关于碰撞检测的 Matlab 实例实现

概述。在求解最优化的实际问题中，嵌入在 Matlab 中的 CVX 工具是比较好的选择。在 C/C++的代码中也有，如 Gurobi，Mosek 等(闭源商用，注册后可用)，Python 也有 CVXPY 工具。本小节通过对 CVX 求解两个实例来验证方法和结果的正确性。终究使用 Matlab 是更方便和快捷(不仅计算结果，还以图形显示结果)。

一、 CVX 的功能和安装

1. CVX 的功能。运行在 Matlab 上的一个求解凸优化的强大工具。它能求解大部分凸问题，基本是最优化教学的标准专用软件。
2. 下载是使用见链接 https://cvxr.com/cvx/。如何使用 Matlab 和 CVX，请自行查阅。

二、 两个三角形之间的最短距离

(1) 距离方程表示。就是一个二次型，在一个三角形上的点为x_1，x_2，在另一个三角形上的点为x_3，x_4，在距离函数表示为式 7.1

$$(x_1 - x_3)^2 - (x_2 - x_4)^2 \qquad \text{式 7.1}$$

(2) 可以使用矩阵表示为：如式 7.2

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad \text{式 7.2}$$

(3) 第一个三角形方程，蓝色线条显示。顶点分别为：(0,0)，(2,0)，(0,1)，决策变量x_1，x_2，在三角形内部或边界上，使用方程表示为，式 7.3。见图 1

$$\begin{gathered} x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 + 2x_2 \leq 2 \end{gathered} \qquad \text{式 7.3}$$

(4) 第二个三角形方程，绿色线条显示。顶点分别为：(1,2)，(2,2)，(0,3)，决策变量x_3，x_4在三角形区域内部或边界上，使用方程表示为，式 7.4。见图 1

$$\begin{gathered} x_4 \geq 2 \\ x_3 + x_4 \geq 3 \\ x_3 + 2x_4 \leq 6 \end{gathered} \qquad \text{式 7.4}$$

(5) Matlab 代码：

```
%% 求解两个三角形之间的最短距离
%% 使用二次规划方法，在 CVX 中实现
%% 清屏清内存
clc;
clear;
%% 距离矩阵
%  1  0 -1  0
%  0  1  0 -1
% -1  0  1  0
%  0 -1  0  1
P0 = [ 1 , 0 , -1 , 0 ;
        0 , 1 , 0 , -1 ;
       -1 , 0 , 1 , 0 ;
        0 , -1 , 0 , 1 ];
q0 = [ 0 , 0 , 0 , 0 ]';
r0 = 0;
%% 6 条直线组成的 2 个三角形
%  1  0  0  0
%  0  1  0  0
% -1 -2  0  0
%  0  0  0  1
%  0  0  1  1
%  0  0 -1 -2
A = [ 1 , 0 , 0 , 0 ;       % x1 >= 0          属于第一个三角形，蓝色
```

```
      0 , 1 , 0 , 0 ;       % x2 >= 0         属于第一个三角形，蓝色
     -1 ,-2 , 0 , 0 ;       % x1 + x2 <= 2    属于第一个三角形，蓝色
      0 , 0 , 0 , 1 ;       % x4 >=2             属于第二个三角形，绿色
      0 , 0 , 1 , 1 ;       % x3 + x4 >=3        属于第二个三角形，绿色
      0 , 0 , -1 ,-2 ];     % x3 + 2 * x4 <= 6   属于第二个三角形，绿色

b = [ 0 , 0 , -2 , 2 , 3 , -6 ]'; % 不等式右端值

n = 4;
%% cvx
fprintf( 1 , '计算 QP 的最优值...\n');

cvx_begin
    cvx_precision best     % 使用最好的精度
    cvx_solver sedumi      % 使用 sedumi 求解器
    variable x( n )
    minimize ( quad_form( x , P0 ) + q0' * x + r0 ); % 目标函数
    subject to
    A * x >= b;                 % 不等式约束
cvx_end
%显示结果
format long;                    % 设置显示高精度数据
display( sqrt( cvx_optval ));   % 显示求解结果
display( sqrt( quad_form( x , P0 ) + q0' * x + r0 )); % 显示最小值
display( x );                   % 显示目标值

%% 画三角形
x1 = [ 0.0   2.0   0.0  0.0 ];  %属于第一个三角形，蓝色，x 坐标
y1 = [ 0.0   0.0   1.0  0.0 ];  %属于第一个三角形，蓝色，y 坐标
x2 = [ 1.0 , 2.0 , 0.0  1.0 ];  %属于第二个三角形，绿色，x 坐标
y2 = [ 2.0 , 2.0 , 3.0  2.0 ];  %属于第二个三角形，绿色，y 坐标
h = plot( x1 , y1 , 'k' , 'linewidth' , 2 );    % 画第一个三角形
set( h , 'Color' , 'Blue' );                     % 设置为蓝色
hold on;
h = plot( x2 , y2 , 'k' , 'linewidth' , 2 );    % 画第二个三角形
set( h , 'Color' , 'Green' );                    % 设置为绿色
hold on;
h = line([ x( 1 ) , x( 3 )],[ x( 2 ) , x( 4 )]);  % 画距离线段
set( h , 'Color' , 'Red' );                        % 设置为红色
grid on;
axis equal;
axis([ -1 3 -1 4 ]);
```

(6) 求解结果最短距离，红色线条显示。

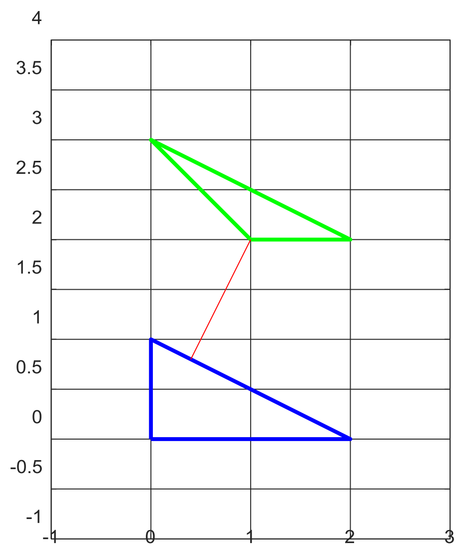

图 1

三、 两个椭圆之间的最短距离

(1) 描述距离的二次型方程，见式 7.1。

(2) 第一个椭圆方程，蓝色显示见图 2，决策变量x_1，x_2在绿色椭圆上，如式 7.5

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0.25 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} - 0.75 \le 0 \qquad \text{式 7.5}$$

(3) 第二个椭圆方程，绿色显示，决策变量x_3，x_4在绿色椭圆上见图 2，如式 7.6

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0.625 & 0.375 \\ 0.375 & 0.625 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -5.5 \\ -6.5 \end{bmatrix} + 17.5 \le 0 \qquad \text{式 7.6}$$

(4) 写成 Matlab 代码

```
%% 求解两个椭圆之间的最短距离
%% 使用二次约束二次规划方法，在 CVX 中实现
%% 清屏清内存
clc;
clear;
%% 距离二次型矩阵
% x'* P0 * x + q0 * x + r0
%  1  0 -1  0
%  0  1  0 -1
% -1  0  1  0
%  0 -1  0  1
P0 = [ 1 , 0 , -1 , 0 ;
       0 , 1 , 0 , -1 ;
      -1 , 0 , 1 , 0 ;
       0 ,-1 , 0 , 1 ];
q0 = [ 0 , 0 , 0 , 0 ]';
r0 = 0;
%% 第一个椭圆，蓝色
% x'* P1 * x + q1 * x + r1
% 0.25  0  0  0
%   0  1  0  0
%   0  0  0  0
%   0  0  0  0
P1 = [ 0.25 , 0 , 0 , 0 ;
          0 , 1 , 0 , 0 ;
          0 , 0 , 0 , 0 ;
          0 , 0 , 0 , 0 ];
q1 = [ -0.5 , 0 , 0 , 0 ]';
r1 = -0.75;
%% 第二个椭圆，绿色
% x'* P1 * x + q1 * x + r1
%   0  0    0  0
%   0  0    0  0
%   0  0  5/8  3/8
%   0  0  3/8  5/8
P2 = [  0 , 0 , 0 , 0 ;
        0 , 0 , 0 , 0 ;
        0 , 0 , 0.625 , 0.375 ;
        0 , 0 , 0.375 , 0.625 ];
q2 = [ 0 , 0 , -5.5 , -6.5 ]';
r2 = 17.5;
%
n = 4;
%% cvx
fprintf( 1 , '计算 QCQP 的最优值...\n');

cvx_begin
    cvx_precision best            % 设置显示高精度数据
    cvx_solver sedumi             % 显示求解结果
    variable x( n )
    minimize ( quad_form( x , P0 ) + q0' * x + r0 );  % 目标函数
    subject to
    quad_form( x , P1 ) + q1' * x + r1 <= 0;  % 第一个二次约束
    quad_form( x , P2 ) + q2' * x + r2 <= 0;  % 第二个二次约束
cvx_end
```

```
%显示结果
format long;
display( sqrt( cvx_optval ));
display( sqrt( quad_form( x , P0 ) + q0' * x + r0 ));
display( x );

%% 画椭圆和的直线
% 第一个蓝色椭圆
h = ezplot( '0.25 * x^2 + y^2 - 0.5 * x  = 0.75 ' );
set( h , 'Color' , 'Blue' , 'LineWidth' , 2 );%, '' ,
hold on;
% 第二个绿色椭圆
h = ezplot( '0.625 * x^2 + 0.75 * x * y + 0.625 * y^2 - 5.5 * x - 6.5 * y = -17.5 ' );
set( h , 'Color' , 'Green' , 'LineWidth' , 2 );
hold on;
% 第三个红色直线
h = line([ x( 1 ) , x( 3 )],[ x( 2 ) , x( 4 )]);
set( h , 'Color' , 'Red' );
% xy 坐标轴相等，范围，显示坐标网格
axis equal;
axis([ -3 5 -2 6 ]);
grid on;
```

(5) 求解结果，红色是最短距离。见图 2

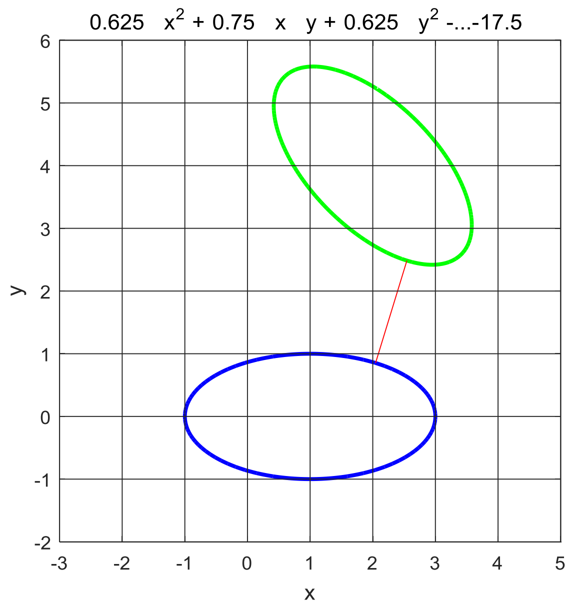

图 2

四、 最优化的方向和一些拙见

(1) 当前最优化的发展方向是锥规划。国内教材或教学中一般以 SQP(序列二次规划)作为书籍结束，但真正面临实际问题时，约束函数的形式远不至于此，因此有了更广泛的锥规划(Cone Programming)，覆盖了二次约束二次规划(QCQP)，二阶锥规划(SOCP)，半定规划(SDP)。其应用请参看[1][2]。

(2) 实例代码的简单说明。代码写得烂，但足够简短，注解充分容易理解。

(3) Matlab 代码转 C。可以使用 Matlab 的应用程序->代码生成->MATLAB Coder，未尝试过是否可以将 CVX 代码转成 C。

五、 参考文章和提供代码

(1) Applications of Second-Order Cone Programming

(2) Applications of Semideinite Programming

(3) QP_2Triangle.m

(4) QCQP_2Ellipse.m。