MATH 227B: Mathematical Biology
Homework 3

Karthik Desingu

February 4, 2026

Code availability

All the code used for these problems is available publicly at this GitHub repository.

Problem 1

a) Develop a code using Newton's method to solve a system of two equations with two unknowns: $f_i(x_1, x_2) = 0$ for $i = 1, 2$.

b) Discuss how you test your code and why your computed solution is correct.

c) Construct two examples to show the iteration converges quadratically.

d) Apply your method to find the steady state solutions of the following two-gene network with cross-activation:

$$\frac{dx}{dt} = \alpha_{min} + (\alpha_{max} - \alpha_{min})\frac{y^n}{e_{cy}^n + y^n} - \alpha_{deg}x$$
$$\frac{dy}{dt} = \beta_{min} + (\beta_{max} - \beta_{min})\frac{x^n}{e_{cx}^n + x^n} - \beta_{deg}y$$

e) Discuss how you test your code. Why do you believe your code is correct?

f) Find at least one parameter regime that shows bistability of the system, meaning for particular, there are two stable solutions. Note that x, y are the unknowns and all the rest are parameters for you to choose.

We study Newton's method for solving systems of equations of the form

$$f_i(x_1, x_2) = 0, \quad i = 1, 2.$$

Newton's Method: Background

For a system

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix},$$

Newton's method constructs a sequence $\mathbf{x}^{(k)}$ by solving

$$J(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)}),$$

and updating

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)},$$

where $J(\mathbf{x})$ is the Jacobian matrix

$$J(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{pmatrix}.$$

If $\mathbf{x}^*$ is a root of $\mathbf{F}$ and the Jacobian is nonsingular at $\mathbf{x}^*$, then Newton's method converges quadratically provided the initial guess is sufficiently close to $\mathbf{x}^*$. *At each iteration*, a linear system is solved to compute the Newton update. The specific function implementing Newton's method is shown in Listing 1.

Listing 1: Newton solver for a system of two equations.

```
def newton_system(F, x0, J, x_true=None, tol=1e-15, max_iter=50):
    """
    Solve F(x) = 0 using Newton's method for systems.

    Parameters
    ----------
    F : callable
        Function F(x) returning array-like of shape (n,).
    x0 : array-like
        Initial guess (n,).
    J : callable, optional
        Function J(x) returning Jacobian matrix (n, n).
        If None, finite-difference Jacobian is used.
    x_true : array-like, optional
        Ground-truth solution x*. If None, final iterate is used.
    tol : float
        Convergence tolerance on ||F(x)||.
    max_iter : int
        Maximum number of iterations.

    Returns
    -------
    x : ndarray
        Approximate solution.
    info : dict
        Dictionary with convergence info and histories.
    """

    x = np.asarray(x0, dtype=float)

```

```
    residuals_l = []      # ||F(x_k)||
    iterates = []         # store x_k for later error computation

    for k in range(max_iter):
        Fx = np.asarray(F(x), dtype=float)
        normF = np.linalg.norm(Fx)
        residuals_l.append(normF)
        iterates.append(x.copy())

        if normF < tol:
            break
        Jx = np.asarray(J(x), dtype=float)

        try:
            delta = np.linalg.solve(Jx, -Fx)
        except np.linalg.LinAlgError:
            return x, {
                "converged": False,
                "reason": "Jacobian singular",
                "residual_history": residuals_l
            }
        x = x + delta

    # Decide reference solution x*
    if x_true is None:
        x_star = iterates[-1]
    else:
        x_star = np.asarray(x_true, dtype=float)

    # Compute error history ||x_k - x*||
    error_history = [np.linalg.norm(xk - x_star) for xk in iterates
        ]

    return x, {
        "converged": residuals_l[-1] < tol,
        "iterations": len(residuals_l),
        "residual": residuals_l[-1],
        "residual_history": residuals_l,
        "error_history": error_history,
        "trajectory": np.array(iterates)
    }
```

The iteration is terminated when either

$$\|\mathbf{F}(\mathbf{x}^{(k)})\| < \texttt{tolerance}$$

or the maximum number of iterations is reached. The tolerance is set to the limit of *machine precision*, which is 10^{-15}.

Verification of the Solver

To validate the correctness of the Newton solver, we implemented a suite of unit tests. These tests are divided into three categories: *corner and trivial edge cases*, *linear systems*, and *nonlinear systems*. Each test is assigned a unique identifier for ease of reference. The full set of tests is summarized in Table 1.

- **Corner and Trivial Edge Cases (T1–T4):** These tests ensure the solver correctly handles degenerate or extreme inputs, including trivial solutions (T1), singular Jacobians (T2), invalid or NaN inputs (T3), and empty systems (T4). These cases check robustness, proper error reporting, and numerical stability.
- **Linear Systems (L1–L4):** Linear systems provide a baseline for correctness because exact solutions are known analytically.
  - L1 and L2 verify convergence from different initial guesses.
  - L3 tests decoupled variables via diagonal systems.
  - L4 ensures correct handling of triangular system structure.
- **Nonlinear Systems (N1–N4):** These tests assess the solver on genuinely nonlinear problems:
  - N1 demonstrates convergence to multiple solutions from initial guesses near each root, highlighting sensitivity to initial conditions.
  - N2 tests a system with four isolated solutions, verifying the solver can find all roots given suitable starting points.
  - N3 combines algebraic and transcendental terms, showing that the solver handles mixed nonlinearities accurately.
  - N4 includes logarithmic and multiplicative coupling, providing a challenging functional form for iterative solving.

Visual Verification While unit tests provide a quantitative check of correctness, visual verification offers intuitive confirmation of solver behavior. In this context, we define:

- **Residual:** $\|F(x_k)\|$, the Euclidean norm of the function at the current iterate x_k, which indicates how close the iterate is to satisfying the system.
- **Error:** $\|x_k - x^*\|$, the distance between the current iterate and the true solution x^*. This quantifies actual convergence to the known solution.

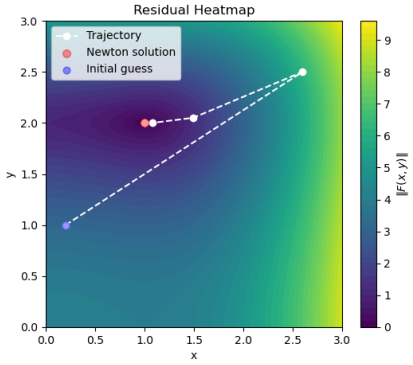
To illustrate correctness, we focus on two representative nonlinear systems from Table 1: N2 ($F(x) = [x^2 - 1, y^2 - 4]$) and N4 ($F(x) = [\log(x + 2) + y^2 - 3, xy - 1]$). For each, we select one solution and generate two plots:

1. A *residual heatmap* over a 2D grid around the convergence point, with the trajectory of Newton iterates overlaid. This demonstrates how the solver navigates the residual landscape from the initial guess to the final solution.
2. A plot of *error and residual vs iteration*, which quantitatively illustrates convergence behavior.

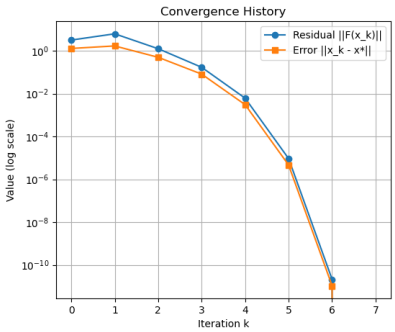
Table 1: Unit tests for the Newton solver. All obtained values match the expected. Tests are grouped into three categories.

ID	Test Case	Expected	Obtained
	Corner and Trivial Edge Cases		
T1	$F(x) = 0$, $x_0 = [0, 0]$	$[0, 0]$	$[0, 0]$
T2	$F(x) = [x_0^2, x_1^2]$, $J = 0$	$[1, 1]$	$[1, 1]$
T3	$F(x) = x$, $x_0 = [\text{NaN}, 0]$	$[\text{NaN}, 0]$	$[\text{NaN}, 0]$
T4	$F(x) =$ `<empty>`, $x_0 =$ `<empty>`	`<empty>`	`<empty>`
	Linear Systems		
L1	$F(x) = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} x - [1, 2]$	$[0, 2/3]$	$[0, 2/3]$
L2	$F(x) = \begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix} x - [6, 5]$	$[1, 1]$	$[1, 1]$
L3	$F(x) = \begin{bmatrix} 5 & 0 \\ 0 & 3 \end{bmatrix} x - [10, 6]$	$[2, 2]$	$[2, 2]$
L4	$F(x) = \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} x - [6, 5]$	$[2, 1]$	$[2, 1]$
	Nonlinear Systems		
N1	$F(x) = [x^2 + y^2 - 1, x - y]$	$[\sqrt{0.5}, \sqrt{0.5}]$ $[-\sqrt{0.5}, -\sqrt{0.5}]$	same
N2	$F(x) = [x^2 - 1, y^2 - 4]$	$[1, 2], [1, -2]$ $[-1, 2], [-1, -2]$	same
N3	$F(x) = [x^2 + y^2 - 5, e^x + y - 3]$	$[-0.250953, 2.221941]$	same
N4	$F(x) = [\log(x + 2) + y^2 - 3, xy - 1]$	$[0.706329, 1.415771]$	same

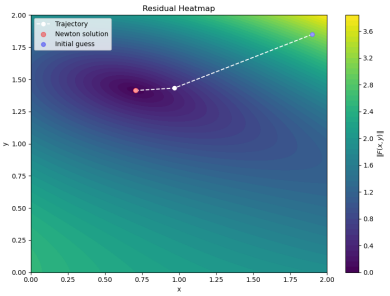
Summary Taken together, the unit tests in Table 1 and the visual verification plots in Figure 1 provide a rigorous demonstration of correctness. Unit tests ensure the solver produces expected solutions under a variety of functional forms and initial conditions, while the visual plots verify convergence trajectories, monotonic decrease of residuals, and approach to true solutions, particularly for nonlinear problems with multiple or complex roots.



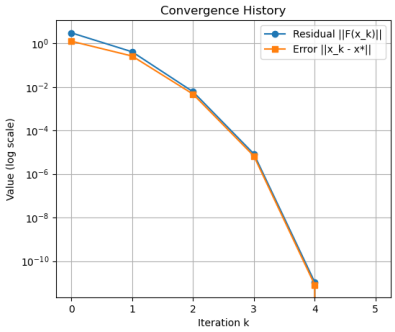
N2: Residual heatmap



N2: Convergence



N4: Residual heatmap



N4: Convergence

Figure 1: Visual verification of Newton solver convergence for selected nonlinear unit tests (Table 1, IDs N2 and N4). Heatmaps show the residual $\|F(x)\|$ around the convergence point with the dotted trajectory of Newton iterates from initial guess to final solution. Line plots show the evolution of both residual and error $\|x_k - x^*\|$ across iterations, highlighting monotonic convergence.

Quadratic Convergence Tests

To verify that the Newton solver converges quadratically, we test it on three nonlinear systems of equations. Each system is chosen to include different types of nonlinearities (polynomial, trigonometric, and exponential) while remaining smooth and well-behaved near the solution. These examples allow us to assess whether the numerical convergence rate matches the theoretical prediction.

Let $\mathbf{F} : R^2 \to R^2$ be continuously differentiable, and let $\mathbf{x}^*$ be a solution of $\mathbf{F}(\mathbf{x}) = 0$. If the Jacobian $J(\mathbf{x}^*)$ is nonsingular and the initial guess $\mathbf{x}^{(0)}$ is

sufficiently close to $\mathbf{x}^*$, then Newton's method generates iterates $\mathbf{x}^{(k)}$ such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq C\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2,$$

for some constant $C > 0$. This inequality shows that the error is squared at each iteration, which is known as *quadratic convergence*.

Test System S_1: Polynomial and Trigonometric Nonlinearity

$$f_1(x, y) = x^2 + y - 2,$$
$$f_2(x, y) = x + \sin(y) - 1.$$

This system combines a quadratic polynomial with a trigonometric function. It is a good test case because the functions are smooth, continuously differentiable, and the Jacobian contains both linear and nonlinear terms. The solution is not available in closed form, so iteration is necessary. This tests the method on mixed algebraic and transcendental nonlinearities.

Test System S_2: Exponential and Polynomial Nonlinearity

$$f_1(x, y) = e^x + y - 3,$$
$$f_2(x, y) = x + y^2 - 2.$$

The exponential term introduces rapid curvature in the solution surface and a strongly varying Jacobian, making this system a sensitive test of quadratic convergence.

Test System S_3: Exponential and Circular Geometry

$$f_1(x, y) = e^x - y - 1,$$
$$f_2(x, y) = x^2 + y^2 - 3.$$

Here, the second equation defines a circle, while the first introduces an exponential constraint. The solution involves the intersection of a nonlinear curve with a circle, making it geometrically nontrivial and providing a robust test of the solver.

Convergence Diagnostics

For each test system, two diagnostic plots are generated:

- **Error vs Iteration Plot:** The norm of the error

  $$e_k = \|\mathbf{x}^{(k)} - \mathbf{x}^*\|$$

  is plotted against iteration k on a semi-logarithmic scale. For quadratic convergence, the error decreases rapidly, producing a steep downward curve after only a few iterations.

- **Quadratic Ratio Plot:** The quantity

$$\frac{e_{k+1}}{e_k^2} = \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k)}\|^2}$$

  is plotted on a log-log scale along with a reference line of slope 2. According to the quadratic convergence theorem, this ratio should approach a constant, confirming that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \approx C\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2.$$

As shown in Fig. 2, the Newton solver exhibits clear quadratic convergence for all three test systems. The semi-log error plots demonstrate the rapid decay of the error, while the log-log ratio plots closely follow the reference slope 2, confirming that the theoretical convergence rate is achieved. These diagnostics provide strong evidence that the solver is correctly implemented and reliable for smooth nonlinear systems, including the two-gene network studied in this work.

d) Apply your method to find the steady state solutions of the following two-gene network with cross-activation:

$$\frac{dx}{dt} = \alpha_{min} + (\alpha_{max} - \alpha_{min})\frac{y^n}{e_{cy}^n + y^n} - \alpha_{deg}x$$
$$\frac{dy}{dt} = \beta_{min} + (\beta_{max} - \beta_{min})\frac{x^n}{e_{cx}^n + x^n} - \beta_{deg}y$$

e) Discuss how you test your code. Why do you believe your code is correct?

f) Find at least one parameter regime that shows bistability of the system, meaning for particular, there are two stable solutions. Note that x, y are the unknowns and all the rest are parameters for you to choose.

In the given system, x, y are gene expression levels; $\alpha_{\max}, \beta_{\max}$ control activation strength; e_{cx}, e_{cy} control activation thresholds; and n is the Hill coefficient.

Steady states satisfy

$$\frac{dx}{dt} = 0, \quad \frac{dy}{dt} = 0.$$

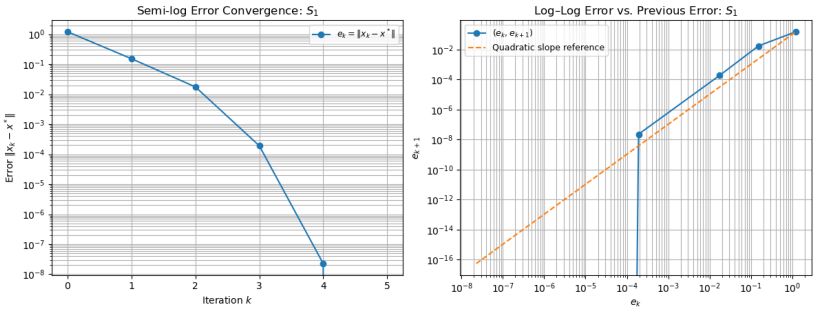
From $\frac{dx}{dt} = 0$,

$$x = \frac{\alpha_{\min} + (\alpha_{\max} - \alpha_{\min})\frac{y^n}{e_{cy}^n + y^n}}{\alpha_{\deg}},$$

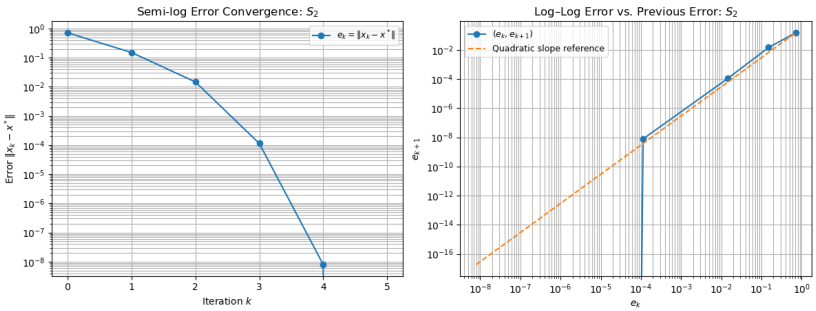
and similarly for y:

$$y = \frac{\beta_{\min} + (\beta_{\max} - \beta_{\min})\frac{x^n}{e_{cx}^n + x^n}}{\beta_{\deg}}.$$

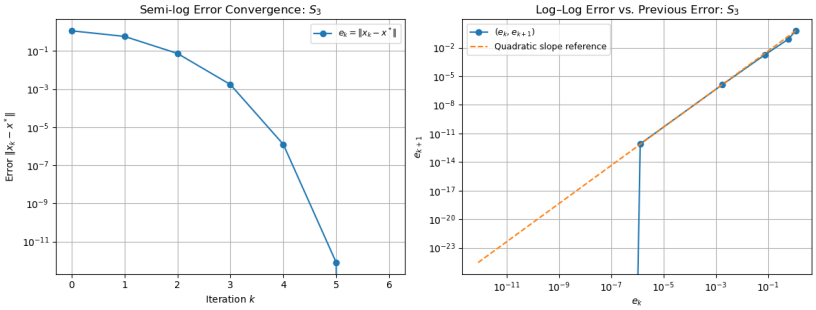
These define the nullclines of the system. Steady states occur at intersections of these curves.



System S_1 (polynomial + trigonometric)



System S_2 (exponential + polynomial)



System S_3 (exponential + circular geometry)

Figure 2: Quadratic convergence verification for three representative test systems. Each row shows two plots for a given system: (1) error versus iteration (semi-log) demonstrating rapid decay, and (2) log-log plot of the ratio e_{k+1}/e_k^2 compared with a reference slope of 2, confirming quadratic convergence.

Application of Newton's Method to the Two-Gene Network

To compute steady states of the two-gene network, Newton's method is applied to the nonlinear system

$$F_1(x, y) = 0, \quad F_2(x, y) = 0,$$

where F_1 and F_2 correspond to the right-hand sides of the differential equations. All results in this section use the parameter set

$$(\alpha_{\min}, \alpha_{\max}, \alpha_{\deg}) = (0.1,\ 5.5,\ 1.0),$$

$$(\beta_{\min}, \beta_{\max}, \beta_{\deg}) = (0.1,\ 4.5,\ 0.9), \quad (e_{cx}, e_{cy}, n) = (1.0,\ 1.5,\ 4),$$

which produces a bistable regime with two stable steady states and one unstable steady state.

The use of *asymmetric parameters* between the two genes (e.g., $\alpha_{\max} \neq \beta_{\max}$ and $e_{cx} \neq e_{cy}$) is intentional. This breaks the symmetry of the system and avoids degenerate cases in which nullclines intersect at symmetric or structurally identical points. As a result, the steady states are nontrivially positioned in phase space, making convergence more sensitive to the local Jacobian and thus providing a stronger test of the Newton implementation.

Verification Using Phase Portrait and Initial Guesses. Initial guesses were chosen near visually identified intersections of the nullclines. For each initial guess, Newton's method converged to a steady state that lies at the intersection of the nullclines and at a fixed point of the vector field. These steady states were also classified using the Jacobian eigenvalues, confirming the presence of two stable solutions and one unstable solution. This agreement between analytical nullclines, numerical solutions, and local stability provides direct evidence that the implementation correctly identifies steady states, as illustrated in Fig. 3(left).

Newton Basins of Attraction and Phase-Space Geometry. To further test correctness, Newton's method was applied over a dense grid of initial guesses spanning the phase plane:

$$x \in [0, 7], \quad y \in [0, 7],$$

using a grid of size 500×500 points. Each initial condition was colored according to which steady state Newton's method converged to. The resulting basin plot was overlaid with:

- the nullclines,
- the steady states,
- the vector field of the differential equations.

This overlay shows that most initial guesses converge to the steady state toward which the vector field points locally. In particular, regions where the vector field flows toward the low (or high) steady state correspond closely to the Newton basin for that state, as shown in Fig. 3(right).

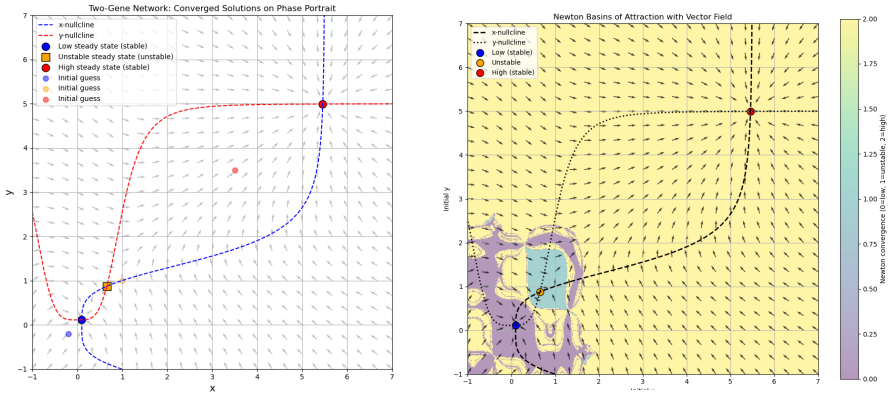


Figure 3: Application of Newton's method to the two-gene network. Left: phase portrait showing nullclines, vector field, and three steady states (two stable and one unstable) obtained from different initial guesses using Newton's method. Right: Newton basin-of-attraction plot overlaid with nullclines and vector field, showing how initial guesses in different regions converge to different steady states.

Quadratic Convergence Verification. To further justify correctness, the convergence of Newton's method for each of the three steady states was analyzed using standard measures of quadratic convergence. Specifically, for each steady state we plotted, once again, the (1) *error vs. iteration semi-log plot* and the (2) *quadratic ratio log-log plot*. These plots for the two-gene network are shown in Figure 4.

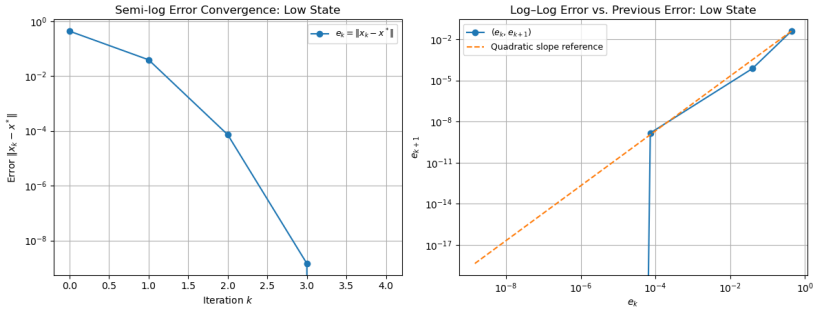
Relationship between Newton's Method and the Vector Field. Although both Newton's method and the vector field depend on the same functions F_1 and F_2, they describe different processes. The vector field describes the time evolution:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{F}(x, y),$$

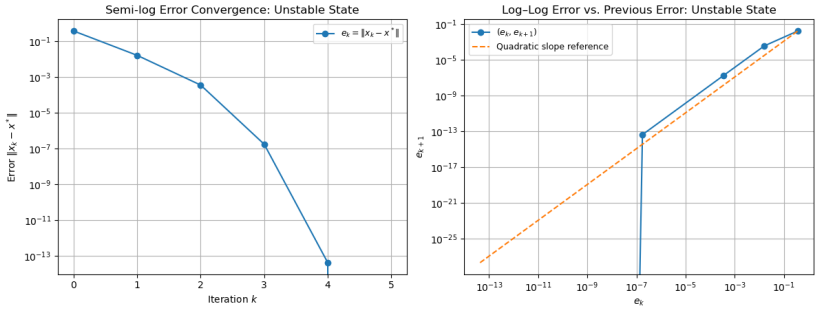
while Newton's method defines the iteration:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}).$$

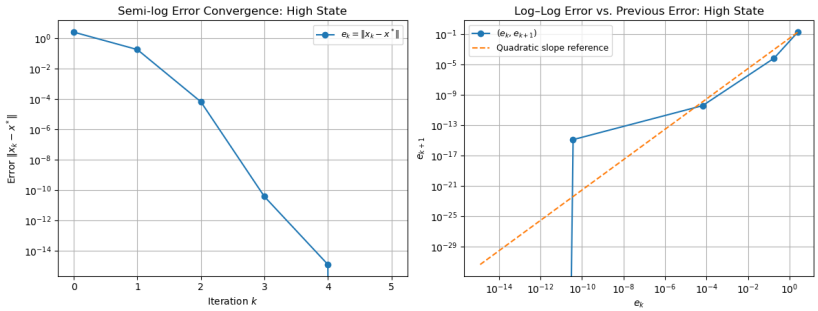
Thus, Newton's method does not move in the direction of $\mathbf{F}$ itself, but instead rescales this direction using the inverse Jacobian. This means Newton steps are



Low steady state



Unstable steady state



High steady state

Figure 4: Quadratic convergence verification for the three steady states of the two-gene network at the parameter set $(\alpha_{\min}, \alpha_{\max}, \alpha_{\deg}) = (0.1, 5.5, 1.0)$, $(\beta_{\min}, \beta_{\max}, \beta_{\deg}) = (0.1, 4.5, 0.9)$, $(e_{cx}, e_{cy}, n) = (1.0, 1.5, 4)$. Left: error versus iteration (semi-log). Right: log-log ratio plot of consecutive errors squared, compared to reference slope 2, confirming quadratic convergence for all three solutions. Each row corresponds to one of the steady states in Figure 3 (left).

designed to jump toward the nearest root of $\mathbf{F}$, not to follow solution trajectories of the differential equation.

As a result, Newton iterates may sometimes move in directions that differ from the local flow of the vector field. These discrepancies are expected mathematically because the vector field governs dynamics in time, whereas Newton's method solves a static nonlinear equation. Agreement between the two is therefore qualitative rather than exact.

Difficulty of Converging to the Unstable Steady State. The unstable steady state is observed to be difficult to reach using Newton's method unless the initial guess is very close to it. This behavior is consistent with its saddle structure. Linearizing near the unstable steady state $\mathbf{x}^*$ gives

$$\frac{d\mathbf{z}}{dt} = J(\mathbf{x}^*)\mathbf{z},$$

where $J(\mathbf{x}^*)$ has eigenvalues of opposite sign. This implies that trajectories are attracted along one direction (the stable manifold) and repelled along another (the unstable manifold).

Newton's method converges to this steady state only if the initial guess lies sufficiently close to its stable manifold. Since this manifold is one-dimensional in the plane, the set of such initial guesses has measure zero. Consequently, almost all initial guesses converge instead to one of the two stable steady states, which is clearly evident from the basin plot in Fig. 3.

Justification of Implementation Correctness. The combination of the phase portrait, basin-of-attraction plot (Fig. 3), and quadratic convergence plots (Fig. 4) provides a rigorous validation of the Newton implementation. Phase portraits and basins demonstrate that the method correctly identifies stable and unstable steady states in phase space, while the convergence plots show that each solution is obtained with the theoretically expected quadratic rate at later iterations. Together, these analyses confirm both the accuracy and reliability of the implementation across all branches of the system.

Parameter Sweeps and Bifurcation Analysis

To further validate the Newton implementation and explore the dynamical behavior of the two-gene network, we performed systematic parameter sweeps, varying one parameter at a time while holding the others fixed at the asymmetric bistable regime

$$(\alpha_{\min}, \alpha_{\max}, \alpha_{\deg}) = (0.1, 5.5, 1.0),$$
$$(\beta_{\min}, \beta_{\max}, \beta_{\deg}) = (0.1, 4.5, 0.9), \quad (e_{cx}, e_{cy}, n) = (1.0, 1.5, 4).$$

This parameter set produces two stable steady states and one unstable steady state, as discussed in the previous section.

For each parameter sweep, the initial guess for each subsequent parameter value was set to the Newton-converged solution from the previous parameter value for that branch. Only the first guess was set manually. This approach ensures that the initial guess remains close to the corresponding steady state, improving convergence and allowing consistent tracking of each branch across the sweep. It also eliminates the need for adjusting initial guesses as the phase space transitions.

For each sweep, we generated two complementary types of plots:

1. **Bifurcation plots** showing steady-state values x^* and y^* as a function of the swept parameter, illustrating how stable and unstable branches evolve.
2. **Phase portraits** for a few representative parameter values (low, middle, high), showing nullclines and Newton-converged steady states to provide geometric intuition and visual verification of the computed solutions.

Sweep of Hill Coefficient n. The Hill coefficient n was varied from 1 to 10. For low values $n = 1, 2$, the system is monostable, possessing a single globally attracting steady state. For $n \geq 3$, bistability emerges, with two stable steady states separated by an unstable saddle point.

Steady states for each n were computed numerically using Newton's method, with initial guesses chosen near the low or high branch, informed by the phase-plane geometry. In the **monostable regime**, all reasonable initial guesses converged to the single solution, demonstrating robustness. In the **bistable regime**, convergence depended on the initial guess, consistent with the basin-of-attraction analysis in Fig. 3.

The bifurcation plot of x^* and y^* versus n (Fig. 5, left) clearly illustrates the *transition from monostability* ($n = 1, 2$) *to bistability* ($n \geq 3$), showing two stable branches and the intermediate unstable branch. Phase portraits for three representative values of n (lowest, middle, highest) are shown in Fig. 5, right, displaying nullclines and the Newton-converged steady states. These plots confirm that the numerical solutions align with the analytical intersections of the nullclines across both monostable and bistable regimes.

Sweep of Maximum Activation $\alpha_{\max}$. The parameter $\alpha_{\max}$ was varied evenly in $[1.5, 6]$ using 25 points. Newton's method was applied for each value to compute all relevant steady states, with initial guesses chosen near the low and high branches.

The bifurcation plot of x^* and y^* versus $\alpha_{\max}$ (Fig. 6, left) shows smooth, monotonic shifts in the high and low stable branches as $\alpha_{\max}$ increases. The low branch remains relatively unchanged, while the high branch increases in x^* and y^*, consistent with the expected effect of enhanced gene activation. Phase portraits for three representative $\alpha_{\max}$ values (Fig. 6, right) illustrate

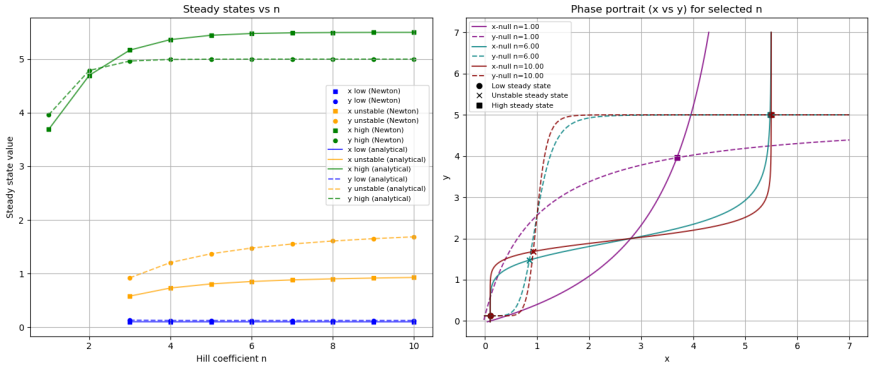


Figure 5: Bifurcation analysis for varying Hill coefficient n. *Left*: steady-state values x^* and y^* versus n, showing the transition from monostability ($n = 1, 2$) to bistability ($n \geq 3$) with two stable branches and one unstable branch. *Right*: phase portraits for three representative values of n (lowest, middle, highest), showing nullclines and Newton-converged steady states.

the corresponding shifts in nullclines and steady states, further demonstrating that Newton's method accurately tracks parameter-dependent changes in the solutions.

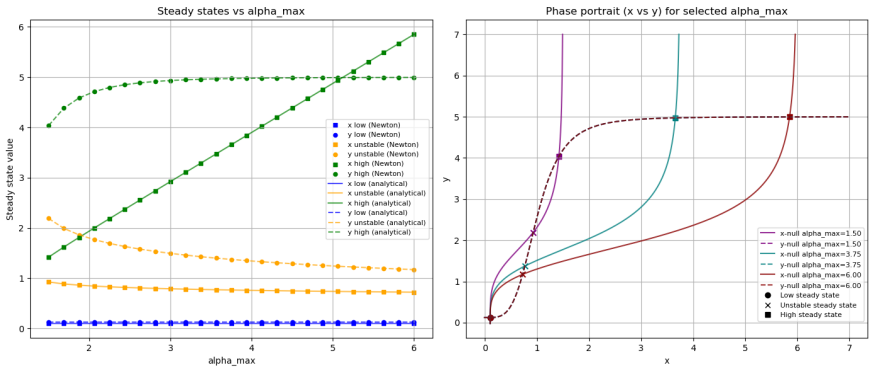


Figure 6: Bifurcation analysis for varying maximum activation $\alpha_{\max}$. *Left*: steady-state values x^* and y^* versus $\alpha_{\max}$, showing smooth shifts in the low and high stable branches. *Right*: phase portraits for three representative values of $\alpha_{\max}$ (lowest, middle, highest), showing nullclines and Newton-converged steady states. *Note that y-null does not change as α_{max} is varied.*

Sweep of Cross-Activation Threshold e_{cx}. The cross-activation threshold e_{cx} was varied evenly in $[1, 4]$ using 20 points. This parameter modulates the sensitivity of gene y to activation by gene x.

The bifurcation plot of x^* and y^* versus e_{cx} (Fig. 7, left) shows that increasing e_{cx} shifts the high steady state to higher x^* values while slightly lowering the low branch, consistent with the Hill function. Phase portraits for three representative values of e_{cx} (Fig. 7, right) illustrate how the nullclines move and how the Newton-converged solutions follow these shifts.

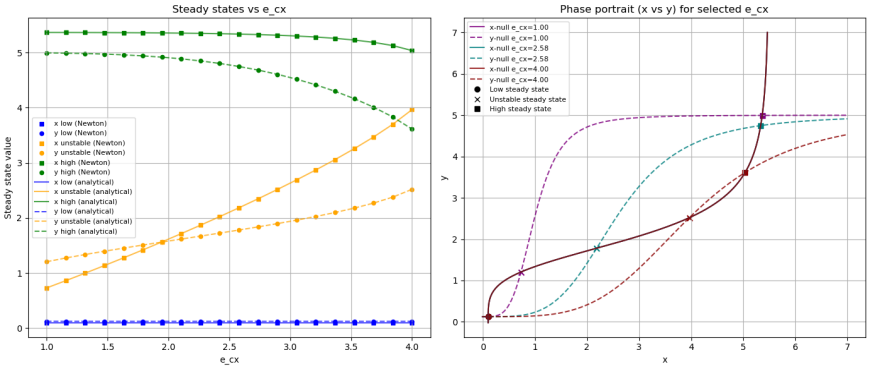


Figure 7: Bifurcation analysis for varying cross-activation threshold e_{cx}. *Left*: steady-state values x^* and y^* versus e_{cx}, showing that higher e_{cx} shifts the high branch upward while slightly lowering the low branch. *Right*: phase portraits for three representative values of e_{cx} (lowest, middle, highest), showing nullclines and Newton-converged steady states. *Note that x-null does not change as e_{cx} is varied.*

Interpretation and Conclusion

These analyses provide multiple lines of evidence supporting the correctness and robustness of the Newton implementation:

- Analytical steady-state solutions align precisely with nullcline intersections in all phase portraits.
- The method captures both monostable ($n < 3$) and bistable ($n \geq 3$) regimes, with convergence behavior determined by initial guesses in accordance with the basin-of-attraction analysis.
- Continuous parameter sweeps ($\alpha_{\max}$ and e_{cx}) produce smooth, physically consistent shifts in steady-state values, confirming that the solver tracks parameter-dependent roots correctly.
- The unstable branch is consistently identified when initial guesses are chosen near the saddle, validating the solver's accuracy even for delicate solutions, while also demonstrating the difficulty of converging to the unstable solution (note narrow basins in Figure 3 (right) for the unstable node).

Overall, these results demonstrate that Newton's method, in combination with careful selection of initial conditions guided by phase-plane geometry, reliably computes all relevant steady states across both monostable and bistable parameter regimes.

Thus, we developed and tested a Newton solver for nonlinear systems and further applied it to a biologically motivated model. The method successfully identified bistability and demonstrated quadratic convergence. The correctness of the solver is supported through unit tests, visual and analytical checks, convergence plots, and phase-plane geometry.

Code availability

All the code used for these problems is available publicly at this GitHub repository.