran, ADOL-C для C і відповідний транслятор для мов моделювання;
  - дані пересилаються по e-mail.
- NIMBUS (http://nimbus.math.jyu.fi/). Система оптимізації недиференційовних функцій.
- Numerische Mathematik Interaktiv (http://fb0445.mathematik.tu-darmstadt.de:8081/) і OptiW³ (http://fb0445.mathematik.tu-darmstadt.de:8081/optiwww/themen.html). Системи навчання методам лінійної і нелінійної оптимізації (німецький сайт).

- UniCalc (http://www.rriai.org.ru/UniCalc/calculate.html). Розв'язує ряд конкретних задач лінійної і нелінійної оптимізації.

Отже, різноманітність ресурсів з методів оптимізації в Internet надзвичайна. Постійно виникають не тільки нові ресурси, присвячені даній темі, але й нові види таких ресурсів. Усе говорить про те, що дана сфера дуже бурхливо розвивається.

СПИСОК ЛІТЕРАТУРИ

1. Божко О.І., ЛюбчакВ.О. СЕРВІСИ INTERNET: Навчальний посібник. - Суми: Вид-во СумДУ, 2000. – 95 с.

2. Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ. – М.:Мир, 1998. – 575с.

3. LP88. Eastern Software Products, Inc.,1986 (P.O. Box 15328 Alexandria, Virginia 22309 (703) 549-5469).

4. Бояринов А.И., Кафаров В.В. Методы оптимизации в химической технологии. - М.: Химия, 1975.

5. Сухарев А.Г., Тимохов А.В., Федоров В.В. Курс методов оптимизации. - М.: Наука, 1986.

6. Хачиян Л.Г. Сложность задач линейного программирования. - М.: Знание, 1987.- №10.

7. Фиакко А., Мак-Кормик Г. Нелинейное программирование. Методы последовательной безусловной минимизации. - М.: Мир, 1972.

8. Линейное и нелинейное программирование /Ляшенко И.И., Карагодова Е.А., Черникова Н.В. и др. - К.: Вища школа, 1975.

9. Карманов В.Г. Математическое программирование. - М.: Наука, 1986.

10. Банди Б. Методы оптимизации. Вводный курс. - М.: Радио и связь, 1988.

11. Реклейтис Г., Рейвиндран А., Рэгсдел К. Оптимизация в технике: В 2-х кн./ Пер. с англ.- М.: Мир, 1986. - Кн.1.

12. Реклейтис Г., Рейвиндран А., Рэгсдел К. Оптимизация в технике: В 2-х кн./ Пер. с англ.- М.: Мир, 1986. - Кн.2.

13. Атманов С.А. Линейное программирование. - М., 1981.

14. Акулич И.Л. Математическое программирование в примерах и задачах. - М.: Высшая школа, 1986.

15. Методы оптимизации в статистических задачах управления / Под ред. Соколенко А.Н. - М.: Машиностроение, 1974.

16. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. - Томск: МП "Раско", 1991. -272с.

17. Химмельблау Д. Прикладное нелинейное программирование. – М.:Мир, 1975. – 536 с.

18. Задачи оптимизации: Пособие для факультативных занятий 10-11 кл. /Л.Н.Вывальнюк, А.И.Соколенко, Ю.В.Костарчук и др. - К.:Рад.шк., 1991.- 175с.

19. Геминтерн В.И. Методы оптимального проектирования. - М.,1980.

20. Поляк Б.Т. Введение в оптимизацию. - М.:Наука, 1983. – 384 с.

21. Базара М., Шетти К. Нелинейное программирование. Теория и алгоритмы: Пер.с англ. – М.:Мир, 1982. – 583 с.

22. Математический анализ в вопросах и задачах функции нескольких переменных: Учеб. пособие /В.Ф.Бутузов, Н.Г.Крутицкая.- М.: Высш. шк., 1988.- 288с.

23. Боглаев Ю.П. Вычислительная математика и программирование. – М.:Высш.шк., 1990. – 544 с.

24. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. – М.: Наука, 1984.

25. Турбо Паскаль в курсе высшей математики: Учеб. пособие/ Е.П.Путятин, Д.М.Смагин, В.П.Степанов. - Харьков: Каравелла, 1997. - 352с., ил.

26. Turbo Pascal: Алгоритмы и программы: численные методы в физике и математике: Учеб. пособие /А.Б.Бартков, Я.Т.Гринчишин. - К.: Вища школа, 1992. - 247с.

27. Свидерский Э.А. Решение технологических задач в машиностроении с применением микрокалькуляторов. - М.: Машиностроение, 1987. - 160с.:ил.

28. Горстко А.Б. Познакомтесь с математическим моделированием. - М.: Знание, 1991. - 160 с.

29. Пономаренко О.Г., Пономаренко В.О. Системні методи в економіці: менеджменті та бізнесі: Навч. посібн.- К.: Либідь,1995.

30. Вознесенский В.А. Численные методы. Решение строительно-технологических задач на ЭВМ. – К.:Вища шк., 1989. – 483 с.

31. ДКН "Курс молодого Web-дизайнера" *http://www.sumdu.edu.ua/cources/html/*

32. "Оптимізація хіміко-технологічних процесів" - розділ конспекту лекцій з курсу Математичне моделювання хіміко-технологічних процесів (Одеський політехнічний інститут) *http://www.opu.odessa.ua/konsp/MMXTP/RAZDEL5/kosp_5.htm*

33. Розробка кафедри "Інформаційні процеси та управління" Тамбовського державного технічного університету* *http://ahp.tstu.ru/Matiss/*

34. Ralph W. Pike. Optimization for Engineering Systems and Solutions Manual (Louisiana State University) *http://www.mpri.lsu.edu/OES_text_solns/index.htm*

35. Pravin Varaiya. Lecture Notes on Optimization (University of California, Berkeley) *http://www.path.berkeley.edu/~varaiya/papers_ps.dir/NOO.pdf*

36. Нурминський Євген. Лінійне програмування. - Курс лекцій. *http://www.dvo.ru/studio/linpro/*

37. Новікова Н.М. Курс лекцій з методів оптимізації.1995. *http://0000.nm.ru/cs/00010001.htm*

38. Левин В.И. Нелинейное программирование в условиях интервальной неопределенности (Пензенский технологический институт) *http://inftech.webservis.ru/it/conference/scm/2000/session1/levin.htm*

---

* При використанні web-адрес будьте уважні. Існує імовірність, що не всі web-адреси на момент роботи з посібником будуть доступні. Причин може бути декілька – відключення потрібного серверу від мережі, зміна web-адреси сайту, зміна розміщення файлу, що запитується. Слід зважати на повідомлення броузера при звертанні до конкретного файлу. Якщо в повідомленні про помилку зазначено "Неможливо знайти сторінку", ми пропонуємо спробувати такі дії:

запам'ятати назву потрібного файлу чи папки (наприклад, *Matiss*);

звернутися до титульної сторінки сайту (тобто видалити всі дані до першого одинарного слешу, наприклад, *http://ahp.tstu.ru/*);

спробувати знайти потрібне посилання самостійно.

39. НТП "Техно-Пульсар". Особенности технологии оптимизации и ее использование *http://www.orc.ru/~pulsar/features_r.html*

40. Компьютерная графика в науке и искусстве: Сборник статей. Владивосток: ДВГМА, 1996. *http://dvgma.vld.ru/Cbornik.htm*

41. *http://www.ece.nwu.edu/OTC/*

42. *http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html*

43. *http://www.wkap.nl/book.htm/0-7923-5801-5*

44. *http://www.mcs.anl.gov/%7Emore/cops/*

45. *http://www.cse.clrc.ac.uk/Activity/CUTE*

46. *ftp://plato.la.asu.edu/pub/donlp2/testenviron.tar.gz*

47. *http://plato.la.asu.edu/bench.html*

48. *http://www.uni-bayreuth.de/departments/math/%7Ekschittkowski/demo_probs.htm*

49. Навчальні матеріали з нормативного курсу “Методи оптимізації” (Київський національний університет імені Тараса Шевченка, факультет кібернетики) *http://unicyb.kiev.ua/Library/DO/index.html*