Exercise 11, Tasks 11.1 - 11.5 (parametric linear program)

Contents

- Problem setup
- Task 11.1 (Numerical solution)
- Task 11.2 (Parametric solution)
- Task 11.3 (Evaluation of the parametric solution)
- Task 11.4 (Plotting of the parametric solution)
- Task 11.5

Problem setup

```
% Decision variables
sdpvar x1 x2 r1 r2

% Objective function
obj = x1+6*x2;

% Constraints
con = [ 0 <= x1 <= 200; ...
    0 <= x2 <= r1; ...
    x1+x2 <= r2; ...
    0 <= r1 <= 500; ...
    0 <= r2 <= 700 ];
```

Task 11.1 (Numerical solution)

Notice the minus in front of the objective function since we want to maximize it

```
solvesdp(con+[ r1==300; r2==400 ], -obj);
xopt_numerical = value([x1; x2])
```

```
Optimize a model with 11 rows, 4 columns and 14 nonzeros
Presolve removed 10 rows and 2 columns
Presolve time: 0.00s
Presolved: 1 rows, 2 columns, 2 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -2.0000000e+03   1.000000e+02   0.000000e+00      0s
       1   -1.9000000e+03   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds
Optimal objective -1.900000000e+03

xopt_numerical =

   100
   300
```

Task 11.2 (Parametric solution)

Here we solve the problem parametrically to obtain x1opt and x2opt as functions of the resources r1 and r2.

```
% Vector of free parameters
params = [r1; r2];
% Vector of decision variables
decs = [x1; x2];
% Let YALMIP solve the parametric linear program via MPT
ysol = solvemp(con, -obj, [], params, decs);
% Convert the YALMIP solution to the MPT format
msol = mpt_mpsol2pu(ysol);
```

```
mpt_plcp: 3 regions
-> Generated 1 partition.
```

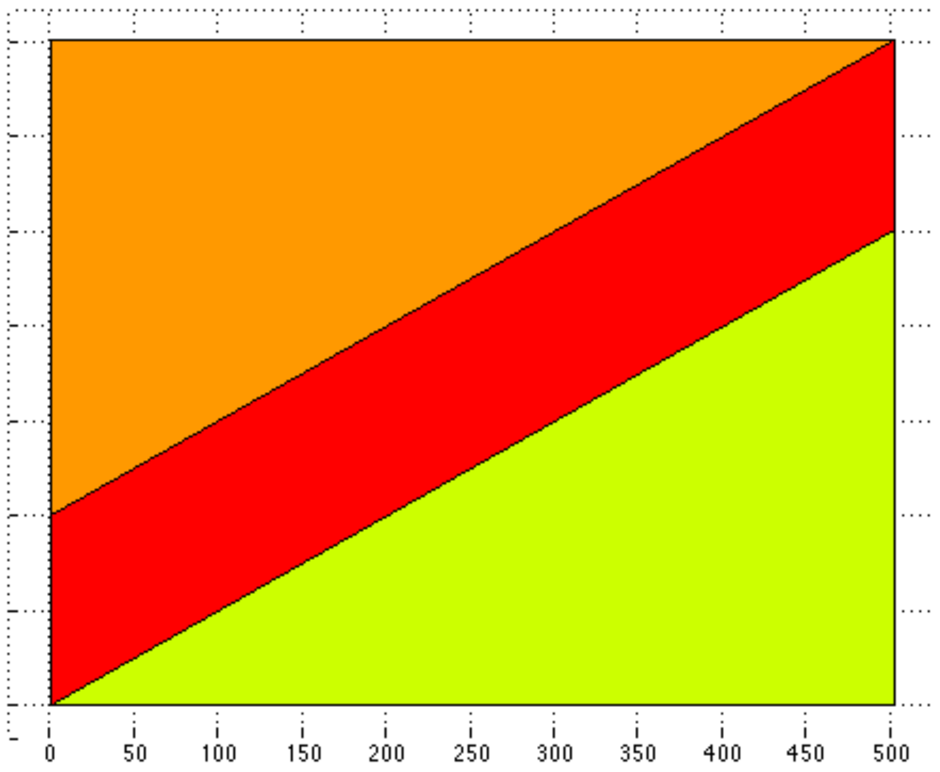
Task 11.3 (Evaluation of the parametric solution)

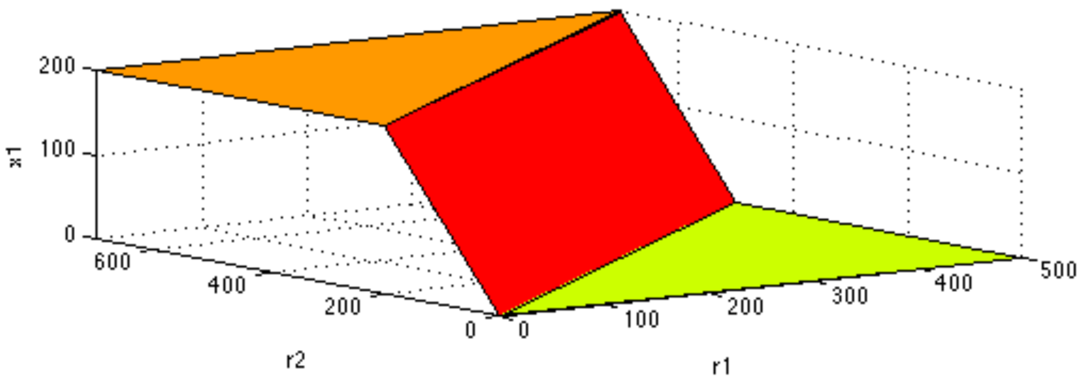
```
% Vector of parameters to evaluate the solution at
r = [300; 400];
% Obtain the primal optimizer for a particular value of the parameters
xopt_parametric = msol.feval(r, 'primal')
```

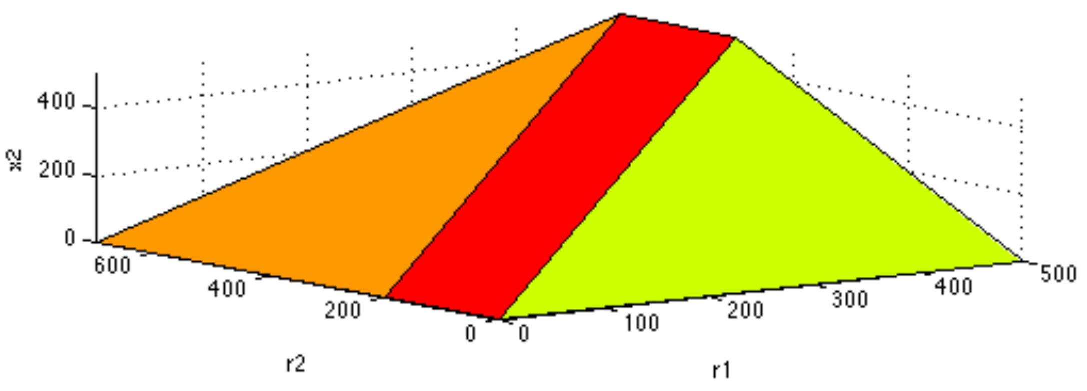
```
xopt_parametric =

   100
   300
```

Task 11.4 (Plotting of the parametric solution)

```
% Plot the critical regions
figure; msol.plot();
% Plot the functions x1(r1, r2) and x2(r1, r2)
figure;
subplot(2, 1, 1); msol.fplot('primal', 'position', 1);
xlabel('r1'); ylabel('r2'); zlabel('x1');
subplot(2, 1, 2); msol.fplot('primal', 'position', 2);
xlabel('r1'); ylabel('r2'); zlabel('x2');
% In each critical region, the optimizer is an affine function of the
% parameters, i.e., xopt = F_i*params+g_i if params \in R_i
F = {}; g = {};
for i = 1:msol.Num
    % iterate through each region and extract the primal optimizer
    F{i} = msol.Set(i).Functions('primal').F;
    g{i} = msol.Set(i).Functions('primal').g;
end
```


700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500


200
100
0
x1
600
400
200
0
r2
0
100
200
300
400
500
r1

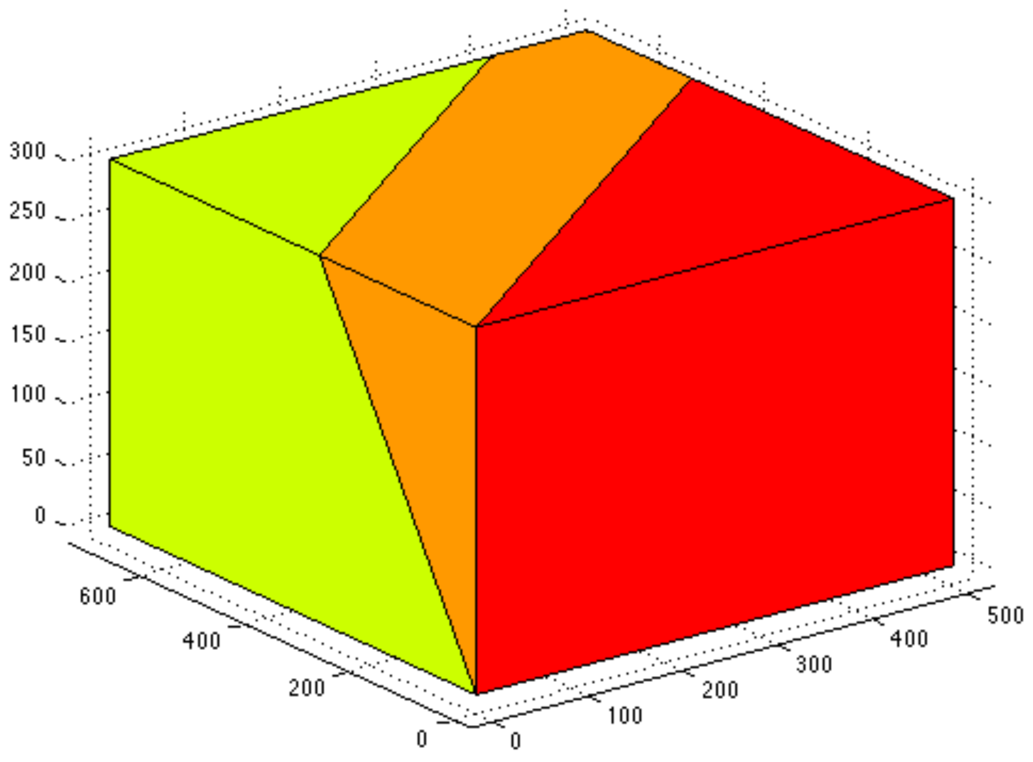

400
200
0
x2
600
400
200
0
r2
0
100
200
300
400
500
r1

Task 11.5

Decision variables

```
sdpvar x1 x2 r1 r2 r3
% Objective function
obj = x1+6*x2;
% Constraints
con = [ 0 <= x1 <= r3; ...
    0 <= x2 <= r1; ...
    x1+x2 <= r2; ...
    0 <= r1 <= 500; ...
    0 <= r2 <= 700; ...
    0 <= r3 <= 300 ];
% Vector of free parameters
params = [r1; r2; r3];
% Vector of decision variables
decs = [x1; x2];
% Let YALMIP solve the parametric linear program via MPT
ysol3d = solvemp(con, -obj, [], params, decs);
% Convert the YALMIP solution to the MPT format
msol3d = mpt_mpsol2pu(ysol3d);
% Plot the critical regions
figure; msol3d.plot();
% Evaluate the parametric solution for r1=200, r2=400, r3=300
xopt = msol3d.feval([200; 400; 300], 'primal')
```

```
mpt_plcp: 3 regions
-> Generated 1 partition.

xopt =

   200
   200
```



Published with MATLAB® R2013a