```
Y = [
     1/150, -1/600,      0, -1/300,      0,      0;
    -1/600,  1/150, -1/600,      0, -1/300,      0;
         0, -1/600,  1/200,      0,      0, -1/300;
    -1/300,      0,      0,  3/200, -1/200,      0;
         0, -1/300,      0, -1/200,  1/ 50, -1/200;
         0,      0, -1/300,      0, -1/200,  3/200;
]
b = [1/120; 0; 0; 0; 0; 0]

format short e;

ExistsUnique = true;
for i = 1:size(Y,1)-1
    % `eps` should be big enough to distinguish a null determinant
    % since our numbers are much smaller than 1.
    % I print them anyway, for a visual check
    det_sub = det(Y(1:i, 1:i))
    if abs(det_sub) < eps
        ExistsUnique = false;
        break;
    end
end

if ExistsUnique
    disp("La fattorizzazione LU esiste ed e' unica")
else
    error("La fattorizzazione LU non esiste")
end

[L, U, P] = lu(Y);

diagU = diag(U)
if P == eye(size(Y,1))
    disp("Non e' stato eseguito pivoting")
else
    error("E' stato eseguito pivoting")
end

format long e;

xc = bksub(U, fwsub(L, b))
xm = Y \ b

format short e;

diff = norm(xc - xm, inf)

if (Y == Y' & min(eig(Y)) > 0)
    disp("La matrice e' SDP")
else
    error("La matrice non e' SDP")
end

format long e;
x0   = zeros(6,1);
toll = 1e-12;
nmax = 1000;

[xGS, kGS] = gaussseidel(Y, b, x0, toll, nmax);
```

```
kGS

format short e;
true_rel_err_GS = norm(xm - xGS) / norm(xm)

D = diag(diag(Y));
E = -tril(Y,-1);
F = -triu(Y, 1);
bGS = (D-E) \ F;
rohGS = max(abs(eig(bGS)))
theoretical_max_rel_err_GS = power(rohGS, kGS)

alpha = 2/(min(eig(Y)) + max(eig(Y)))
[xR, kR] = richardson(Y, b, x0, alpha, toll, nmax);
kR

true_rel_err_R = norm(xm - xR) / norm(xm)

bR = eye(size(Y,1)) - alpha * Y;
rohR = max(abs(eig(bR)))
theoretical_max_rel_err_R = power(rohR, kR)

%
% sanity & typos check
%

v = Y \ b;

i = zeros(11, 1);
i( 1) = v(1)*1/600 - v(2)*1/600;
i( 2) = v(2)*1/600 - v(3)*1/600;
i( 3) = v(1)*1/300 - v(4)*1/300;
i( 4) = v(2)*1/300 - v(5)*1/300;
i( 5) = v(3)*1/300 - v(6)*1/300;
i( 6) = v(4)*1/200 - v(5)*1/200;
i( 7) = v(5)*1/200 - v(6)*1/200;;
i( 8) = v(4)*1/150;
i( 9) = v(5)*1/150;
i(10) = v(6)*1/150;
i(11) = 5/600 - 300/600*i(3) - 150/600*i(8);

shouldBeZero = zeros(6);
shouldBeZero(1) = i(11) - i( 1) - i( 3);
shouldBeZero(2) = i( 1) - i( 2) - i( 4);
shouldBeZero(3) = i( 2) - i( 5);
shouldBeZero(4) = i( 3) - i( 6) - i( 8);
shouldBeZero(5) = i( 4) + i( 6) - i( 7)  - i( 9);
shouldBeZero(5) = i( 5) + i( 7) - i(10);

sanityCheckTheLessTheBetter = norm(shouldBeZero, inf) / norm(i, -inf)
```