Research Questions:

Can we verify a mathematical chemical kinetics model of microtubule oscillations by systematically varying parameters and comparing the computational results with experimental data?

Introduction :

What are microtubules:

Cytoskeletal Framework Microtubules are one of the three components of the cell's cytoskeleton. Actin, microtubules, and intermediate filaments collectively form the cytoskeleton that maintains cellular structure and organization. The first two are dynamic structures that perform various cellular processes, while intermediate filaments serve only for structural support.

Structure of Microtubules Globular heterodimeric tubulin, composed of alpha and beta subunits, serves as the fundamental building block of microtubules. Tubulin dimers assemble into linear protofilaments through head-to-tail polymerization. These protofilaments associate laterally, with thirteen protofilaments forming a single microtubule that creates a hollow cylindrical structure.

Dynamic Properties Microtubules are polar, dynamic structures with distinct plus and minus ends that exhibit different assembly kinetics. This polarity is essential for their functional interactions with motor proteins and regulatory factors. Their dynamic nature allows for rapid reorganization in response to cellular needs.

Key Functions Cell division represents a critical function where microtubules form the mitotic spindle apparatus necessary for chromosome segregation. Additionally, microtubules serve as transport highways throughout the cell, providing tracks along which motor proteins "walk" to facilitate endocytosis and exocytosis of cellular cargo.

Microtubule Dynamics:

GTP Binding and Hydrolysis Both dimers of tubulin are bound to GTP, but only the GTP bound to beta tubulin is hydrolysable to GDP. Microtubules grow only from the plus end where beta tubulin with exposed GTP is incorporated. Microtubules can exist in T form (GTP-bound with strong binding) or D form (GDP-bound with weakened binding affinity), with hydrolysis occurring faster when tubulin is incorporated in a microtubule.

Polymerization Behavior The D form tends to depolymerize due to reduced binding strength between subunits. Conversely, the T form tends to polymerize as GTP-bound tubulin maintains stable inter-dimer interactions. This nucleotide-dependent switching between stable and unstable states drives the alternating phases of microtubule dynamics.

Individual Microtubules Display Rapid growth occurs through addition of GTP tubulin to the plus end, creating a stabilizing cap. Rapid shrinking follows depolymerization when the plus end "cap" hydrolyzes to GDP, destabilizing the entire structure. The rapid interconversion between growing and shrinking phases is called dynamic instability.

Why? Dynamic instability allows for rapid reorganization of microtubules as needed in chromosome segregation and cell division. This mechanism enables cells to quickly restructure their cytoskeletal networks in response to functional demands during critical cellular processes.

Mechanism of Dynamic Instability:

Tubulin + GTP → Stable Growth GTP provides the necessary energy for tubulin assembly into stable microtubule structures. The binding of GTP creates favorable conditions that promote polymerization and maintain structural integrity by overcoming entropic barriers in protein assembly.

GTP → GDP → Unstable Structure Hydrolysis of GTP to GDP results in energy loss that makes GDP-bound tubulin adopt a curved conformation. This conformational change weakens inter-subunit interactions and destabilizes the microtubule lattice structure.

GTP "Cap" Protects Microtubule The GTP cap forms a protective layer at the plus end that prevents structural collapse. This cap maintains straight tubulin conformation and provides stability to the underlying GDP-rich microtubule body.

Cap Lost → Rapid Collapse When the GTP cap is lost through complete hydrolysis, curved GDP-tubulin subunits splay outward and rapidly dissociate. This leads to catastrophic depolymerization and rapid structural collapse of the entire microtubule.

What are Microtubule Oscillations:

A Complex Phenomenon Observed on a Systems Level Microtubule dynamics represent a complex phenomenon that emerges at the systems level, distinct from individual microtubule behavior. Ensembles of microtubules undergo large cyclical changes in the total amount of polymerized tubulin within the cellular system. These manifest as periodic oscillations of the whole microtubule network rather than individual filament dynamics.

Oscillation Characteristics These oscillations eventually stop as the system reaches equilibrium conditions. The oscillations slowly dampen over time and occur before a non-zero steady state is established in the microtubule population. Even after oscillations cease, dynamic instability of individual microtubules continues at the molecular level.

Oscillations Do Not Account for Individual MT Dynamics Individual microtubules display dynamic instability as previously described, involving rapid switching between growth and shrinkage phases. However, system-level oscillations involve redistribution in both the length and number of total microtubules present in the ensemble. This redistribution represents a collective behavior that emerges from but is distinct from individual microtubule dynamics.

Different Scales, Different Mechanisms The phenomena operate at different organizational scales with distinct underlying mechanisms. Molecular-level processes govern individual microtubule dynamic instability, while system-level phenomena result from collective interactions and feedback mechanisms across the entire microtubule population.

Individual vs Systems level:

Individual Level: Dynamic Instability At the individual level, single microtubules randomly grow and shrink through dynamic instability processes. This behavior is rapid and random (stochastic), with individual microtubules switching unpredictably between growth and shrinkage phases. The stochastic nature means that predicting the behavior of any single microtubule is impossible without statistical approaches.

System Level: Two Pathways Possible At the system level, microtubule populations can follow two distinct pathways toward equilibrium. Monotonic behavior involves steady increase to equilibrium without oscillations, while oscillatory behavior exhibits cyclic changes before reaching steady state. The pathway taken depends on initial conditions and system parameters.

Different Timescales These phenomena operate on fundamentally different temporal scales with distinct persistence characteristics. Dynamic instability continues forever at the individual level, with microtubules perpetually switching between growth and shrinkage. In contrast, oscillations eventually stop when the system reaches steady state equilibrium.

Relationship Unclear The relationship between individual dynamic instability and system-level oscillations is not fully understood. Either phenomenon could simply be a byproduct of the other, or they may be governed by independent mechanisms that happen to coexist in microtubule systems.

Why are we studying this:

Oscillations Present an Interesting Higher-Order Phenomenon Individual microtubules display stochastic (random) growth and shrinking behavior at the molecular level. However, an ensemble of these randomly behaving components produces predictable and rhythmic oscillations of total assembled tubulin. This emergence of order from randomness represents a fascinating example of collective behavior in biological systems.

New Research Perspective This phenomenon requires a focus on the system rather than just individual components, embodying the principle that the whole is greater than the sum of its parts. In vivo systems operate collectively, with cells depending on coordinated microtubule networks for proper function. Understanding these emergent properties is crucial for comprehending cellular organization.

Cellular Processes Might Be Utilizing This Systems Level Property Cells may exploit this systems-level property of microtubules for predictable resource allocation among individual microtubules. Chromosome oscillations during mitosis may use the same mechanisms as described in theoretical models. This suggests that cellular processes have evolved to harness collective microtubule dynamics for critical functions.

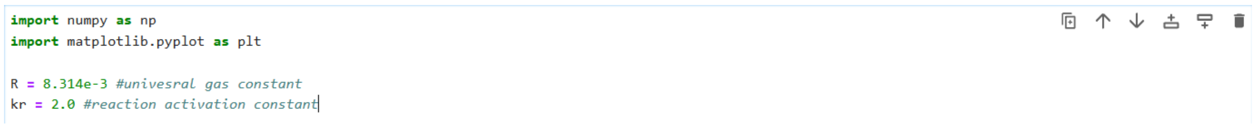
Broader Understanding and Prediction Fully understanding and computing microtubule oscillations will broaden our understanding of systems-level dynamics based on and verified by existing experimental data. A robust model can then be used to predict long-term behavior and future experimental outcomes, creating a cycle where existing data informs model development, which is then verified against data and used for future predictions.

Methodology:

1. We obtained the ODEs and rate constants describing the processes for Nucleation, Growth, Catastrophe from the paper: Microtubule Oscillations (Septet al 1998)
2. This paper explained the 'autocatalytic' component of the differential euqations that gives rise to the oscillations in the quantities of Assembled tubulin, available GTP and available GDP GDP->GTP in free tubulin
3. Under a fixed value of temperature, for a list of initial concentrations of Tubulin, using Eulers method we obtained solutions for the ODEs for the following quantities: N(number density), Ta(Assembled Tubulin), Td(Free GDP), Tt(Free GTP)
4. Analysed the plots of Assembled Tubulin against time, and inferred the qualitative nature of all the plots for constant temp and constant concentration
5. On changing the temperature, the rate constant changes, we conducted the same process mentioned above for different values of the Temperature of the System

Logic of the program:

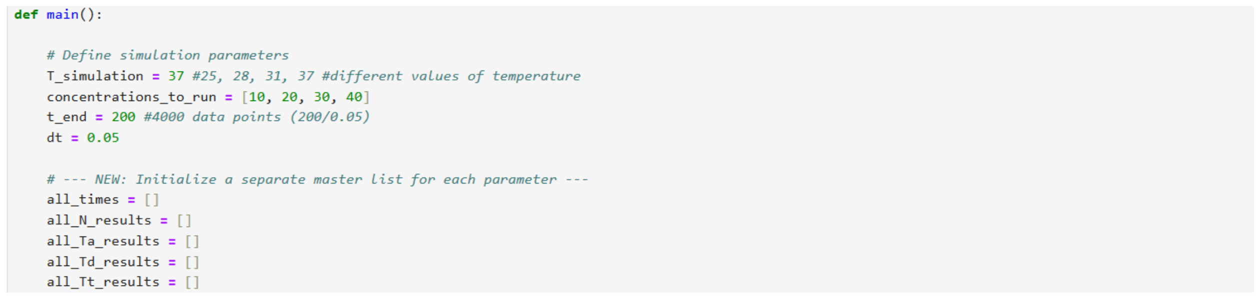
- Main calls the ‘run_simulation’ function, and eventually calls the ‘plot_simulation_lists’ function to visualise the results
- ‘run_simulation’ function uses various functions: ‘rates’ to obtain the rate constants, these rate constants are used to calculate the derivatives
- The ‘run_simulation’ function uses a helper function called ‘microtubule_odes’ to calculate the derivatives
- Main calls ‘plot_simulation_lists’, which uses the final lists containing the values of each of the parameters



```
import numpy as np
import matplotlib.pyplot as plt

R = 8.314e-3 #univesral gas constant
kr = 2.0 #reaction activation constant
```

Importing Libraries and Defining Constants:

- We import numpy for mathematical calculations, we import matplotlib.pyplot for data visualisation
- We define the universal gas constant, this is required in the calculator of the different rates of the reaction
- We define kr= 2, as determined by the authors, this value is constant throughout the simulation but can be tuned differently to observe a change in the oscillations

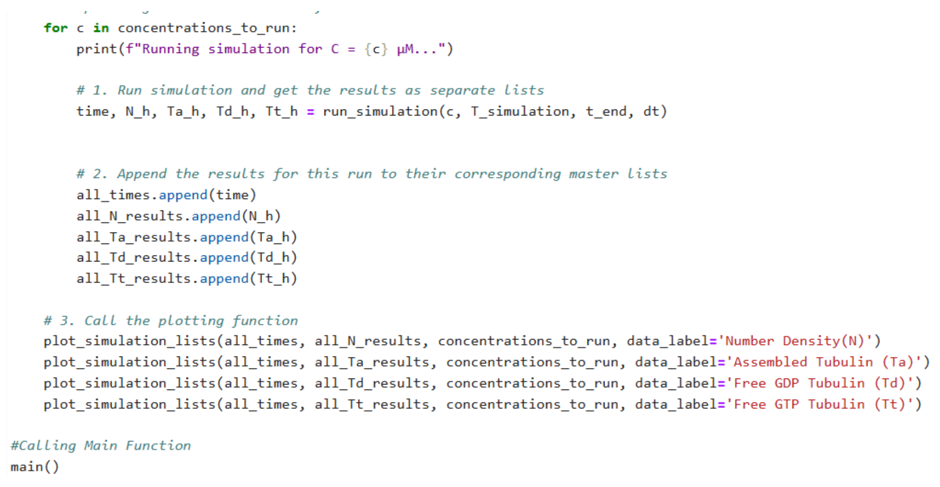


```python
def main():

    # Define simulation parameters
    T_simulation = 37 #25, 28, 31, 37 #different values of temperature
    concentrations_to_run = [10, 20, 30, 40]
    t_end = 200 #4000 data points (200/0.05)
    dt = 0.05

    # --- NEW: Initialize a separate master list for each parameter ---
    all_times = []
    all_N_results = []
    all_Ta_results = []
    all_Td_results = []
    all_Tt_results = []
```

Main Function:
Does not take any parameters, calls run_simulation function, calls plotting function

Code Flow:

- We define the various initial values of the parameters of the system: Temperatures (25, 28, 31,37 degrees C), concentrations of Tubulin (10,20,30,40 micro M), Time of total simulation (200s), Time steps(dt= 0.05s)
- We create the 'Master Lists' for all the values we are recording for each value concentration through the simulation



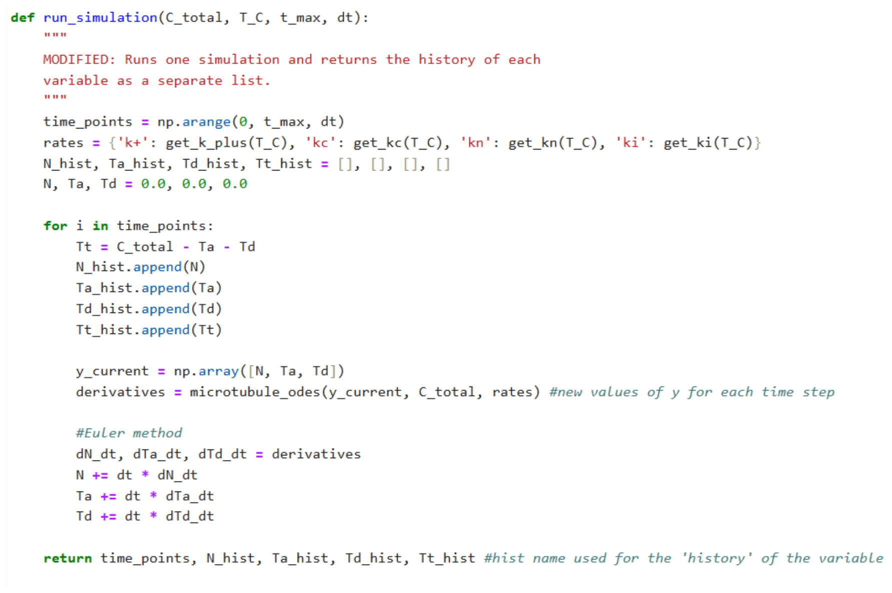
```python
    for c in concentrations_to_run:
        print(f"Running simulation for C = {c} µM...")

        # 1. Run simulation and get the results as separate lists
        time, N_h, Ta_h, Td_h, Tt_h = run_simulation(c, T_simulation, t_end, dt)


        # 2. Append the results for this run to their corresponding master lists
        all_times.append(time)
        all_N_results.append(N_h)
        all_Ta_results.append(Ta_h)
        all_Td_results.append(Td_h)
        all_Tt_results.append(Tt_h)

    # 3. Call the plotting function
    plot_simulation_lists(all_times, all_N_results, concentrations_to_run, data_label='Number Density(N)')
    plot_simulation_lists(all_times, all_Ta_results, concentrations_to_run, data_label='Assembled Tubulin (Ta)')
    plot_simulation_lists(all_times, all_Td_results, concentrations_to_run, data_label='Free GDP Tubulin (Td)')
    plot_simulation_lists(all_times, all_Tt_results, concentrations_to_run, data_label='Free GTP Tubulin (Tt)')

#Calling Main Function
main()
```

- for each value 'c' in the list of concentrations, we call the function 'run_simulation' and pass the parameters:
    - c: the concentrations [10,20,30,40]

        - T_simulation: the temperature of the system, can be different values, example: 20,25,30,37
        - t_end: the total time of the simulation (200s)
        - dt: each increment of time, timestep (0.05s)

- The function ‘run_simulation’ returns the values Number Density (N), Assembled Tubulin (Ta), Free GTP Tubulin (Tt), Free GDP Tubulin (Td)
- These values are appended to the master list for one value of the concentration, similarly, the simulation runs for four different values of concentration and generates four different lists of the values N, Ta, Tt, and Td

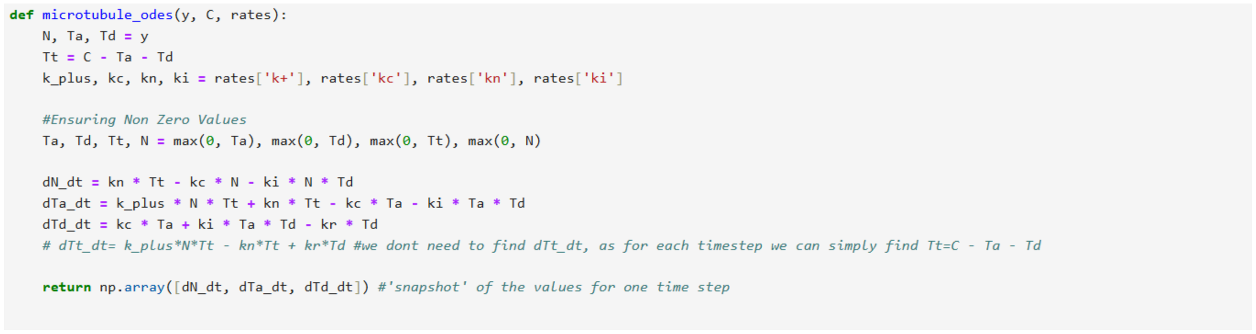
- The main function calls the plotting function passing the values of time, the results for one particular parameter(N, Ta, Tt, Tt) collected over different values of concentration but a fixed value of Temperature
- After plotting the results of the different results, we explicitly call the main function to run the entire simulation for one fixed value of the temperature and a list of concentrations.


```
def run_simulation(C_total, T_C, t_max, dt):
    """
    MODIFIED: Runs one simulation and returns the history of each
    variable as a separate list.
    """
    time_points = np.arange(0, t_max, dt)
    rates = {'k+': get_k_plus(T_C), 'kc': get_kc(T_C), 'kn': get_kn(T_C), 'ki': get_ki(T_C)}
    N_hist, Ta_hist, Td_hist, Tt_hist = [], [], [], []
    N, Ta, Td = 0.0, 0.0, 0.0

    for i in time_points:
        Tt = C_total - Ta - Td
        N_hist.append(N)
        Ta_hist.append(Ta)
        Td_hist.append(Td)
        Tt_hist.append(Tt)

        y_current = np.array([N, Ta, Td])
        derivatives = microtubule_odes(y_current, C_total, rates) #new values of y for each time step

        #Euler method
        dN_dt, dTa_dt, dTd_dt = derivatives
        N += dt * dN_dt
        Ta += dt * dTa_dt
        Td += dt * dTd_dt

    return time_points, N_hist, Ta_hist, Td_hist, Tt_hist #hist name used for the 'history' of the variable
```

Run Simulation function:
Calls ‘microtuble_odes’ function, calls individual functions to obtain various rates
Parameters accepted: Concentration (C_total), Temperature (T_C), t_max (total time=200s), time steps (dt=0.05s)
Parameters returned: time datapoints for the x-axis (time_points), values of Number Density (N_hist), Assembled Tubulin (Ta_hist), Free GTP Tubulin (Tt_hist), Free GDP Tubulin (Td_hist) in the form of lists for one particular value of concentration.

Code Flow:

- We calculate the time data points using tmax and dt, the result is 4000 data points (200/0.05)
- We get the rates from each of the independent functions defined for all the rates, each of the rates are functions of Temperature
- We initialize the empty values and empty lists for only N, Ta, Td, but not Tt. As we can find the value of Tt from: C = Ta+Tt+Td, therefore: Tt= C - Ta- Td
- We run the loop for each of the time points, hence we generate 4000 data points for each of the parameters of system corresponding to time
- We append the initial values N, Ta, Td, Tt to their lists
- Tt initial value is the concentration value (10,20,30,40 micro M)
- The variable of y_current acts as a snapshot of the current values [N, Ta, Td], this makes it easier to pass the values to the function ‘microtubule_odes’
- We use the helper function of ‘microtubule_odes’ to calculate the derivatives of each of the differential equations described by Equation 10 of the paper
- On using the values of the derivatives, we use Euler’s Method to update the value of each derivative for the differential equations, and the loop goes through another iteration
- These updated values are appended to the lists and using these updated values, new values of the derivatives are calculated
- After the loop has run through all the iterations, we return the values

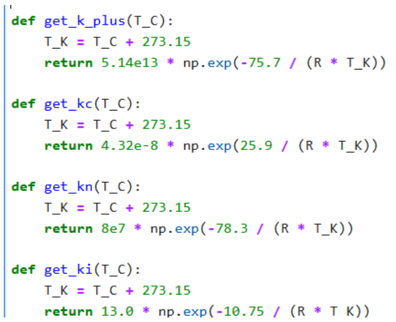


```python
def microtubule_odes(y, C, rates):
    N, Ta, Td = y
    Tt = C - Ta - Td
    k_plus, kc, kn, ki = rates['k+'], rates['kc'], rates['kn'], rates['ki']

    #Ensuring Non Zero Values
    Ta, Td, Tt, N = max(0, Ta), max(0, Td), max(0, Tt), max(0, N)

    dN_dt = kn * Tt - kc * N - ki * N * Td
    dTa_dt = k_plus * N * Tt + kn * Tt - kc * Ta - ki * Ta * Td
    dTd_dt = kc * Ta + ki * Ta * Td - kr * Td
    # dTt_dt= k_plus*N*Tt - kn*Tt + kr*Td #we dont need to find dTt_dt, as for each timestep we can simply find Tt=C - Ta - Td

    return np.array([dN_dt, dTa_dt, dTd_dt]) #'snapshot' of the values for one time step
```

Microtubule ODEs (Helper Function):
Parameters accepted: snapshot/state vector of [N,Ta,Td] (y), particular value of concentration [10,20,30,40], list of all the rate constants (rates)
Parameters returned: an array of all the updated derivatives obtained from the differential equations

Code Flow:
- We unpack the array y
- Calculate Tt from: Tt= C- Ta-Td
- Ensure nonzero values of all the parameters
- Calculate the derivatives from the differential equations for the current timestep and update the values
- We comment out the equation of dTt_dt as the value of Tt is determined from the concentration, hence we can avoid one entire differential equation to compute

- The values of all the updated derivatives are returned to the ‘run_simulation’ function

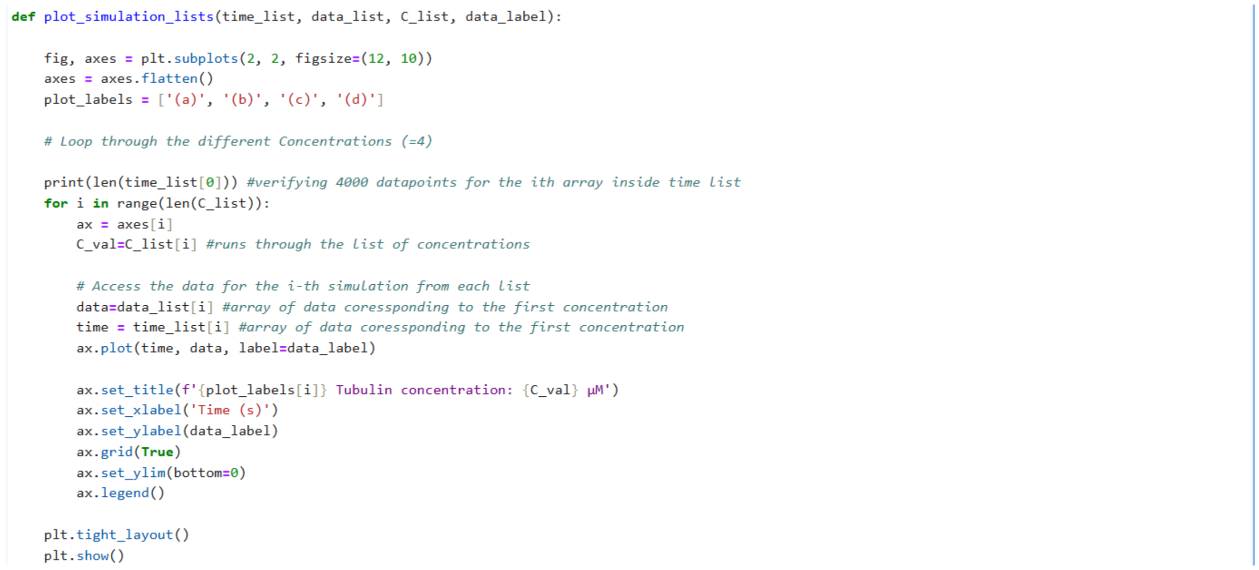


```python
def get_k_plus(T_C):
    T_K = T_C + 273.15
    return 5.14e13 * np.exp(-75.7 / (R * T_K))

def get_kc(T_C):
    T_K = T_C + 273.15
    return 4.32e-8 * np.exp(25.9 / (R * T_K))

def get_kn(T_C):
    T_K = T_C + 273.15
    return 8e7 * np.exp(-78.3 / (R * T_K))

def get_ki(T_C):
    T_K = T_C + 273.15
    return 13.0 * np.exp(-10.75 / (R * T_K))
```

Rate Functions:
Parameters Accepted: Temperature of the concentration (T_C)
Parameter Returned: The values of all the rates:

- K_plus: Rate constant for microtubule growth
- K_c : Rate constant for microtubule collapse
- K_n : Nucleation Rate constant, creating new microtubule
- K_i: Rate constant for Induced Collapse (‘Autocatalytic’ component)

Kr is the reaction rate constant, it is separately defined at the start of the program, and remains constant throughout the simulation. According to the Authors, the value 2 was chosen as a representative value, allowing the model to produce oscillations consistent with experimental observations



```python
def plot_simulation_lists(time_list, data_list, C_list, data_label):

    fig, axes = plt.subplots(2, 2, figsize=(12, 10))
    axes = axes.flatten()
    plot_labels = ['(a)', '(b)', '(c)', '(d)']

    # Loop through the different Concentrations (=4)

    print(len(time_list[0])) #verifying 4000 datapoints for the ith array inside time list
    for i in range(len(C_list)):
        ax = axes[i]
        C_val=C_list[i] #runs through the list of concentrations

        # Access the data for the i-th simulation from each list
        data=data_list[i] #array of data coressponding to the first concentration
        time = time_list[i] #array of data coressponding to the first concentration
        ax.plot(time, data, label=data_label)

        ax.set_title(f'{plot_labels[i]} Tubulin concentration: {C_val} µM')
        ax.set_xlabel('Time (s)')
        ax.set_ylabel(data_label)
        ax.grid(True)
        ax.set_ylim(bottom=0)
        ax.legend()

    plt.tight_layout()
    plt.show()
```

Plotting simulation lists:
Parameters Accepted: time data points (time_list), data we are plotting against time (data_list), list of values of concentration(C_list), the label of the data we are plotting (data_label)
Parameters Returned: four subfigure plots of the particular parameter we are plotting

Code Flow:

- We create a 2x2 subfigure plots for the four values of concentrations we have
- We use axes.flatten to create a 1D array for the subfigures as this is easier to index through an iterative loop
- Each of the data lists the function receives is a list of 4 arrays, each of the arrays contains 4000 data points corresponding to each of the time points we are calculating the differential equations for
- The loop runs 4 times, as len(C_list)=4, each time we access the data and time arrays corresponding to one value of concentration and plot 4 subfigures with the passed label of the data
- We set the title of the plots for each value of concentration
- We set the x axis and y axis titles as time and data label respectively
- After the four subfigures are created we run plt.show and plt.tight layout

Assumptions:

- **Complete Microtubule Collapse:** The model assumes that microtubules collapse entirely, which simplifies the model by ignoring the length distribution of the microtubule ensemble. This still captures the crucial aspect of releasing tubulin back into the solution.
- **Simplified Nucleation (nc=1):** The model assumes a single tubulin dimer is needed for nucleation. While this is a simplification, the inclusion of nucleation itself is what matters for oscillations, not the specific degree of nonlinearity.
- **Simplified Reactivation (Tu−GDP→Tu−GTP):** The conversion of GDP-tubulin to GTP-tubulin is known to be a multi-step process, but the model's simplified representation still successfully produces oscillations.
- These simplifying assumptions were made to create a **simple model with fewer free parameters**, providing clearer insight into the phenomenon. The model can be expanded later to be more realistic.

Limitations:

1. Ignores length distribution - By assuming complete collapse, the model cannot account for the actual length distribution of microtubules in the ensemble
2. Oversimplified nucleation kinetics - The linear nucleation assumption (nc=1) may not accurately represent the true nonlinear nature of microtubule nucleation
3. Missing intermediate steps - The single-step GDP-to-GTP conversion ignores potentially important intermediate chemical states and reactions
4. Limited predictive power for partial collapse scenarios - The model cannot predict or explain situations where microtubules undergo partial rather than complete collapse
5. Reduced biological realism - The simplifications, while mathematically convenient, may limit the model's ability to capture the full complexity of real microtubule dynamics

6. Potential parameter fitting issues - While the authors argue for fewer parameters, the simplifications might require compensation through less realistic parameter values

The authors acknowledge these limitations but justify them by arguing that the simplifications preserve the essential oscillatory behavior while providing clearer theoretical insights with fewer free parameters.

Future Outlook:

1. Mechanisms for oscillations
    - Synchronized fluctuations: All MTs move together
2. What relationship it probably has to dynamic instability
3. Possible biological relevance Identify biochemical triggers to these oscillations
4. Understanding if it is related to the chromosome oscillations during mitosis

References:

1) Obermann H, E M Mandelkow, G Lange, E Mandelkow," Microtubule oscillations. Role of nucleation and microtubule number concentration." 1990 https://www.sciencedirect.com/science/article/pii/S0021925819395766
2) Sept et al, 1998, A Chemical Kinetics Model for Microtubule Oscillations
3) https://doi.org/10.1016/0968-0004(87)90024-7
4) https://www.cell.com/current-biology/fulltext/S0960-9822(06)01771-4
5) https://www.pnas.org/doi/pdf/10.1073/pnas.84.15.5257
6) doi: 10.1073/pnas.84.15.5257
7) https://www.sciencedirect.com/science/article/pii/S0006349524000328
8) https://pubmed.ncbi.nlm.nih.gov/7068758/
9) https://pmc.ncbi.nlm.nih.gov/articles/PMC553776/pdf/emboj00251-0050.pdf