Homework 1 - KenwanCheung

1 - Consider the following maximization problem.

max $x_1 + \frac{1}{4}x_2$

$\frac{1}{2}x_1 + x_2 \leq 1$

$2x_1 + x_2 \leq 2$

$x_1 >= 0, x_2 >= 0$

A. Guess the solution. Plot the constraints and the objective function. Justify your guess.

B. Using Julia define and solve the above problem 'as is'.

C. Rewrite it in the standard form.

D. Using Julia define and solve the above problem in the standard form.

E. Compare the solutions in item B and D.

F. Rework items A and B if the objective function is x1+x2x1+x2 .

1A

In [1]:
```
using JuMP
using GLPKMathProgInterface
using PyPlot
```

In [21]:
```
myModel = Model(solver=GLPKSolverLP())
```

```
@variable(myModel, x1 >= 0)
@variable(myModel, x2 >= 0)
@constraint(myModel, 0.5*x1 + x2 <= 1)
@constraint(myModel, 2*x1 + x2 <= 2)
@objective(myModel, Max, x1+0.25*x2)
myModel
```

Out[21]:

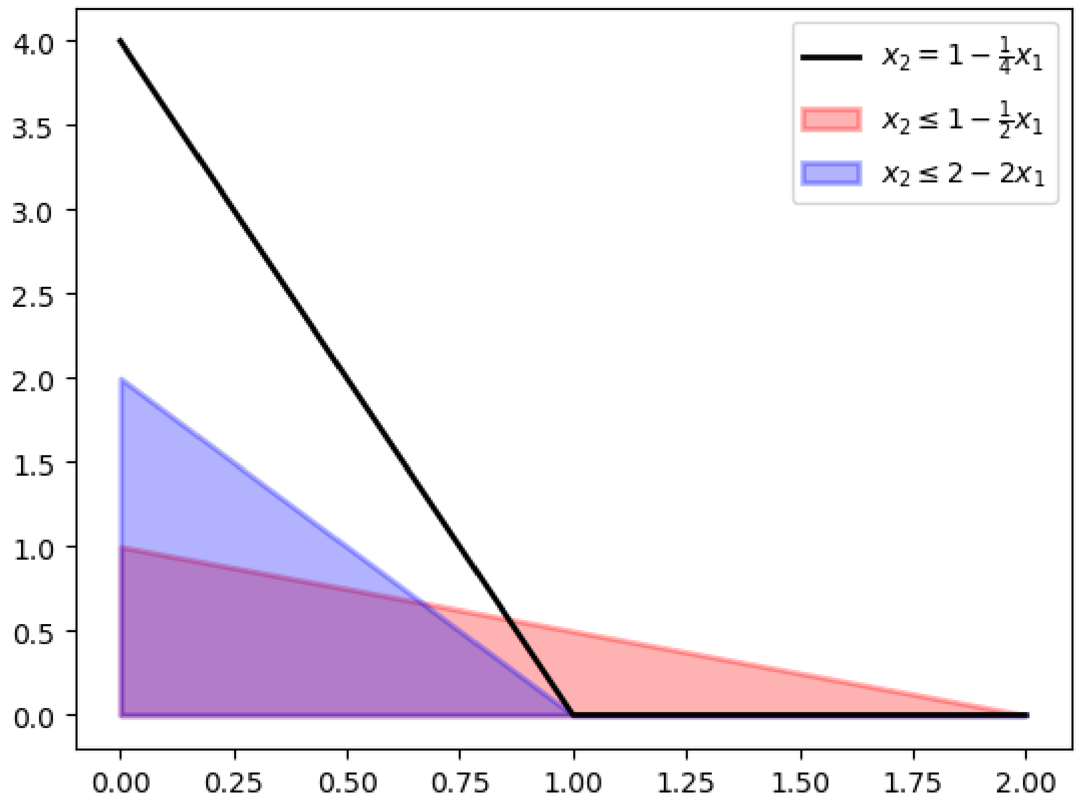
$$
\begin{aligned}
\max \quad & x1 + 0.25x2\\
\text{Subject to} \quad & 0.5x1 + x2 \leq 1\\
& 2x1 + x2 \leq 2\\
& x1 \geq 0\\
& x2 \geq 0
\end{aligned}
$$

In [15]:

```
# plot the constraints

x1 = collect(0:0.1:2)
x2a = 1-0.5*x1
x2b = ifelse(2-2*x1.>=0,2-2*x1,0)
x2c = ifelse(4-4*x1.>=0,4-4*x1,0)

fig, ax = subplots()
ax[:fill_between](x1,x2a,color="red",linewidth=2,label=L"x_{2} \leq 1 - \frac{1}{2}x_{1}",alpha=0.3)
ax[:legend](loc="upper right")
ax[:fill_between](x1,x2b,color="blue",linewidth=2,label=L"x_{2} \leq 2 - 2x_{1}",alpha=0.3)
ax[:legend](loc="upper right")
ax[:plot](x1,x2c,color="black",linewidth=2,label=L"x_{2} = 1 - \frac{1}{4}x_{1}",alpha=1)
ax[:legend](loc="upper right")
```




```
WARNING: ifelse(c::AbstractArray{Bool}, x::AbstractArray, y) is depreca
ted, use ifelse.(c, x, y) instead.
Stacktrace:
 [1] depwarn(::String, ::Symbol) at ./deprecated.jl:70
 [2] ifelse(::BitArray{1}, ::Array{Float64,1}, ::Int64) at ./deprecate
d.jl:57
 [3] include_string(::String, ::String) at ./loading.jl:522
 [4] include_string(::Module, ::String, ::String) at /Users/kenwancheun
g/.julia/v0.6/Compat/src/Compat.jl:174
 [5] execute_request(::ZMQ.Socket, ::IJulia.Msg) at /Users/kenwancheun
g/.julia/v0.6/IJulia/src/execute_request.jl:154
 [6] (::Compat.#inner#16{Array{Any,1},IJulia.#execute_request,Tuple{ZM
Q.Socket,IJulia.Msg}})() at /Users/kenwancheung/.julia/v0.6/Compat/src/
```

```
Compat.jl:496

 [7] eventloop(::ZMQ.Socket) at /Users/kenwancheung/.julia/v0.6/IJulia/src/eventloop.jl:8
 [8] (::IJulia.##14#17)() at ./task.jl:335
while loading In[15], in expression starting on line 5
WARNING: ifelse(c::AbstractArray{Bool}, x::AbstractArray, y) is deprecated, use ifelse.(c, x, y) instead.
Stacktrace:
 [1] depwarn(::String, ::Symbol) at ./deprecated.jl:70
 [2] ifelse(::BitArray{1}, ::Array{Float64,1}, ::Int64) at ./deprecated.jl:57
 [3] include_string(::String, ::String) at ./loading.jl:522
 [4] include_string(::Module, ::String, ::String) at /Users/kenwancheung/.julia/v0.6/Compat/src/Compat.jl:174
 [5] execute_request(::ZMQ.Socket, ::IJulia.Msg) at /Users/kenwancheung/.julia/v0.6/IJulia/src/execute_request.jl:154
 [6] (::Compat.#inner#16{Array{Any,1},IJulia.#execute_request,Tuple{ZMQ.Socket,IJulia.Msg}})() at /Users/kenwancheung/.julia/v0.6/Compat/src/Compat.jl:496
 [7] eventloop(::ZMQ.Socket) at /Users/kenwancheung/.julia/v0.6/IJulia/src/eventloop.jl:8
 [8] (::IJulia.##14#17)() at ./task.jl:335
while loading In[15], in expression starting on line 6
```

Out[15]:
```
PyObject <matplotlib.legend.Legend object at 0x138086f10>
```

I'm going to guess the solution is x1 = 1 and x2 = 0. I would justify it as we need to get far more x2 to get the same impact (4x) of x1.

1B

In [22]:
```
@time begin
    status = solve(myModel)
end
println("Objective value: ", getobjectivevalue(myModel))
```

```
println("x1 = ", getvalue(x1))
println("x2 = ", getvalue(x2))
```

```
  0.000499 seconds (75 allocations: 5.047 KiB)
Objective value: 1.0
x1 = 1.0
x2 = 0.0
```

1C

minimize

$x_1 + \frac{1}{4}x_2$

subject to

$\frac{1}{2}x_1 + x_2 + s_1 = 1$

$2x_1 + x_2 + s_2 = 2$

$x_1, x_2, s_1, s_2 >= 0$

1D

In [25]:

```
sfLpModel = Model(solver=GLPKSolverLP())
c = [1; 1/4; 0; 0]
b = [1;2]
A= [
     0.5 1 1 0;
     2 1 0 1
    ]
m, n = size(A)
@variable(sfLpModel, x[1:n] >= 0)
for i=1:m
    @constraint(sfLpModel, sum{A[i,j]*x[j] , j=1:n} == b[i])
end
```

```
@objective(sfLpModel, Max, sum{c[j]*x[j], j=1:n})
println("The optimization problem to be solved is:")
print(sfLpModel)
```

```
The optimization problem to be solved is:
```

```
WARNING: The curly syntax (sum{},prod{},norm2{}) is deprecated in favor
 of the new generator syntax (sum(),prod(),norm()).
WARNING: Replace sum{A[i, j] * x[j], j = 1:n} with sum(A[i, j] * x[j] f
or j = 1:n).
WARNING: Replace sum{c[j] * x[j], j = 1:n} with sum(c[j] * x[j] for j =
 1:n).
```

```
Max x[1] + 0.25 x[2]
Subject to
 0.5 x[1] + x[2] + x[3] = 1
 2 x[1] + x[2] + x[4] = 2
 x[i] ≥ 0 ∀ i ∈ {1,2,3,4}
```

In [26]:

```
# Solve

@time begin
status = solve(sfLpModel)
end
println("Objective value: ", getobjectivevalue(sfLpModel))
println("Optimal solution is x = \n", getvalue(x))
```

```
  0.000505 seconds (83 allocations: 5.516 KiB)
Objective value: 1.0
Optimal solution is x =
[1.0, 0.0, 0.5, 0.0]
```

1E

The results for x1 and x2 are {1,0} for both.

Only difference is aat one point the presence of a slack variable in s_{1} when its an equality

1F changed objective function

In [27]:
```
myModel = Model(solver=GLPKSolverLP())
@variable(myModel, x1 >= 0)
@variable(myModel, x2 >= 0)
@constraint(myModel, 0.5*x1 + x2 <= 1)
@constraint(myModel, 2*x1 + x2 <= 2)
@objective(myModel, Max, x1+x2)
myModel
```

Out[27]:

$$\begin{aligned} \max \quad & x1 + x2 \\ \text{Subject to} \quad & 0.5x1 + x2 \leq 1 \\ & 2x1 + x2 \leq 2 \\ & x1 \geq 0 \\ & x2 \geq 0 \end{aligned}$$

In [28]:
```
@time begin
    status = solve(myModel)
end
println("Objective value: ", getobjectivevalue(myModel))
println("x1 = ", getvalue(x1))
println("x2 = ", getvalue(x2))
```

```
  0.000858 seconds (75 allocations: 5.047 KiB)
Objective value: 1.3333333333333335
x1 = 0.6666666666666667
x2 = 0.6666666666666666
```

Because we now weight them in the objective function equally, the slope of the solution line changes, and x1,x2 change to {2/3,2/3}

In [ ]: