Range: Undergraduate Student Journal

Mathematical Modeling of Metabolic Interactions in the Citric Acid Cycle
Isaac Clark (Faculty Mentor: Andre Cherkaev)
Department of Mathematics

Abstract

The human body consists of highly complex and organized biochemical systems. There are countless mechanisms at work that make the body function. In this study, we formulate a system of ordinary differential equations (ODEs) to model the concentration gradients of the various chemicals found in the Citric Acid cycle. The Citric Acid cycle, also known as the Krebs or TCA cycle, is the primary energy creation mechanism in the cell efficiently breaking down food sources into the necessary components for Adenosine Tri-phosphate (ATP) synthesis. The model considers the interactions of each of these chemicals, and the speed of the reactions, necessary to maintain a stable equilibrium. We assume a semi-closed system contained in the inner membrane of the mitochondria to simplify the model and reduce the number of variables. Using approximate steady-state concentrations for each chemical while at rest, we optimize the reaction rate constants such that the concentrations remain steady-state. This allows us to observe, mathematically, the aspects of the body that are used to govern the production of ATP in the body and maintain homeostasis.

Introduction

The Citric Acid cycle is a central mechanism in cellular respiration, playing a critical role in energy production by converting biochemical energy from nutrients into usable energy in the form of Adenosine Triphosphate (ATP). This series of enzymatic reactions occurs in the mitochondrial matrix. For a graphical representation of the Citric Acid cycle, see Appendix A.

Despite its complexity, the Citric Acid cycle operates with incredible consistency, maintaining a equilibrium among the concentrations of its intermediates and products. This equilibrium is essential for sustaining life, as disruptions in the cycle can lead to metabolic imbalances and cellular dysfunction. Understanding the underlying dynamics that govern this stability is therefore of significant interest.

In this project, we approach the Citric Acid cycle from a mathematical perspective. By constructing a system of ordinary differential equations (ODEs), which takes great inspiration from Bocharov et al., 2020 and pharmacokinetics, we aim to simulate the concentration gradients of the various chemicals within the cycle under the assumption of a semi-closed mitochondrial environment. That is, we have defined influx and outflux of certain metabolites and none for others, which is not necessarily the case in an applied setting. This is done for simplification as an open system would be too numerically complex and chaotic to be feasible mathematically.

Our objective is to identify reaction rate constants that result in a steady-state system, where the concentrations of the chemicals remain constant over time. This optimization problem not only shows the quantitative relationships necessary for metabolic stability but also highlights the utility of mathematical modeling in understanding complex biological systems. Through this study, we hope to provide a framework that deepens our insight into how the Citric Acid cycle works to maintains homeostasis and supports energy production.

Methods

The core of our system of ODEs to simulate the Citric Acid cycle is based around the kinetic reaction equation which is defined as:

$$\frac{dx_1}{dt} = k_0 x_0 - k_1 x_1 \tag{1}$$

where the change in the value of x_1 over the change in time t is equal to the rate at which x_1 is produced by the reaction $k_0 x_0$, minus the rate at which x_1 is consumed by the reaction $k_1 x_1$. For the system, we will assign each product in the cycle a variable as follows:

$x_1 = $ *Acetyl-CoA*, $x_2 = $ *Citrate*, $x_3 = $ *Isocitrate*, $x_4 = $ *α-ketoglutarate*, $x_5 = $ *Succinyl-CoA*, $x_6 = $ *Succinate*, $x_7 = $ *Fumarate*, $x_8 = $ *Malate*, $x_9 = $ *Oxaloacetate*, $x_{10} = $ *NADH*, $x_{11} = $ *ATP*.

There are also other substances produced and consumed by the system such as CO_2, H_2O, ADP, NAD+, HS-CoA enzymes, and other catalyzing enzymes, however, we will assume that these are in great enough abundance that their concentrations have a negligible effect on the core system itself and we will ignore these for the purposes of our model, establishing a semi-closed system.

The first step in the cycle is the introduction of Acetyl-CoA into the inner membrane of the mitochondria. This is achieved through various means such as pyruvate decarboxylation and beta-oxidation. For the purposes of our model, we simply combine all these methods and assign it a single rate constant variable k_0:

$$\xrightarrow{k_0} Acetyl\text{-}CoA = k_0. \tag{2}$$

Acetyl-CoA is then reacted with Oxaloacetate in the first step of the citric acid cycle to form Citrate and release an HS-CoA enzyme, which we represent as:

$$Acetyl\text{-}CoA + Oxaloacetate \xrightarrow{k_1} Citrate + HS\text{-}CoA$$
$$= k_1 x_1 x_9. \tag{3}$$
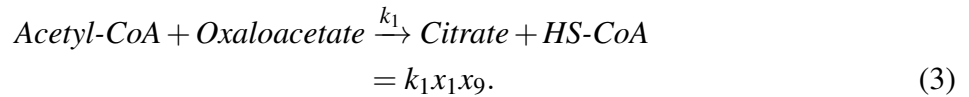
Citrate reacts with a catalyzing enzyme and is reconstituted into Isocitrate and is represented as:

$$Citrate \xrightarrow{k_2} Isocitrate$$
$$= k_2 x_2. \tag{4}$$

Isocitrate is combined with an NAD+ molecule and produces CO_2, NADH, and α-ketoglutarate. NADH production will be accounted for in subsequent modeling steps. We will model this mathematically as:

$$Isocitrate + NAD+ \xrightarrow{k_3} \alpha\text{-}ketoglutarate + NADH + CO_2$$
$$= k_3 x_3. \tag{5}$$
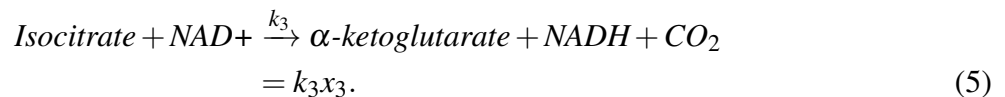
α-ketoglutarate reacts with another NAD+ molecule and HS-CoA enzyme to produce NADH, CO_2, and Succinyl-CoA. Again, NADH production will be accounted for in subsequent steps and model this reaction with:

$$\alpha\text{-ketoglutarate} + HS\text{-}CoA + NAD+ \xrightarrow{k_4} Succinyl\text{-}CoA + NADH + CO_2$$
$$= k_4 x_4. \tag{6}$$
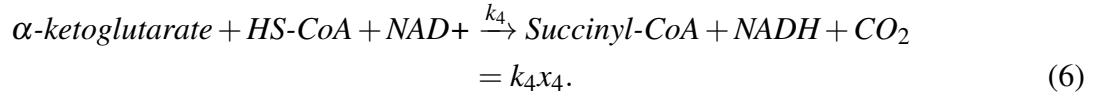
Succinyl-CoA reacts with a GDP molecule and H_2O, releases the HS-CoA enzyme and produces GTP and Succinate. GTP is energetically equivalent so for the purposes of our model, it will be modeled as ATP. Like NADH, we'll account for this in subsequent steps. This process is modeled as:

$$Succinyl\text{-}CoA + H_2O + GDP \xrightarrow{k_5} Succinate + GTP + HS\text{-}CoA$$
$$= k_5 x_5. \tag{7}$$
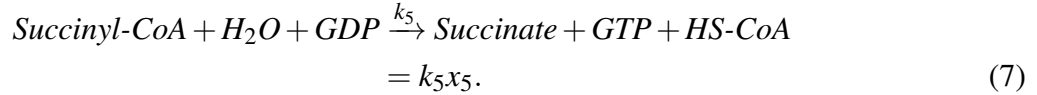
The Succinate then reacts with FAD and produces Fumarate and $FADH_2$. $FADH_2$ is similar enough to NADH that we will group them together and model them under one variable. We'll ignore the $FADH_2$ for now and return to it later with the other NADH molecules. We'll model this with:

$$Succinate + FAD \xrightarrow{k_6} Fumarate + FADH_2$$
$$= k_6 x_6. \tag{8}$$
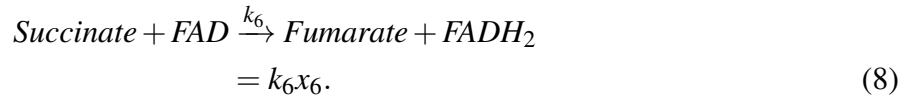
Fumarate reacts with H_2O to become Malate:

$$Fumarate + H_2O \xrightarrow{k_7} Malate$$
$$= k_7 x_7. \tag{9}$$

We finally reach the end of the cycle to where Malate reacts with a NAD+ molecule to produce NADH and Oxaloacetate:

$$Malate + NAD+ \xrightarrow{k_8} Oxaloacetate + NADH$$
$$= k_8 x_8 \tag{10}$$
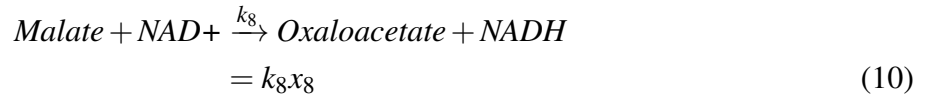
where Oxaloacetate is once again consumed with Acetyl-CoA to start the cycle over again:

$$Oxaloacetate \xrightarrow{k_9}$$
$$= k_9 x_9. \tag{11}$$

Now we can revisit NADH. Because NADH (and its equivalents) are a byproduct of the reactions in the Citric Acid cycle, it's production is already modeled in the above equations, so the concentration of NADH is simply the product of those reactions which is modeled with the ODE 23 in our final system.

ATP is similar, however, it includes a reaction component as NADH is consumed to generate ATP which is modeled as:

$$NADH \xrightarrow{k_{10}} ATP$$
$$= k_{10} x_{10}. \tag{12}$$
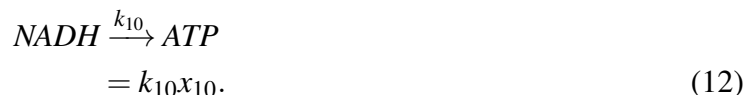
This is a gross oversimplification of oxidative phosphorylation, however, for the purposes of the model and showing the behaviors of the concentration gradients it is "good enough." The consumption and transportation of ATP out of the mitochondria is similarly simplified with:

$$ATP \xrightarrow{k_{11}}$$
$$= k_{11}x_{11}. \tag{13}$$

The rational for this oversimplification being that since the body is so incredibly complex that, under the already existing assumptions such as operating in a closed system, this simplification is moot and the model provides a close enough approximation to provide the deeper understanding of how the Citric Acid cycle works that it primarily sets out to do.

With all variables and equations defined, we can assemble them into a system of ODEs, predicated upon equation 1:

$$\frac{dx_1}{dt} = k_0 - k_1 x_1 x_9 \tag{14}$$

$$\frac{dx_2}{dt} = k_1 x_1 x_9 - k_2 x_2 \tag{15}$$

$$\frac{dx_3}{dt} = k_2 x_2 - k_3 x_3 \tag{16}$$

$$\frac{dx_4}{dt} = k_3 x_3 - k_4 x_4 \tag{17}$$

$$\frac{dx_5}{dt} = k_4 x_4 - k_5 x_5 \tag{18}$$

$$\frac{dx_6}{dt} = k_5 x_5 - k_6 x_6 \tag{19}$$

$$\frac{dx_7}{dt} = k_6 x_6 - k_7 x_7 \tag{20}$$

$$\frac{dx_8}{dt} = k_7 x_7 - k_8 x_8 \tag{21}$$

$$\frac{dx_9}{dt} = k_8 x_8 - k_9 x_9 \tag{22}$$

$$\frac{dx_{10}}{dt} = k_3 x_3 + k_4 x_4 + k_8 x_8 - k_{10} x_{10} \tag{23}$$

$$\frac{dx_{11}}{dt} = k_5 x_5 + 5 k_{10} x_{10} - k_{11} x_{11}. \tag{24}$$

This is the final system of ODEs that will constitute our simulation for the Citric Acid cycle. Since our goal here is to identify the reaction rate constants k_i such that our system maintains equilibrium at steady-state concentrations of the variables x_i, we will define an optimization problem as:

$$\min_{k} \sum_{i=0}^{11} k_i \tag{25}$$

$$\text{s.t.} \sum_{i=1}^{11} \left| \frac{dx_i}{dt} \right| = 0,$$

$$k_i \geq 0.001, \quad \forall i \in \{0, 1, \ldots, 11\}.$$

This ensures we get real, non-zero, values for k_i such that the derivatives of all equations are zero. We assume that the body strives to create the needed amount of ATP with the least exertion. Under this assumption, the optimization problem attempts to estimate the minimum reaction rates k_i so that it maintains constant concentrations of metabolites x_i. Using known and estimated values of metabolite concentrations in the body, solving and simulating the system revolves around this optimization problem.

Results

The model was simulated using Python. Using the following steady-state concentration values shown in table 1.

Variable	Metabolite	Concentration (mM)
x_1	Acetyl-CoA	0.012
x_2	Citrate	0.382
x_3	Isocitrate	0.085
x_4	α-ketoglutarate	0.05
x_5	Succinyl-CoA	0.025
x_6	Succinate	0.368
x_7	Fumarate	0.087
x_8	Malate	0.365
x_9	Oxaloacetate	0.012
x_{10}	NADH	0.5
x_{11}	ATP	3.5

Table 1: Steady-State Metabolite Concentrations (Gibala et al., 1998)

The system of ODEs is optimized using 25 and the kinetic reaction rate constants k_i are estimated as the values in table 2.

Parameter	Value (s^{-1})
k_0	0.0011412
k_1	0.2742360
k_2	0.0034839
k_3	0.0135017
k_4	0.0261262
k_5	0.0441452
k_6	0.0033095
k_7	0.0137874
k_8	0.0034206
k_9	0.0163994
k_{10}	0.0068624
k_{11}	0.0052111

Table 2: Reaction Rate Constant

Running the simulation using the above k_i values and randomly generated initial concentrations x_i between $[0, 1]$, we plot the change in concentrations over time and we get the following graph:
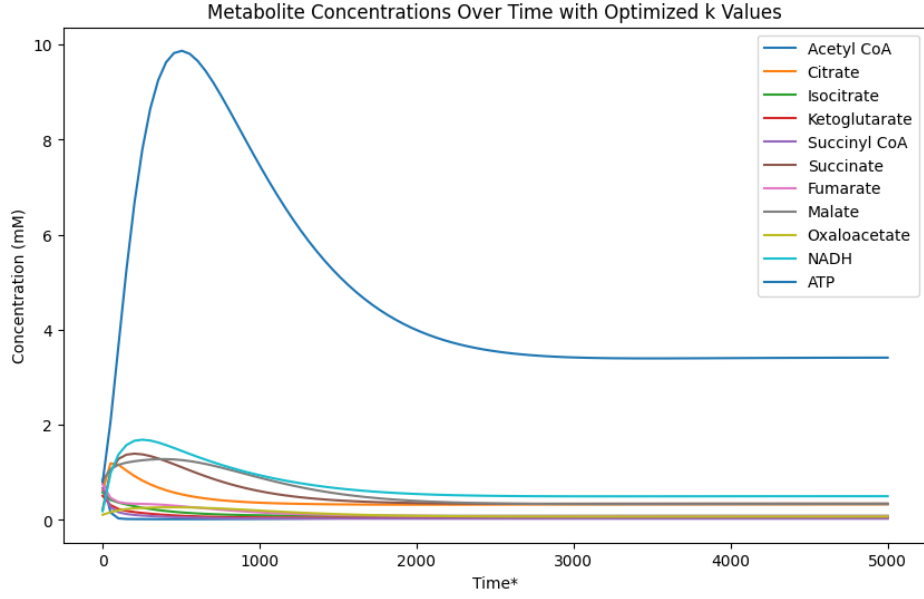

Figure 1: Concentrations

Here we observe a unique property of the system that no matter what concentration values we start with we always return to equilibrium at our steady-state concentrations in Table 1. To show this mathematically, and evaluate the stability of the system, we can compute the Jacobian matrix:

$$\mathbf{J} = \begin{bmatrix}
-k_1 x_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_1 x_1 & 0 & 0 \\
k_1 x_9 & -k_2 & 0 & 0 & 0 & 0 & 0 & 0 & k_1 x_1 & 0 & 0 \\
0 & k_2 & -k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_3 & -k_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & k_4 & -k_5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & k_5 & -k_6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & k_6 & -k_7 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & k_7 & -k_8 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_8 & -k_9 & 0 & 0 \\
0 & 0 & k_3 & k_4 & 0 & 0 & 0 & k_8 & 0 & -k_{10} & 0 \\
0 & 0 & 0 & 0 & k_4 & 0 & 0 & 0 & 0 & 5k_{10} & -k_{11}
\end{bmatrix} \tag{26}$$

If we evaluate the Jacobian with the values in tables 1 and 2, and compute the eigenvalues, we get the following:

Symbol	Eigenvalue
λ_1	$-0.0441 + 5.4464 \times 10^{-34}i$
λ_2	$-0.0260 + 8.7044 \times 10^{-36}i$
λ_3	$-0.0185 - 2.0647 \times 10^{-34}i$
λ_4	$-0.0134 - 0.0038i$
λ_5	$-0.0134 + 0.0038i$
λ_6	$-0.0045 - 0.0033i$
λ_7	$-0.0015 - 0.0011i$
λ_8	$-0.0015 + 0.0011i$
λ_9	$-0.0045 + 0.0033i$
λ_{10}	-0.0069
λ_{11}	-0.0052

Table 3: Jacobian Eigenvalues

Because the real part of all the eigenvalues are negative, this shows that the system of ODEs is asymptotically stable and will always converge to an equilibrium point with the optimized k_i values.

Conclusion

Our results show that by optimizing reaction rate constants, the Citric Acid Cycle can be modeled as an asymptotically stable system, consistently returning to steady-state concentrations despite initial disturbances. This confirms the inherent optimization and stability of cellular metabolic regulation, even under simplified assumptions.

The steady-state optimization reflects the body's biological preference for metabolic efficiency, suggesting that the availability and regulation of enzymatic catalysts may have evolved to achieve minimal energetic cost. This model may help explain how cells dynamically regulate energy production in response to demand.

This model necessarily simplifies a highly complex biochemical system. Factors such as feedback inhibition, transport of metabolites across mitochondrial membranes, and the role of oxygen and electron transport were not considered. These elements, while critical in reality, were excluded to focus on the cycle's core dynamics and ensure feasibility.

Future work should be done, as more complex and realistic model could provide greater insight into how our bodies functions. With applications to general biology, medicine, kinesiology, sports science, and various other fields that study energy systems, this is an important area of study with unrealized benefits to science as a whole.

References

Alberts, B., Hopkin, K., Johnson, A., Morgan, D., Raff, M., Roberts, K., & Walter, P. (2019). *Essential cell biology* (5th). W. W. Norton & Company.

Bocharov, G., Jaeger, W., Knoch, J., Neuss-Radu, M., & Thiel, M. (2020). A mathematical model of hif-1 regulated cellular energy metabolism. *Vietnam Journal of Mathematics*, *49*(1), 119–141. https://doi.org/10.1007/s10013-020-00426-y

Gibala, M. J., MacLean, D. A., Graham, T. E., & Saltin, B. (1998). Tricarboxylic acid cycle intermediate pool size and estimated cycle flux in human muscle during exercise. *AJP Endocrinology and Metabolism*, *275*(2), E235–E242. https://doi.org/10.1152/ajpendo.1998.275.2.e235

**ChatGPT was used as a tool to aid in the ideation and drafting of this paper. All content and ideas are the original creations of the author.

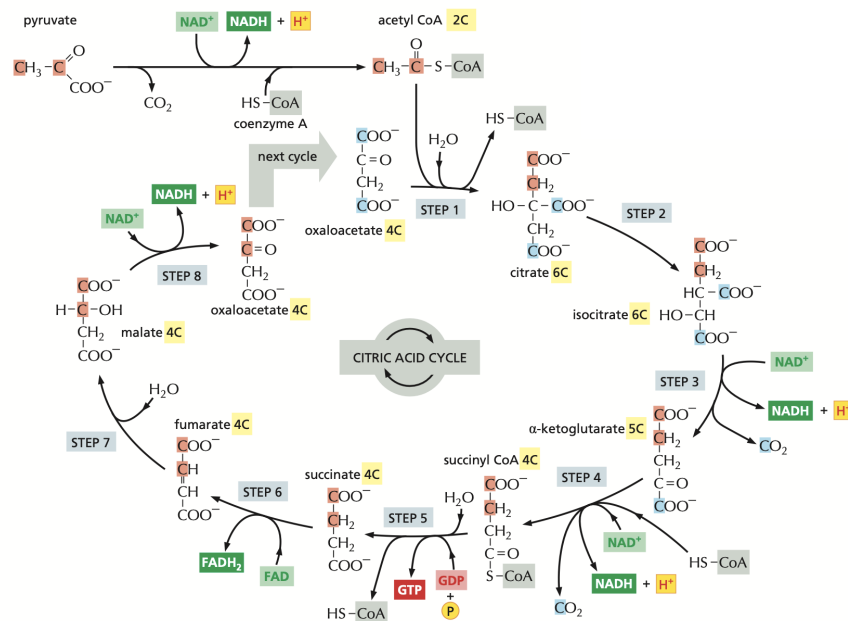## Appendix A: Citric Acid Cycle Diagram

Figure 2: The Citric Acid cycle (Alberts et al., 2019)

Appendix B: Python Code

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

```python
from scipy.integrate import solve_ivp
from scipy.optimize import minimize
import sympy as sp
import warnings

warnings.filterwarnings('ignore')

# System of ODEs
def TCA_cycle_model(t, y, k):
    Acetyl_CoA, Citrate, Isocitrate, Ketoglutarate, Succinyl_CoA,
        Succinate, Fumarate, Malate, Oxaloacetate, NADH, ATP = y

    dAcetyl_CoA = (k[0]) - (k[1] * Acetyl_CoA * Oxaloacetate)
    dCitrate = (k[1] * Acetyl_CoA * Oxaloacetate) - (k[2] * Citrate)
    dIsocitrate = (k[2] * Citrate) - (k[3] * Isocitrate)
    dKetoglutarate = (k[3] * Isocitrate) - (k[4] * Ketoglutarate)
    dSuccinyl_CoA = (k[4] * Ketoglutarate) - (k[5] * Succinyl_CoA)
    dSuccinate = (k[5] * Succinyl_CoA) - (k[6] * Succinate)
    dFumarate = (k[6] * Succinate) - (k[7] * Fumarate)
    dMalate = (k[7] * Fumarate) - (k[8] * Malate)
    dOxaloacetate = (k[8] * Malate) - (k[9] * Oxaloacetate)
    dNADH = (k[3] * Isocitrate) + (k[4] * Ketoglutarate) + (k[8] *
        Malate) - (k[10] * NADH)
    dATP = (k[4] * Succinyl_CoA) + (5 * k[10] * NADH) - (k[11] * ATP)

    return [dAcetyl_CoA, dCitrate, dIsocitrate, dKetoglutarate,
        dSuccinyl_CoA, dSuccinate, dFumarate, dMalate, dOxaloacetate,
        dNADH, dATP]

# Rate constant optimization error function
def steady_state_error(k, y0):
    sol = solve_ivp(lambda t, y: TCA_cycle_model(t, y, k), (0, 100),
        y0, t_eval=[100])
    final_values = sol.y[:, -1]
    derivatives = TCA_cycle_model(0, final_values, k)
    return np.sum(np.abs(derivatives))

# Rate constant optimization function
def optimize_k(y0):
    initial_k = np.ones(12)
    bounds = [(0.001, 10) for _ in range(12)]
    result = minimize(lambda k: np.sum(k), initial_k, bounds=bounds,
        constraints={'type': 'eq', 'fun': lambda k: steady_state_error(
        k, y0)})
    return result.x

# Given initial conditions
constants = [0.012, 0.362, 0.085, 0.05, 0.025, 0.368, 0.087, 0.365,
    0.012, 0.5, 3.5]
optimal_k = optimize_k(constants)
print('Optimized k:', optimal_k)

# Simulation space
t_span = (0, 5000)
t_eval = np.linspace(*t_span, 100)
```

```python
# Random initial values
y0 = np.random.rand(11)

# Solution to the system
solution = solve_ivp(lambda t, y: TCA_cycle_model(t, y, optimal_k),
    t_span, y0, t_eval=t_eval)

# Plot the results
plt.figure(figsize=(10, 6))
metabolites = ['Acetyl-CoA', 'Citrate', 'Isocitrate', 'Ketoglutarate',
    'Succinyl-CoA', 'Succinate', 'Fumarate', 'Malate', 'Oxaloacetate',
    'NADH', 'ATP']
for i, label in enumerate(metabolites):
    plt.plot(solution.t, solution.y[i], label=label)

plt.xlabel('Time*')
plt.ylabel('Concentration (mM)')
plt.title('Metabolite Concentrations Over Time with Optimized k Values
    ')
plt.legend()
plt.show()

# Variables
x = sp.symbols('x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11')
k = sp.symbols('k0:12')

# System of ODEs
dxdt = [
    k[0] - k[1]*x[0]*x[8],
    k[1]*x[0]*x[8] - k[2]*x[1],
    k[2]*x[1] - k[3]*x[2],
    k[3]*x[2] - k[4]*x[3],
    k[4]*x[3] - k[5]*x[4],
    k[5]*x[4] - k[6]*x[5],
    k[6]*x[5] - k[7]*x[6],
    k[7]*x[6] - k[8]*x[7],
    k[8]*x[7] - k[9]*x[8],
    k[3]*x[2] + k[4]*x[3] + k[8]*x[7] - k[10]*x[9],
    k[4]*x[4] + 5*k[10]*x[9] - k[11]*x[10]
]

J = sp.Matrix(dxdt).jacobian(x)

steady_state_values = {
x[0]: 0.012, x[1]: 0.362, x[2]: 0.085, x[3]: 0.05, x[4]: 0.025, x[5]:
    0.368,
x[6]: 0.087, x[7]: 0.365, x[8]: 0.012, x[9]: 0.5, x[10]: 3.5
}

parameter_values = {
    k[0]: 0.0011412, k[1]: 0.2742360, k[2]: 0.0034839, k[3]:
        0.0135017, k[4]: 0.0261262, k[5]: 0.0441452,
    k[6]: 0.0033095, k[7]: 0.0137874, k[8]: 0.0034206, k[9]:
        0.0163994, k[10]: 0.0068624, k[11]: 0.0052111
}
```

```python
# Evaluate the Jacobian
J_evaluated = J.subs(steady_state_values).subs(parameter_values)

# Compute eigenvalues
eigenvalues = J_evaluated.eigenvals()

# Display results
for i, (eig, multiplicity) in enumerate(eigenvalues.items(), start=1):
    print(f'Eigenvalue {i}: {eig}')
```