**FLOW CHART**

no

END

yes

START

**NEWTON RAPHSON METHOD**

**EXAMPLE ONE**

Start timer

Set initial guess x0 = 2

Define function f(x) = x^3 - x - 2

Define derivative df(x) = 3\*x^2 - 1

Set tolerance err = 0.0005

Compute first Newton–Raphson update:

x = x0 - f(x0)/df(x0)

Compute error = |x - x0|

While error > tolerance:

Update x0 = x

Compute next Newton–Raphson update:

x = x0 - f(x0)/df(x0)

Compute error = |x - x0|

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Newton-Raphson.mat"

tic;

x0 = 2;

f = @(x0) x0^3 - x0 - 2;

df = @(x0) (3\*x0^2)-1;

err = 0.0005;

x = x0-(f(x0)/df(x0))

x = 1.6364

calc\_err = abs(x-x0);

while calc\_err>err

x0 = x;

x = x0-(f(x0)/df(x0))

calc\_err = abs(x-x0);

end

x = 1.5304

x = 1.5214

x = 1.5214

timetaken = toc

timetaken = 0.1385

xref = fzero(@(x0) x0^3 - x0 - 2,1.6);

save("Newton-Raphson.mat","x","timetaken","xref");

**EXAMPLE TWO**

Start timer

Set initial guess x0 = 0

Define function f(x) = x^3 + x - 1

Define derivative df(x) = 3\*x^2 + 1

Set tolerance err = 0.0005

Compute first Newton–Raphson update:

x = x0 - f(x0)/df(x0)

Compute error = |x - x0|

While error > tolerance:

Update x0 = x

Compute next Newton–Raphson update:

x = x0 - f(x0)/df(x0)

Compute error = |x - x0|

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Newton-Raphson\_2.mat"

tic;

x0 = 0;

f = @(x0) x0^3 + x0 - 1;

df = @(x0) (3\*x0^2)+1;

err = 0.0005;

x = x0-(f(x0)/df(x0))

x = 1

calc\_err = abs(x-x0);

while calc\_err>err

x0 = x;

x = x0-(f(x0)/df(x0))

calc\_err = abs(x0-x);

end

x = 0.7500

x = 0.6860

x = 0.6823

x = 0.6823

timetaken = toc

timetaken = 0.0540

xref = fzero(@(x0) x0^3 + x0 - 1,1);

save("Newton-Raphson\_2.mat","x","timetaken","xref");

**SECANT METHOD**

**EXAMPLE ONE**

Start timer

Set initial guesses:

x0 = 1

x1 = 2

Define function f(x) = x^3 - x - 2

Compute slope (approx derivative) between (x0, f(x0)) and (x1, f(x1)):

df = (f(x1) - f(x0)) / (x1 - x0)

Compute first update:

x = x1 - f(x1) / df

Compute error = |x - x1|

While error > tolerance (0.0005):

Update x1 = x

Recompute slope df = (f(x1) - f(x0)) / (x1 - x0)

Compute next update:

x = x1 - f(x1) / df

Update error = |x - x1|

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Secant.mat"

tic;

x0 = 1;

x1 = 2;

f0 = @(x0) x0^3 - x0 - 2;

f = @(x1) x1^3 - x1 - 2;

df = @(x1) (f(x1)-f0(x0))/(x1-x0);

err = 0.0005;

x = x1-(f(x1)/df(x1))

x = 1.3333

calc\_err = abs(x-x1);

while calc\_err>err

x1 = x;

x = x1-(f(x1)/df(x1))

calc\_err = abs(x-x1);

end

x = 1.6429

x = 1.4606

x = 1.5565

x = 1.5026

x = 1.5318

x = 1.5157

x = 1.5245

x = 1.5197

x = 1.5223

x = 1.5209

x = 1.5217

x = 1.5212

timetaken = toc

timetaken = 0.0812

xref = fzero(@(x0) x0^3 - x0 - 2,1.6);

save("Secant.mat","x","timetaken","xref");

**EXAMPLE TWO**

Start timer

Set initial guesses:

x0 = 0

x1 = 1

Define function f(x) = x^3 + x - 1

Compute slope (approx derivative) between (x0, f(x0)) and (x1, f(x1)):

df = (f(x1) - f(x0)) / (x1 - x0)

Compute first update:

x = x1 - f(x1) / df

Compute error = |x - x1|

While error > tolerance (0.0005):

Update x1 = x

Recompute slope df = (f(x1) - f(x0)) / (x1 - x0)

Compute next update:

x = x1 - f(x1) / df

Update error = |x - x1|

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Secant\_2.mat"

tic;

x0 = 0;

x1 = 1;

f0 = @(x0) x0^3 + x0 - 1;

f = @(x1) x1^3 + x1 - 1;

df = @(x1) (f(x1)-f0(x0))/(x1-x0);

err = 0.0005;

x = x1-(f(x1)/df(x1))

x = 0.5000

calc\_err = abs(x-x1);

while calc\_err>err

x1 = x;

x = x1-(f(x1)/df(x1))

calc\_err = abs(x-x1);

end

x = 0.8000

x = 0.6098

x = 0.7290

x = 0.6530

x = 0.7011

x = 0.6705

x = 0.6899

x = 0.6775

x = 0.6854

x = 0.6804

x = 0.6836

x = 0.6815

x = 0.6828

x = 0.6820

x = 0.6825

x = 0.6822

timetaken = toc

timetaken = 0.1129

xref = fzero(@(x0) x0^3 + x0 - 1,1);

save("Secant\_2.mat","x","timetaken","xref");

**BISECTOR METHOD**

**EXAMPLE ONE**

Start timer

Define function f(x) = x^3 - x - 2

Set initial interval [a, b] = [1, 2]

Set tolerance err = 0.0005

While (b - a) > err:

Compute midpoint c = (a + b) / 2

If f(a) \* f(c) < 0:

Root lies in [a, c], so set b = c

Else:

Root lies in [c, b], so set a = c

End loop

Set approximate root x = (a + b) / 2

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Bisection.mat"

tic;

f = @(x) x^3 - x - 2;

a = 1;

b = 2;

err = 5e-4;

while (b-a)>err

c=(a+b)/2;

if f(a)\*f(c)<0

b = c;

else

a = c;

end

end

x = (a+b)/2

x = 1.5212

timetaken = toc

timetaken = 0.0491

xref = fzero(@(x) x^3 - x - 2,1.6);

save("Bisection.mat","x","timetaken","xref");

**EXAMPLE TWO**

Start timer

Define function f(x) = x^3 + x - 1

Set initial interval [a, b] = [0, 1]

Set tolerance err = 0.0005

While (b - a) > err:

Compute midpoint c = (a + b) / 2

If f(a) \* f(c) < 0:

Root lies in [a, c], so set b = c

Else:

Root lies in [c, b], so set a = c

End loop

Set approximate root x = (a + b) / 2

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Bisection\_2.mat"

tic;

f = @(x) x^3 + x - 1;

a = 0;

b = 1;

err = 5e-4;

while (b-a)>err

c=(a+b)/2;

if f(a)\*f(c)<0

b = c;

else

a = c;

end

end

x = (a+b)/2

x = 0.6824

timetaken = toc

timetaken = 0.1038

xref = fzero(@(x) x^3 - x - 2,1);

save("Bisection\_2.mat","x","timetaken","xref");

**FIXED POINT ITERATION METHOD**

**EXAMPLE ONE**

Start timer

Define function f(x) = x^3 - x - 2

Rearrange into x = g(x), e.g., g(x) = (x + 2)^(1/3)

Set initial guess x0

Set tolerance err = 0.0005

Initialize iteration list x\_hist with x0

Compute first iterate:

x = g(x0)

Compute error = |x - x0|

While error > tolerance:

Update x0 = x

Compute next iterate:

x = g(x0)

Compute error = |x - x0|

Append x to iteration list

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Fixed\_point\_iteration.mat"

tic;

f = @(x) x^3 - x - 2;

g = @(x) (x+2).^(1/3);

x = 1;

err = 5e-4;

while true

x1 = g(x);

if abs(x1-x)<err

break

end

x = x1

end

x = 1.4422

x = 1.5099

x = 1.5197

x = 1.5211

timetaken = toc

timetaken = 0.0500

xref = fzero(@(x) x^3 - x - 2,1.6);

save("Fixed\_point\_iteration.mat","x","timetaken","xref");

**EXAMPLE TWO**

Start timer

Define function f(x) = x^3 + x - 1

Rearrange into x = g(x), e.g., g(x) = (1 - x)^(1/3)

Set initial guess x0

Set tolerance err = 0.0005

Initialize iteration list x\_hist with x0

Compute first iterate:

x = g(x0)

Compute error = |x - x0|

While error > tolerance:

Update x0 = x

Compute next iterate:

x = g(x0)

Compute error = |x - x0|

Append x to iteration list

End loop

Record final root value x

Stop timer and record timetaken

Compute high-precision reference root xref using fzero

Save variables (x, timetaken, xref) into file "Fixed\_point\_iteration\_2.mat"

tic;

f = @(x) x^3 + x - 1;

g = @(x) (1-x).^(1/3);

x = 0.5;

err = 5e-4;

while true

x1 = g(x);

if abs(x1-x)<err

break

end

x = x1

end

x = 0.7937

x = 0.5909

x = 0.7424

x = 0.6363

x = 0.7138

x = 0.6590

x = 0.6986

x = 0.6704

x = 0.6907

x = 0.6763

x = 0.6866

x = 0.6792

x = 0.6845

x = 0.6807

x = 0.6835

x = 0.6815

x = 0.6829

x = 0.6819

x = 0.6826

x = 0.6821

timetaken = toc

timetaken = 0.0755

xref = fzero(@(x) x^3 + x - 1,1);

save("Fixed\_point\_iteration\_2.mat","x","timetaken","xref");

**GROUPING DATA FOR PLOTTING**

Define list of methods = [Newton-Raphson, Secant, Bisection, Fixed Point]

Define list of problems = [Example1, Example2]

Initialize empty table for computation times

For each problem in problems:

Open new figure

For each method in methods:

Load the saved data file for this method and problem

Extract iterates (x), true root, and time

Compute error at each iteration = |x - true root|

Plot error vs iteration

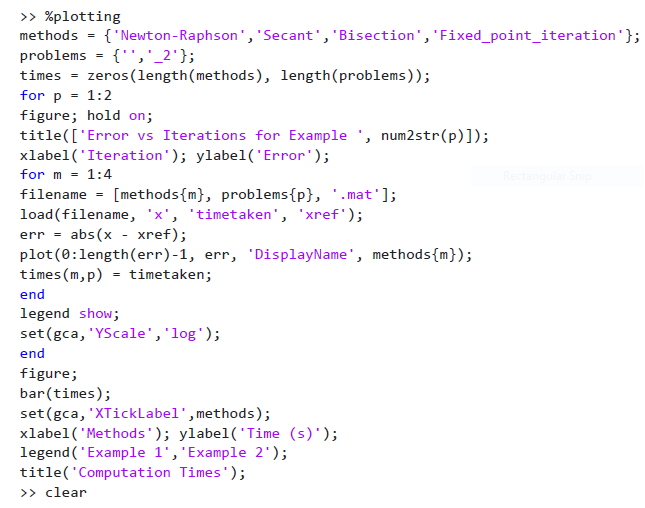
Store time in the times table

Show legend and set Y axis to log scale

Open new figure

Make bar chart of computation times for all methods and both problems

Label axes and add title



**GRAPH COMPARING ANALYTICAL SOLUTIONS TO SOLUTIONS OBTAINED WITH COMPUTATION TIME**

For each method and problem:

Plot error vs iteration curve

Use log scale for error axis

After looping:

Create bar chart comparing times

