Chapter 1

hw1 12235005 谭焱

1.14. Solution.

- Truncation or discretization:

  Some features of a mathematical model may be omitted or simplified (e.g., replacing derivatives by finite differences or using only a finite number of terms in an infinite series).

- Rounding:

  Whether in hand in computation, a calculator, or digital computer, the representation of real numbers and arithmetic operations upon them is ultimately limited to some finite amount of precision and thus is generally inexact.

1.51. Solution.

- If the coefficients are very large or very small, then b^2 or $4ac$ may overflow or underflow.
- Cancellation insider the square root, when the discriminant is small relative to the coefficients.

1.10. Solution. Code as follow,

- When $-b \pm \sqrt{b^2 - 4ac}$ or $2 * a$ too small, we should use second formula.
- Same as $-b$ and square, $2 * c$ too small, consider first formula.

```
function [r1, r2] = exRoot(a, b, c)

bound = max(abs([a, b, c]));
a = a / bound;
b = b / bound;
c = c / bound;

m = sqrt(b * b - 4 * a * c);
if real(m) * b > 0
    if a > 10^-300
        r1 = (-b + m) / (2 * a);
    else
        r1 = (2 * c) / (-b - m);
    end
    if imag(r1) ~= 0
        r2 = abs(imag(r1));
```

```
        r1 = real(r1);
disp(["image result. r1 :", num2str(r1), "r2 :", num2str(r2)]);
        return;
    end
    r2 = (2 * c) / (-b + m);
else
    if  a > 10^-300
        r1 = (-b - m) / (2 * a);
    else
        r1 = (2 * c) / (-b + m);
    end
    if imag(r1) ~= 0
        r2 = abs(imag(r1));
        r1 = real(r1);
disp(["image result. r1 :", num2str(r1), "r2 :", num2str(r2)]);
        return;
    end
    r2 = (2 * c) / (-b - m);
end
disp(["Real result. r1 :", num2str(r1), "r2 :", num2str(r2)]);
end
```

2.39. Solution. (a) 4 ,(b) 6 ,(c) -10.

2.40. Solution.

- In finite-precision arithmetic the choice should be made with some care to minimize propagation of numerical error. In particular, we wish to limit the magnitudes of the multipliers so that previous rounding errors will not be amplified when remain portion of the matrix and right-hand side are multiplied by each elementary elimination matrix.

- Let E is the backward error in the matrix A. For LU factorization by Gaussian elimination, a bound of form

$$\frac{\|E\|}{A} \leq \rho n^2 \epsilon_{mach}$$

  holds, where ρ, called the growth factor, is the ratio of the largest entry of A in magnitude. Without pivoting ρ can be arbitrary large, and hence Gaussian elimination without pivoting is unstable.

2.61. Solution.

(a) $cond_1 = 10^{2}0$, is ill-conditioned.

(b) $cond_1 = 1$, is well-conditioned.

(c) $cond_1 = 1$, is well-conditioned.

(d) $cond = \infty$, is ill-conditioned.

2.17. Solution.

$$
\begin{aligned}
\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} =: LU
\end{aligned}
$$

2.28. Solution.

$$
\begin{aligned}
&(A - UV^T)(A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}) \\
=&I + U(I - V^TA^{-1}U)^{-1}V^TA^{-1} - UV^TA^{-1} - UV^TA^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1} \\
=&I + UV^TA^{-1} - UV^RA^{-1} \\
=&I.
\end{aligned}
$$

Similarly, $(A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1})(A - UV^T) = I$.

So, $(A^{-1} + A^{-1}U(I - V^TA^{-1}U)^{-1}V^TA^{-1}) = (A - UV^T)^{-1}$.

2.34. Solution.

(a) Suppose A is singular, there is exist x such that $Ax = 0$. Hence $x^TAx = 0$, which conflict with A is positive definite matrix.

(b) Suppose A^{-1} is not positive definite. There is exist x such that $x^TA^{-1}x <= 0$.

Since A is not singular, $Ay = x$ have a solution y. Which means $0 >= x^TA^{-1}x = y^TA^TA^{-1}Ay = y^TA^Ty$. That is conflict with A, A^T is positive definite.

2.7. Solution.

(a) When partial pivoting is used, nothing differences with no pivoting using. During eliminating the matrix, entries of the transformed matrix not grow. When complete pivoting is used, some right col permutation matrix Q will be generated, number of transformed matrix won't change.

$$
\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 0 & 2 \\ 0 & -1 & -1 & 1 & 2 \\ 0 & -1 & -1 & -1 & 2 \end{bmatrix}
$$

$$
= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 & -1 \\ 0 & 2 & -1 & 1 & -1 \\ 0 & 2 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}
$$

$\ldots$

(b) using program as following

```
function [x] = GaussEliminateSolver(A, r)
n = size(A, 1);
L = eye(n, n);
for k = 1: n-1
    [~, p] = max(abs(A(k : n, k)));
    p = p + k - 1;
    if A(p, k) == 0
        continue;
    end
    if p ~= k
        swapM(A, k, p, 1);
        swapM(L, k, p, 1);
        swapM(r, k, p, 1);
    end
    for i = k + 1: n
        L(i, k) = A(i, k) / A(k, k);
        A(i, :) = A(i, :) - L(i, k) * A(k, :);
    end
end

x = zeros(n, 1);
y = zeros(n, 1);
for i = 1:n
    y(i) = (r(i) - L(i, :) * y);
end
for i = n:-1:1
    x(i) = (y(i) - A(i, :) * x) / A(i, i);
end

end

function [A] = swapM(A, i, j, dim)
    if dim == 1
        tmp = A(i, :);
        A(i, :) = A(j, :);
        A(j, :) = tmp;
    elseif dim == 2
        tmp = A(:, i);
        A(:, i) = A(:, j);
        A(:, j) = tmp;
```

```
    end
end
```

```
k = 20;
m = rand(k, 1);
for n = 5:2:k;

A = tril(-ones(n));
A = A + 2 * eye(n);
A(:, end) = 1;
x = m(1:n, 1);
r = A * x;
b = GaussEliminateSolver(A, r);
disp([ num2str(n), ' & ' , num2str(log(norm(b - x, 2)), 10), ' & ', num2str(log(norm(A * b - r, 2)), 10), ' & ', num2str(cond(A), 10), ' \\
    ']);

end
```

.

We get result in table , Which imply about linear relation between $\log(error)$ and size n.

n	$\log\|\|E\|\|$	$\log\|\|R\|\|$	$cond(A)$
5	-34.72412472	-33.9800862	2.22392344
7	-34.40508102	-34.0399868	3.075424567
9	-34.16596879	-33.90532033	3.945777218
11	-31.96870812	-31.62615731	4.825981291
13	-32.13881932	-31.68500675	5.711914731
15	-28.68323595	-28.30497086	6.601454215
17	-27.98571253	-27.25548417	7.493403319
19	-25.99587455	-25.60176018	8.387039353

表 1.1: matrix size, error ,residual and condition number.