Розв'язання алгебраїчних рівнянь і пошук екстремумів функцій MATLAB в першу чергу орієнтовано рішення задач лінійної алгебри, проте в ньому є і засоби для вирішення нелінійних алгебраїчних завдань, таких як відшукання коренів нелінійних рівнянь, пошук екстремумів функцій однієї або декількох змінних, вирішення задач апроксимації . Опишемо деякі з них. Розв'язання нелінійних рівнянь Одна з поширених задач алгебри – пошук коренів рівняння f(x)=0. Для чисельного відшукання коренів найпростіше побудувати графік функції y=f(x) і знайти точки його перетину з віссю абсцис (MATLAB це можна зробити за допомогою команд plot, fplot, ezplot). Аналогічно можна надійти і у разі системи двох рівнянь із двома невідомими f(x, у)=0, g(x, y)=0, побудувавши на площині (х, у) графіки цих функцій і знайшовши точки їх перетину один з одним. Складніша справа з пошуком комплексного коріння, тут потрібно залучення спеціальних методів. Раніше вже було описано застосування команди roots для коріння поліномів. Для пошуку коренів складніших рівнянь з однією змінною, наприклад, що включають логарифмічні, тригонометричні, експоненційні залежності, застосовують команду fzero. Її вхідними аргументами є ім'я функції, що обчислює ліву частину рівняння f (x) 0, і початкове наближення x0.

приклад.

Візьмемо поліном F (x) = x^2 + 3x + 2

з корінням -1 та -2.

У MATLAB їх можна знайти двома шляхами: за допомогою команди roots та за допомогою команди fzero. В останньому випадку потрібно сформувати допоміжну функцію ff у вигляді окремого m-файлу або створити тимчасову функцію (на період даного сеансу MATLAB) за допомогою команди inline. Команда fzero, що використовує чисельні методи, повертає один із коренів полінома залежно від початкового наближення.

>> р=[1 3 2];

>> roots(p) ans = -2 -1

>>f(x)= inline('x^2+3\*x+2';

>> fzero f(x), -3))

function y=ff(x) y=x^2+3\*x+2;

55 ans = -2.0000

>> fzero(f(x),0)

ans = -1

>> fzero(@ff,0)

ans = -1

У команді fzero та інших командах, які розглядаються у цьому параграфі, є можливість задавати структуру опцій вирішувача. Її можна описати вручну, але зручніше використовувати команди optimget та optimset. Для отримання повного списку опцій та значень за замовчуванням використовують формат optimset('ім'я вирішувача'), наприклад opts=optimset('fzero'). Для зміни опцій використовується команда opts=optimset(ім'я параметра,значення\_параметра). Зменшимо точність обчислень у попередньому прикладі, змінюючи опції вирішувача:

>> o=optimset('fzero'); % опции команды fzero по умолчанию >>

optimget(o,'TolX') % точность вычислений по умолчанию

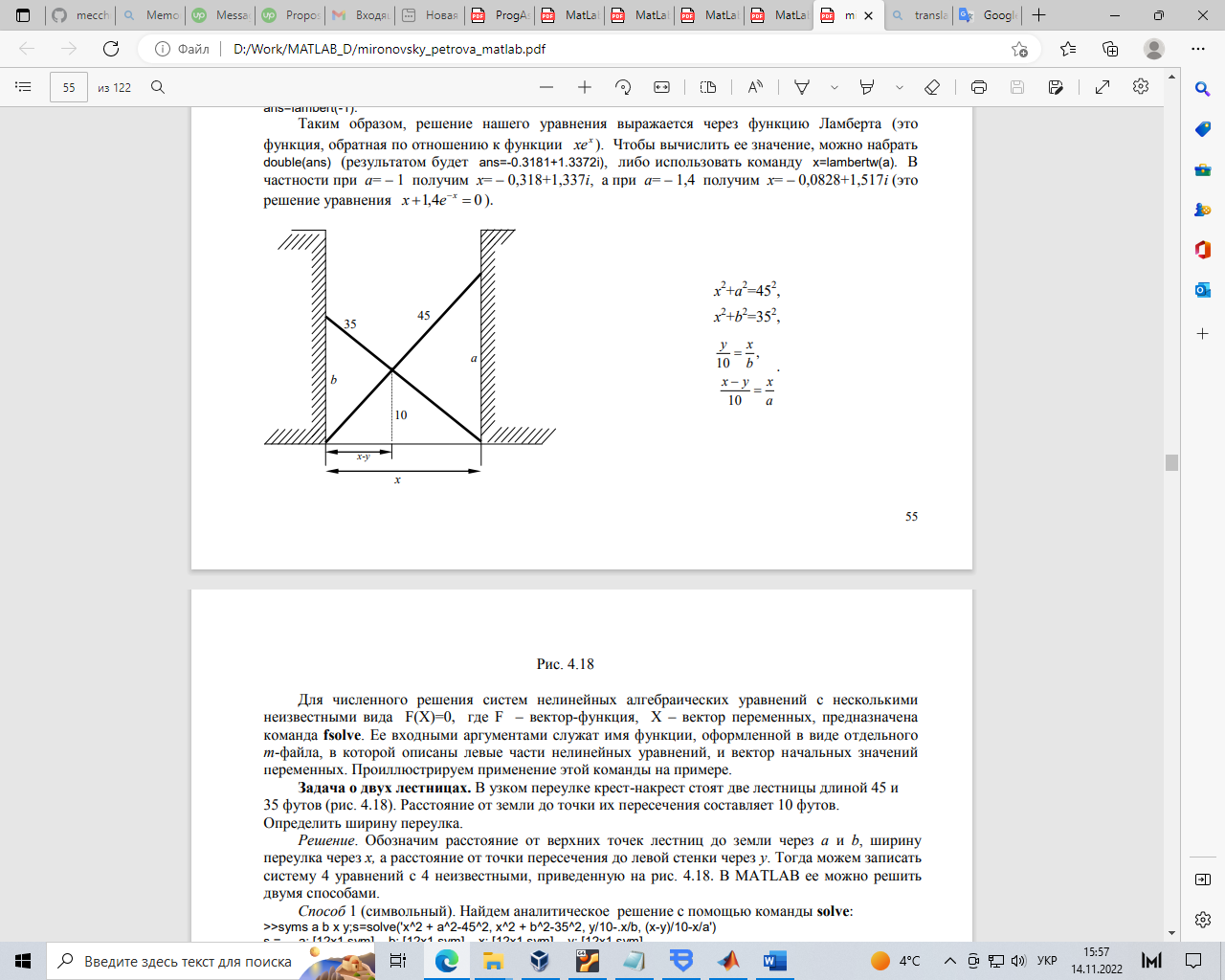
ans = 2.2204e-016

>> fzero(@ff,0) % оптимизация с точностью 2.22е-16

ans = -1 >> o=optimset(o,'TolX',1e-1); % понизили точность до 0.1

>> fzero(@ff,0,o) % оптимизация с точностью 0.1 ans = -1.0240

Для чисельного розв'язання систем нелінійних рівнянь алгебри з кількома невідомими виду F(X)=0, де F – вектор-функція, Х – вектор змінних, призначена команда fsolve. Її вхідними аргументами є ім'я функції, оформленої у вигляді окремого m-файлу, в якій описані ліві частини нелінійних рівнянь, і вектор початкових значень змінних. Проілюструємо застосування цієї команди з прикладу. Завдання про дві сходи. У вузькому провулку хрест-навхрест стоять дві сходи завдовжки 45 і 35 футів.



Відстань від землі до точки їх перетину становить 10 футів. Визначити ширину провулку.

Рішення. Позначимо відстань від верхніх точок сходів до землі через a та b, ширину провулка через x, а відстань від точки перетину до лівої стінки через y. Тоді можемо записати систему 4 рівнянь із 4 невідомими, У MATLAB її можна вирішити двома способами.

Спосіб 1 (символьний). Знайдемо аналітичне рішення за допомогою команди solve:

syms a b x y;

s=solve(x^2 + a^2-45^2==0, x^2 + b^2-35^2==0, y/10-x./b==0, (x-y)/10-x/a==0)

s =

struct with fields:

a: [12×1 sym]

b: [12×1 sym]

x: [12×1 sym]

y: [12×1 sym]

У результаті отримуємо структуру s містить 12 аналітичних варіантів рішення.

Однак спроба виведення їх на дисплей, звертаючись до полів >> s.x, s.y, показує, що рішення містить багатоповерхове коріння, громіздко і незручно для огляду. Його можна уявити у чисельному вигляді, набравши:

>>[double(s.x) double(s.y) double(s.a) double(s.b)]

>ans=31.8175 21.8189 31.8222 14.5825

На дисплей буде виведено всі 12 варіантів рішення, але серед них тільки одне має фізичний зміст: >>ans=31.8175 21.8189 31.8222 14.5825 Спосіб 2 (чисельний). Вирішимо систему у чисельному вигляді за допомогою команди fsolve. Попередньо складемо допоміжну

Спосіб 2 (чисельний). Вирішимо систему у чисельному вигляді за допомогою команди fsolve. Попередньо складемо допоміжну функцію ladder, що містить інформацію про рівняннях

function fn=ladder(p)

x=p(1); y=p(2); a=p(3); b=p(4);

f(1)=x^2 + a^2 - 45^2; f(2)=x^2 + b^2 - 35^2;

f(3)=y/10 - x/b; f(4)=(x-y)/10 - x/a;

fn=f(:);

x=fsolve('ladder',[10; 10; 20; 20])

Тепер замість точної відповіді х1 = - 1 отримано наближене значення х1 = - 1024. Інша можливість розв'язання нелінійних рівнянь алгебри (і систем таких рівнянь) пов'язана з використанням команди solve тулбокса SYMBOLIC. Проілюструємо її простому прикладі. приклад.

*Потрібно знайти коріння трансцендентного рівняння*

*x+e^{-x} = 0*

*Спроба його графічного рішення до успіху не наводить – графік функції x+e^{-x} = 0 жодного разу не перетинає вісь абсцис. Це означає, що дійсних коренів немає. Для пошуку комплексного коріння скористаємося функцією solve. Набираючи solve('x+exp(-x)'), отримаємо відповідь ans=lambert(-1). Таким чином, рішення нашого рівняння виражається через функцію Ламберта (це функція, зворотна по відношенню до функції x xe). Щоб обчислити її значення, можна набрати double(ans) (результатом буде ans=-0.3181+1.3372i) або використовувати команду x=lambertw(а).*

*Зокрема при а = - 1 отримаємо x = - 0,318 +1,337i, а при а = - 1,4 отримаємо x = - 0,0828 +1,517i (це рішення рівняння 1,4 0 x e). x 2 +a 2 =452 , x 2 +b 2 =352 , a x y x b y x = − 10 , 10 . 45 35 a b 10 x x 56 Мал. 4.18*

*Отримуємо відповідь: x=31.8175; y=21.8189; a = 31.8222; b = 14.5825. Бачимо, що обидва способи дають той самий результат.*

Пошук екстремумів

Поширена група завдань пов'язана з пошуком екстремумів функцій одного або кількох аргументів. Широко відомі аналітичні методи вирішення кінцевих екстремальних завдань – метод Ферма (рецепт "взяти похідну і прирівняти нулю") та метод множників Лагранжа. Перший їх застосовується на вирішення завдань безумовної оптимізації, коли потрібно знайти екстремуми функції

Y = f(x\_1, …, x\_n)

Необхідні умови екстремуму мають вигляд

∂f / ∂x1 =0 , ..., ∂f / ∂xn =0,

їх можна записати в компактному вигляді grad у = 0. Після відшукання коренів цієї системи алгебраїчних рівнянь перевірки достатні умови екстремуму.

Для чисельного вирішення тих самих завдань застосовують методи половинного поділу та золотого перерізу (одномірний пошук), методи градієнта та якнайшвидшого спуску, методи цілісного, лінійного та нелінійного програмування. У пакеті MATLAB ці методи реалізовані у командах fminsearch, fminunc, fmincon, fminbnd. Для задач лінійного програмування призначена команда linprog. Мінімум одновимірної функції шукають за допомогою команди fminsearch. Для пошуку максимуму функції f (x) достатньо знайти мінімум функції f (x) , тому спеціальної функції для пошуку максимумів у MATLAB не існує. приклад. Знайдемо точку мінімуму полінома

F (x} = x^2 + 3x + 2

прикладу 1 і переконаємося в правильності результату, прирівнюючи похідну нулю. Похідну беремо за допомогою функції polyder, її корінь знаходимо командою fzero.

Для безусловной минимизации функций от нескольких переменных используют функцию fminunc (от слова unconstrained – без ограничений). Ее первый входной аргумент – имя минимизируемой функции, второй – координаты начальной точки для поиска. Пример. Найдем минимум функции двух переменных

X^2 + Y^2

который, очевидно, достигается в точке x1 = x2 = 0. Предварительно нужно в отдельном m-файле описать минимизируемую функцию, назовем ее fff

function y=fff(x)

y=x(1)^2+x(2)^2;

return

x=fminunc(@fff,[3 3])

Відповідь отримана з досить гарною точністю. У той же час, MATLAB рекомендує замість лінійного пошуку застосувати градієнтний метод. Для цього потрібно, по-перше, переписати функцію fff так, щоб вона повертала не тільки змінну y, а й її приватні похідні (вектор градієнта), і, по-друге, підключити градієнтний метод за допомогою команди optimset

function [y,dy]=fff1(x)

y=x(1)^2+x(2)^2; % функци

dy=[2\*x(1),2\*x(2)]; % градиент

fminunc(@fff1,[3 3],optimset('Gradobj','on'))

використовуючи line-search метод instead.

x = 1.0e-008\* [-0.9290 -0.9290]

Відповідь отримана з досить гарною точністю. У той же час, MATLAB рекомендує замість лінійного пошуку застосувати градієнтний метод. Для цього потрібно, по-перше, переписати функцію fff так, щоб вона повертала не тільки змінну y, але і її похідні приватні (вектор градієнта), і, по-друге, підключити градієнтний метод за допомогою команди

optimset.

function [y,dy]=fff1(x)

y=x(1)^2+x(2)^2; % функци dy=[2\*x(1),2\*x(2)]; % градиент

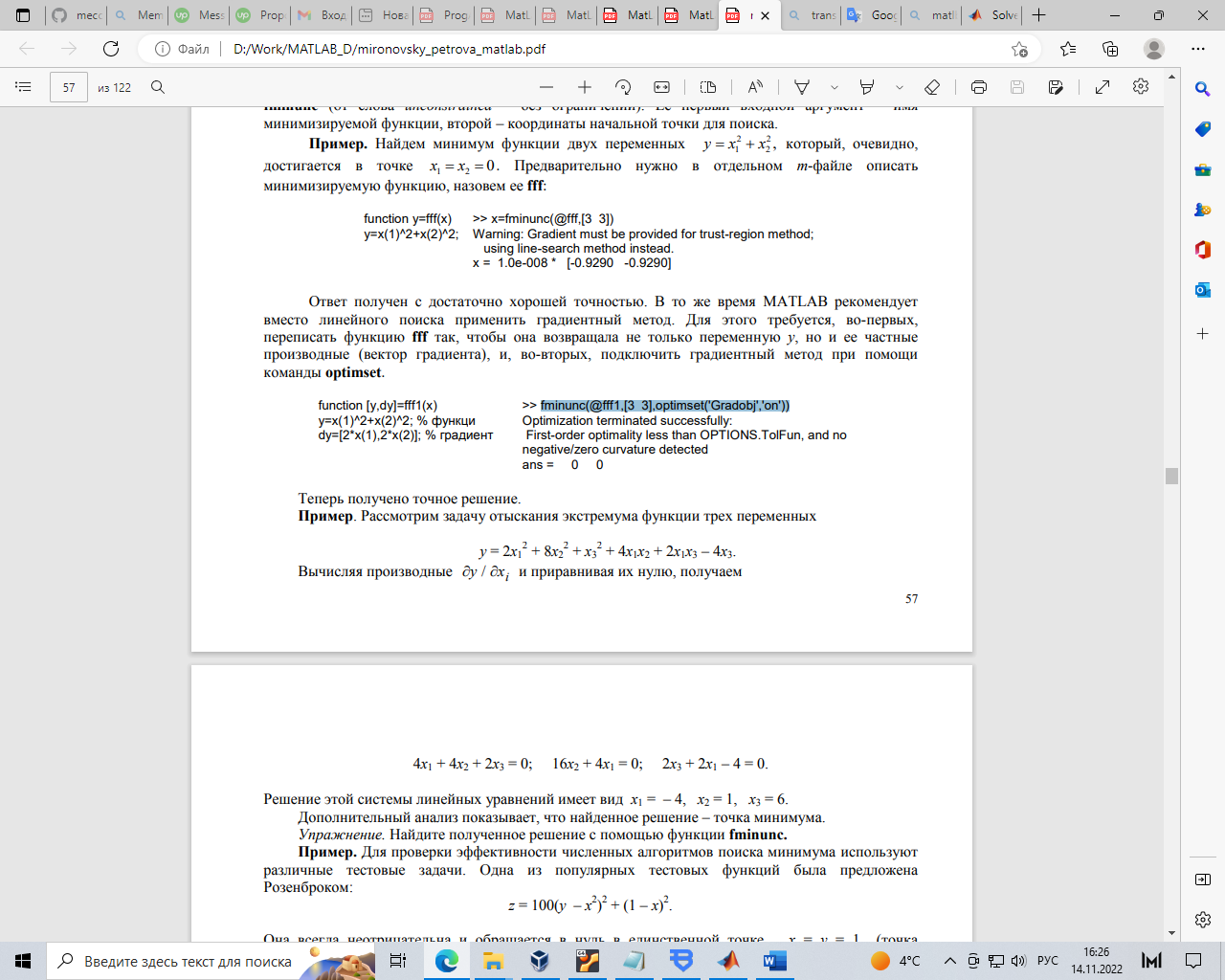
>> fminunc(@fff1,[3 3],optimset('Gradobj','on')) Optimization terminated successfully: First-order optimality less than OPTIONS.TolFun, and no negative/zero curvature detected

ans = 0 0

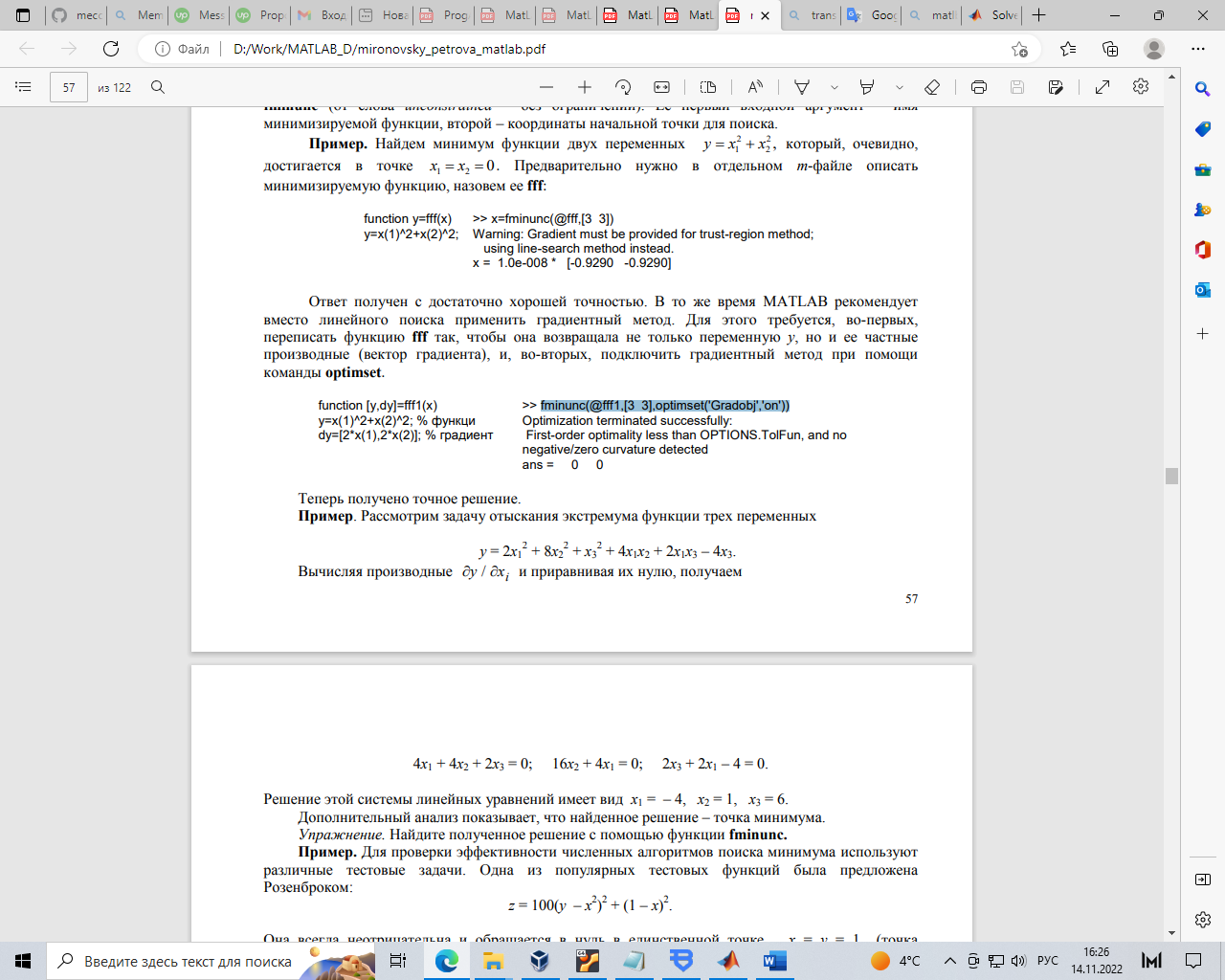
Наразі отримано точне рішення.

Приклад.

Розглянемо завдання пошуку екстремуму функції трьох змінних



Обчислюючи похідні ∂y ∂xi / та прирівнюючи їх нулю, отримуємо



Вирішення цієї системи лінійних рівнянь має вигляд x1 = – 4, x2 = 1, x3 = 6.

Додатковий аналіз показує, що знайдене рішення – точка мінімуму.

Знайдемо мінімум за допомогою команди fminunc, використовуючи графдієнтний пошук

x= -1.9;

y=2.0;

f='100\*(x(2)-x(1)^2)^2+(1-x(1))^2'; % описание функции

GRAD='[100\*(4\*x(1)^3-4\*x(1)\*x(2))+2\*x(1)-2; 100\*(2\*x(2)-2\*x(1)^2)]'; % описание градиента >>OPTIONS = optimset(OPTIONS,'gradobj','on'); % подключение опции градиентного поиска

[x,fval] = fminunc({f,GRAD},[-1.9, 2],OPTIONS) x = 1.0000 1.0000 fval = 1.2371e-015 % результат

Для вирішення оптимізаційних задач з обмеженнями використовуються команди fminbnd та fmincon.

Команда fminbnd (від bound – кордон) призначена для мінімізації функції однієї змінної на інтервалі x1 < x < x2 .

Наприклад,

x=fminbnd(inline('x^2+3\*x+2'),0,1)

дасть відповідь x = 0.

Для мінімізації функцій від кількох змінних з обмеженнями складнішого вигляду застосовують команду fmincon.

Список вхідних параметрів функції досить значний:

x = fmincon (fun, x0, A, B, Aeq, Veq, Lb, Ub, nonlcon, options, p1, p2, ...).

Перший параметр, зазвичай, ім'я функції fun, другий – початкове наближення x0.

Інші параметри: опції options,

5-9 визначають обмеження.

За потреби можна вказати додаткові параметри p1, p2, ....

Якщо завдання немає будь-який тип обмежень, відповідні аргументи замінюються порожнім масивом [ ].

Типи обмежень.

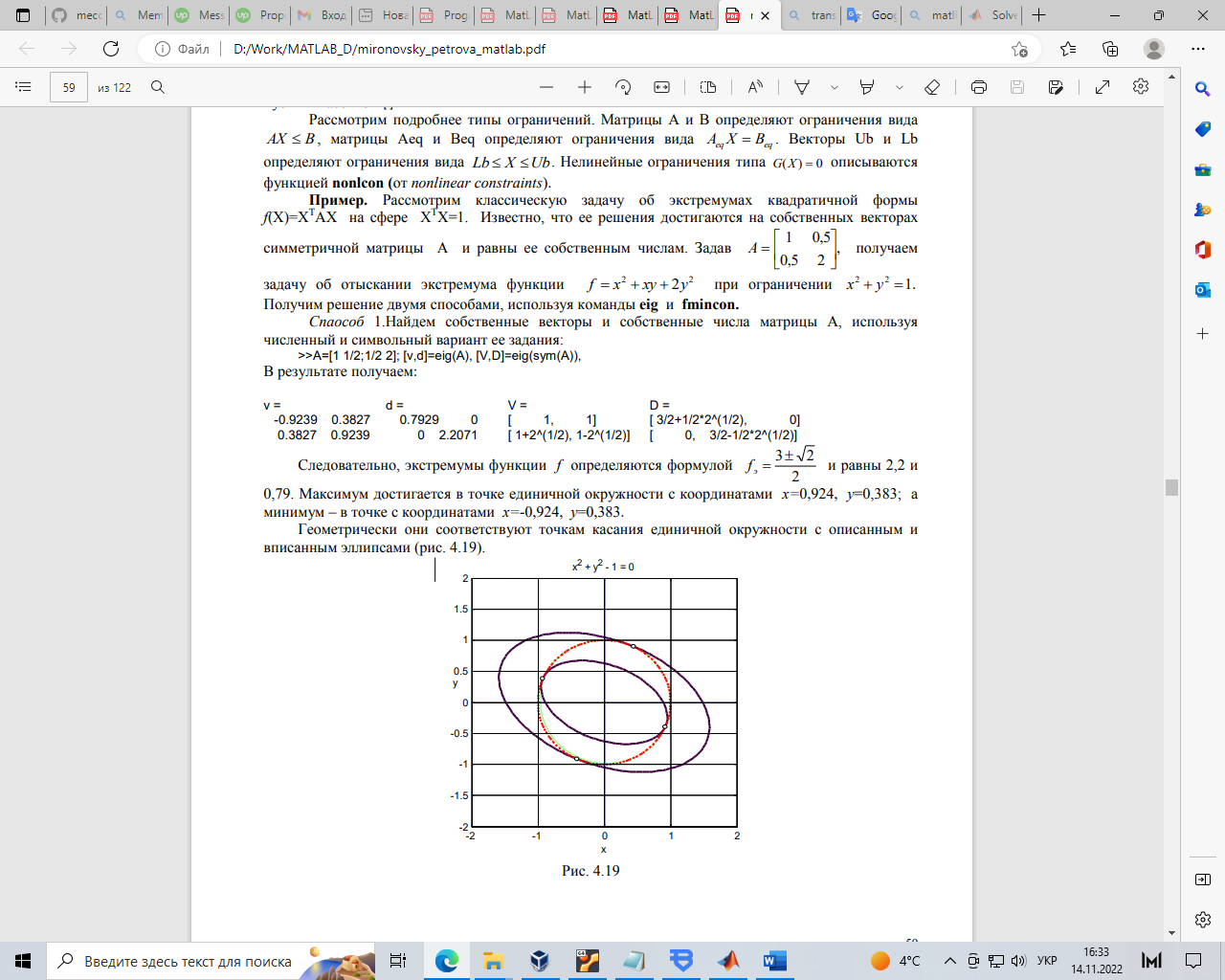
Матриці A і B визначають обмеження виду AX ≤ B,

матриці Aeq та Beq визначають обмеження виду AeqX = Beq.

Вектори Ub і Lb визначають обмеження виду Lb ≤ X ≤ Ub .

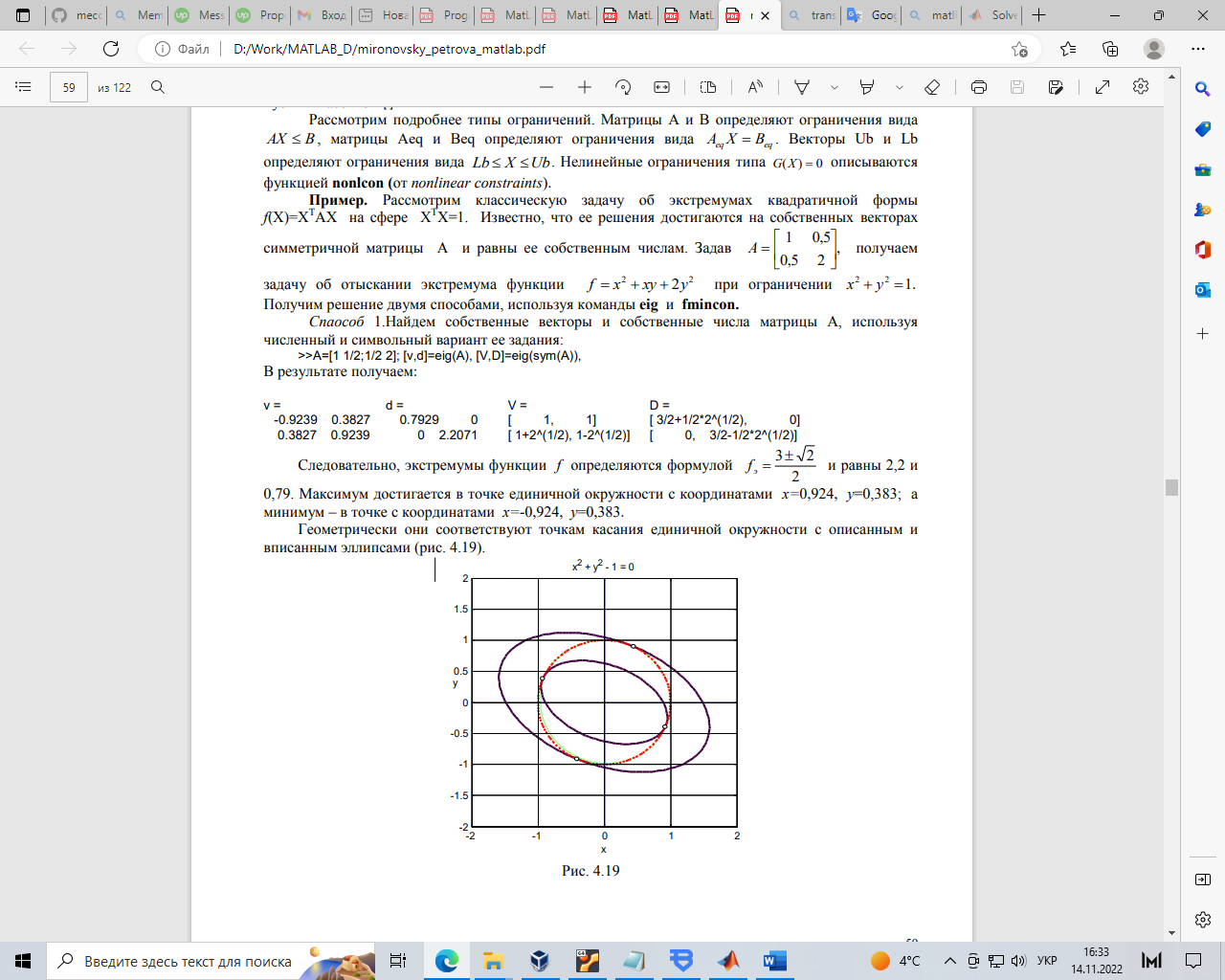
Нелінійні обмеження типу G(X) 0 описуються функцією nonlcon (від nonlinear constraints).

Приклад. Розглянемо класичне завдання екстремуми квадратичної форми f

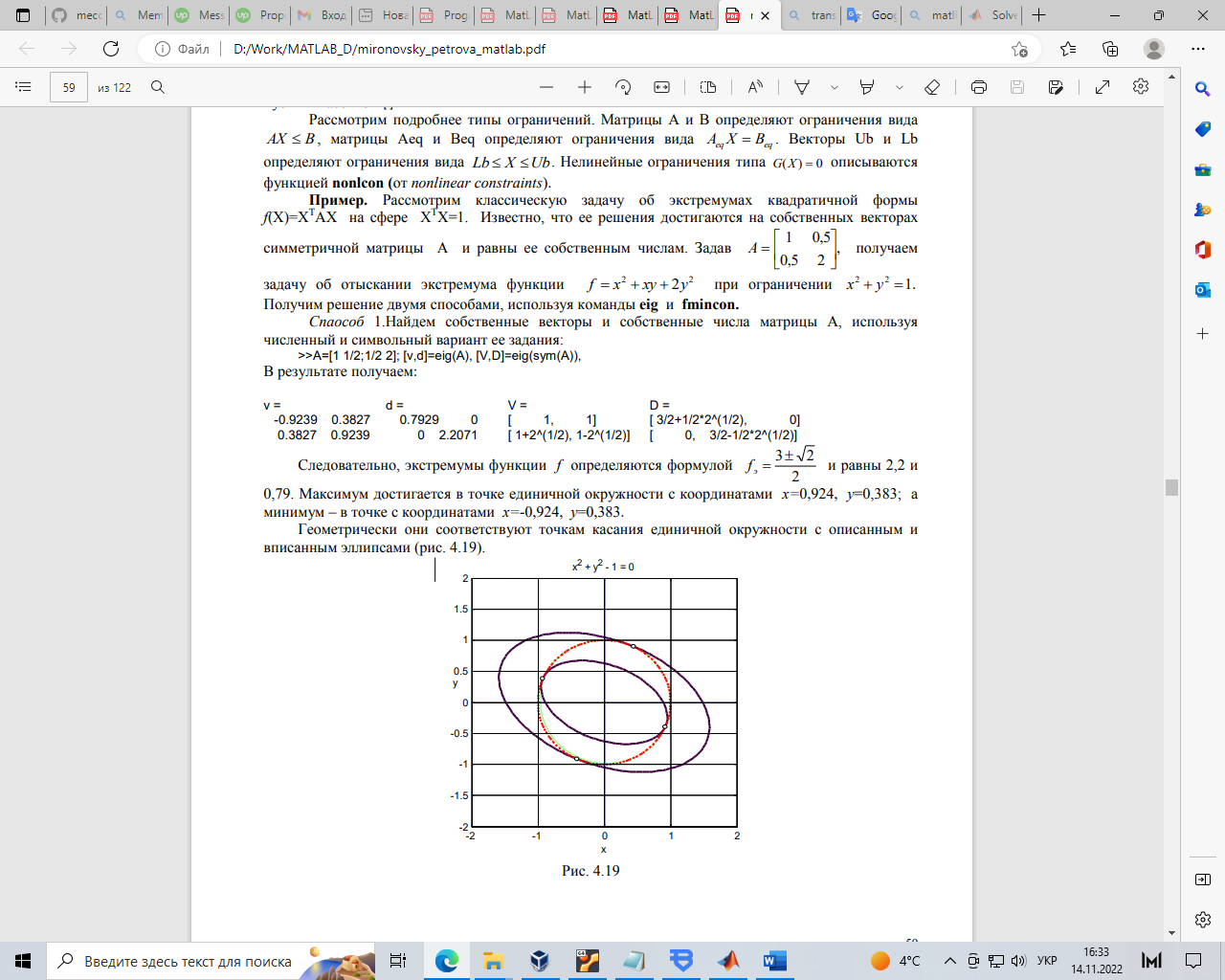


Відомо, що її рішення досягаються на власних векторах симетричної матриці А і дорівнюють її власним числам.

Задавши ,



отримуємо завдання про віднайдення екстремуму функції



Спосіб 1.Знайдемо власні вектори та власні числа матриці А, використовуючи чисельний та символьний варіант її завдання:

>>А=[1 1/2;1/2 2]; [v,d]=eig(А), [V,D]=eig(sym(А)),

A =

1.0000 0.5000

0.5000 2.0000

v =

-0.9239 0.3827

0.3827 0.9239

d =

0.7929 0

0 2.2071

V =

[- 2^(1/2) - 1, 2^(1/2) - 1]

[ 1, 1]

D =

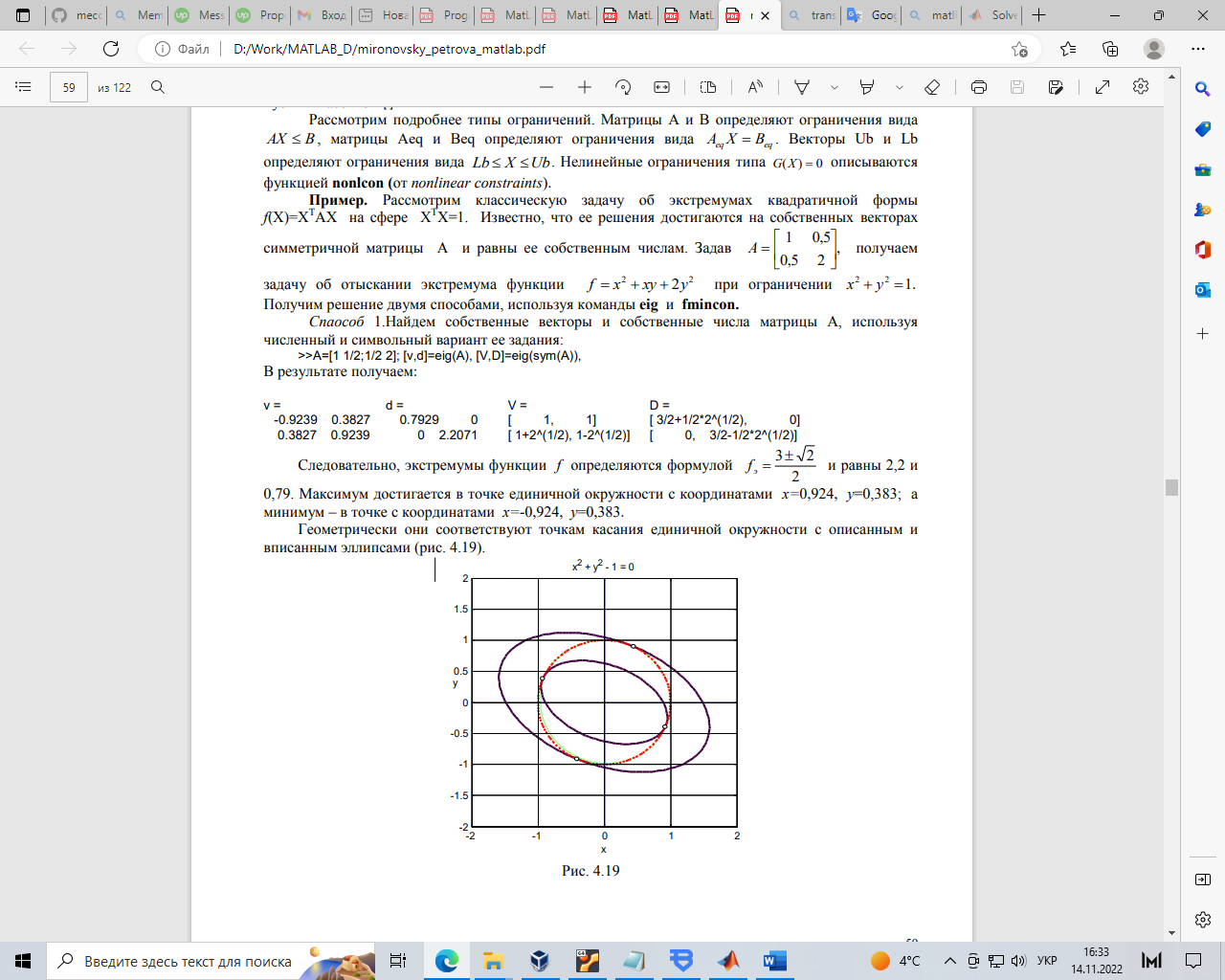
[3/2 - 2^(1/2)/2, 0]

[ 0, 2^(1/2)/2 + 3/2]

Отже, екстремуми функції f визначаються формулою 2 3± 2 f е = і дорівнюють 2,2 та 0,79.

Максимум досягається в точці одиничного кола з координатами х = 0,924, у = 0,383; а мінімум - у точці з координатами х = -0,924, у = 0,383.

Геометрично вони відповідають точкам торкання одиничного кола з описаним та вписаним еліпсами

Спосіб 2. Для застосування команди fmincon потрібно описати функцію 

Для опису першої з них скористаємося командою inline, для другої створимо m-файл nlcon.m:

function [h,g] = nlcon(x)

h = [];

g = x(1)^2+x(2)^2-1;

Після цього набираємо основну команду:

>> f=inline('x(1)^2+x(1)\*x(2)+2\*x(2)^2');

x=fmincon(@(x) f(x),[1;1],[],[],[],[],[],[],[[],@(x)nlcon(x)]) x = -0.9239 0.3827

Ми одержали координати точки мінімуму. Щоб отримати координати точки максимуму, змінюємо знак функції f:

f1=inline('-(x(1)^2+x(1)\*x(2)+2\*x(2)^2)');

x=fmincon(@(x) f1(x),[1;1],[],[],[],[],[],[],[[],@(x)nlcon(x)] )

x = 0.3827 0.9239

Бачимо, що обидва способи вирішення призвели до однакових результатів. Приклад використання функції fmincon для вибору оптимальних параметрів схеми моделювання,

Усі перелічені команди можуть повертати один, два та більше вихідних аргументів:

>> x=fminunc(@fff,[3 3]);

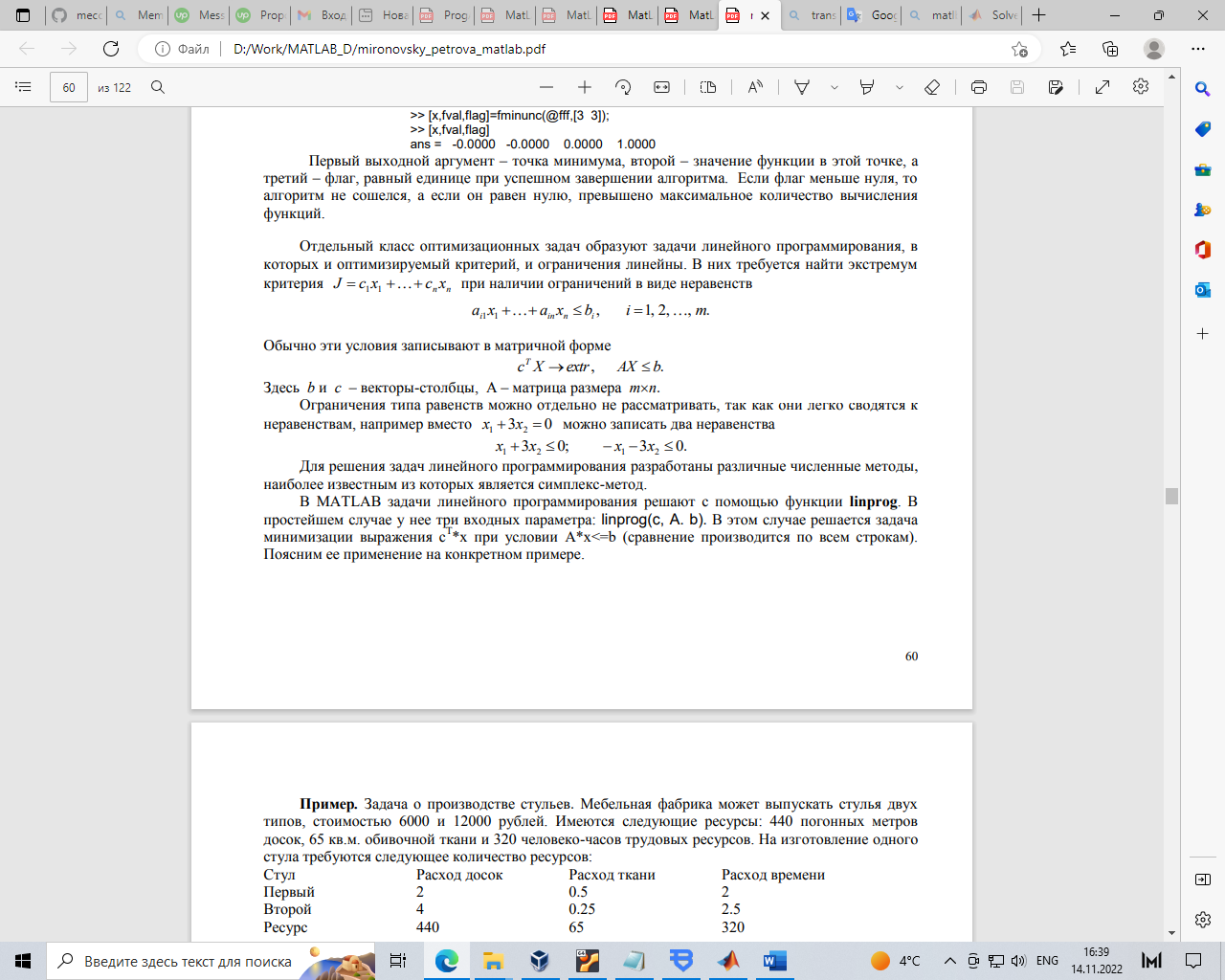
>> [x, fval] = fminunc (@ fff, [3 3]);

>> [x, fval, flag] = fminunc (@ fff, [3 3]);

>> [x,fval,flag] ans = -0.0000 -0.0000 0.0000 1.0000

Перший вихідний аргумент – точка мінімуму, другий – значення функції у цій точці, а третій – прапор, рівний одиниці за успішного завершення алгоритму. Якщо прапор менший за нуль, то алгоритм не зійшовся, а якщо він дорівнює нулю, перевищено максимальну кількість обчислення функцій.

Окремий клас оптимізаційних завдань утворюють завдання лінійного програмування, в яких і критерій, що оптимізується, і обмеження лінійні. У них потрібно знайти екстремум критерію за наявності обмежень у вигляді нерівностей :



Тут b та c – вектори-стовпці, А – матриця розміру m×n.

Обмеження типу рівностей можна окремо не розглядати, оскільки вони легко зводяться до нерівностей, наприклад замість

У MATLAB задачі лінійного програмування вирішують за допомогою функції linprog.

У найпростішому випадку вона має три вхідні параметри: linprog(с, A. b). У цьому випадку вирішується задача мінімізації виразу з

T \*x за умови A \* x <= b (порівняння проводиться у всіх рядках).

Приклад. Завдання про виробництво стільців. Меблева фабрика може випускати стільці двох типів, вартістю 6000 та 12000 рублів. Є такі ресурси: 440 погонних метрів дощок, 65 кв. оббивної тканини та 320 людино-годин трудових ресурсів.

На виготовлення одного випорожнення потрібна наступна кількість ресурсів:

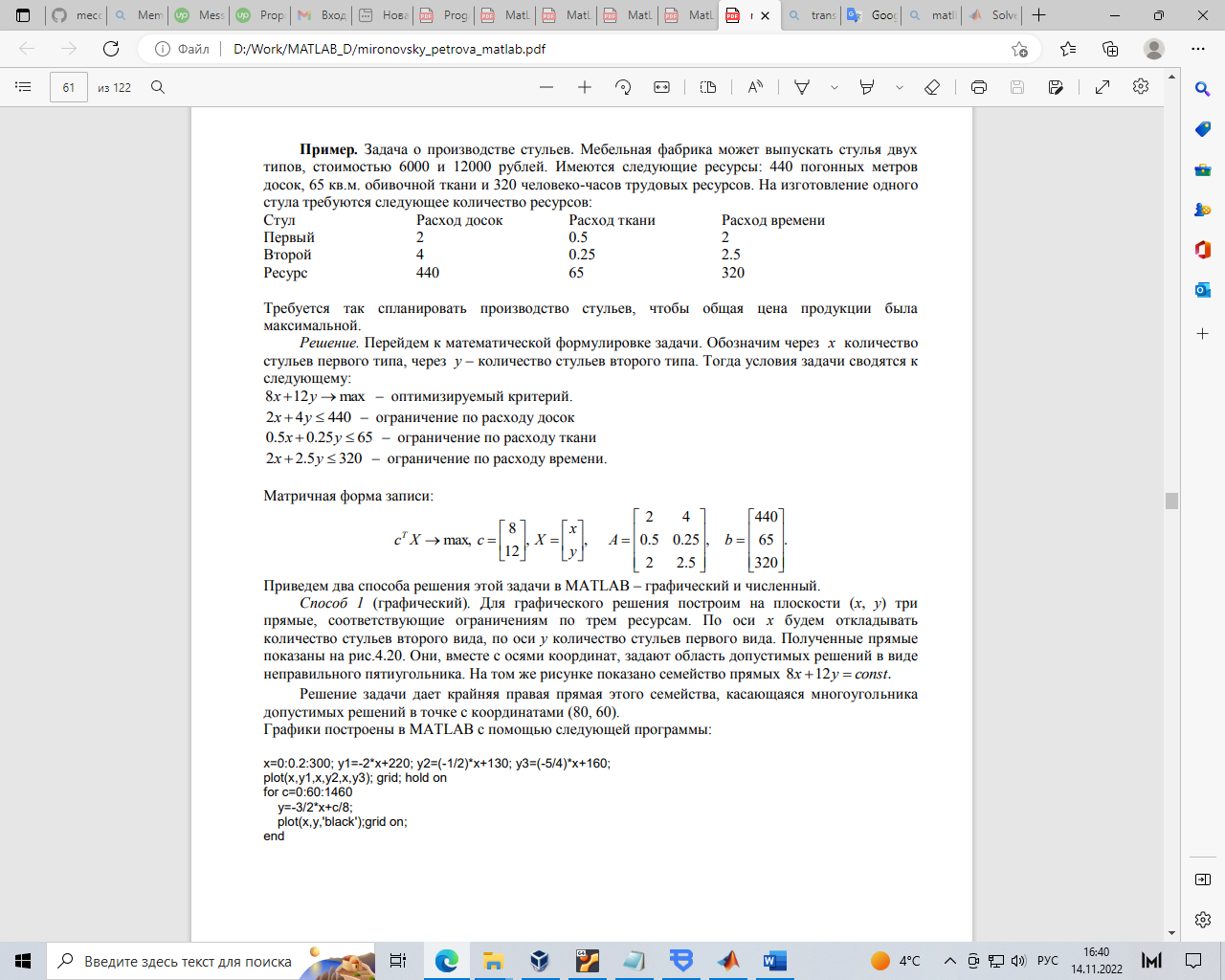
8x +12y → max – оптимізований критерий.

2x + 4y ≤ 440 – обмеження по расходу дошок

0.5x + 0.25y ≤ 65 – ограничение по расходу тканини

2x + 2.5y ≤ 320 – ліміт по часу.

Перейдемо до математичного формулювання завдання. Позначимо через x кількість стільців першого типу, через у – кількість стільців другого типу. Тоді умови завдання зводяться до наступного: 8x +12y → max – критерій, що оптимізується. 2x + 4y ≤ 440 – обмеження витрат дощок 0.5x + 0.25y ≤ 65 – обмеження витрат тканини 2x + 2.5y ≤ 320 – обмеження витрат часу



Для графічного рішення побудуємо на площині (x, y) три прямі, що відповідають обмеженням трьох ресурсів. По осі x відкладатимемо кількість стільців другого виду, по осі y кількість стільців першого виду. Отримані прямі показано на рис.4.20. Вони разом з осями координат задають область допустимих рішень у вигляді неправильного п'ятикутника. На тому ж малюнку показано сімейство прямих 8x +12y= const. Розв'язання задачі дає крайня права пряма цього сімейства, що стосується багатокутника допустимих рішень у точці з координатами (80, 60). Графіки побудовані МАТLAB за допомогою наступної програми:

x=0:0.2:300;

y1=-2\*x+220;

y2=(-1/2)\*x+130;

y3=(-5/4)\*x+160;

plot(x, y1, x, y2, x, y3); grid;

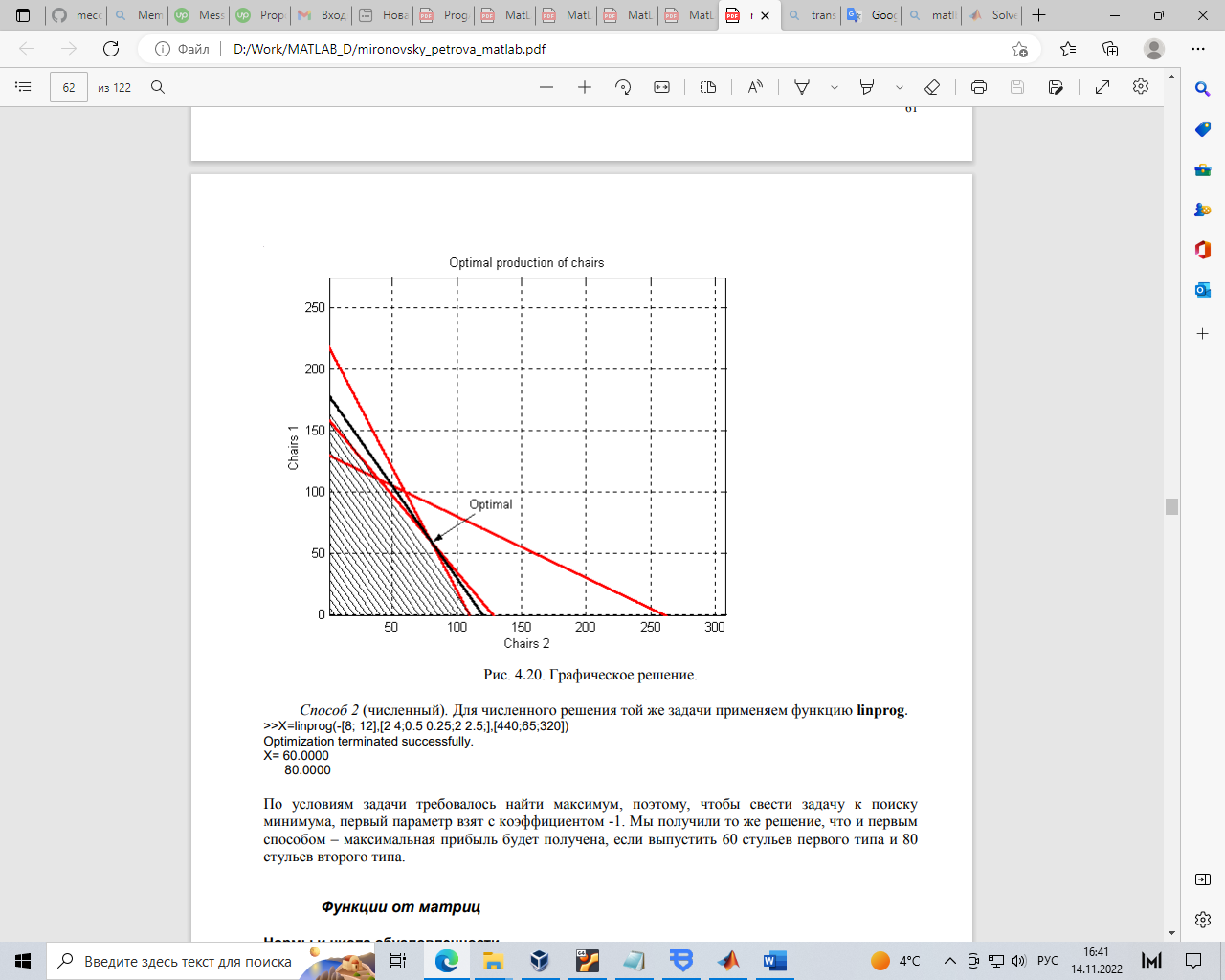
hold on for c=0:60:1460

y=-3/2\*x+c/8;

plot(x,y,'black');

grid on;

end



Спосіб 2 (чисельний). Для чисельного вирішення тієї ж задачі застосовуємо функцію linprog.

>>X=linprog(-[8; 12],[2 4;0.5 0.25;2 2.5;],[440;65;320])

Optimization terminated successfully.

X = 60.0000 80.0000

За умовами завдання потрібно знайти максимум, тому, щоб звести завдання до пошуку мінімуму, перший параметр взятий з коефіцієнтом -1. Ми отримали те ж рішення, що і першим способом – максимальний прибуток буде отримано, якщо випустити 60 стільців першого типу та 80 стільців другого ти