## Formal Logic

Background

This is a branch of mathematics that studies and develops numerical methods.

They help us find ad accept **approximate** answers to mathematical problems

numerically:

try x=2

try x=2.5

Analytically:

add +1

simplify

exact solution

Software for this course:

Octave

Numerical methods are widely applicable across various fields, and they are essential for solving mathematical problems that cannot be solved analytically. Here are some common application fields where numerical methods find extensive use:

|  |  |
| --- | --- |
| **Linear Systems** | NMLibforOctave  - Gauss elimination  - LU factorization  - Cholesky factorization  - Jacobi  - Gauss-Seidel  - Conjugate Gradient  - MINRES  - GMRES |
| **Nonlinear Equations**  **(Chapter 2)** | Many real-world problems involve nonlinear equations that cannot be solved algebraically. Used to approximate solutions to nonlinear equations. Applications include finding roots of functions, optimization problems, and modelling complex systems.  NMLibforOctave  - Bisection  - Fixed-point  - Newton-Raphson  - Secant  - Newton’s method for systems of nonlinear equations |
| **Interpolation** and Approximation | Numerical methods are used to approximate functions or data points between known values. Interpolation methods are used in fields such as computer graphics, signal processing, and data analysis  NMLibforOctave  - Monomial basis  - Lagrange interpolation  - Newton interpolation |
| **Numerical Integration** and Differentiation | Approximate definite integrals and compute derivatives of functions.  Used in physics, engineering, finance, and scientific computing  NMLibforOctave  - Trapezoid’s rule  - Simpson’s rule  - Newton-Cotes’ rule |
| Ordinary Differential Equations (ODEs):  Initial value problems | ODEs are ubiquitous in modelling dynamic systems across various disciplines such as physics, chemistry, biology, and engineering.  NMLibforOctave  - Euler  - Implicit Euler  - Modified Euler  - Fourth-order Rounge-Kutta  - Fourth-order predictor |
| Ordinary Differential Equations (ODEs):  Boundary value problems | NMLibforOctave  - Shooting method  - Finite difference method  - Colocation method |
| Partial Differential Equations (PDEs): | PDEs arise in many areas of science and engineering, including fluid dynamics, heat transfer, electromagnetism, and quantum mechanics.  Used to approximate solutions to PDEs. NMLibforOctave  - Method of lines (for Heat equation)  - Finite difference method for time-dependant PDEs (2-D solver for Advection, Heat  and Wave equations) :  – explicit method for Advection equation  – implicit method for Advection equation  – explicit method for Heat equation  – implicit method for Heat equation  – explicit method for Wave equation  – implicit method for Wave equation  - Finite difference method for time-independent PDEs (2-D solver for the Poisson  equation) |
| Optimization | Used to find the minimum or maximum of a function, subject to constraints. Optimization problems arise in various fields such as engineering design, operations research, finance, and machine learning.  NMLibforOctave  - Newton’s method  - Conjugate gradient method  - Lagrange multipliers |
| Eigenvalue Problems | Encountered in various scientific and engineering applications, such as structural analysis, quantum mechanics, and vibration analysis.  Used to compute eigenvalues and eigenvectors of matrices.  NMLibforOctave  - Power iteration  - Inverse method  - Rayleigh quotient iteration  - Orthogonal iteration  - QR iteration |

**DIRS FOR OCATAVE FILES:**

**001\_linear\_systems**

**002\_nonlinear\_equations**

**003\_interpolation\_and\_approximation**

**004\_numerical\_integration\_and\_differentiation**

**005\_ODE\_initial\_value\_problems**

**006\_ODE\_boundary\_value\_problems**

**007\_PDE**

**008\_optimization**

**009\_eigenvalue\_problems**

|  |  |
| --- | --- |
| **Nonlinear Equations**  **(Chapter 2)** | Many real-world problems involve nonlinear equations that cannot be solved algebraically. Used to approximate solutions to nonlinear equations. Applications include finding roots of functions, optimization problems, and modelling complex systems.  NMLibforOctave  - Bisection  - Fixed-point  - Newton-Raphson  - Secant  - Newton’s method for systems of nonlinear equations |

**Bisection (Binary search method)**

[Root finding]

used to find the roots of a continuous function within a given interval. The purpose of the bisection method is to iteratively narrow down the interval in which a root lies until a desired level of accuracy is achieved

[1] **Select an Interval:**

Choose an interval [a, b] such that the function changes sign within that interval, indicating that a root exists between a and b. Mathematically, this means

*f(a) should be a different sign from f(b)*

[2] **Evaluate the Function:**

Compute the function value at the midpoint of the interval. The midpoint, c, is given by

*Recurrence Relation??*

[3] **Check for Root:**

Determine if the midpoint c is a root of the function.

If , then c is the root and the process is complete.

[4] **Update Interval:**

Determine which half of the interval (either [a, c] or [c, b]) contains the root by checking the sign of

If , then the root lies in the left half of the interval; otherwise, it lies in the right half.

[5] **Repeat Iteratively:**

Update the interval to the new half-interval containing the root and repeat steps 2-4 until the width of the interval becomes smaller than a predetermined tolerance or until a desired level of accuracy is achieved

Variations

[ ] given interval

[ ] no given interval (Find , , , …)

[ ] fixed no. of iterations

With a given Interval

[1] **Initialize the interval**

*Find points and such that and*

[2] **Calculate the midpoint:**

find next value

Recurrence Relation:

**[3] Evaluate**

if then  is an exact root

- if or , stop

else if then ,

- if has the same sign as , set

else if then ,

- if has the same sign as , set

[4] **Repeat steps 2-3 until convergence**

until then  OR

until

Example: JUNE 2015 Q1.2

With a given Interval

Find the roots of

Given and

Octave

f: 3\*x^3 - 10\*x^2 + 13/4 + 15/2

a: 0

b: 2

tol: 1e-6

max\_iter: 100

[1] Initialize the interval [a,b]

function changes sign within the interval

f(a): 10.75

f(b): -5.25

-------------------------------------------------------- iter: 0

[2] Calculate the midpoint:

c=(a+b)/2 = 1

[3] Evaluate f(c)

- if f(c) has the same sign as f(a), set a=c

- if f(c) has the same sign as f(b), set b=c

f(c)= f(1) = (3.75)

f(a)= f(1) = (3.75)

root lies in the right half of the interval

--------------------------------------------------------

[4] Repeat steps 2-3 until convergence, stop

no. of iterations: 21

Root found using bisection method: 1.3412

|  |  |
| --- | --- |
| **Nonlinear Equations**  **(Chapter 2)** | Many real-world problems involve nonlinear equations that cannot be solved algebraically. Used to approximate solutions to nonlinear equations. Applications include finding roots of functions, optimization problems, and modelling complex systems.  NMLibforOctave  - Bisection  - Fixed-point  - Newton-Raphson  - Secant  - Newton’s method for systems of nonlinear equations |

**Newton-Raphson**

[Root finding]

used to find the root of a real-valued function.

The "root" of a function is a value x for which f(x)=0

[1] **Choose an initial guess:**

Start with an initial guess for the root of the function .

This can be any value close to the actual root, as the method may not converge if the initial guess is far from the true root.

[2] **Iterate:**

Use the formula derived from Newton's method to update the guess for the root. The formula for theguess, is given by:

Where:

- is the value of the function at the guess.

- is the derivative of the function evaluated at the guess.

[3] **Repeat:**

Continue iterating using the updated guess until a desired level of accuracy is achieved or until convergence criteria are met. Typically, convergence is considered to have occurred when the difference between consecutive iterations is smaller than a predefined tolerance level.

[4] **Check for Convergence:**

Monitor the sequence of iterates to ensure convergence. If the method converges, the iterates should get closer to the actual root with each iteration. If the method diverges, the iterates may oscillate or move away from the root.

[5] **Terminate:**

Once a satisfactory approximation to the root is obtained, terminate the iterations, and report the final value of the root.

With a given Interval

[1] **Initialize the interval**

*Find points and such that and*

[2] **Calculate the midpoint:**

find next value

**[3] Evaluate &**

find

find

Recurrence relation

[4] **Evaluate**

if then  is an exact root

- if or , stop

else

[5] **Repeat steps 3-4 until convergence**

until then  OR

until

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

Find the roots of ,

Given and

Octave

f: x^3 + 2\*x - 2

f\_derivative: 3\*x^2 + 2

initial\_guess: 1

tolerance: 1e-6

max\_iterations: 100

-------------------------------------------------------- iter: 0

[2] Calculate the midpoint:

x\_0 = 1

[3] Evaluate f(c) & f'(c)

f(x\_0): 1

f'(x\_0): 5

x\_1: 0.8

[4] Evaluate f(c)

if f(x\_0) then x\_0 is an exact root

-------------------------------------------------------- iter: 1

[2] Calculate the midpoint:

x\_0 = 0.8

[3] Evaluate f(c) & f'(c)

f(x\_0): 0.112

f'(x\_0): 3.92

x\_1: 0.771429

[4] Evaluate f(c)

if f(x\_0) then x\_0 is an exact root

-------------------------------------------------------- iter: 2

[2] Calculate the midpoint:

x\_0 = 0.771429

[3] Evaluate f(c) & f'(c)

f(x\_0): 0.00193586

f'(x\_0): 3.78531

x\_1: 0.770917

[4] Evaluate f(c)

if f(x\_0) then x\_0 is an exact root

-------------------------------------------------------- iter: 3

[2] Calculate the midpoint:

x\_0 = 0.770917

[3] Evaluate f(c) & f'(c)

f(x\_0): 6.05155e-07

f'(x\_0): 3.78294

x\_1: 0.770917

[4] Evaluate f(c)

if f(x\_0) then x\_0 is an exact root

Approximate root using Newton's method:

0.7709