

Lab Session 5

MA581: Numerical Computations Lab

R. Alam

September 02, 2025

Switch to format long e for all experiments

1. Solve the equation $xe^x - 2 = 0$ near $x = 1$ as follows.

```
f = @(x) x.*exp(x) - 2;
df = @(x) exp(x).*(x+1);
format long e, r = fzero(f,1)
```

Now perform the following steps (Newton iteration)

```
x = 1;
for k = 1:6
x(k+1) = x(k) - f(x(k)) / df(x(k));
end
```

Next, compute the errors with r computed by `fzero` taken as the exact zero and check the quadratic convergence of the Newton method

```
format short e
err = x' - r
semilogy(abs(err),'.-')
xlabel('k'), ylabel('|x_k-r|')
title('Quadratic convergence')
```

2. For a given function f write a function program $y = \text{newton}(f, df, x, N, tol)$ that performs Newton iterations to find a column vector y of length N such that $|f(y(N))| < tol$ with initial guess x . The f and df should be the function handles of f and its first derivative f' .

Organize your program in such a way that it terminates if either $|f(y)| < tol$ or the iterations exceed N . If $|f(y(N))| \geq tol$, and the iterations exceed N , then the program should produce an error message “zero could not be found for the given tolerance within the given iterations”.

3. Find $2^{\frac{1}{4}}$ correct up to 7 decimal places by using the function $f(x) = x^4 - 2$ and initial estimate $x = 1$ by

[a] Bisection [b] Regula-Falsi [c] Secant [d] Newton

methods. Also perform fixed point iterations with iteration function

$$[e] g(x) = \frac{x}{3} + \frac{4}{3x^3} \quad [f] g(x) = \frac{3x}{4} + \frac{1}{2x^3}$$

Record the number of iterations in each case and answer the following questions.

- (i) Does the number of iterations necessary to produce the desired accuracy in theory match with your output for Bisection method?

- (ii) Does any method fail to converge? If so, provide justification from theory for the failure.
 - (iii) Which method requires the least number of iterations?
 - (iv) Does Newton's Method exhibit quadratic convergence? Justify your answer with data.
4. Let $a > 0$. Then the square root $\alpha = \sqrt{a}$ is the zero of $f(x) = x^2 - a$. The Newton's method yields

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right), \quad n = 0, 1, 2, \dots$$

This scheme converges globally, that is, $x_n \rightarrow \sqrt{a}$ for any $x_0 > 0$. To verify global convergence, compute $\sqrt{173373}$ for various values of x_0 . Compare your computed result with the result obtained by using MATLAB command `sqrt(173373)`. Determine the number of iterations required to achieve $|x_n - \sqrt{173373}| \leq \text{tol}$, for $\text{tol} = 10^{-8}, 10^{-12}$. Do the results show quadratic order of convergence?

5. This problem discusses inverse interpolation which gives another method to find the zero of a function. Let $f : [a, b] \rightarrow \mathbb{R}$ be continuous and has only one zero at α in the interval, that is, $f(\alpha) = 0$ and $f(x) \neq 0$ for $x \neq \alpha$. Also assume that f has an inverse. Let x_0, x_1, \dots, x_n be $n + 1$ distinct nodes in $[a, b]$ with $f(x_j) = y_j$, $j = 0 : n$. Construct an interpolating polynomial $p_n(x)$ for $f^{-1}(x)$ by taking your data points as (y_j, x_j) , $j = 0 : n$. Write a MATLAB program for constructing $p_n(x)$. Observe that $f^{-1}(0) = \alpha$, the zero we are trying to find. Then, approximate the zero α , by evaluating the interpolating polynomial for f^{-1} at 0, that is, $p_n(0) \approx \alpha$. Use this method to find an approximation to the solution of $\log x = 0$ using the following data:

x	0.4	0.8	1.2	1.6
$\log x$	-0.92	-0.22	0.18	0.47

Next solve $\log(x) = 0$ using Newton's method with tolerance $\text{tol} = 10^{-6}$ and compare the result with that obtained by inverse interpolation. Estimate the errors for both the methods.

6. In neutron transport theory, the critical length of a fuel rod is determined by the solutions of the equation $\cot(x) = (x^2 - 1)/(2x)$. Use a zero finder (your own program) to determine the smallest positive solution of this equation. Compare your result with that obtained by using MATLAB function `fzero`.
7. Consider the problem of finding the smallest positive solution of the nonlinear equation $\cos(x) + 1/(1 + e^{-2x}) = 0$. Investigate, both theoretically and empirically, the following iterative schemes for solving this problem using the starting point $x_0 := 3$. For each scheme, you should show that it is indeed an equivalent fixed-point problem, determine analytically whether it is locally convergent and its expected convergence rate, and then implement the method to confirm your results.
- (a) $x_{k+1} = \arccos(-1/(1 + e^{-2x_k}))$.
 - (b) Newton's method.
8. The natural frequencies of vibration of a uniform beam of unit length, clamped on one end and free on the other, satisfy the equation $\tan(x) \tanh(x) + 1 = 0$. Use a zero finder (your own program) to determine the smallest positive solution of this equation. Compare your result with that obtained by using MATLAB function `fzero`.

9. Write a function program $[y, T] = \text{NewtonS}(f, J, x, \text{iter}, \text{tol})$ to find an approximate solution $y = (y(1), y(2))$ of a nonlinear system of equations $f(u, v) = 0$ via Newton's method. The inputs should be the function handles of f and its Jacobian matrix J , initial guess x , maximum number of iterations N and tolerance tol such that $\text{norm}(f(y(1), y(2))) < \text{tol}$.

Your program should terminate if either the iterations exceed N or $\text{norm}(f(y(1), y(2))) < \text{tol}$. If termination happens because the norm of $f(y(1), y(2))$ is not less than tol but the iterations exceed N , then the program should provide an error message "zero could not be found for the given tolerance within the given iterations".

10. Sketch the two curves on the u-v plane and find all solutions exactly via simple algebra.

(a)

$$\begin{aligned} u^2 + v^2 &= 1 \\ (1-u)^2 + v^2 &= 1 \end{aligned}$$

(b)

$$\begin{aligned} u^2 + 4v^2 &= 4 \\ v^2 + 4u^2 &= 4 \end{aligned}$$

Denoting each system by $F(u, v) = 0$, perform Newton's Method until $\text{norm}(F(u, v)) < 10^{-7}$ with starting guess $x = [1/2, 1/2]^\top$. Report the number of iterations required and the computed value of $[u, v]^\top$ in each case.

*** End ***