# Numerical Methods I

## System of Linear Algebraic Equations: Jacobi Iterative Method

**program** mainIterative

**implicit** **none**

**integer** :: n

**real**, **dimension**(:,:), **allocatable** :: a

**real**, **dimension**(:), **allocatable** :: x

**real**, **dimension**(:), **allocatable** :: xGuess

**real**, **parameter** :: tolerance = 1.0e-6

n = 5

**allocate**(a(n, n+1))

**allocate**(x(n))

**allocate**(xGuess(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

xGuess = 0

**call** jacobi(a, n, xGuess, tolerance, x)

**write**(\*,\*) "Solution by Jacobi Iterative Method:"

**call** printMatrix2D(x, n, 1)

**end** **program** mainIterative

**subroutine** jacobi(aIn, n, xGuess, tolerance, x)

**implicit** **none**

**logical**, **external** :: diagonallyDominant

**integer**, **intent**(in) :: n

**real**, **dimension**(n, (n+1)), **intent**(in) :: aIn

**real**, **dimension**(n), **intent**(in) :: xGuess

**real**, **intent**(in) :: tolerance

**real**, **dimension**(n), **intent**(out) :: x

**real**, **dimension**(n, (n+1)) :: a

**real**, **dimension**(n, n) :: coefficientMatrix

**real**, **dimension**(n) :: xPrevious

**real**, **dimension**(n) :: error

**real** :: factor

**logical** :: solutionFound = .**false**.

**integer** :: iteration = 0

**integer** :: rowCount, columnCount

**write**(\*,\*)

**write**(\*,\*) "System of Linear Algebraic Equations"

**write**(\*,\*) "Method: Jacobi Iterative"

**write**(\*,\*)

a = aIn

**write**(\*,\*) "The Augmented Matrix"

**call** printMatrix2D(a, n, (n+1))

**do** rowCount = 1, n

**do** columnCount = 1, n

coefficientMatrix(rowCount, columnCount) = a(rowCount, columnCount)

**end** **do**

**end** **do**

**if**(diagonallyDominant(coefficientMatrix, n, n) .**eqv**. .**false**.) **then**

**stop** "Error (Jacobi Method): The coefficient matrix is not diagonally dominant."

**end** **if**

x = xGuess

xPrevious = x

**do** **while**(solutionFound .**eqv**. .**false**.)

iteration = iteration + 1

**do** rowCount = 1, n

factor = 0

**do** columnCount = 1, n

**if**(columnCount /= rowCount) **then**

factor = factor + a(rowCount, columnCount) \* xPrevious(columnCount)

**end** **if**

**end** **do**

x(rowCount) = (1 / a(rowCount, rowCount)) \* (a(rowCount, (n+1)) - factor)

**end** **do**

**do** rowCount = 1, n

error(rowCount) = *abs*((x(rowCount) - xPrevious(rowCount)) / (x(rowCount)))

**end** **do**

**write**(\*,10) "Iteration #", iteration

**write**(\*,\*) "Estimated Solution: "

**call** printMatrix2D(x, n, 1)

**write**(\*,\*) "Error: "

**call** printMatrix2D(error, n, 1)

**if**(*maxval*(error) < tolerance) **then**

solutionFound = .**true**.

**end** **if**

xPrevious = x

**end** **do**

10 **format**(a11, i2)

**end** **subroutine** jacobi

**logical** **function** diagonallyDominant(matrix, rows, columns)

**implicit** **none**

**integer**, **intent**(in) :: rows, columns

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

**integer** :: rowCount, columnCount

**real** :: diagonalElement

**real** :: sumOfNondiagonalElements

diagonallyDominant = .**true**.

**do** rowCount = 1, rows

diagonalElement = *abs*(matrix(rowCount, rowCount))

sumOfNondiagonalElements = 0

**do** columnCount = 1, columns

**if**(columnCount /= rowCount) **then**

sumOfNondiagonalElements = sumOfNondiagonalElements + *abs*(matrix(rowCount, columnCount))

**end** **if**

**end** **do**

**if**(sumOfNondiagonalElements > diagonalElement) **then**

diagonallyDominant = .**false**.

**return**

**end** **if**

**end** **do**

**end** **function** diagonallyDominant

**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: Jacobi Iterative

The Augmented Matrix

15.00 -1.00 2.00 -3.00 4.00 8.00

2.00 23.00 -1.00 5.00 -2.00 82.40

-1.00 3.00 92.00 -5.00 1.00-764.90

1.00 2.00 1.00 27.00 3.00 -8.90

-4.00 -6.00 -2.00 8.00 41.00-201.90

Iteration # 1

Estimated Solution:

0.53

3.58

-8.31

-0.33

-4.92

Error:

1.00

1.00

1.00

1.00

1.00

Iteration # 2

Estimated Solution:

3.13

2.82

-8.39

0.24

-4.69

Error:

0.83

0.27

0.01

2.37

0.05

Iteration # 3

Estimated Solution:

3.14

2.49

-8.31

0.18

-4.66

Error:

0.00

0.13

0.01

0.35

0.01

Iteration # 4

Estimated Solution:

3.09

2.50

-8.30

0.20

-4.69

Error:

0.02

0.01

0.00

0.09

0.01

Iteration # 5

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.02

0.00

Iteration # 6

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration # 7

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration # 8

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration # 9

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration #10

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration #11

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Iteration #12

Estimated Solution:

3.10

2.50

-8.30

0.20

-4.70

Error:

0.00

0.00

0.00

0.00

0.00

Solution by Jacobi Iterative Method:

3.10

2.50

-8.30

0.20

-4.70