Lab03 - River 1, Pawel Drapiewski, 11.03.2018 r.

Table of Contents

Searching for place to build bridge - RIVER 1

In this problem we will be searching for the most optimal place for the bridge between city A and city B.

```
clear all;
close all;
```

Define initial variables

```
plot_bound_x_low = 0;
plot_bound_x_high = 5;
% Define the normal vectors of the river bounds (r1 and r1)
A = [-2 1; -2 1];
b = [-1 -3]';
% define the cities
a_point = [1.5, 4.5];
b_point = [3.5, 0.5];
% find the f line (perpendicular to the river)
f_normal_vec = [A(1, 2), -1 * A(1, 1)];
```

Find the points where the bridge should be builded.

Find the points for the bridge, where the cost will be the smallest. To gain it, we should minimize the total length of the route. And we know that building the bridge is very expesive, so the best scenarios is when bridge is perpendicular to river (f line). Because of that fact that bridge lenght is constans we can ommit the bridge length in minimalizing. So overall we need to find only optimal b value of the f line.

```
cvx_begin
    variable f_b(1)
    variable p1(2)
    variable p2(2)
    minimize norm(a_point' - p1) + norm(b_point' - p2);
        subject to
```

```
            % find points where straight f is crossing with r1 (upper
 river bound) and r2 (bottom river bound)
            p1 == find_intersection([A(1, 1) A(1, 2) b(1)],
 [f_normal_vec(1), f_normal_vec(2), f_b(1)])' % crossing of the r1 and
 f function
            p2 == find_intersection([A(2, 1) A(2, 2) b(2)],
 [f_normal_vec(1), f_normal_vec(2), f_b(1)])' % crossing of the r2 and
 f function
cvx_end

disp('Found route has the length: ')
disp(norm(a_point' - p1) + norm(b_point' - p1) + norm(p2 - p1))
disp('Bridge points are: ');
disp(p1);
disp(p2);


Calling SDPT3 4.0: 6 variables, 3 equality constraints
------------------------------------------------------------

 num. of constraints =  3
 dim. of socp   var  =  6,   num. of socp blk  =  2
*******************************************************************
   SDPT3: Infeasible path-following algorithms
*******************************************************************
 version  predcorr  gam  expon  scale_data
    NT      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj
 cputime
-------------------------------------------------------------------
 0|0.000|0.000|7.5e-01|1.3e+00|1.2e+01| 3.746632e+00  0.000000e+00|
 0:0:00| chol  1  1
 1|1.000|0.675|7.4e-07|4.7e-01|4.3e+00| 5.629080e+00  5.787117e+00|
 0:0:01| chol  1  1
 2|1.000|1.000|3.9e-07|5.9e-03|3.6e-01| 4.075972e+00  3.755145e+00|
 0:0:01| chol  1  1
 3|0.984|0.983|2.4e-08|6.8e-04|5.7e-03| 3.798800e+00  3.797487e+00|
 0:0:01| chol  1  1
 4|0.959|0.953|6.1e-08|8.8e-05|2.4e-04| 3.794860e+00  3.795186e+00|
 0:0:01| chol  1  1
 5|0.810|0.922|1.5e-08|6.8e-06|3.6e-05| 3.794754e+00  3.794762e+00|
 0:0:01| chol  1  1
 6|0.958|1.000|2.2e-09|3.1e-09|4.3e-06| 3.794736e+00  3.794731e+00|
 0:0:01| chol  1  1
 7|0.976|0.950|1.3e-10|6.0e-10|2.2e-07| 3.794733e+00  3.794733e+00|
 0:0:01| chol  1  2
 8|0.927|0.986|2.7e-11|3.5e-11|1.9e-08| 3.794733e+00  3.794733e+00|
 0:0:01|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   =  8
 primal objective value =  3.79473320e+00
 dual   objective value =  3.79473319e+00
 gap := trace(XZ)       = 1.88e-08
 relative gap           = 2.18e-09
 actual relative gap    = 2.16e-09
 rel. primal infeas (scaled problem)   = 2.71e-11
 rel. dual     "        "       "      = 3.49e-11
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 3.8e+00, 1.4e+00, 2.0e+00
 norm(A), norm(b), norm(C) = 4.5e+00, 4.0e+00, 2.4e+00
 Total CPU time (secs)  = 0.91
 CPU time per iteration = 0.11
 termination code       =  0
 DIMACS: 3.1e-11  0.0e+00  4.2e-11  0.0e+00  2.2e-09  2.2e-09
-------------------------------------------------------------------

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +3.79473

Found route has the length:
    5.3910

Bridge points are:
    2.0000
    3.0000

    2.8000
    2.6000
```

Prepare data to be visulased on the plot

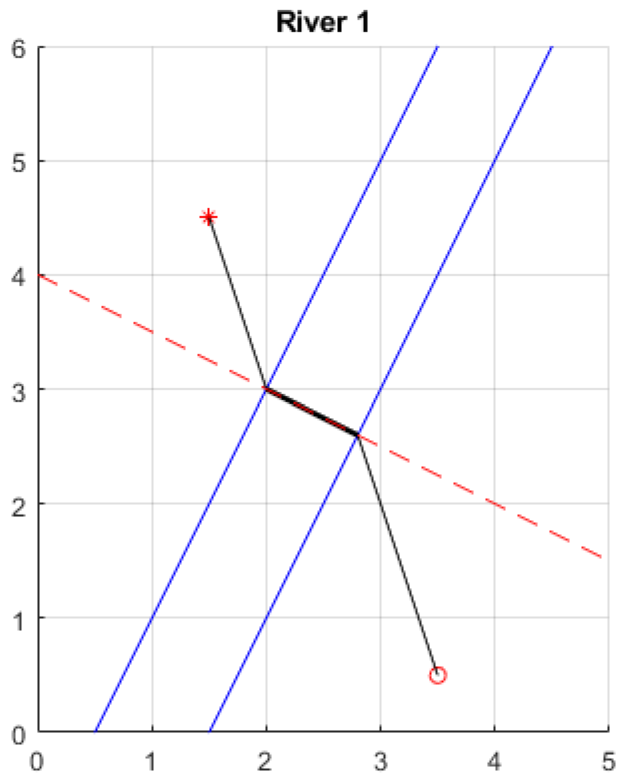
find river coordinates, so it could be drawn on the plot

```
x_low = plot_bound_x_low; x_high = plot_bound_x_high;
y_low = (b - A(:,1)*x_low)./A(:,2);
y_high = (b - A(:,1)*x_high)./A(:,2);
matrix_A_size = size(A, 1);
X = [x_low; x_high]*ones(1, matrix_A_size);
Y = [y_low' ; y_high'];
% find cooridnates for the plotting the f line (red dashed)
% find f (red dashed line) coordinates, so it could be drawn on the
 plot
x_low_f = plot_bound_x_low; x_high_f = plot_bound_x_high;
y_low_f = (f_b - f_normal_vec(:,1)*x_low_f)./f_normal_vec(:,2);
y_high_f = (f_b - f_normal_vec(:,1)*x_high_f)./f_normal_vec(:,2);
matrix_A_size = size(f_normal_vec, 1);
f_X = [x_low_f; x_high_f]*ones(1, matrix_A_size);
f_Y = [y_low_f' ; y_high_f'];

% draw the plot
hold on;
title("River 1");
daspect([1 1 1]);
```

```
xlim([plot_bound_x_low plot_bound_x_high]);
ylim([0 6]);
grid on;

plot(a_point(1), a_point(2) ,'r*');
plot(b_point(1), b_point(2) ,'ro');
plot([x_low(1), x_high(1)], [y_low(1), y_high(1)], 'blue')
plot([x_low(1), x_high(1)], [y_low(2), y_high(2)], 'blue')
plot([p1(1), p2(1)], [p1(2), p2(2)], 'black', 'LineWidth', 2)
plot([a_point(1), p1(1)], [a_point(2), p1(2)], 'black')
plot([b_point(1), p2(1)], [b_point(2), p2(2)], 'black')
plot([x_low_f(1), x_high_f(1)], [y_low_f(1), y_high_f(1)], 'red--')
hold off;
```



Solve the problem using fmincon function

```
d = norm(p2 - p1);
fmin_A = [ A(1, 1) A(1, 2) 0 0;
           0 0 A(1, 1) A(1, 2);
           A(2, 1) A(2, 2) 0 0;
           0 0 A(2, 1) A(2, 2)];
b = [-1; -3; d; d];
f_descriptor = @(x) double(min_route(x));
x0 = [0, 0, 0 ,0];
X = fmincon(f_descriptor, x0, [], [], fmin_A, b);
```

```
disp("Solution using fmincon: ");
disp("p1: ");
disp(X(1:2));
disp("p2: ");
disp(X(3:4));

% As we can see, this result is only aproximation of the proper value
 and
% the cvx was better in that case.

% function to find the intersection point of two lines
function result = find_intersection(line_1, line_2)
    y = (line_2(3) - line_2(1) * line_1(3) / line_1(1)) / (line_2(2) -
 line_2(1) * line_1(2) / line_1(1));
    x = line_1(3) / line_1(1) - line_1(2) / line_1(1) * y;
    result = [x, y];
end

% This is hardcoded problem of the upper example. This function exist
 for
% fmincon calculation only.
function res = min_route(x)
    a_point = [1.5, 4.5];
    b_point = [3.5, 0.5];
    res = norm([a_point(1)-x(1), a_point(2)-x(2)]) + norm([b_point(1)-
x(3) b_point(2)-x(4)]) + norm([x(1)-x(3), x(2)-x(4)]);
end
```

```
Converged to an infeasible point.

fmincon stopped because the size of the current step is less than
the default value of the step size tolerance but constraints are not
satisfied to within the default value of the constraint tolerance.



Solution using fmincon:
p1:
    1.8882    3.7236

p2:
    2.1382    3.2236
```

Published with MATLAB® R2017b