**Lesson 0**

Root Finding – Bisection Method

Example: JUNE 2015 Q1.2

With a given Interval

Find the roots of

Given and

Recurrence Relation:

*This can be interpreted as*

[1] Find , , , …

OR

Use the starting points given.

Given and

[2] Get our interval. This is where the sign changes between two points

*In this example, and are different signs*

[3] Compute iterations

**Iteration 1**

*Sub the above answer into the function i.e.*

**Iteration 2**

*Sub the above answer into the function i.e.*

Since and , the root is between 1 and 2

Example: <https://www.youtube.com/watch?v=qEecNyRa5o4>

Without a given Interval

Find the roots of , Three iterations

Recurrence Relation:

*This can be interpreted as*

[1] Start with an initial guess i.e. Find , , , …

OR

Use the starting points given.

[2] Get our interval. This is where the sign changes between two points

Our sign changes between the last two

Our interval is (2,3)

[3] Compute iterations

*Since we don’t have or , we need to find these manually*

**Iteration 1**

interval is

*Sub the above answer into the function i.e.*

negative value, replace a

**Iteration 2**

interval is now

*Sub the above answer into the function i.e.*

positive value, replace b

**Iteration 3**

interval is now

*Sub the above answer into the function i.e.*

Therefore, our root is

Which is within our initial interval

**Lesson 0**

Root Finding – Newton’s (Newton-Raphson) Method

Example: <https://www.youtube.com/watch?v=PIPiv6gn_Ls>

Find the roots of , Three iterations

Given and

Recurrence Relation:

[1] Find

[2] Start with an initial guess i.e., Find , , , …

OR

Use the starting points given.

Given and

Let

**Iteration 1**

**Iteration 2**

**Iteration 3**

Where solution converges is our answer. It converges around

**Wolfram Alpha**

jacobian of (x^2+y^2z, 2.zxy+z^3, xy^2+3z^2y)

**Lesson 0**

Root Finding – Regula Falsi (False Position) Method

**Lesson 0**

Solving

<http://www.math.iit.edu/~fass/477577_Chapter_13.pdf>

Virtually all methods for solving or require operations. In practical applications A often has a certain structure and/or is sparse, i.e. contains many zeros.

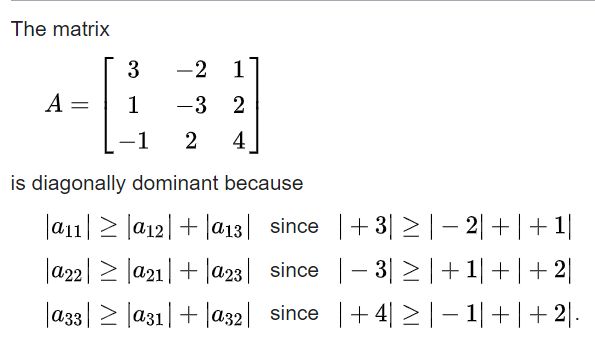
We use a variety of iteration methods to solve , usually opting for the one which is has the least cost.

**Glossary**

Diagonally dominant matrix

*Biggest element is the middle one in each row*

<https://en.wikipedia.org/wiki/Diagonally_dominant_matrix>



Convergence

**Lesson 0**

Method 1: The splitting approach

<https://en.wikipedia.org/wiki/Matrix_splitting>

The basic iterative scheme to solve will be of the form:

[1]

Iterative methods are based on a splitting of the matrix A of the form

Therefore, [1] becomes

**Lesson 0**

Method 2: Jacobi Method

<https://www.maa.org/press/periodicals/loci/joma/iterative-methods-for-solving-iaxi-ibi-jacobis-method>

Perhaps the simplest iterative method for solving Ax = b is Jacobi’s Method. **Good**: relatively easy to understand and thus is a good first taste of iterative methods;

**Bad**: because it is not typically used in practice

Still, it is a good starting point for learning about more useful, but more complicated, iterative methods

To choose and , we can use the Jacobi method.

Iterative algorithm.

Each diagonal element is solved for, and an approximate value is plugged in.

The process is then iterated until it converges.

**INPUT:**

initial guess to the solution,

(diagonal dominant) matrix

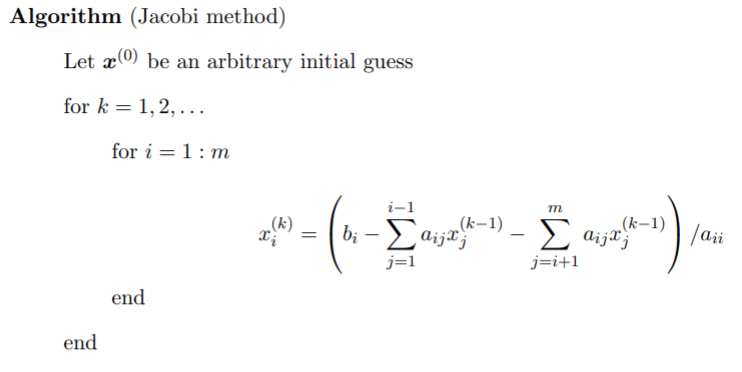
right-hand side vector

Convergence criterion

**OUTPUT:**

Solution when convergence is reached

The algorithm is formed from a LDU decomposition of a matrix

****

Method 2.1: Weighted Jacobi Method

<https://en.wikipedia.org/wiki/Jacobi_method#Python_example>

The weighted Jacobi iteration uses a parameter to compute the iteration as

which means our matrix A is not diagonally dominant.

Therefore, we don't know if the Jacobi method will converge or not.

We have , therefore

initial guess to the solution,`

(diagonal dominant) matrix

right-hand side vector

**example: ASS2 2021 Q2.1 b**

Iteration 1:

Iteration 2:

Iteration 3:

Solution:

[-49.0266 -34.1144 -34.4728 -31.9344]

Error:

[ -8.734426 -12.156534 -12.322124 -11.44533 ]

**Lesson 0**

Method 3: Gauss-Seidel Method

<https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method>

An improvement on the Jacobi method.

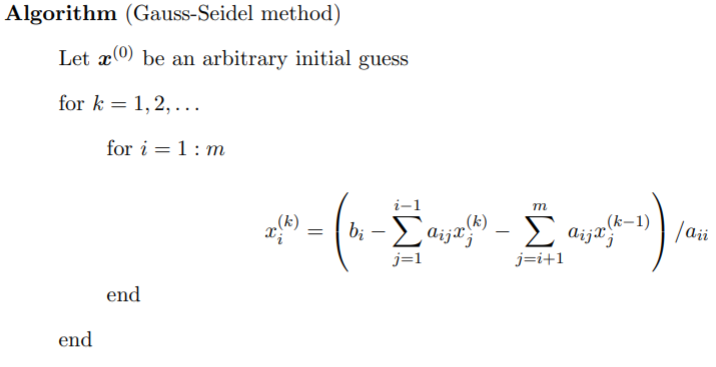
convergence is only guaranteed if the matrix is either strictly diagonally dominant, or symmetric and positive definite

**INPUT:**

(diagonal dominant) matrix

right-hand side vector

**OUTPUT:**

****

**Lesson 0**

Method 4: Successive Over-Relaxation (SOR)

Variant of Gauss-Seidel, results in faster convergence

<https://en.wikipedia.org/wiki/Successive_over-relaxation>

**INPUT:**

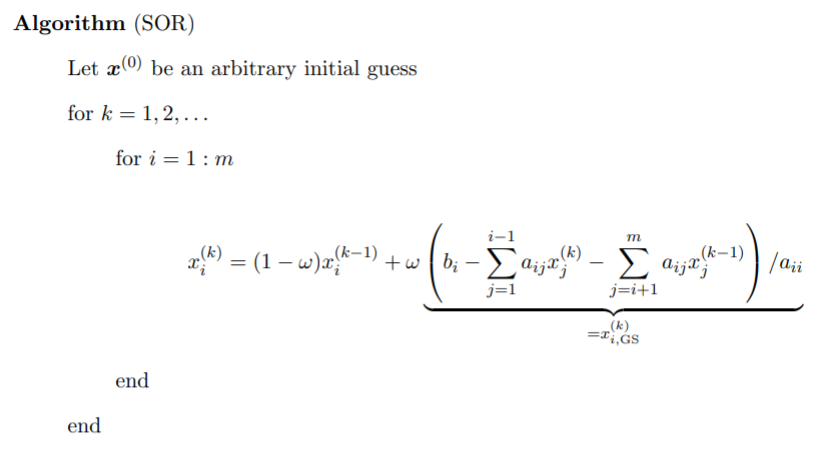
initial guess to the solution,

(diagonal dominant) matrix

right-hand side vector

relaxation factor

**OUTPUT:**

****

iteration 1 :

iteration 2 :

iteration 3 :

Method 4.1: Symmetric Successive Over-Relaxation (SSOR)

**Lesson 0**

Gaussian Elimination

**1 Gaussian Elimination without pivoting**

*(Naïve Gaussian Elimination algorithm)*

Forward Elimination

* Number of steps of forward elimination is , where is the number of equations

Back Substitution

example: ASS2 Q1

Given

Goal: elements under main diagonal to 0

Step 1: Write equation in the form

Augmented matrix

Number of steps:

FORWARD ELIMINATION

Step 1: calculate multipliers for each row

*Use 0.05 as the denominator*

*FROMT THIS POINT ON, INCREASE THE PRECISION TO FOUR DECIMAL PLACES AND USE THAT FOR EVERY CALCULATION*

:

:

:

Step 2: Apply algorithm from second row

Step 3: Apply algorithm to third row

:

*The first column will be 0.00, don’t bother calculating this*

Step 4: Apply algorithm from fourth row

:

BACK SUBSTITUTION

**2 Gaussian Elimination with pivoting**

Pivoting is basically swapping rows with others in the matrix beforehand. This can have a ‘profound’ effect on the results.

Again, the steps to achieve this:

Forward Elimination

* Number of steps of forward elimination is , where is the number of equations

Back Substitution

Compute for each iteration

Compute Compute

**3 Gaussian Elimination with scaled partial pivoting**

Given

:

:

:

Iteration 1:

*the first column*

*computed column that contains the biggest values from each row*

Since the row in with the largest value is , no row interchange is required

*If was the biggest, we would swap with*

:

Iteration 2:

Since the row in with the largest value is , no row interchange is required

:

Iteration 3:

Since the row in with the largest value is , no row interchange is required

Lesson 0

**4 LU decomposition**

lower–upper (LU) decomposition or factorization

L a lower triangular matrix and

U an upper triangular matrix

ASS2 2014

Step 1:

Step 2:

Step 3:

Step 4:

ASS2 2021

Step 1:

Step 2:

Step 3:

Step 4:

**5 LDU decomposition**

L a lower triangular matrix and

D is a diagonal matrix

U an upper triangular matrix

A Lower-diagonal-upper (LDU) decomposition is a decomposition of the form

Lesson 0

**Gauss-Siedel Method**

**ASS 2 2021 Q2.2 b**

iteration 1:

iteration 2:

iteration 3: