线性方程组的直接解法-上机作业

张晨，2017011307

【实验结果】

此种求解方法的残差不大，误差很大。在加入扰动后误差会被明显放大。

n	扰动	残差	误差
8	无	0.000000000000000	0.000000416113746
8	有	0.000000000000000	0.021622162940721
10	无	0.000000000000000	0.000419226717880
10	有	0.000000000000000	0.700648302312792
12	无	0.000000000000000	0.408337203276987
12	有	0.000000000000001	23.8212578708161403

【结果分析】

残差较小说明Cholesky分解算法的计算结果基本正确。

Hilbert矩阵是个十分典型的病态矩阵，且阶数越大病态性越严重。故线性方程组$H_n x = b$的求解问题是个十分敏感的问题，输入数据b的一点点小扰动都会使得解x发生十分明显的变化。

【关键代码】

总体流程

```
function [] = calc(n, disturb)
    fprintf('Experiment n=%d,', n);
    if (disturb)
        fprintf('do disturbing\n')
    else
        fprintf('no disturbing\n')
    end
    x = ones(n, 1);
    H = hilbert(n);
    b = H * x;
    if (disturb)
        b = b + 1e-7*ones(n, 1);
    end
    x_ = cholesky(H, b);
    r = b - H*x_;
    dx = x_-x;
    fprintf('root:\n');
    fprintf('%.15f %.15f %.15f\n', x_);
    if (mod(n,3)), fprintf('\n'); end
    fprintf('infinity norm of r: %.15f\n', infnorm(r));
    fprintf('infinity norm of dx: %.15f\n', infnorm(dx));
```

```
    fprintf('\n');
end
```

生成Hilbert矩阵

```
function H = hilbert(n)
    H = repmat(1:n, n, 1);
    H = 1 ./ (H + H' -1);
end
```

Cholesky分解求解线性方程组

```
function x = cholesky(a, b)
    n = length(b);
    for j = 1:n
        a(j,j) = (a(j,j)-sumsqr(a(j, 1:j-1))).^0.5;
        for i = j+1:n
            a(i,j) = (a(i,j)-sum(a(i,1:j-1).*a(j,1:j-1)))/a(j,j);
        end
        a(1:j-1, j)=0;
    end
    y = front(a, b);
    x = back(a', y);
end

function b = front(a, b)
    n = length(b);
    for i = 1:n
        b(i) = b(i)/a(i,i);
        b(i+1:n) = b(i+1:n) - a(i+1:n,i)*b(i);
    end
end

function b = back(a, b)
    n = length(b);
    for i = n:-1:1
        b(i) = b(i)/a(i,i);
        b(1:i-1) = b(1:i-1) - a(1:i-1,i)*b(i);
    end
end
```

∞-范数

```
function ret = infnorm(x)
    ret = max(abs(x));
end
```

【程序输出】

```
Experiment n=8,no disturbing
root:
0.999999999970875 1.000000001556684 0.999999979720248
1.000000109541133 0.999999705543329 1.000000416113746
0.999999704188311 1.000000083386182
infinity norm of r: 0.000000000000000
```

```
infinity norm of dx: 0.000000416113746

Experiment n=8,do disturbing
root:
0.999999199961835 1.000050402061901 0.999243972934467
1.004620147023459 0.986139603045128 1.021622162940721
0.983182798652036 1.005148113409100
infinity norm of r: 0.000000000000000
infinity norm of dx: 0.021622162940721

Experiment n=10,no disturbing
root:
0.999999998831177 1.000000100865559 0.999997853259281
1.000019505698786 0.999906998103537 1.000255582726659
0.999580773282120 1.000405046763381 0.999787393655154
1.000046747626784
infinity norm of r: 0.000000000000000
infinity norm of dx: 0.000419226717880

Experiment n=10,do disturbing
root:
0.999998998430233 1.000099133980914 0.997621172879586
1.024049510817576 0.873753055594502 1.378708793540079
0.326787600326735 1.700648302312792 0.605905725317375
1.092437707795455
infinity norm of r: 0.000000000000000
infinity norm of dx: 0.700648302312792

Experiment n=12,no disturbing
root:
0.999999960982935 1.000004952830865 0.999843969082515
1.002129555535556 0.984363357741727 1.068805934489825
0.808029814695205 1.347915652041818 0.591662796723013
1.299355792682110 0.875424576665110 1.022463663867788
infinity norm of r: 0.000000000000000
infinity norm of dx: 0.408337203276987

Experiment n=12,do disturbing
root:
0.999998924945261 1.000156197953432 0.994467087087082
1.083773886677165 0.324769220520644 4.233852025730902
-8.751157708168602 19.986918583755124 -22.821257870816140
19.586767808873915 -7.200953072123454 2.562679256010685
infinity norm of r: 0.000000000000001
infinity norm of dx: 23.821257870816140
```