Jeadel Deguia

2.1.1

$\alpha = 0 \Rightarrow \begin{cases} X_1 + 4X_2 = 6 \\ 2X_1 - X_2 = 3 \Rightarrow X_2 = 2X_1 - 3 \quad (1.1) \\ 3X_2 + X_3 = 5 \quad (1.2) \end{cases} \Rightarrow X_1 + 4(2X_1 - 3) = 6$

$\Rightarrow X_1 + 8X_1 - 12 = 6 \Rightarrow 9X_1 = 18 \Rightarrow X_1 = 2$

$(1.1) \Rightarrow X_2 = 2(2) - 3 \Rightarrow X_2 = 1 \quad (1.2) \Rightarrow X_3 = 2$

$\therefore \boxed{\begin{matrix} X_1 = 2 \\ X_2 = 1 \\ X_3 = 2 \end{matrix}}$

---

$\alpha = -1 \Rightarrow \begin{cases} X_1 + 4X_2 - X_3 = 6 \\ 2X_1 - X_2 - 2X_3 = 3 \\ -X_1 + 3X_2 + X_3 = 5 \end{cases} \xRightarrow{\text{augmented matrix}} A := \left[\begin{array}{ccc|c} 1 & 4 & -1 & 6 \\ 2 & -1 & -2 & 3 \\ -1 & 3 & 1 & 5 \end{array}\right]$

$\Rightarrow A = \left[\begin{array}{ccc|c} 1 & 4 & -1 & 6 \\ 0 & -9 & 0 & -9 \\ -1 & 3 & 1 & 5 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 4 & -1 & 6 \\ 0 & -9 & 0 & -9 \\ 0 & 7 & 0 & 11 \end{array}\right] \Rightarrow \begin{matrix} X_1 + 4X_2 - X_3 = 6 \\ -9X_2 = -9 \\ 7X_2 = 11 \end{matrix}$

$\Rightarrow A = \left[\begin{array}{ccc|c} 1 & 4 & -1 & 6 \\ 0 & -9 & 0 & -9 \\ 0 & 0 & 0 & 4 \end{array}\right] \Rightarrow 0 = 4 \Rightarrow \boxed{\text{no Solution}}$

2.1.1 continued ("$\infty$": infinitely many)

$\alpha=1 \Rightarrow \begin{cases} x_1+4x_2+x_3=6 \\ 2x_1-x_2+2x_3=3 \\ x_1+3x_2+x_3=5 \end{cases} \Rightarrow A:=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 6 \\ 2 & -1 & 2 & 3 \\ 1 & 3 & 1 & 5 \end{array}\right]$

$\Rightarrow A=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 6 \\ 0 & -9 & 0 & -9 \\ 1 & 3 & 1 & 5 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 6 \\ 0 & -9 & 0 & -9 \\ 0 & -1 & 0 & -1 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 6 \\ 0 & -9 & 0 & -9 \\ 0 & 0 & 0 & 0 \end{array}\right]$

$\Rightarrow x_1+4x_2+x_3=6, x_2=1 \Rightarrow x_1=2-x_3$

where $x_3$ can be any real number

so $\boxed{\text{there are } \infty \text{ solutions}}$

---

For RHS replaced by 0's: $\begin{cases} x_1+4x_2+\alpha x_3=0 \\ 2x_1-x_2+2\alpha x_3=0 \\ \alpha x_1+3x_2+x_3=0 \end{cases}$

$\alpha=0 \Rightarrow \left[\begin{array}{ccc|c} 1 & 4 & 0 & 0 \\ 2 & -1 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 0 & 0 \\ 0 & -9 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 0 & 0 \\ 0 & -9 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \Rightarrow \boxed{\begin{array}{l} x_1=0 \\ x_2=0 \\ x_3=0 \end{array}}$

$\alpha=-1 \Rightarrow \left[\begin{array}{ccc|c} 1 & 4 & -1 & 0 \\ 2 & -1 & -2 & 0 \\ -1 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & -1 & 0 \\ 0 & -9 & 0 & 0 \\ -1 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & -1 & 0 \\ 0 & -9 & 0 & 0 \\ 0 & 7 & 0 & 0 \end{array}\right]$

$\Rightarrow x_2=0,\ x_1=x_3\in\mathbb{R} \Rightarrow \boxed{\infty \text{ solutions}}$

$\alpha=1 \Rightarrow \left[\begin{array}{ccc|c} 1 & 4 & 1 & 0 \\ 2 & -1 & 2 & 0 \\ 1 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 0 \\ 0 & -9 & 0 & 0 \\ 1 & 3 & 1 & 0 \end{array}\right]=\left[\begin{array}{ccc|c} 1 & 4 & 1 & 0 \\ 0 & -9 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{array}\right]$

$\Rightarrow x_2=0,\ x_1=-x_3\in\mathbb{R} \Rightarrow \boxed{\infty \text{ solutions}}$

2.1.5

$$A = \begin{bmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{bmatrix} \quad b = \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix}$$

$$\tilde{r} = A\tilde{x} - b = \begin{bmatrix} 0.2157 \\ 0.2529 \end{bmatrix} - \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix} = \begin{bmatrix} -0.001343 \\ -0.001572 \end{bmatrix}$$

$$\hat{r} = A\hat{x} - b = \begin{bmatrix} 0.216999 \\ 0.254 \end{bmatrix} - \begin{bmatrix} 0.217 \\ 0.254 \end{bmatrix} = \begin{bmatrix} (216{,}999 - 217{,}000) \times 10^{-6} \\ 0 \end{bmatrix}$$

exact solution:

$$x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\Rightarrow \hat{r} = \begin{bmatrix} -1 \times 10^{-6} \\ 0 \end{bmatrix} \Rightarrow$$

$\boxed{\text{residual vectors indicate } \hat{x} \text{ is the better solution}}$

$$\tilde{e} = \tilde{x} - x = \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.001 \\ -0.001 \end{bmatrix}$$

$$\hat{e} = \hat{x} - x = \begin{bmatrix} 0.341 \\ -0.087 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.659 \\ 0.913 \end{bmatrix}$$

$\Rightarrow$ $\boxed{\text{error vectors indicate } \tilde{x} \text{ is the better solution}}$

$\Rightarrow$ $\boxed{\text{Basing the better solution on the smaller residual vector is not always a reliable method}}$

2.1.6 (a) $b_1 = 1$, $b_2 = 2 \Rightarrow \begin{cases} 10^{-4}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$

$$\Rightarrow \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right] = \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & -9999 & -9998 \end{array}\right]$$

$$\Rightarrow x_2 = \frac{9998}{9999}, \quad 10^{-4}x_1 + \frac{9998}{9999} = 1$$

relative error: $100\frac{|x_{approx} - x_{exact}|}{|x_{exact}|}$

$$\Rightarrow x_2 \approx 1.00 \Rightarrow 10^{-4}x_1 + 1 \approx 1 \Rightarrow x_1 \approx 0.00$$

exact

$x_1 = \frac{10{,}000}{9999}$

$x_2 = \frac{9998}{9999}$

$$\Rightarrow \boxed{x_1 \approx 0.00, x_2 \approx 1.00}$$

$\Rightarrow$ relative error for $x_1$ is %100; relative error for $x_2$ is $\sim$ %0.01

(b) interchange $\Rightarrow \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 10^{-4} & 1 & 1 \end{array}\right] = \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 0.9999 & 0.9998 \end{array}\right]$

$$\Rightarrow x_2 = \frac{9998}{9999} \approx 1.00 \Rightarrow x_1 + 1.00 = 2 \Rightarrow x_1 = 1.00$$

$$\Rightarrow \boxed{x_1 \approx 1.00, x_2 \approx 1.00}$$

relative error for both $x_1$ and $x_2$ is $\sim$ %0.01

(c) Let $b_1 = 1$, $b_2 = 1$, then $\left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & -9999 & -9999 \end{array}\right]$

$$\Rightarrow x_2 = \frac{-9999}{-9999} = 1.00 \Rightarrow 10^{-4}x_1 + 1.00 = 1.00 \Rightarrow x_1 = 0.00$$

$\Rightarrow$ Answer is exact; $\boxed{b_1 = 1, b_2 = 1}$

2.3.1 ~~[illegible]~~ A banded system $A\vec{x}=\vec{b}$ subjected to Gaussian elimination with partial pivoting will make A become an upper-triangular matrix such that it is able to solve the system via backwards substitution provided A is not close to being singular.

2.3.5

(a) Starting from $i=4$, the first $i-3$ entries are zero, that is, for $4 \le i \le n$, $1 \le j \le i-3$, $a_{ij}=0$

forms triangle of zeros

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & & a_{1n} \\ a_{21} & a_{22} & \cdots & & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ 0 & a_{42} & a_{43} & \cdots & a_{4n} \\ 0 & 0 & a_{53} & \cdots & a_{5n} \\ 0 & 0 & 0 & a_{64} \cdots & a_{6n} \\ & & \cdots & & \end{bmatrix}$$

(b) For each row "$i$", there are $n-(i+1)$ zeros starting from column $j=i+2$; i.e. $a_{ij}=0$ where $1 \le i \le n$, $i+2 \le j \le n$ until $i+2>n$.

also forms triangle of zeros

$$A=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \\ & & & & & 0 & 0 & 0 \\ & & & & & & 0 & 0 \\ & & & & & & & 0 \\ & & \cdots & & & & & \vdots \end{bmatrix}$$

**Computer Exercise 2.1.3**

The following program will use the *naive_gauss* algorithm to solve $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is an $n \times n$ matrix defined by $[\mathbf{A}]_{ij} = i + j$ and $\mathbf{b}$ is an $n \times 1$ column vector defined by $b_i = i + 1$.

```
%seed for random integer
format default
rng('default')
s = rng;
%iniate parameters
n=randi([3, 6]);
A = zeros(n, n);
b = zeros(n, 1);

%satisfy conditions A(i, j) = i+j
% and b(i) = i + 1
for i = 1:n
    b(i) = i + 1;
    for j = 1:n
        A(i, j) = i + j;
    end
end
%run naive gauss algorithm
x = naive_gauss(A, b);
x
```

```
x = 6×1
   NaN
   NaN
   NaN
   NaN
   NaN
   NaN
```

We see that $\mathbf{x}$ yields an infinite amount of solutions. This is to be expected since $det(\mathbf{A}) = 0$ as can be checked by the following command:

```
det(A)
```

```
ans = 0
```

```
function x = naive_gauss(A, b)
    n = length(b);
%forward elimination
    for k = 1:(n-1)
        for i = (k+1):n
            xmult = (A(i, k))/(A(k, k));
            A(i, k) = 0;
            for j = (k+1):n
                A(i, j) = A(i, j) - xmult*A(k, j);
                if abs(A(i, j)) < 10^(-12)
```

```
                    A(i, j) = 0;
                end
            end
            b(i) = b(i) - xmult*b(k);
        end
    end
%backwards substitution
    x = zeros(n, 1);
    x(n) = (b(n))/(A(n,n));
    for u = (n-1):-1:1
        sum = 0;
        for v = (u+1):n
            sum = sum + (A(u, v)*x(v));
        end
        x(u) = (b(u) - sum)/(A(u, u));
    end
end
```

**Computer Exercise 2.1.8**

The following program will use the *naive_gauss* algorithm to solve $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{b}$ is defined such that $\mathbf{x} = (1, 2, \ldots, n)^T$ where $n$ is the dimension of the square matrix $\mathbf{A}$. We will first define a preliminary array $\mathbf{x}_0 = (1, 2, \ldots, n)^T$ then use matrix multiplication $\mathbf{Ax}_0$ to acquire $\mathbf{b}$. *naive_gauss* will then be applied to $\mathbf{A}$ and $\mathbf{b}$ to determine if we end up having $\mathbf{x} = \mathbf{x}_0$.

```
%seed for random number generator
format default
rng('default')
s = rng;

n = 5;
A = randi([4, 20]).*rand(n,n);
x0 = (1:n)';
b = A*x0;
x = naive_gauss(A, b);
x
```

```
x = 5×1
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

We see that indeed $\mathbf{x} = \mathbf{x}_0$.

```
function x = naive_gauss(A, b)
    n = length(b);
%forward elimination
    for k = 1:(n-1)
        for i = (k+1):n
            xmult = (A(i, k))/(A(k, k));
            A(i, k) = 0;
            for j = (k+1):n
                A(i, j) = A(i, j) - xmult*A(k, j);
                if abs(A(i, j)) < 10^(-12)
                    A(i, j) = 0;
                end
            end
            b(i) = b(i) - xmult*b(k);
        end
    end
%backwards substitution
    x = zeros(n, 1);
    x(n) = (b(n))/(A(n,n));
    for u = (n-1):-1:1
        sum = 0;
        for v = (u+1):n
```

```
            sum = sum + (A(u, v)*x(v));
        end
        x(u) = (b(u) - sum)/(A(u, u));
    end
end
```

**Computer Exercise 2.3.1**

The following program will consist of a rewritten version of *Tri* using four arrays instead of five (the way it is written in the textbook). It will then test the new algorithm on both a nonsymmetric and symmetric tridiagonal system.

```
%seed for random number generator
format default
rng('default')
s = rng;

%nonsymmetric
n = randi([3, 6]);
a = randi([4, 20], n, 1);
d = randi([4, 20], n, 1);
c = randi([4, 20], n, 1);
b = randi([4, 20], n, 1);

b = tri(n, a, d, c, b)
```

```
b = 6×1
   66.8656
  -70.7710
    9.4103
   40.0701
  -36.4742
    9.8685
```

```
%symmetric
a = randi([4, 20], n, 1);
d = randi([4, 20], n, 1);
c = a;
b = randi([4, 20], n, 1);

b = tri(n, a, d, c, b)
```

```
b = 6×1
    0.6773
    0.3932
    0.3761
   -0.3795
    0.6091
    0.8897
```

Here, array "a" and array "c" are equal.

```
function b = tri(n, a, d, c, b)
   for i = 2:n
       xmult = a(i-1)/d(i-1);
       d(i) = d(i) - (xmult*c(i-1));
       b(i) = b(i) - (xmult*b(i-1));
   end
   b(n) = b(n)/d(n);
```

```
    for i = (n-1):-1:1
        b(i) = (b(i) - c(i)*b(i+1))/d(i);
    end
end
```

## Computer Exercise 2.3.5

The following program will use *Tri* to solve the following system where $n = 100$

$4x_1 - x_2 = -20$

$x_{j-1} - 4x_j + x_{j+1} = 40 \quad (2 \le j \le n-1)$

$-x_{n-1} + 4x_n = -20$

```
%seed for random number generator
format default
rng('default')
s = rng;

n = 100;
a = ones(n, 1);
d = -4*ones(n, 1);
c = ones(n, 1);
b = 40*ones(n, 1);
[a(n-1), d(1), d(n), c(1), b(1), b(n)] = deal(-1, 4, 4, -1, -20, -20);
b = tri(n, a, d, c, b);
b1 = b(1:20);
b2 = b(21:40);
b3 = b(41:60);
b4 = b(61:80);
b5 = b(81:100);
T = table(b1, b2, b3, b4, b5, 'VariableNames', ...
    {'b1 to b20', 'b21 to b40', 'b41 to b60', 'b61 to b80', 'b81 to b100'});
disp(T)
```

| b1 to b20 | b21 to b40 | b41 to b60 | b61 to b80 | b81 to b100 |
|---|---|---|---|---|
| -9.282 | -20 | -20 | -20 | -20 |
| -17.128 | -20 | -20 | -20 | -20 |
| -19.23 | -20 | -20 | -20 | -20 |
| -19.794 | -20 | -20 | -20 | -20 |
| -19.945 | -20 | -20 | -20 | -20 |
| -19.985 | -20 | -20 | -20 | -20 |
| -19.996 | -20 | -20 | -20 | -20 |
| -19.999 | -20 | -20 | -20 | -20 |
| -20 | -20 | -20 | -20 | -20 |
| -20 | -20 | -20 | -20 | -20 |
| -20 | -20 | -20 | -20 | -20 |
| -20 | -20 | -20 | -20 | -20 |
| -20 | -20 | -20 | -20 | -19.999 |
| -20 | -20 | -20 | -20 | -19.996 |
| -20 | -20 | -20 | -20 | -19.985 |
| -20 | -20 | -20 | -20 | -19.945 |
| -20 | -20 | -20 | -20 | -19.794 |
| -20 | -20 | -20 | -20 | -19.23 |
| -20 | -20 | -20 | -20 | -17.128 |
| -20 | -20 | -20 | -20 | -9.282 |

It appears the result is symmetric in its entries.

```
function b = tri(n, a, d, c, b)
   for i = 2:n
       xmult = a(i-1)/d(i-1);
       d(i) = d(i) - (xmult*c(i-1));
       b(i) = b(i) - (xmult*b(i-1));
   end
   b(n) = b(n)/d(n);
   for i = (n-1):-1:1
       b(i) = (b(i) - c(i)*b(i+1))/d(i);
   end
end
```

**Computer Exercise 2.3.6**

The following program will solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ for 50 different values of $\mathbf{b}$ where each $\mathbf{b}$ is a different cyclic permutation of $[1, 2, \ldots, 49, 50]^T$ and $\mathbf{A}$ is a $50 \times 50$ tridiagonal matrix given by:

$$\begin{bmatrix} 5 & -1 & & & & \\ -1 & 5 & -1 & & & \\ & -1 & 5 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 5 & -1 \\ & & & & -1 & 5 \end{bmatrix}$$

Only the first ten elements of each solution $\mathbf{x}_i$ for each $\mathbf{b}_i$ $(1 \le i \le 50)$ will be displayed

```
%initiate values
n = 50;
a = -1*ones(n, 1);
d = 5*ones(n, 1);
c = -1*ones(n, 1);

b = (1:n);
X = zeros(n, n);
%permute b and store x in X
for i = 1:n
    b = circshift(b, i);
    X(i, :) = tri(n, a, d, c, b);
end

%each row of X is a solution x_i
%display first 10 elements for each x_i
for i = 1:n
    array = X(i, 1:10);
    fmt=['first 10 elements of x%d: ' repmat(' %3.3f ',1,numel(array)) '\n'];
    fprintf(fmt, i, array)
end
```

```
first 10 elements of x1:  10.505  2.526  1.124  1.096  1.353  1.671  2.001  2.334  2.667  3.000
first 10 elements of x2:  12.610  15.051  13.647  3.182  1.261  1.124  1.359  1.672  2.001  2.334
first 10 elements of x3:  11.938  14.689  15.507  15.847  15.727  13.788  3.211  1.267  1.125  1.359
first 10 elements of x4:  10.884  13.419  14.212  14.641  14.994  15.327  15.640  15.875  15.733  13.789
first 10 elements of x5:  9.565  11.825  12.561  12.978  13.329  13.666  14.000  14.333  14.666  14.999
first 10 elements of x6:  7.982  9.912  10.579  10.982  11.330  11.666  12.000  12.333  12.667  13.000
first 10 elements of x7:  6.136  7.681  8.267  8.653  8.997  9.333  9.667  10.000  10.333  10.667
first 10 elements of x8:  4.026  5.130  5.624  5.991  6.331  6.666  7.000  7.333  7.667  8.000
first 10 elements of x9:  1.652  2.261  2.652  2.997  3.333  3.667  4.000  4.333  4.667  5.000
first 10 elements of x10:  12.197  14.987  15.738  15.704  13.783  3.210  1.267  1.125  1.359  1.672
first 10 elements of x11:  9.301  11.506  12.230  12.645  12.996  13.332  13.666  14.000  14.333  14.666
first 10 elements of x12:  6.136  7.681  8.267  8.653  8.997  9.333  9.667  10.000  10.333  10.667
first 10 elements of x13:  2.707  3.536  3.973  4.328  4.665  5.000  5.333  5.667  6.000  6.333
first 10 elements of x14:  12.197  14.987  15.738  15.704  13.783  3.210  1.267  1.125  1.359  1.672
first 10 elements of x15:  8.246  10.231  10.909  11.314  11.663  11.999  12.333  12.667  13.000  13.333
first 10 elements of x16:  4.026  5.130  5.624  5.991  6.331  6.666  7.000  7.333  7.667  8.000
first 10 elements of x17:  12.610  15.051  13.647  3.182  1.261  1.124  1.359  1.672  2.001  2.334
first 10 elements of x18:  7.982  9.912  10.579  10.982  11.330  11.666  12.000  12.333  12.667  13.000
```

```
first 10 elements of x19:  2.971  3.855  4.303  4.660  4.999  5.333  5.667  6.000  6.333  6.667
first 10 elements of x20:  10.884  13.419  14.212  14.641  14.994  15.327  15.640  15.875  15.733  13.789
first 10 elements of x21:  5.345  6.724  7.276  7.655  7.997  8.333  8.667  9.000  9.333  9.667
first 10 elements of x22:  12.610  15.051  13.647  3.182  1.261  1.124  1.359  1.672  2.001  2.334
first 10 elements of x23:  6.664  8.318  8.927  9.318  9.663  9.999  10.333  10.667  11.000  11.333
first 10 elements of x24:  0.333  0.667  1.000  1.333  1.667  2.000  2.333  2.667  3.000  3.333
first 10 elements of x25:  6.927  8.637  9.258  9.651  9.997  10.333  10.667  11.000  11.333  11.667
first 10 elements of x26:  10.505  2.526  1.124  1.096  1.353  1.671  2.001  2.334  2.667  3.000
first 10 elements of x27:  6.136  7.681  8.267  8.653  8.997  9.333  9.667  10.000  10.333  10.667
first 10 elements of x28:  11.938  14.689  15.507  15.847  15.727  13.788  3.211  1.267  1.125  1.359
first 10 elements of x29:  4.290  5.449  5.955  6.324  6.665  7.000  7.333  7.667  8.000  8.333
first 10 elements of x30:  9.565  11.825  12.561  12.978  13.329  13.666  14.000  14.333  14.666  14.999
first 10 elements of x31:  1.388  1.942  2.321  2.664  2.999  3.333  3.667  4.000  4.333  4.667
first 10 elements of x32:  6.136  7.681  8.267  8.653  8.997  9.333  9.667  10.000  10.333  10.667
first 10 elements of x33:  10.620  13.100  13.882  14.309  14.661  14.998  15.328  15.640  15.875  15.733
first 10 elements of x34:  1.652  2.261  2.652  2.997  3.333  3.667  4.000  4.333  4.667  5.000
first 10 elements of x35:  5.609  7.043  7.606  7.987  8.331  8.666  9.000  9.333  9.667  10.000
first 10 elements of x36:  9.301  11.506  12.230  12.645  12.996  13.332  13.666  14.000  14.333  14.666
first 10 elements of x37:  12.610  15.051  13.647  3.182  1.261  1.124  1.359  1.672  2.001  2.334
first 10 elements of x38:  2.707  3.536  3.973  4.328  4.665  5.000  5.333  5.667  6.000  6.333
first 10 elements of x39:  5.609  7.043  7.606  7.987  8.331  8.666  9.000  9.333  9.667  10.000
first 10 elements of x40:  8.246  10.231  10.909  11.314  11.663  11.999  12.333  12.667  13.000  13.333
first 10 elements of x41:  10.620  13.100  13.882  14.309  14.661  14.998  15.328  15.640  15.875  15.733
first 10 elements of x42:  12.610  15.051  13.647  3.182  1.261  1.124  1.359  1.672  2.001  2.334
first 10 elements of x43:  1.388  1.942  2.321  2.664  2.999  3.333  3.667  4.000  4.333  4.667
first 10 elements of x44:  2.971  3.855  4.303  4.660  4.999  5.333  5.667  6.000  6.333  6.667
first 10 elements of x45:  4.290  5.449  5.955  6.324  6.665  7.000  7.333  7.667  8.000  8.333
first 10 elements of x46:  5.345  6.724  7.276  7.655  7.997  8.333  8.667  9.000  9.333  9.667
first 10 elements of x47:  6.136  7.681  8.267  8.653  8.997  9.333  9.667  10.000  10.333  10.667
first 10 elements of x48:  6.664  8.318  8.927  9.318  9.663  9.999  10.333  10.667  11.000  11.333
first 10 elements of x49:  6.927  8.637  9.258  9.651  9.997  10.333  10.667  11.000  11.333  11.667
first 10 elements of x50:  6.927  8.637  9.258  9.651  9.997  10.333  10.667  11.000  11.333  11.667
```

```matlab
function b = tri(n, a, d, c, b)
   for i = 2:n
       xmult = a(i-1)/d(i-1);
       d(i) = d(i) - (xmult*c(i-1));
       b(i) = b(i) - (xmult*b(i-1));
   end
   b(n) = b(n)/d(n);
   for i = (n-1):-1:1
       b(i) = (b(i) - c(i)*b(i+1))/d(i);
   end
end
```