Project1

Consider the given data on the trajectory of Michaelis-Menten nonlinear ODE system over the combination of phase-spaces in (X, P, V, S, E). Compute the most probable values of eta, kappa, and epsilon along the arclength in the respective phase-spaces for the given start and end times.
Note: scan be parameter in the range (0.1, 1.5) with an increment of 0.1

Project2

The goal of this project is to **analyze the movement of proteins along a DNA strand** and determine key interaction points, such as **binding sites and first landing positions**. The simulation uses **Euclidean distance calculations** to measure how proteins interact with DNA.

Reading DNA & Protein Coordinates:

- The `readc()` function reads the `DNA.csv` and `PROTIEN.csv` files, extracting the **X, Y, Z** coordinates.
- These coordinates represent the spatial positions of **DNA base pairs** and **protein molecules** in a simulated system.

Simulating Protein-DNA Interaction:

The `solve()` function calculates the **distances between proteins and DNA**.

It identifies sites where a protein comes **within a threshold distance (`t`)** of DNA.

It records:

- **Total site visits (`nv`)** – How many times the protein gets close to DNA.
- **First landing position (`flp`)** – The first DNA base where a protein binds.
- **First landing step (`fls`)** – The step at which the first binding event occurs.
- **Max range (`mr`)** – The spread of binding interactions over the DNA sequence.

Final Output & Insights:

- The script **simulates protein motion** and outputs critical information about DNA-protein interactions.

Project3

Comparing Two Molecular Structures (`inda` and `indb`)

- `inda` and `indb` are **two sets of 3D coordinates**, likely representing different conformations of a biomolecule.
- Each row in these arrays corresponds to a **point in 3D space** (e.g., atoms or residues in a protein structure).
- The goal is to **find common spatial positions** between `inda` and `indb`.

2. Identifying Types of Structural Contacts

We check if coordinates in `inda` exist in `indb`, and classify them as:

1. **Correct contacts (`cr`)**
    - When the **same residue and same segment** exist in both structures.
    - This means the structural alignment is **unchanged**.
2. **Trapped contacts (`tr`)**
    - When the **same residue but different segments** are found.
    - This suggests a **structural shift** has occurred.
3. **Incorrect contacts (`incr`)**
    - When a coordinate match doesn't follow the above rules.
    - This means significant **structural reorganization** has occurred.

3. Computing the Radius of Gyration (Rg)

- The **Radius of Gyration (Rg)** measures the **compactness** of the structure.

Project4

Understanding the Variables

- `tau_t`: A **target response time dataset** (likely experimental or theoretical).
- `rin`: A range of input signal levels (`0.09` to `0.95` in steps of `0.01`).
- `dt = 1e-5`: A small **time step** for numerical integration.
- `Ps = 1`: Steady-state probability (likely used in the model).
- `errorbest = 1e5`: A large initial error value to track optimization progress.
- `vbest, hillbest, wbest, sigmabest, mubest`: Best parameter values (to be optimized).
- `value`: Determines whether the calculation is based on `P = 1` or `M = 2` (used in `responsetimeneg` function).
- `calc = 1`: Control variable for computation.

---

2. Parameter Optimization Using Nested Loops

The **goal** is to find the best-fitting parameters (`v, w, mu, sigma, hill`) that minimize the error between **model predictions (`rt(:,2)`) and experimental data (`tau_t`)**.

- The code systematically **varies each parameter** in a predefined range:
  - **Hill coefficient (`hill`)**: Controls cooperativity (1 to 10).
  - **Degradation rate (`mu`)**: Affects stability (`0.001` to `0.01`).
  - **Activation rate (`v`)**: Represents speed (`0.0001` to `0.001`).
  - **Decay parameter (`w`)**: Regulates adaptation (`0.1` to `1`).
  - **Sensitivity (`sigma`)**: Controls response steepness (`1` to `10`).
- For each parameter set, the `responsetimeneg` function is called to compute **predicted response times (`rt`)**.
- The **error** is computed as the sum of squared differences: error=∑(model response time−target response time)2\text{error} = \sum (\text{model response time} - \text{target response time})^2error=∑(model response time−target response time)2
- The **best parameters** are stored when a lower error is found.

---

3. Functions in the Code

(a) `psteadystate(mu, hillc, tol)`

- **Finds the steady-state concentration (`ps`)** of a regulatory molecule.
- Uses an iterative method to solve: μ=ps(μ+pshillc)\mu = ps (\mu + ps^{\text{hillc}})μ=ps(μ+pshillc)
- This equation is common in **biochemical reaction models**.

(b) `responsetimeneg(v, w, mu, sigma, hill, dt, tol, r, optmp)`

- **Simulates the response time** of a biochemical system.
- Uses **numerical integration** to evolve state variables over time:
  - **xtx_txt (active state variable)** → Changes based on `v` and `mu`.
  - **mtm_tmt (intermediate state variable)** → Modulated by `w`.
  - **ptp_tpt (output response)** → Controlled by `sigma` and `hill` function.
  - Computes **time (τ\tauτ)** for `p_t` or `m_t` to reach a threshold.
- Depending on `optmp`:
  - If `optmp = 1`: The system tracks `p_t`.
  - If `optmp = 2`: The system tracks `m_t`.

Project5

Curve-Fitting Model (`lm_func`)

- The function models the dataset using an exponential equation with multiple terms.
- It takes an independent variable `t` and parameters `p`, then returns the estimated values.

Jacobian Calculation (`lm_FD_J`)

- Computes the Jacobian matrix (partial derivatives) using finite differences.
- The Jacobian helps optimize the parameter updates.

Broyden's Rank-1 Update (`lm_Broyden_J`)

- If full Jacobian computation is expensive, this method updates it iteratively using the previous Jacobian.

Matrix Calculation for Optimization (`lm_matx`)

- Computes:
  - The Hessian matrix (second-order derivatives)

    - The change in Chi-squared (error metric)
    - The updated function evaluation.

Levenberg-Marquardt Algorithm (`lm`)

- Iteratively optimizes the parameter values to minimize the error.
- Uses gradient descent and the Hessian matrix to update `p`.
- Tracks convergence, error reduction, and correlation between parameters.