Numerical Methods I

System of Linear Algebraic Equations: LU Decomposition (Crout)

```
program mainLU
    implicit none

    integer :: n
    real, dimension(:,:), allocatable :: a
    real, dimension(:), allocatable :: x

    n = 5
    allocate(a(n, n+1))
    allocate(x(n))

    a(1,1) = 15
    a(1,2) = -1
    a(1,3) = 2
    a(1,4) = -3
    a(1,5) = 4
    a(1,6) = 8

    a(2,1) = 2
    a(2,2) = 23
    a(2,3) = -1
    a(2,4) = 5
    a(2,5) = -2
    a(2,6) = 82.4

    a(3,1) = -1
    a(3,2) = 3
    a(3,3) = 92
    a(3,4) = -5
    a(3,5) = 1
    a(3,6) = -764.9

    a(4,1) = 1
    a(4,2) = 2
    a(4,3) = 1
    a(4,4) = 27
    a(4,5) = 3
    a(4,6) = -8.9

    a(5,1) = -4
    a(5,2) = -6
    a(5,3) = -2
    a(5,4) = 8
    a(5,5) = 41
    a(5,6) = -201.9
```

```
    call lucrout(a, n, x)
    write(*,*) "Solution by LU Decomposition:"
    call printMatrix2D(x, n, 1)
end program mainLU
```

```
subroutine lucrout(aIn, n, x)
    implicit none

    integer, intent(in) :: n
    real, dimension(n, n+1), intent(in) :: aIn
    real, dimension(n), intent(out) :: x

    real, dimension(n, n) :: a
    real, dimension(n) :: b
    real, dimension(n, n) :: l
    real, dimension(n, n) :: u
    real, dimension(n, n) :: productOfLU
    real, dimension(n) :: y

    integer :: rowCount, columnCount
    real :: summation
    integer :: termCount

    write(*,*)
    write(*,*) "System of Linear Algebraic Equations"
    write(*,*) "Method: LU Decomposition (Crout)"
    write(*,*)

    do rowCount = 1, n
        do columnCount = 1, n
            a(rowCount, columnCount) = aIn(rowCount, columnCount)
        end do
        b(rowCount) = aIn(rowCount, (n + 1))
    end do

    l = 0
    u = 0

    write(*,*) "The Coefficient Matrix"
    call printMatrix2D(a, n, n)
    write(*,*) "The RHS"
    call printMatrix2D(b, n, 1)
    write(*,*) "The Lower Triangular Matrix"
    call printMatrix2D(l, n, n)
    write(*,*) "The Upper Triangular Matrix"
    call printMatrix2D(u, n, n)

    write(*,*) "Part 1 of 3: Finding L and U"
    write(*,*) "Part 1A: Set u(i,i) = 1 (diagonal elements of u)"
    do rowCount = 1, n
        u(rowCount, rowCount) = 1.0
    end do

    write(*,*) "The Lower Triangular Matrix"
    call printMatrix2D(l, n, n)
    write(*,*) "The Upper Triangular Matrix"
    call printMatrix2D(u, n, n)

    write(*,*) "Part 1B: Set l(i,1) = a(i,1) (elements in the first column of l)"
```

```
    do rowCount = 1, n
        l(rowCount, 1) = a(rowCount, 1)
    end do

    write(*,*) "The Lower Triangular Matrix"
    call printMatrix2D(l, n, n)
    write(*,*) "The Upper Triangular Matrix"
    call printMatrix2D(u, n, n)

    write(*,*) "Part 1C: Set u(1,j) = a(1,j) / l(1,1) (elements in the first row of
u)"
    do columnCount = 2, n
        u(1, columnCount) = a(1, columnCount) / l(1, 1)
    end do

    write(*,*) "The Lower Triangular Matrix"
    call printMatrix2D(l, n, n)
    write(*,*) "The Upper Triangular Matrix"
    call printMatrix2D(u, n, n)

    write(*,*) "Part 1D: Evaluate the remaining u(i,j) and l(i,j), row by row"
    do rowCount = 2, n
        do columnCount = 2, rowCount
            summation = 0
            do termCount = 1, (columnCount - 1)
                summation = summation + (l(rowCount, termCount) * u(termCount,
columnCount))
            end do
            l(rowCount, columnCount) = a(rowCount, columnCount) - summation
        end do
        do columnCount = (rowCount + 1), n
            summation = 0
            do termCount = 1, (rowCount - 1)
                summation = summation + (l(rowCount, termCount) * u(termCount,
columnCount))
            end do
            u(rowCount, columnCount) = (a(rowCount, columnCount) - summation) /
l(rowCount, rowCount)
        end do
    end do

    write(*,*) "The Lower Triangular Matrix (Final)"
    call printMatrix2D(l, n, n)
    write(*,*) "The Upper Triangular Matrix (Final)"
    call printMatrix2D(u, n, n)

    call matrixProduct(l, u, n, n, n, n, productOfLU)
    write(*,*) "The Coefficient Matrix"
    call printMatrix2D(a, n, n)
    write(*,*) "The Product of L and U"
    call printMatrix2D(productOfLU, n, n)

    write(*,*) "Part 2 of 3: Using forward substitution to solve ly = b"
    call forwardSubstitution(l, b, n, y)
```

```
    write(*,*) "Intermediate Column Vector y"
    call printMatrix2D(y, n, 1)

    write(*,*) "Part 3 of 3: Using backward substitution to solve ux = y"
    call backSubstitution(u, y, n, x)

    write(*,*) "Solution x"
    call printMatrix2D(x, n, 1)
end subroutine lucrout
```

```
subroutine forwardSubstitution(l, b, n, x)
    implicit none

    integer, intent(in) :: n
    real, dimension(n, n), intent(in) :: l
    real, dimension(n), intent(in) :: b
    real, dimension(n), intent(out) :: x

    integer :: rowCount

    real :: summation
    integer :: termCount

    x(1) = b(1) / l(1,1)
    do rowCount = 2, n
        summation = 0
        do termCount = 1, (rowCount - 1)
            summation = summation + (l(rowCount, termCount) * x(termCount))
        end do
        x(rowCount) = (b(rowCount) - summation) / l(rowCount, rowCount)
    end do
end subroutine forwardSubstitution
```

```
subroutine backSubstitution(u, b, n, x)
    implicit none

    integer, intent(in) :: n
    real, dimension(n, n), intent(in) :: u
    real, dimension(n), intent(in) :: b
    real, dimension(n), intent(out) :: x

    integer :: rowCount

    real :: summation
    integer :: termCount

    x(n) = b(n) / u(n,n)
    do rowCount = (n - 1), 1, -1
        summation = 0
        do termCount = (rowCount + 1), n
            summation = summation + (u(rowCount, termCount) * x(termCount))
        end do
        x(rowCount) = (b(rowCount) - summation) / u(rowCount, rowCount)
    end do
end subroutine backSubstitution
```

```
subroutine matrixProduct(a, b, rowsA, columnsA, rowsB, columnsB, ab)
    implicit none

    integer, intent(in) :: rowsA, columnsA, rowsB, columnsB
    real, dimension(rowsA, columnsA), intent(in) :: a
    real, dimension(rowsB, columnsB), intent(in) :: b
    real, dimension(rowsA, columnsB), intent(out) :: ab

    integer :: rowCount, columnCount

    real :: summation
    integer :: termCount

    if(columnsA /= rowsB) then
        stop "Error -- Matrixes cannot be multiplied."
    end if

    do rowCount = 1, rowsA
        do columnCount = 1, columnsB
            summation = 0
            do termCount = 1, columnsA
                summation = summation + ((a(rowCount, termCount)) * (b(termCount,
columnCount)))
            end do
            ab(rowCount, columnCount) = summation
        end do
    end do
end subroutine matrixProduct
```

```
subroutine printMatrix2D(matrix, rows, columns)
    implicit none

    integer, intent(in) :: rows, columns
    real, dimension(rows, columns), intent(in) :: matrix

    integer :: rowCounter, columnCounter

    do rowCounter = 1, rows
        do columnCounter = 1, columns
            write(*,10, advance='no') matrix(rowCounter, columnCounter)
        end do
        write(*,*)
    end do
    write(*,*)

    10 format(f7.2)
end subroutine printMatrix2D
```

Output

```
 System of Linear Algebraic Equations
 Method: LU Decomposition (Crout)

 The Coefficient Matrix
  15.00  -1.00   2.00  -3.00   4.00
   2.00  23.00  -1.00   5.00  -2.00
  -1.00   3.00  92.00  -5.00   1.00
   1.00   2.00   1.00  27.00   3.00
  -4.00  -6.00  -2.00   8.00  41.00

 The RHS
   8.00
  82.40
-764.90
  -8.90
-201.90

 The Lower Triangular Matrix
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00

 The Upper Triangular Matrix
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00

 Part 1 of 3: Finding L and U
 Part 1A: Set u(i,i) = 1 (diagonal elements of u)
 The Lower Triangular Matrix
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00
   0.00   0.00   0.00   0.00   0.00

 The Upper Triangular Matrix
   1.00   0.00   0.00   0.00   0.00
   0.00   1.00   0.00   0.00   0.00
   0.00   0.00   1.00   0.00   0.00
   0.00   0.00   0.00   1.00   0.00
   0.00   0.00   0.00   0.00   1.00
```

```
Part 1B: Set l(i,1) = a(i,1) (elements in the first column of l)
The Lower Triangular Matrix
 15.00   0.00   0.00   0.00   0.00
  2.00   0.00   0.00   0.00   0.00
 -1.00   0.00   0.00   0.00   0.00
  1.00   0.00   0.00   0.00   0.00
 -4.00   0.00   0.00   0.00   0.00

The Upper Triangular Matrix
  1.00   0.00   0.00   0.00   0.00
  0.00   1.00   0.00   0.00   0.00
  0.00   0.00   1.00   0.00   0.00
  0.00   0.00   0.00   1.00   0.00
  0.00   0.00   0.00   0.00   1.00

Part 1C: Set u(1,j) = a(1,j) / l(1,1) (elements in the first row of u)
The Lower Triangular Matrix
 15.00   0.00   0.00   0.00   0.00
  2.00   0.00   0.00   0.00   0.00
 -1.00   0.00   0.00   0.00   0.00
  1.00   0.00   0.00   0.00   0.00
 -4.00   0.00   0.00   0.00   0.00

The Upper Triangular Matrix
  1.00  -0.07   0.13  -0.20   0.27
  0.00   1.00   0.00   0.00   0.00
  0.00   0.00   1.00   0.00   0.00
  0.00   0.00   0.00   1.00   0.00
  0.00   0.00   0.00   0.00   1.00

Part 1D: Evaluate the remaining u(i,j) and l(i,j), row by row
The Lower Triangular Matrix (Final)
 15.00   0.00   0.00   0.00   0.00
  2.00  23.13   0.00   0.00   0.00
 -1.00   2.93  92.29   0.00   0.00
  1.00   2.07   0.98  26.78   0.00
 -4.00  -6.27  -1.81   8.55  40.47

The Upper Triangular Matrix (Final)
  1.00  -0.07   0.13  -0.20   0.27
  0.00   1.00  -0.05   0.23  -0.11
  0.00   0.00   1.00  -0.06   0.02
  0.00   0.00   0.00   1.00   0.11
  0.00   0.00   0.00   0.00   1.00

The Coefficient Matrix
 15.00  -1.00   2.00  -3.00   4.00
  2.00  23.00  -1.00   5.00  -2.00
 -1.00   3.00  92.00  -5.00   1.00
  1.00   2.00   1.00  27.00   3.00
 -4.00  -6.00  -2.00   8.00  41.00

The Product of L and U
 15.00  -1.00   2.00  -3.00   4.00
```

```
 2.00  23.00  -1.00   5.00  -2.00
-1.00   3.00  92.00  -5.00   1.00
 1.00   2.00   1.00  27.00   3.00
-4.00  -6.00  -2.00   8.00  41.00
```

```
Part 2 of 3: Using forward substitution to solve ly = b
Intermediate Column Vector y
  0.53
  3.52
 -8.39
 -0.32
 -4.70

Part 3 of 3: Using backward substitution to solve ux = y
Solution x
  3.10
  2.50
 -8.30
  0.20
 -4.70

Solution by LU Decomposition:
  3.10
  2.50
 -8.30
  0.20
 -4.70
```