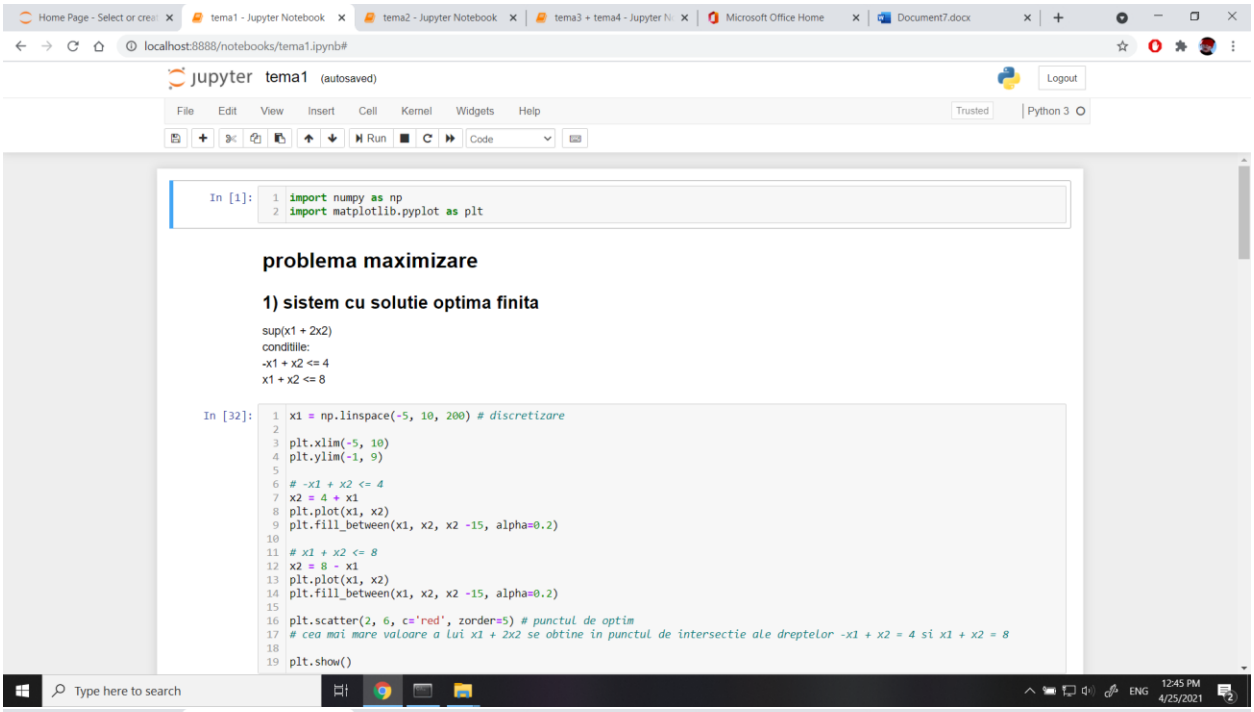
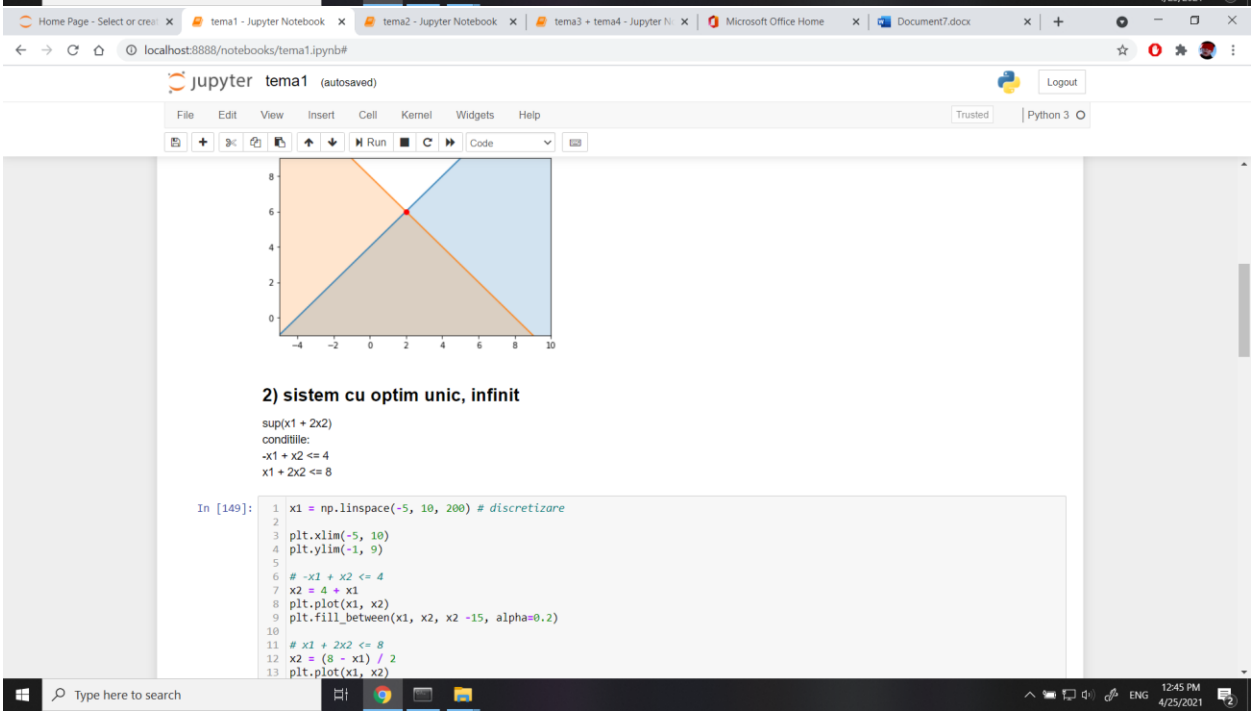
TEME TEHNICI OPTIMIZARE

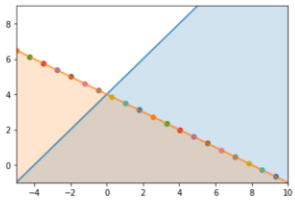
GRUPA 332 LUNGU ANDREI

TEMA 1





```
for i in range(0, len(x1), 10):
    plt.scatter(x1[i], x2[i])
plt.fill_between(x1, x2, x2 -15, alpha=0.2)

plt.show()
## toate punctele de pe dreapta portocalie sunt sol optime
```

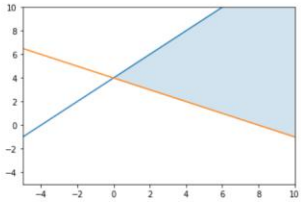


3) sistem cu optim infinit

sup(x1 + 2x2)
conditiile:
-x1 + x2 <= 4
x1 + 2x2 >= 8

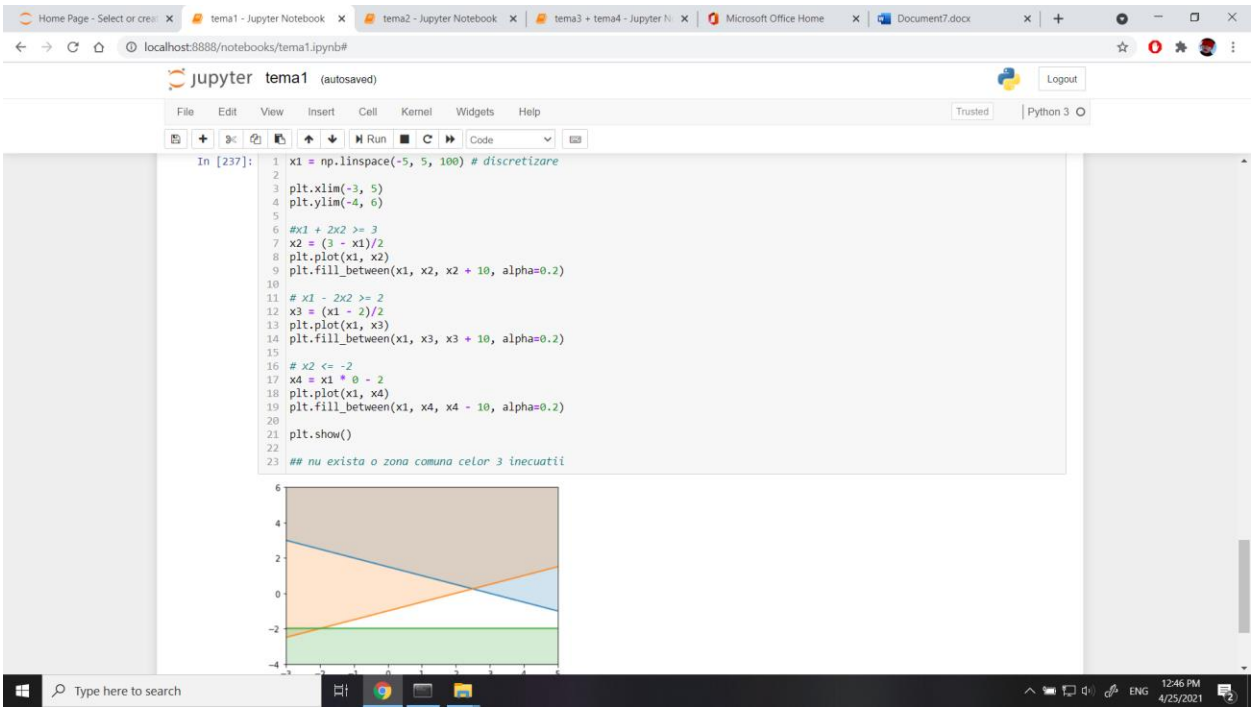
In [172]:
```
x1 = np.linspace(-5, 10, 100) # discretizare

plt.xlim(-5, 10)
plt.ylim(-5, 10)

# x1 + 2x2 >= 3
```

```
# x1 + 2x2 >= 3
x2 = 4+x1
plt.plot(x1, x2)

# x1 - 2x2 >= 2
x3 = (8 - x1)/2
plt.plot(x1, x3)

plt.fill_between(x1, x2, x3, where = x2>x3, alpha = 0.2)

plt.show()
## punctele din zona hasurata sunt sol optime
```



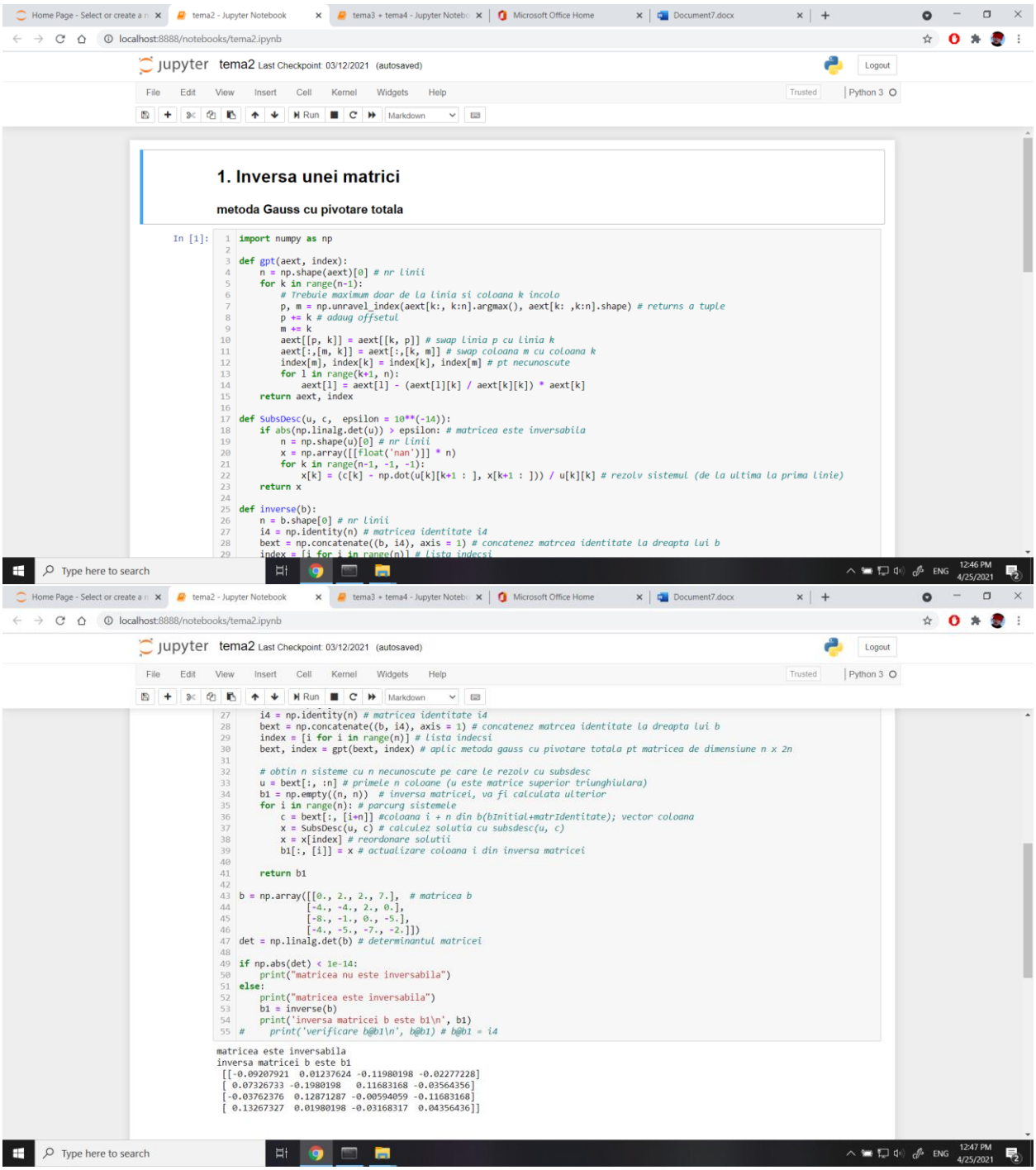
4) sistem incompatibil

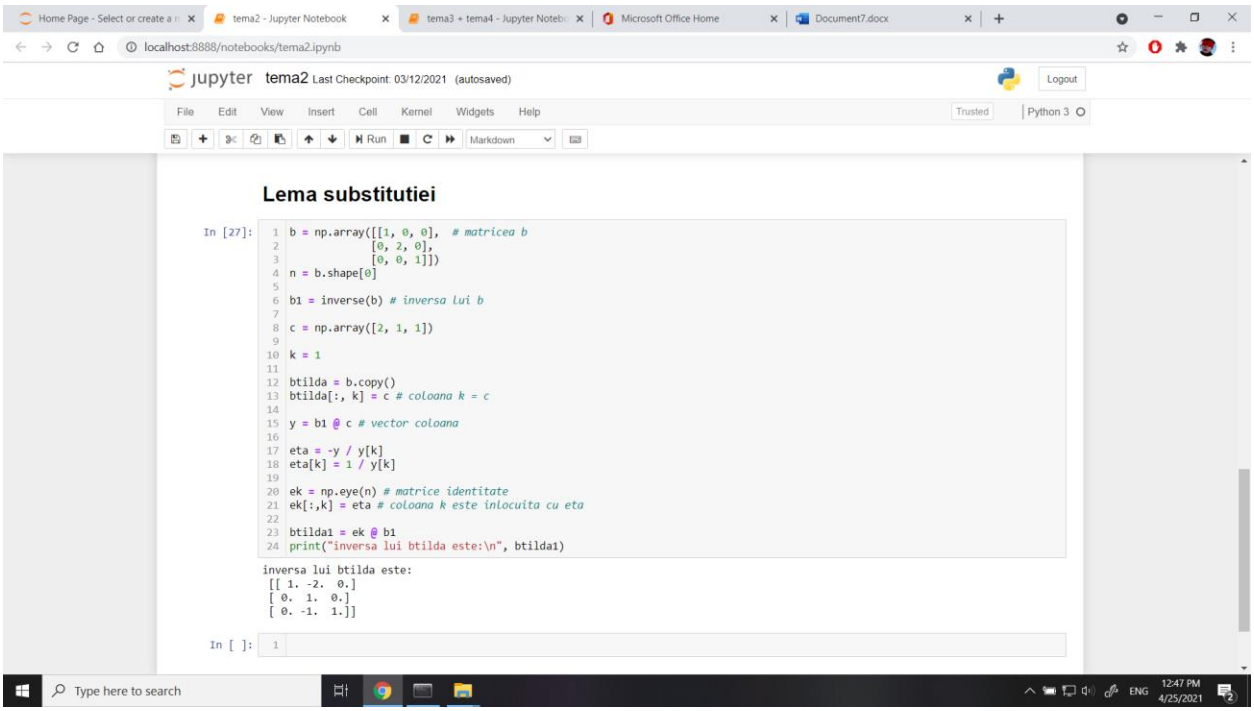
sup(x1 + x2)
conditiile:
x1 + 2x2 >= 3
x1 - 2x2 >= 2
x2 <= -2


Home Page - Select or crea
tema1 - Jupyter Notebook
tema2 - Jupyter Notebook
tema3 + tema4 - Jupyter N
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema1.ipynb#
jupyter tema1 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [237]:
x1 = np.linspace(-5, 5, 100) # discretizare
plt.xlim(-3, 5)
plt.ylim(-4, 6)
#x1 + 2x2 >= 3
x2 = (3 - x1)/2
plt.plot(x1, x2)
plt.fill_between(x1, x2, x2 + 10, alpha=0.2)
# x1 - 2x2 >= 2
x3 = (x1 - 2)/2
plt.plot(x1, x3)
plt.fill_between(x1, x3, x3 + 10, alpha=0.2)
# x2 <= -2
x4 = x1 * 0 - 2
plt.plot(x1, x4)
plt.fill_between(x1, x4, x4 - 10, alpha=0.2)
plt.show()
nu exista o zona comuna celor 3 inecuatii
Type here to search
ENG
12:46 PM
4/25/2021

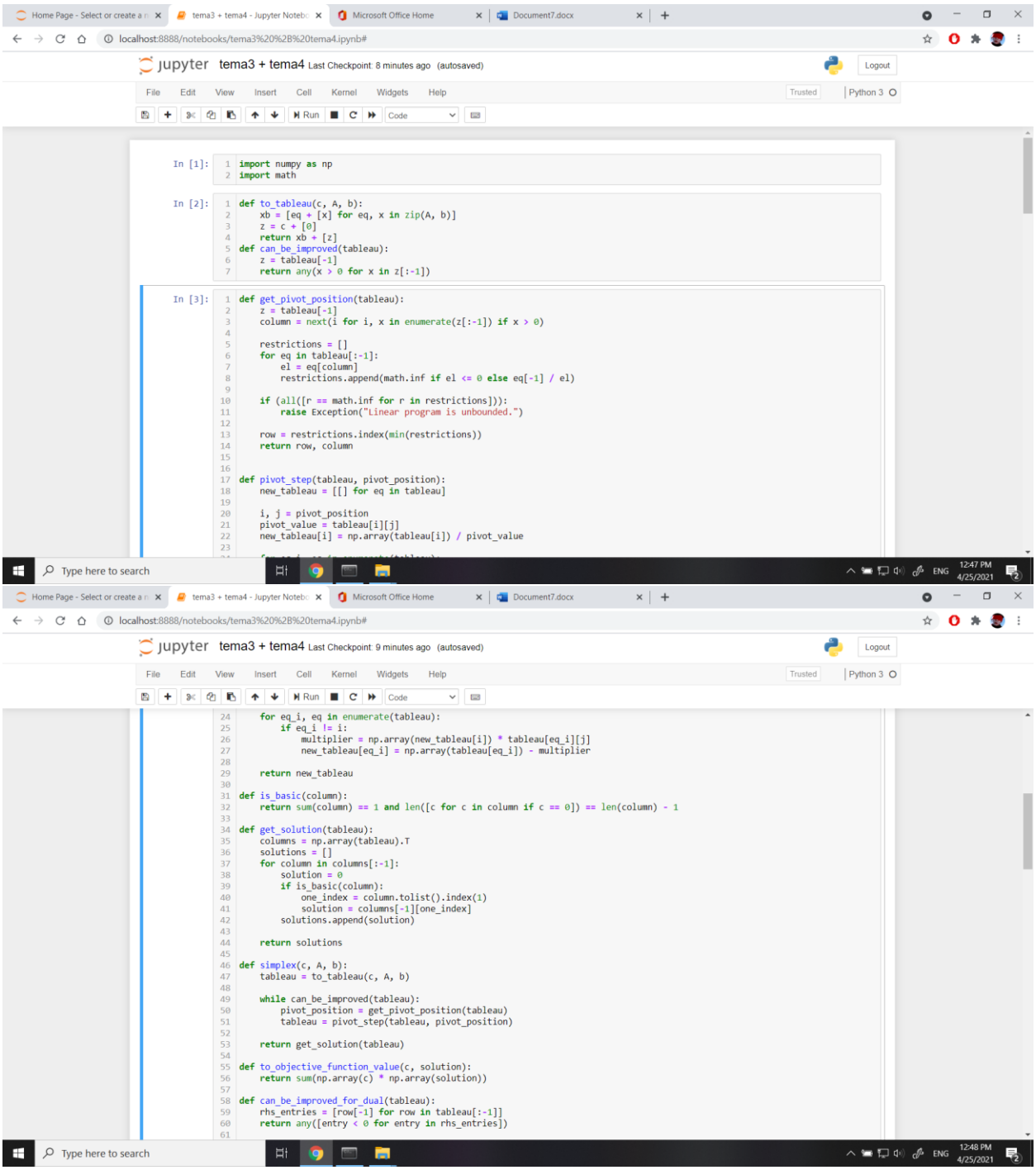
--- --- ---

TEMA 2


Home Page - Select or create a n
tema2 - Jupyter Notebook
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema2.ipynb
jupyter tema2 Last Checkpoint: 03/12/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
1. Inversa unei matrici
metoda Gauss cu pivotare totala
In [1]:
import numpy as np
def gpt(aext, index):
n = np.shape(aext)[0] # nr linii
for k in range(n-1):
# Trebuie maximum doar de la linia si coloana k incolo
p, m = np.unravel_index(aext[k:, k:n].argmax(), aext[k: ,k:n].shape) # returns a tuple
p += k # adaug offsetul
m += k
aext[[p, k]] = aext[[k, p]] # swap linia p cu linia k
aext[:,[m, k]] = aext[:,[k, m]] # swap coloana m cu coloana k
index[m], index[k] = index[k], index[m] # pt necunoscute
for l in range(k+1, n):
aext[l] = aext[l] - (aext[l][k] / aext[k][k]) * aext[k]
return aext, index
def SubsDesc(u, c, epsilon = 10**(-14)):
if abs(np.linalg.det(u)) > epsilon: # matricea este inversabila
n = np.shape(u)[0] # nr linii
x = np.array([[float('nan')]] * n)
for k in range(n-1, -1, -1):
x[k] = (c[k] - np.dot(u[k][k+1 :], x[k+1 :])) / u[k][k] # rezolv sistemul (de la ultima la prima linie)
return x
def inverse(b):
n = b.shape[0] # nr linii
i4 = np.identity(n) # matricea identitate i4
bext = np.concatenate((b, i4), axis = 1) # concatenez matrcea identitate la dreapta lui b
index = [i for i in range(n)] # lista indecsi
Type here to search
ENG
12:46 PM
4/25/2021
Home Page - Select or create a n
tema2 - Jupyter Notebook
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema2.ipynb
jupyter tema2 Last Checkpoint: 03/12/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
i4 = np.identity(n) # matricea identitate i4
bext = np.concatenate((b, i4), axis = 1) # concatenez matrcea identitate la dreapta lui b
index = [i for i in range(n)] # lista indecsi
bext, index = gpt(bext, index) # aplic metoda gauss cu pivotare totala pt matricea de dimensiune n x 2n
# obtin n sisteme cu n necunoscute pe care le rezolv cu subsdesc
u = bext[:, :n] # primele n coloane (u este matrice superior triunghiulara)
b1 = np.empty((n, n)) # inversa matricei, va fi calculata ulterior
for i in range(n): # parcurg sistemele
c = bext[:, [i+n]] #coloana i + n din b(bInitial+matrIdentitate); vector coloana
x = SubsDesc(u, c) # calculez solutia cu subsdesc(u, c)
x = x[index] # reordonare solutii
b1[:, [i]] = x # actualizare coloana i din inversa matricei
return b1
b = np.array([[0., 2., 2., 7.], # matricea b
[-4., -4., 2., 0.],
[-8., -1., 0., -5.],
[-4., -5., -7., -2.]])
det = np.linalg.det(b) # determinantul matricei
if np.abs(det) < 1e-14:
print("matricea nu este inversabila")
else:
print("matricea este inversabila")
b1 = inverse(b)
print('inversa matricei b este b1\n', b1)
# print('verificare b@b1\n', b@b1) # b@b1 = i4
matricea este inversabila
inversa matricei b este b1
[[-0.09207921 0.01237624 -0.11980198 -0.02277228]
[0.07326733 -0.1980198 0.11683168 -0.03564356]
[-0.03762376 0.12871287 -0.00594059 -0.11683168]
[0.13267327 0.01980198 -0.03168317 0.04356436]]
Type here to search
ENG
12:47 PM
4/25/2021


Home Page - Select or create a n
tema2 - Jupyter Notebook
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema2.ipynb
jupyter tema2 Last Checkpoint: 03/12/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Lema substitutiei
In [27]:
b = np.array([[1, 0, 0], # matricea b
[0, 2, 0],
[0, 0, 1]])
n = b.shape[0]
b1 = inverse(b) # inversa lui b
c = np.array([2, 1, 1])
k = 1
btilda = b.copy()
btilda[:, k] = c # coloana k = c
y = b1 @ c # vector coloana
eta = -y / y[k]
eta[k] = 1 / y[k]
ek = np.eye(n) # matrice identitate
ek[:,k] = eta # coloana k este inlocuita cu eta
btilda1 = ek @ b1
print("inversa lui btilda este:\n", btilda1)
inversa lui btilda este:
[[1. -2. 0.]
[0. 1. 0.]
[0. -1. 1.]]
In [ ]:
Type here to search
ENG
12:47 PM
4/25/2021

TEMA 3 + TEMA 4



Home Page - Select or create a n | tema3 + tema4 - Jupyter Noteb | Microsoft Office Home | Document7.docx

localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#

jupyter tema3 + tema4 Last Checkpoint: 8 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [1]:
```
import numpy as np
import math
```

In [2]:
```
def to_tableau(c, A, b):
    xb = [eq + [x] for eq, x in zip(A, b)]
    z = c + [0]
    return xb + [z]
def can_be_improved(tableau):
    z = tableau[-1]
    return any(x > 0 for x in z[:-1])
```

In [3]:
```
def get_pivot_position(tableau):
    z = tableau[-1]
    column = next(i for i, x in enumerate(z[:-1]) if x > 0)

    restrictions = []
    for eq in tableau[:-1]:
        el = eq[column]
        restrictions.append(math.inf if el <= 0 else eq[-1] / el)

    if (all([r == math.inf for r in restrictions])):
        raise Exception("Linear program is unbounded.")

    row = restrictions.index(min(restrictions))
    return row, column


def pivot_step(tableau, pivot_position):
    new_tableau = [[] for eq in tableau]

    i, j = pivot_position
    pivot_value = tableau[i][j]
    new_tableau[i] = np.array(tableau[i]) / pivot_value
```

Type here to search ENG 12:47 PM 4/25/2021

Home Page - Select or create a n | tema3 + tema4 - Jupyter Noteb | Microsoft Office Home | Document7.docx

localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#

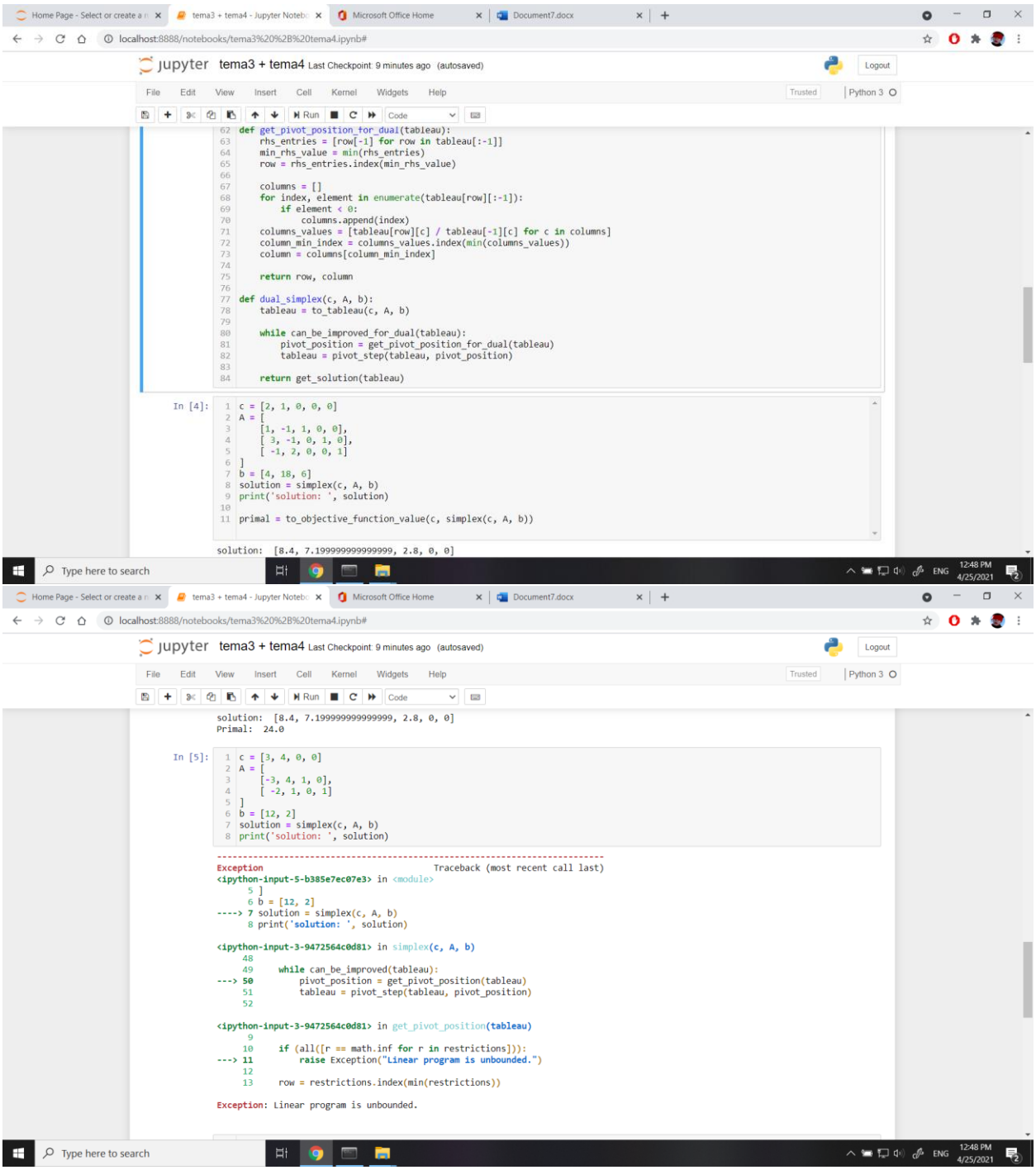
jupyter tema3 + tema4 Last Checkpoint: 9 minutes ago (autosaved) Logout

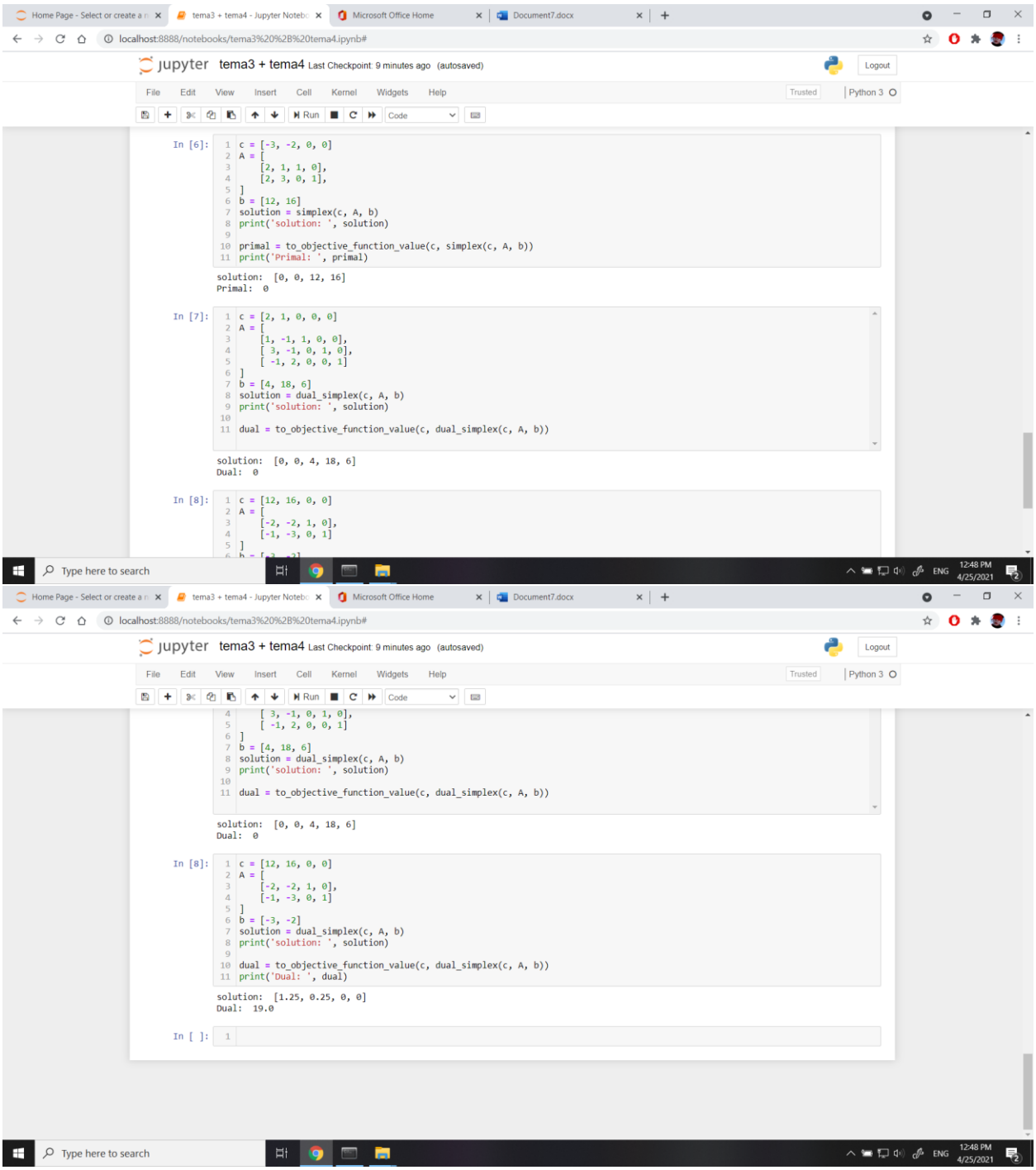
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
    for eq_i, eq in enumerate(tableau):
        if eq_i != i:
            multiplier = np.array(new_tableau[i]) * tableau[eq_i][j]
            new_tableau[eq_i] = np.array(tableau[eq_i]) - multiplier

    return new_tableau

def is_basic(column):
    return sum(column) == 1 and len([c for c in column if c == 0]) == len(column) - 1

def get_solution(tableau):
    columns = np.array(tableau).T
    solutions = []
    for column in columns[:-1]:
        solution = 0
        if is_basic(column):
            one_index = column.tolist().index(1)
            solution = columns[-1][one_index]
        solutions.append(solution)

    return solutions

def simplex(c, A, b):
    tableau = to_tableau(c, A, b)

    while can_be_improved(tableau):
        pivot_position = get_pivot_position(tableau)
        tableau = pivot_step(tableau, pivot_position)

    return get_solution(tableau)

def to_objective_function_value(c, solution):
    return sum(np.array(c) * np.array(solution))

def can_be_improved_for_dual(tableau):
    rhs_entries = [row[-1] for row in tableau[:-1]]
    return any([entry < 0 for entry in rhs_entries])
```

Type here to search ENG 12:48 PM 4/25/2021


Home Page - Select or create a n
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#
jupyter tema3 + tema4 Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
def get_pivot_position_for_dual(tableau):
rhs_entries = [row[-1] for row in tableau[:-1]]
min_rhs_value = min(rhs_entries)
row = rhs_entries.index(min_rhs_value)
columns = []
for index, element in enumerate(tableau[row][:-1]):
if element < 0:
columns.append(index)
columns_values = [tableau[row][c] / tableau[-1][c] for c in columns]
column_min_index = columns_values.index(min(columns_values))
column = columns[column_min_index]
return row, column
def dual_simplex(c, A, b):
tableau = to_tableau(c, A, b)
while can_be_improved_for_dual(tableau):
pivot_position = get_pivot_position_for_dual(tableau)
tableau = pivot_step(tableau, pivot_position)
return get_solution(tableau)
In [4]:
c = [2, 1, 0, 0, 0]
A = [
[1, -1, 1, 0, 0],
[3, -1, 0, 1, 0],
[-1, 2, 0, 0, 1]
]
b = [4, 18, 6]
solution = simplex(c, A, b)
print('solution: ', solution)
primal = to_objective_function_value(c, simplex(c, A, b))
solution: [8.4, 7.199999999999999, 2.8, 0, 0]
Type here to search
ENG
12:48 PM
4/25/2021
Home Page - Select or create a n
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#
jupyter tema3 + tema4 Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
solution: [8.4, 7.199999999999999, 2.8, 0, 0]
Primal: 24.0
In [5]:
c = [3, 4, 0, 0]
A = [
[-3, 4, 1, 0],
[-2, 1, 0, 1]
]
b = [12, 2]
solution = simplex(c, A, b)
print('solution: ', solution)
Exception Traceback (most recent call last)
<ipython-input-5-b385e7ec07e3> in <module>
5]
6 b = [12, 2]
----> 7 solution = simplex(c, A, b)
8 print('solution: ', solution)
<ipython-input-3-9472564c0d81> in simplex(c, A, b)
48
49 while can_be_improved(tableau):
---> 50 pivot_position = get_pivot_position(tableau)
51 tableau = pivot_step(tableau, pivot_position)
52
<ipython-input-3-9472564c0d81> in get_pivot_position(tableau)
9
10 if (all([r == math.inf for r in restrictions])):
---> 11 raise Exception("Linear program is unbounded.")
12
13 row = restrictions.index(min(restrictions))
Exception: Linear program is unbounded.
Type here to search
ENG
12:48 PM
4/25/2021


Home Page - Select or create a n
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#
jupyter tema3 + tema4 Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [6]:
c = [-3, -2, 0, 0]
A = [
[2, 1, 1, 0],
[2, 3, 0, 1],
]
b = [12, 16]
solution = simplex(c, A, b)
print('solution: ', solution)
primal = to_objective_function_value(c, simplex(c, A, b))
print('Primal: ', primal)
solution: [0, 0, 12, 16]
Primal: 0
In [7]:
c = [2, 1, 0, 0, 0]
A = [
[1, -1, 1, 0, 0],
[3, -1, 0, 1, 0],
[-1, 2, 0, 0, 1]
]
b = [4, 18, 6]
solution = dual_simplex(c, A, b)
print('solution: ', solution)
dual = to_objective_function_value(c, dual_simplex(c, A, b))
solution: [0, 0, 4, 18, 6]
Dual: 0
In [8]:
c = [12, 16, 0, 0]
A = [
[-2, -2, 1, 0],
[-1, -3, 0, 1]
]
Type here to search
ENG
12:48 PM
4/25/2021
Home Page - Select or create a n
tema3 + tema4 - Jupyter Noteb
Microsoft Office Home
Document7.docx
localhost:8888/notebooks/tema3%20%2B%20tema4.ipynb#
jupyter tema3 + tema4 Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
[3, -1, 0, 1, 0],
[-1, 2, 0, 0, 1]
]
b = [4, 18, 6]
solution = dual_simplex(c, A, b)
print('solution: ', solution)
dual = to_objective_function_value(c, dual_simplex(c, A, b))
solution: [0, 0, 4, 18, 6]
Dual: 0
In [8]:
c = [12, 16, 0, 0]
A = [
[-2, -2, 1, 0],
[-1, -3, 0, 1]
]
b = [-3, -2]
solution = dual_simplex(c, A, b)
print('solution: ', solution)
dual = to_objective_function_value(c, dual_simplex(c, A, b))
print('Dual: ', dual)
solution: [1.25, 0.25, 0, 0]
Dual: 19.0
In [ ]:
Type here to search
ENG
12:48 PM
4/25/2021