Homework 7

Ben Ridenhour

08-Nov-21

Contents

For the Chapter 7 homework, you were asked to perform the optimization (maximization and minimization) using the Simplex algorithm of various linear programs in the book. The questions are given on pg. 278 of the book, and you were to do problems 8–12. Additionally, you were asked plot each system and the progression of the algorithm at each step.

The Simplex Algorithm

First, while not specified, it would probably be easiest to code the algorithm up as a function. For every system, we need to specify the following:

1. The matrix **A**
2. The cost (optimization) vector **c**
3. The starting coordinates for the optimization **x**
4. Whether we want to perform maximization or minimization

and

5. (Optionally) Whether to return the points the algorithm passed through.

Let's create this function first:

```
import numpy as np
import numpy.linalg as nla

def Simplex(A, c, x, Max = True, vals = False):
  Cols = np.arange(A.shape[1])
  B = np.where(x != 0)[0] #x should be a numpy array for ths to work
  N = np.setdiff1d(Cols,B)
  X = x.copy()
  out = X.copy().reshape(1,-1)

  enter = 1

  while enter >= 0:
    Lambda = nla.inv(A[:,B].transpose()) @ c[B]
```

```
    sn = c[N] - A[:,N].transpose() @ Lambda

    sn_pos = sn[sn > 0]
    sn_neg = sn[sn < 0]

    enter = -1

    if Max & (sn_pos.size >  0): enter = N[np.argmax(sn)]
    if ~Max & (sn_neg.size >  0): enter = N[np.argmin(sn)]

    if enter == -1: break

    d = nla.inv(A[:,B]) @ A[:,enter]
    ratios = X[B]/d #get the ratios for the ratio test

    multiplier = np.min(ratios[d>0]) #value of exiting index
    Exit = B[ratios==multiplier]

    X[enter] = multiplier
    X[B] = X[B] - d*multiplier

    out = np.vstack([out, X])

    B = np.union1d(np.setdiff1d(B,Exit),enter)
    B.sort()
    N = np.setdiff1d(np.arange(A.shape[1]), B)

  if vals: return out

  return out[-1,:]
```

Alright, we should be able to get everything we need from that function

Problem 8

The linear program to optimize is

$$\begin{aligned} \text{Optimize} \quad & 2x + 3y \\ & \text{subject to} \\ & 2x + 3y \geq 6 \\ & 3x - y \leq 15 \\ & -x + y \leq 4 \\ & 2x + 5y \leq 27 \\ & x, y \geq 0 \end{aligned}$$

Note that the first constraint equation is $\geq$ and needs to be converted to $\leq$, giving $-2x - 3y \leq -6$.

```
A =  np.array([
  -2.,  -3, 1, 0, 0, 0,
  3, -1, 0, 1, 0, 0,
  -1, 1, 0, 0, 1, 0,
```

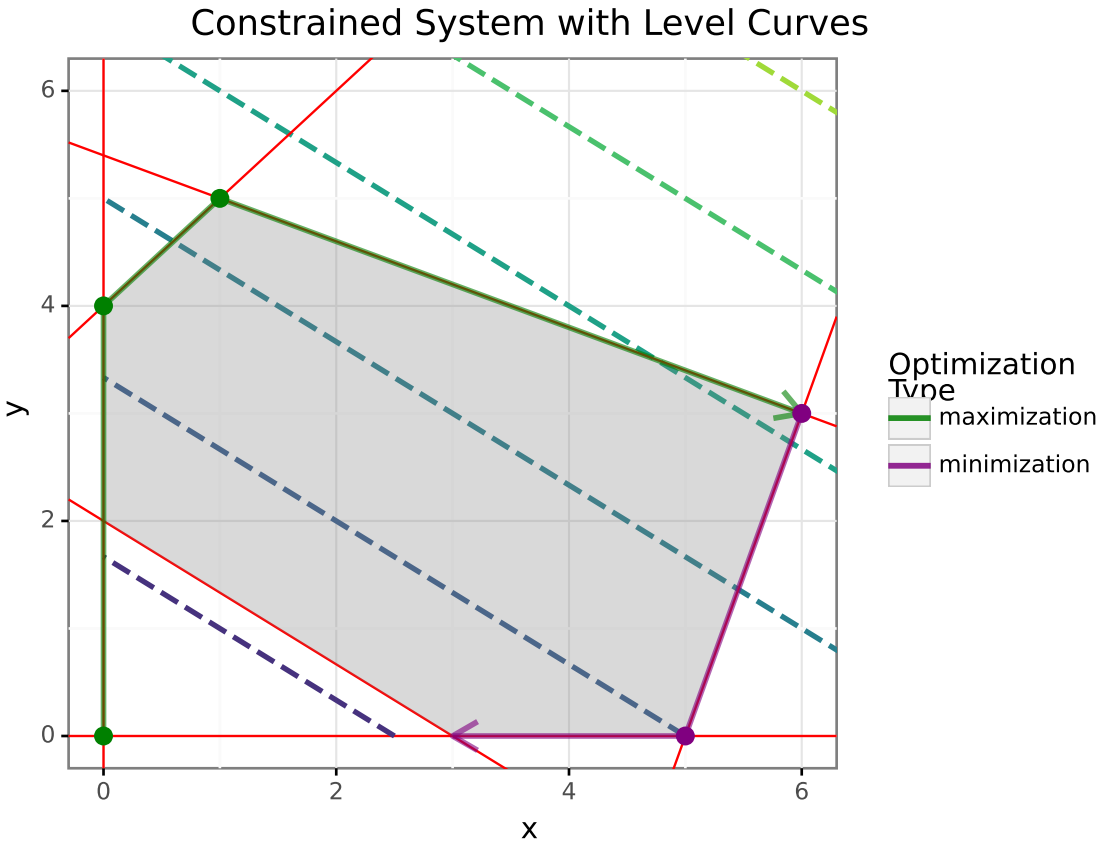
```
  2, 5, 0, 0, 0, 1]).reshape(4,-1)

C = np.array([2., 3, 0, 0, 0, 0])
x = np.array([0, 0, -6., 15, 4, 27])


maximum = Simplex(A,C,x,vals = True) #last row has the maximum
minimum = Simplex(A,C,maximum[-1,:], Max = False, vals = True)

##plot it

import plotnine as p9
import matplotlib.pyplot as plt
import pandas as pd

### A plot of the system
data = pd.merge(pd.DataFrame({'x': np.linspace(0,7,50), 'key':1}), pd.DataFrame({'y': np.linspace(0,7,5
```

```
## <string>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for
```

```
data['z'] = 2*data.x + 3*data.y

shadeMe = pd.DataFrame({'x':[3,5,6,1,0,0], 'y':[0,0,3,5,4,2]})

maxD = pd.DataFrame({'x':maximum[:,0], 'y':maximum[:,1]})
minD = pd.DataFrame({'x':minimum[:,0], 'y':minimum[:,1]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 2, slope = -2/3), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = -15, slope = 3), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = 4, slope = 1), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = 27/5, slope = -2/5), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.geom_path(data=maxD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("green", maxD.shape[0])
  + p9.geom_point(data = maxD.drop(maxD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, col
  + p9.geom_path(data=minD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("purple", minD.shape[0]
  + p9.geom_point(data = minD.drop(minD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, col
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75})+ p9.ggtitle("Constrained System with Leve
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshap
```

```
## <matplotlib.contour.QuadContourSet object at 0x18e284fa0>
```

```
plt.show()
```




```
plt.close('all')
```

The algorithm has appropriate identify the maximum that occurs at $(6, 3)$ and a minimum that occurs at $(3, 0)$. Let's quickly try the minimization starting from a different extreme point, specifically $\mathbf{x}^T = \begin{bmatrix} 1 & 5 & -11 & 17 & 0 & 0 \end{bmatrix}$.

```
x = np.array([1., 5, 11, 17, 0, 0])

minimum = Simplex(A,C,x, Max = False, vals=True) #last row has the minimum
minimum[-1,:]
```

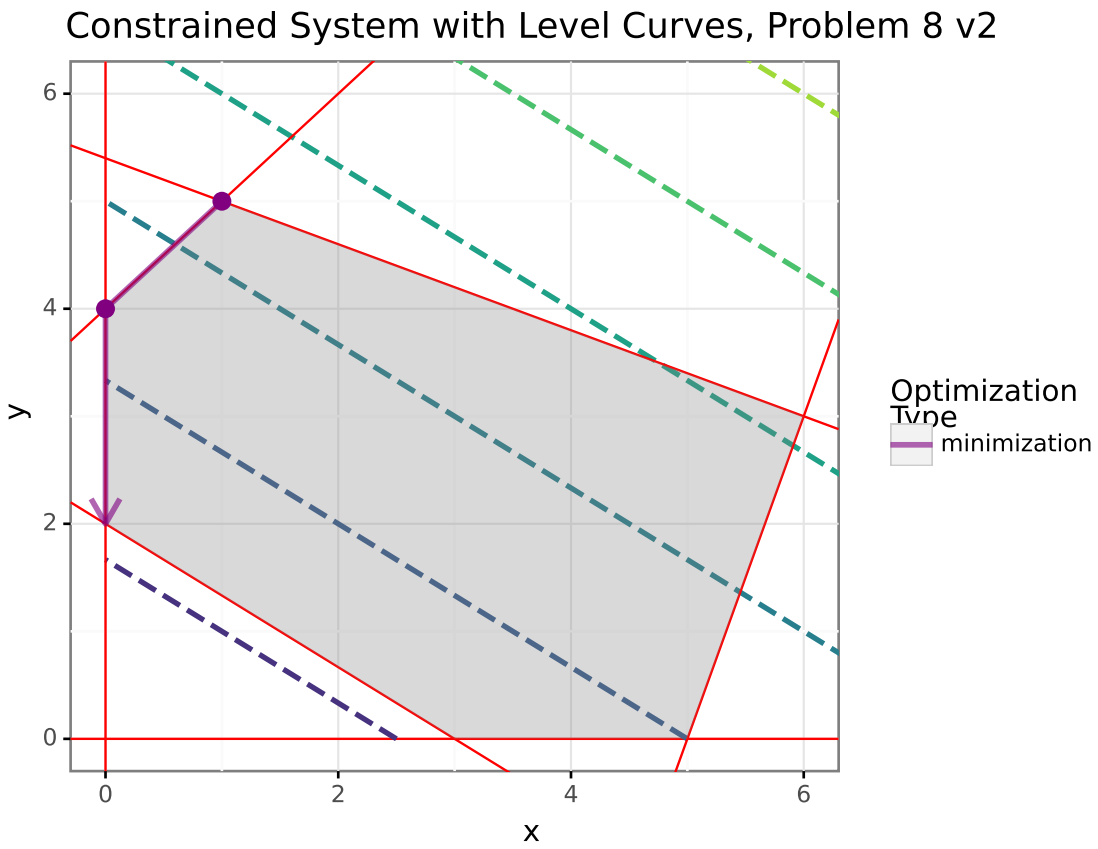
```
## array([ 0.,  2.,  0., 17.,  2., 17.])
```

```
minD = pd.DataFrame({'x':minimum[:,0], 'y':minimum[:,1]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 2, slope = -2/3), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = -15, slope = 3), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = 4, slope = 1), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = 27/5, slope = -2/5), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.geom_path(data=minD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("purple", minD.shape[0]]
  + p9.geom_point(data = minD.drop(minD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, colo
```

```
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Leve
  + p9.scale_color_identity(name = "Optimization\nType",labels = ["minimization"], guide = "legend")
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshape
```

```
## <matplotlib.contour.QuadContourSet object at 0x1906245e0>
```

```
plt.show()
```




```
plt.close('all')
```

Starting from this alternate extreme point, the Simplex algorithm identifies a different minimum $(0, 2)$. Is this okay? Why?

Problem 9

The linear program to optimize is

$$
\begin{aligned}
\text{Optimize} \quad & 6x + 4y \\
\text{subject to} \quad & \\
-x + y &\leq 12 \\
x + y &\leq 24 \\
2x + 5y &\leq 80 \\
x, y &\geq 0
\end{aligned}
$$

Rather than starting the at the origin (which is somewhat boring for this system), let's start at a different extreme point, specifically $\mathbf{x}^T = \begin{bmatrix} 0 & 12 & 0 & 12 & 20 \end{bmatrix}$

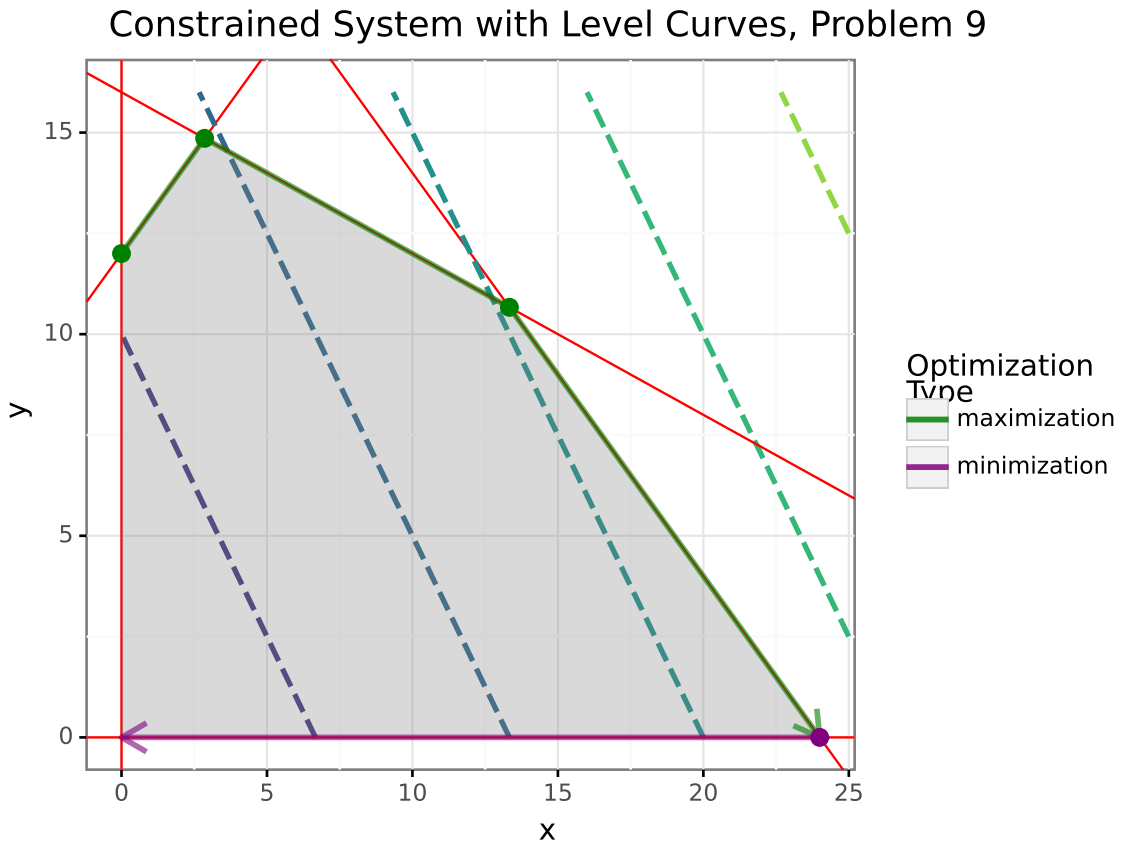
```
A =  np.array([
  -1., 1, 1, 0, 0,
  1, 1, 0, 1, 0,
  2, 5, 0, 0, 1]).reshape(3,-1)

C = np.array([6., 4, 0, 0, 0])
x = np.array([0, 12., 0, 12, 20])

maximum = Simplex(A,C,x,vals = True) #last row has the maximum
###getting some floating point non-sense:
maximum = np.round(maximum,5)
minimum = Simplex(A,C,maximum[-1,:], Max = False, vals = True)

### A plot of the system
data = pd.merge(pd.DataFrame({'x': np.linspace(0,25,50), 'key':1}), pd.DataFrame({'y': np.linspace(0,16
```

```
## <string>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for
```

```
data['z'] = 6*data.x + 4*data.y

shadeMe = pd.DataFrame({'x':[20/7,40/3,24,0,0], 'y':[104/7,32/3,0,0,12]})

maxD = pd.DataFrame({'x':maximum[:,0], 'y':maximum[:,1]})
minD = pd.DataFrame({'x':minimum[:,0], 'y':minimum[:,1]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 12, slope = 1), color = "red")
  + p9.geom_abline(mapping=p9.aes(intercept = 24, slope = -1), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_abline(mapping=p9.aes(intercept = 16, slope = -2/5), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.geom_path(data=maxD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("green", maxD.shape[0])
  + p9.geom_point(data = maxD.drop(maxD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, col
  + p9.geom_path(data=minD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("purple", minD.shape[0]
  + p9.geom_point(data = minD.drop(minD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, col
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Lev
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshap
```

```
## <matplotlib.contour.QuadContourSet object at 0x1906fe3d0>
```

```
plt.show()
```




```
plt.close('all')
```

Again, we see that simple algorithm appropriately identifies both the minimum $(0, 0)$ and maximum $(24, 0)$ by examining the values of the optimization function at different vertices of the convex polytope.

Problem 10

The linear program to optimize is

$$\begin{aligned} \text{Optimize} \quad & 6x + 5y \\ \text{subject to} \quad & \\ & x + y \geq 6 \\ & 2x + y \geq 9 \\ & x, y \geq 0 \end{aligned}$$

Because all of the equations are expressed as $\geq$, let's plot this system first and see what it looks like.

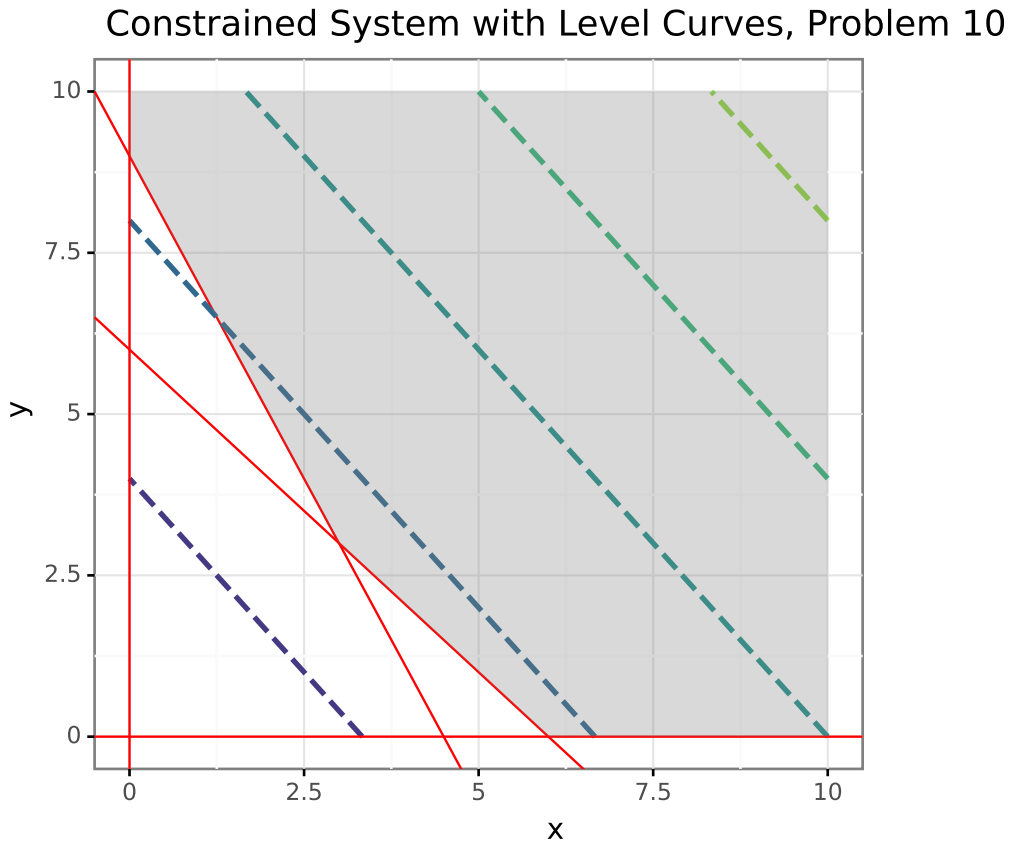
```
### A plot of the system
data = pd.merge(pd.DataFrame({'x': np.linspace(0,10,50), 'key':1}), pd.DataFrame({'y': np.linspace(0,10
```

```
## <string>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for
```

```
data['z'] = 6*data.x + 5*data.y

shadeMe = pd.DataFrame({'x':[3,6,10,10,0,0], 'y':[3,0,0,10,10,9]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 6, slope = -1), color = "red")
  + p9.geom_abline(mapping=p9.aes(intercept = 9, slope = -2), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Leve
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshape
```

```
## <matplotlib.contour.QuadContourSet object at 0x1907401f0>
```

```
plt.show()
```




```
plt.close('all')
```

Obviously, there is a bit of problem with this system: it is **unbounded**. Let's try our algorithm and see

what it does:

```
A =  np.array([
  -1., -1, 1, 0,
  -2, -1, 0, 1
  ]).reshape(2,-1)

C = np.array([6., 5, 0, 0])
x = np.array([0, 0, -6., -9])

maximum <= Simplex(A,C,x,vals = True) #fails due to unbounded problem
```

```
## Error in py_call_impl(callable, dots$args, dots$keywords): ValueError: zero-size array to reduction o
##
## Detailed traceback:
##   File "<string>", line 1, in <module>
##   File "<string>", line 27, in Simplex
##   File "<__array_function__ internals>", line 5, in amin
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return _wrapreduction(a, np.minimum, 'min', axis, None, out,
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

The algorithm fails to find a maximum due to the unbounded nature of the problem. Let's try a minimum starting from the origin:

```
x = np.array([0, 0, -6, -9.])
minimum = Simplex(A,C,x, Max = False, vals=True) #last row has the minimum
minimum[-1,:]
```

```
## array([ 0.,  0., -6., -9.])
```

The algorithm failed to move from the origin. **Why?**

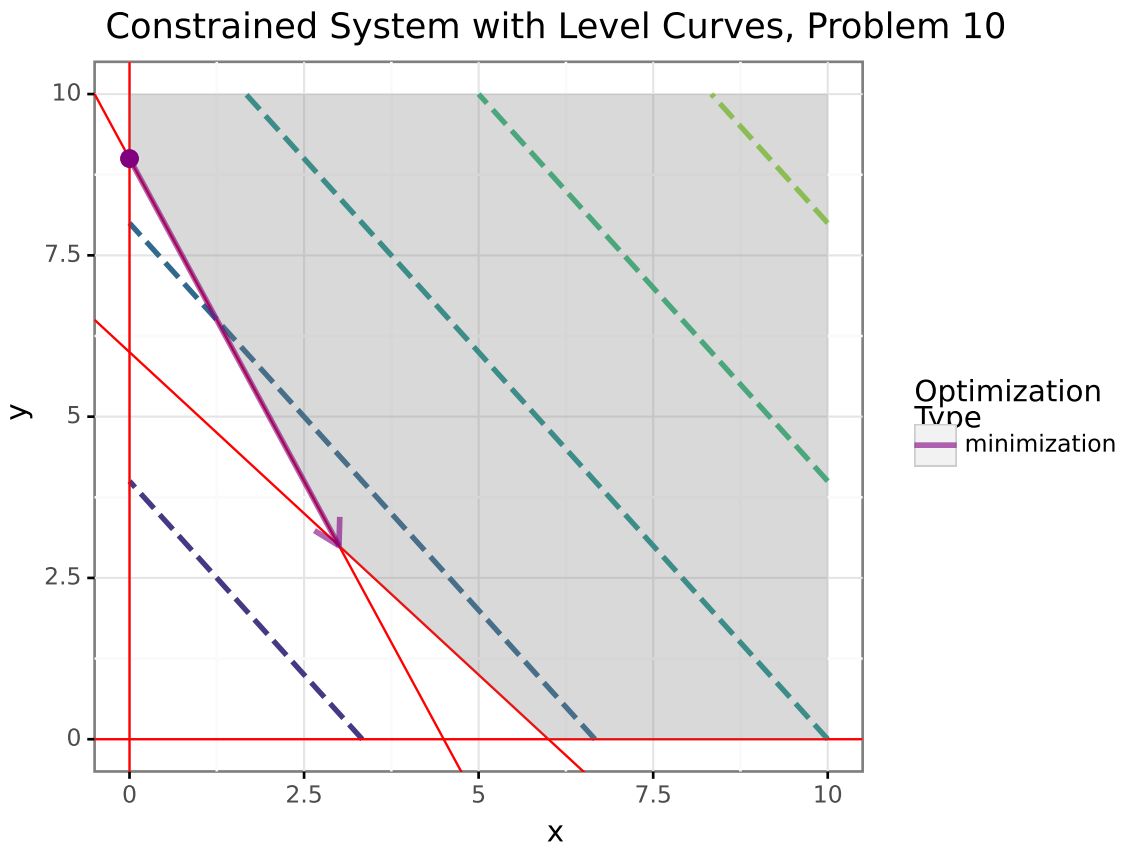
```
x = np.array([0,9,3,0])
minimum = Simplex(A,C,x, Max = False, vals=True) #last row has the minimum
minimum[-1,:]
```

```
## array([3, 3, 0, 0])
```

```
minD = pd.DataFrame({'x' : minimum[:,0], "y" : minimum[:,1]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 6, slope = -1), color = "red")
  + p9.geom_abline(mapping=p9.aes(intercept = 9, slope = -2), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.geom_path(data=minD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("purple", minD.shape[0]]
  + p9.geom_point(data = minD.drop(minD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, colo
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Leve
  + p9.scale_color_identity(name = "Optimization\nType",labels = ["minimization"], guide = "legend")
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshape
```

```
## <matplotlib.contour.QuadContourSet object at 0x1906e4d30>
```

```
plt.show()
```




```
plt.close('all')
```

Our algorithm now works **if** we start it at a feasible extreme point. Starting the algorithm at a non-feasible extreme point erroneously leaves the algorithm at a non-feasible point because the value of the optimization function is less there than at the actual constrained minimum $(3, 3)$.

Problem 11

Problem 11 is a slight modification of Problem 10. The optimization function for Problem 11 is:

$$\text{Optimize} \quad x - y$$

with the same constraints. Let's plot this system:

```
### A plot of the system
data = pd.merge(pd.DataFrame({'x': np.linspace(0,10,50), 'key':1}), pd.DataFrame({'y': np.linspace(0,10
```

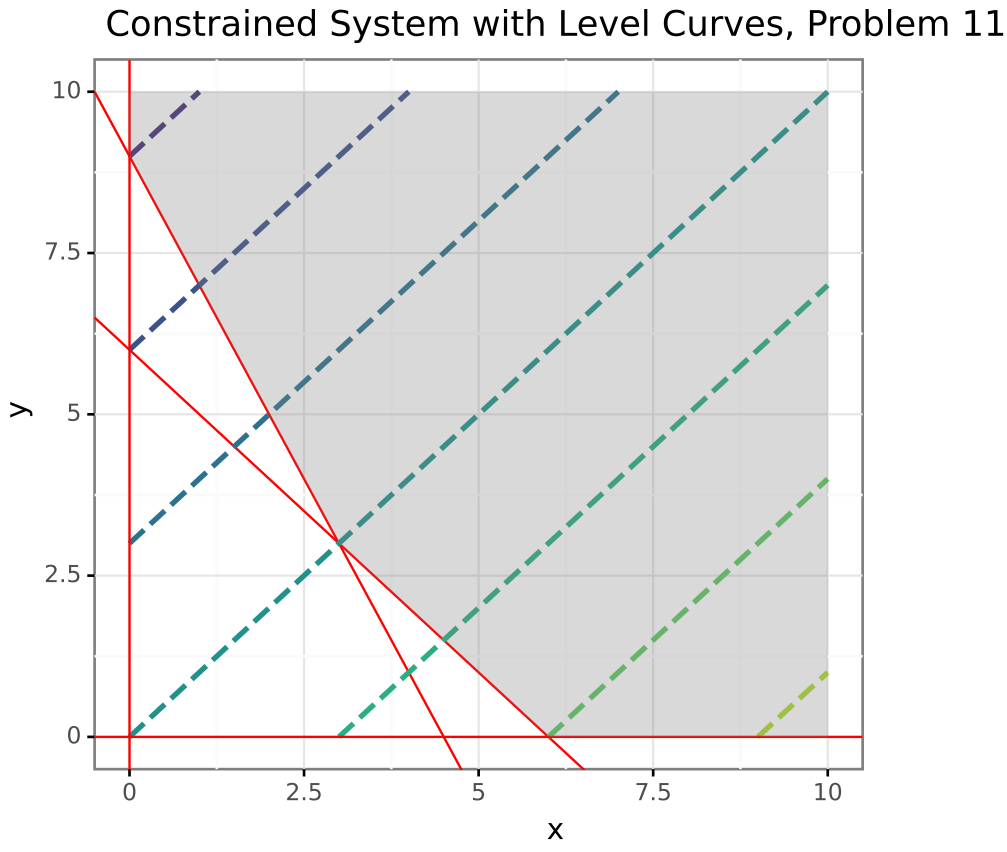
```
## <string>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for
```

```
data['z'] = data.x - data.y
```

```
shadeMe = pd.DataFrame({'x':[3,6,10,10,0,0], 'y':[3,0,0,10,10,9]})

myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 6, slope = -1), color = "red")
  + p9.geom_abline(mapping=p9.aes(intercept = 9, slope = -2), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Leve
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshape
```

```
## <matplotlib.contour.QuadContourSet object at 0x1906febe0>
```

```
plt.show()
```




```
plt.close('all')
```

Clearly, the only difference here are the level curves. Will this matter to our algorithm? To avoid the problem with not starting at a feasible points let's use $\mathbf{x}^T = \begin{bmatrix} 0 & 9 & 3 & 0 \end{bmatrix}$ as our starting point.

```
C = np.array([1,-1, 0, 0])
maximum <- Simplex(A,C,x,vals = True) #fails due to unbounded problem
```

```
## Error in py_call_impl(callable, dots$args, dots$keywords): ValueError: zero-size array to reduction o
##
## Detailed traceback:
##   File "<string>", line 1, in <module>
##   File "<string>", line 27, in Simplex
##   File "<__array_function__ internals>", line 5, in amin
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return _wrapreduction(a, np.minimum, 'min', axis, None, out,
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

So we can see the maximization still fails. Let's try the minimization:

```
minimum <- Simplex(A,C,x,Max = False, vals = True)
```

```
## Error in py_call_impl(callable, dots$args, dots$keywords): ValueError: zero-size array to reduction o
##
## Detailed traceback:
##   File "<string>", line 1, in <module>
##   File "<string>", line 27, in Simplex
##   File "<__array_function__ internals>", line 5, in amin
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return _wrapreduction(a, np.minimum, 'min', axis, None, out,
##   File "/Users/bridenho/opt/anaconda3/envs/r-reticulate/lib/python3.9/site-packages/numpy/core/fromnu
##     return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

The minimization now also fails. **Why?**

Problem 12

The final linear program for optimization was specified as follows:

$$\begin{aligned} \text{Optimize} \quad & 5x + 3y \\ \text{subject to} & \\ 1.2x + 0.6y &\leq 24 \\ 2x + 1.5y &\leq 80 \\ x, y &\geq 0 \end{aligned}$$

Our system looks relatively straight-forward for optimization. Note that the extreme points of the **convex polytope** are $(0, 0)$, $(20, 0)$, and $(0, 40)$. Let's try it:

```
A = np.array([1.2, 0.6, 1, 0,
      2, 1.5, 0, 1
      ]).reshape(2,-1)

C = np.array([5., 3, 0, 0])
x =  np.array([0, 0, 24., 80])

maximum = Simplex(A,C,x,vals=True)

minimum = Simplex(A, C, maximum[maximum.shape[0]-1,:], Max = False, vals = True)
```

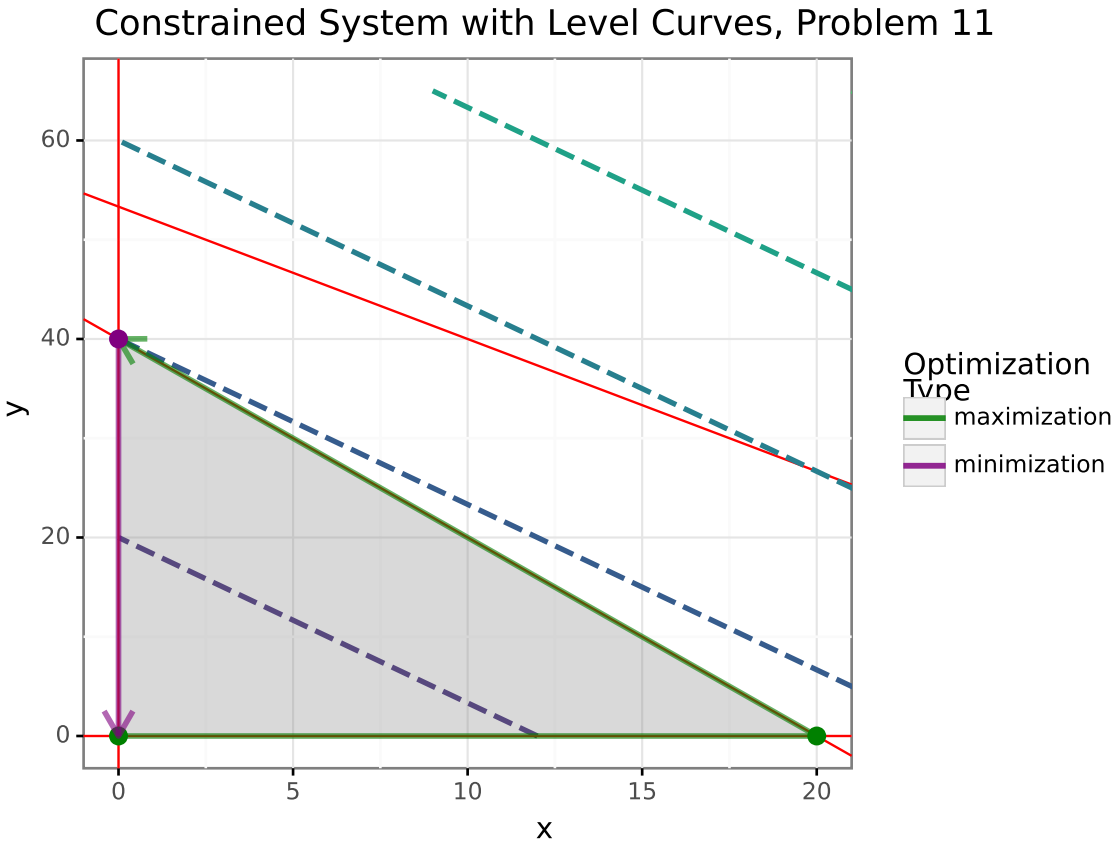
```
data = pd.merge(pd.DataFrame({'x': np.linspace(0,45,50), 'key':1}), pd.DataFrame({'y': np.linspace(0,65
```

```
## <string>:1: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for t
```

```
data['z'] = 5*data.x + 3*data.y

shadeMe = pd.DataFrame({'x':[0, 20, 0], 'y':[0, 0, 40]})

maxD = pd.DataFrame({'x':maximum[:,0], 'y':maximum[:,1]})
minD = pd.DataFrame({'x':minimum[:,0], 'y':minimum[:,1]})


myggplot = (
  p9.ggplot(data, p9.aes(x='x',y='y'))
  + p9.geom_abline(mapping=p9.aes(intercept = 40, slope = -2), color = "red")
  + p9.geom_abline(mapping=p9.aes(intercept = 80/1.5, slope = -2/1.5), color = "red")
  + p9.geom_hline(mapping=p9.aes(yintercept = 0 ), color="red")
  + p9.geom_vline(mapping=p9.aes(xintercept = 0), color="red")
  + p9.geom_polygon(data = shadeMe, mapping=p9.aes(x='x',y='y'), fill = "gray", alpha = 0.3)
  + p9.geom_path(data=maxD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("green", maxD.shape[0])
  + p9.geom_point(data = maxD.drop(maxD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, colo
  + p9.geom_path(data=minD, mapping=p9.aes(x = 'x', y = 'y', color = 'np.repeat("purple", minD.shape[0]]
  + p9.geom_point(data = minD.drop(minD.shape[0] - 1), mapping = p9.aes(x = 'x', y = 'y'), size =3, colo
  + p9.theme_bw() + p9.theme(subplots_adjust={'right': 0.75}) + p9.ggtitle("Constrained System with Leve
  +  p9.scale_color_identity(name = "Optimization\nType",labels = ["maximization","minimization"], guide
)

fig = myggplot.draw()
[ax] = fig.get_axes()
ax.contour(data.x.to_numpy().reshape(50,-1), data.y.to_numpy().reshape(50,-1), data.z.to_numpy().reshape
```

```
## <matplotlib.contour.QuadContourSet object at 0x1908c6d30>
```

```
plt.show()
```




```
plt.close('all')
```