Contents

Abstract

We present a computational framework for analyzing and simulating mitochondrial ATP synthesis using basic thermodynamic and kinetic principles. The framework invokes detailed descriptions of the thermodynamic driving forces associated with the processes of the electron transport chain, mitochondrial ATP synthetase, and mitochondrial phosphate and adenine nucleotide transporters. Assembling models of these discrete processes into an integrated model of mitochondrial ATP synthesis, we illustrate how to analyze and simulate in vitro respirometry experiments and how models identified from in vitro experimental data effectively explain cardiac respiratory control in vivo. Computer codes for these analyses are embedded as Python scripts in a Jupyter Book to facilitate easy adoption and modification of the concepts developed here.

Abbreviations

Description	Abbreviation
Adenosine triphosphate	ATP
Adenosine diphosphate	ADP
Deoxyribonucleic acid	DNA
Flavin adenine dinucleotide (oxidized)	FAD
Flavin adenine dinucleotide (reduced)	$FADH_2$
Inner mitochondrial membrane	IMM
Inorganic phosphate	Pi
Intermembrane space	IMS
Nicotinamide adenine dinucleotide (oxidized)	NAD^+
Nicotinamide adenine dinucleotide (reduced)	NADH
Oxygen consumption rate	OCR
Total adenine nucleotide	TAN
Total enchangeable phosphate	TEP
Tricarboxylic acid	TCA

Introduction

Acquired in the initiating event in the origin of eukaryotes, mitochondria are a defining feature of eukaroytic organisms [13]. They serve as a central hub for eukaryotic energy metabolism, oxidizing carbohydrates, fats, and ketones, producing the reducing equivalents to drive the proton pumps of the respiratory complexes, and synthesizing the majority of adenosine triphosphate (ATP) needed to drive essential cell functions in high metabolic demand tissues (e.g., heart, brain, kidney). Since oxygen (O_2) is the final electron acceptor of the respiratory chain, mitochondrial respiration is the primary reason why vertebrate animals need a heart and vascular system to deliver oxygen to within diffusional distances of nearly every cell in the body. In other words, mitochondria are, in large part, the reason why the mammalian heart must beat continuously throughout the life of the organism. Mitochondria are also the source of the ATP hydrolysis potential that drives the mechanical pumping of the heart. Thus, at a basic level, mitochondria are both the ends and the means of cardiovascular physiology.

In oxidizing primary substrates and synthesizing ATP, mitochondria transduce free energy through a multi-step process: (i) reduced cofactors nicotinatmide adenine dinucleotide (NADH) and flavin adenine dinucleotide ($FADH_2$) are synthesized via the oxidation of fuels such as pyruvate and fatty acids; (ii) oxidation of NADH and $FADH_2$ drives the proton pumps of

the respiratory chain, generating an electrostatic potential across the inner mitochondrial membrane (IMM); and (iii) that electrostatic potential, in turn, drives the phosphorylation of adenosine diphosphate (ADP) to ATP and the electrogenic exchange of a mitochondrial matrix ATP^{4-} for a cytosolic ADP^{3-}. Thus, substrate oxidation drives the transduction of free energy of chemical reactions to free energy in an electrochemical transport gradient, and the electrochemical grandient is transduced back to chemical reaction free energy in the form of the ATP hydrolysis potential. The idea that chemical synthesis of ATP is thermodynamically coupled to the movement of ions across the electrochemical gradient of the mitochondrial membrane was introduced by Peter Mitchell, for which he was awarded the Nobel Prize in 1978 [12].

We aim to develop a computational framework to help understand how these electrochemical processes operate. Kinetic models of mitochondrial ATP synthesis date back to Chance and Williams [16], who observed a Michaelis-Menten type dependence of ATP synthesis on ADP concentration, yielding a phenomenological approach that is able to effectively describe mitochondrial ATP synthesis in human skeletal muscle in vivo [14]. While our approach is to analyze the component processes of oxidative phosphorylation with more mechanistic and thermodynamic fidelity, ultimately our models are in agreement with the overall dependence of ATP synthesis on the concentration of the products of ATP hydrolysis, namely ADP and inorganic phosphate (Pi). Ultimately, we follow the approach of Chance et al. [14] of first using in vitro data to identify a mitochondrial model, and then using the identified model to predict and analyze in vivo energetics.

Basic chemical, electrical, and thermodynamic principles

To develop a quantitative understanding of how these processes work, we start with a set of definitions of the basic quantities and concepts with which we are concerned.

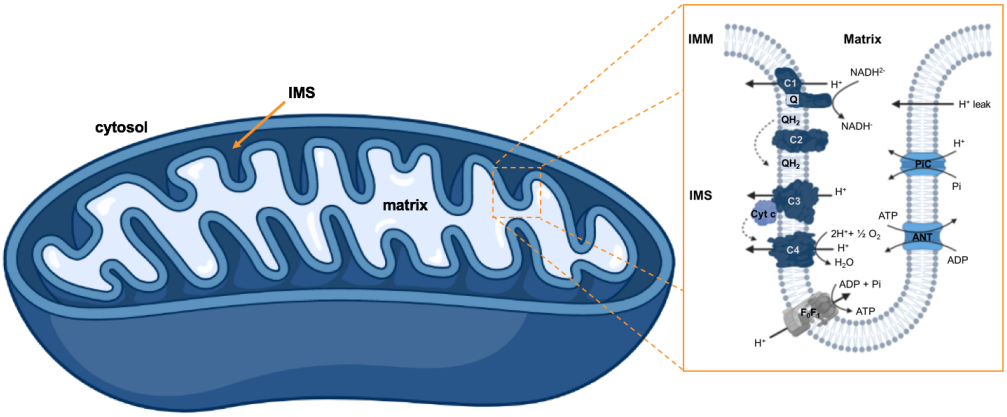


Fig. 1 Diagram of a mitochondrion with the cytosol, intermembrane space (IMS), and matrix indicated. *Inset from left to right:* Protein channels and complexes associated with oxidative phosphorylation in the cristae of the mitochondrion. Complex I (C1) catalyzes the oxidation of $NADH^{2-}$ to NAD^{-} and reduction of ubiquinone (Q) to QH_2. Complex II (C2) catalyzes the oxidation of $FADH_2$ to FAD coupled to the reduction of Q. Complex III (C3) catalyzes the oxidation of QH_2 coupled to the reduction of cytochrome c (Cyt c). Complex IV (C4) catalyzes the oxidation of Cyt c coupled to the reduction of oxygen to water. These redox transfers drive pumping of H^+ ions out of the matrix, establishing the proton motive force across the inner mitochondrial membrane (IMM) that drives ATP synthesis at complex V, or the F_0F_1-ATPase (F_0F_1). The adenine nucleotide translocase (ANT) exchanges matrix ATP for IMS ADP. The inorganic phosphate cotransporter (PiC) brings protons and Pi from the IMS to the matrix. Lastly, there is a passive H^+ leak across the IMM. (Figure created with Biorender.com.)

Mitochondrial anatomy

The mitochondrion is a membrane-bound, rod-shaped organelle that is responsible for generating most of the chemical energy needed to power the cell's biochemical reactions by respiration [11]. Mitochondria are comprised of an outer and inner membrane that are separated by the intermembrane space (IMS) (Fig. 1). The outer mitochondrial membrane is freely permeable to small molecules and ions. The IMM folds inward to make cristae that extend into the matrix. Transmembrane channels called porins and the respiratory complexes involved in oxidative phosphorylation and ATP

synthesis allow for more selective IMM permeability. The IMM encloses the mitochondrial matrix, which contains mitochondrial deoxyribonucleic acid (DNA), the majority of mitochondrial proteins, soluble metabolic intermediates including ATP, ADP, and Pi, and the enzymes catalyzing the tricarboxylic acid (TCA) cycle and β-oxidation.

IMM capacitance

The IMM acts as an electrical capacitor to store energy in an electrostatic potential difference between the milieu on each side. Electrical capacitance of a membrane (C_m) is the proportionality between the rate of charge transport across the membrane, i.e. current (I), to the rate of membrane potential ($\Delta\Psi$) change, that is,

$$C_m \frac{\mathrm{d}\Delta\Psi}{\mathrm{d}t} = I.$$

In the model and associated calculations presented below, we express fluxes in units of moles per unit time per unit volume of mitochondria. Thus, it is convenient to obtain an estimate of C_m in units of mole per volt per volume of mitochondria. Approximating a mitochondrion as a sphere with radius $r = 1\ \mu\mathrm{m}$, we obtain a surface area-to-volume ratio of $3\ \mu\mathrm{m}^{-1}$. Furthermore, we estimate that the IMM has ten-fold greater surface area than the outer membrane, yielding a surface area to volume ratio of $30\ \mu\mathrm{m}^{-1}$ for the IMM. Since the capacitance density of biological membranes ranges from $0.5\text{-}1.0 \mu\mathrm{F\ cm}^{-2}$, or $0.5\text{-}1.0 \times\ 10^{-8}\ \mu\mathrm{F}\ \mu\mathrm{m}^{-2}$ [11], C_m is approximately $3 \times 10^{-8}\ \mu\mathrm{F}\ \mu\mathrm{m}^{-3} = 300\ \mathrm{F\ (L\ mito)}^{-1}$. To convert to the units used in the calculations below, we have

$$C_m = 300\ \frac{\mathrm{F}}{\mathrm{L\ mito}} = 300\ \frac{\mathrm{C}}{\mathrm{V \cdot L\ mito}} \cdot \frac{1}{F}\ \frac{\mathrm{mol}}{\mathrm{C}} = 3.1 \times 10^{-3}\ \frac{\mathrm{mol}}{\mathrm{V \cdot L\ mito}},$$

where $F = 96,485\ \mathrm{C\ mol}^{-1}$ is Faraday's constant.

Gibbs free energy

A *free energy* is a thermodynamic quantity that relates a change in the thermodynamic state of a system to an associated change in total entropy of the system plus its environment. Chemical reaction processes necessarily proceed in the direction associated with a reduction in free energy [11]. When free energy of a system is reduced, total entropy (of the universe) is increased. The form of free energy that is operative in constant-temperature and constant-pressure systems (most relevant for biochemistry) is the Gibbs free energy, or simply the *Gibbs energy*.

For a chemical reaction of reactants A_i and products B_j,

$$\sum_{i=1}^{M} m_i A_i \rightleftharpoons \sum_{j=1}^{N} n_j B_j$$

where M and N are the total number of reactants and products, respectively, and m_i and n_j are the coefficients of reactant i and product j, respectively, the Gibbs energy can be expressed as

$$\Delta_r G = \Delta_r G^\circ + R\mathrm{T} \ln\left(\frac{\prod_{i=1}^{N} [\mathrm{B}_j]^{n_i}}{\prod_{i=1}^{M} [\mathrm{A}_i]^{m_i}} \right), \tag{1}$$

where $\Delta_r G^\circ$ is the reference Gibbs energy for the reaction (a constant at given constant chemical conditions of temperature, pressure, ionic conditions, etc.), $R = 8.314\ \mathrm{J\ mol}^{-1}\ \mathrm{K}^{-1}$ is the gas constant, and $\mathrm{T} = 310.15\ \mathrm{K}$ is the temperature. The second term on the right hand side of Equation (1) governs how changes in concentrations of species affects $\Delta_r G$.

A system is in chemical equilibrium when there is no thermodynamic driving force, that is, $\Delta_r G = 0$. Thus, for this chemical reaction the reference Gibbs energy is related to the equilibrium constant as

$$K_{eq} = \left(\frac{\prod_{i=1}^{N} [\mathrm{B}_j]^{n_i}}{\prod_{i=1}^{M} [\mathrm{A}_i]^{m_i}} \right)_{eq} = \exp\left\{ -\frac{\Delta_r G^\circ}{R\mathrm{T}} \right\}. \tag{2}$$

Membrane potential and proton motive force

Free energy associated with the oxidation of primary fuels is transduced to generate the chemical potential across the IMM known as the *proton motive force*, which is used to synthesize ATP in the matrix and transport ATP out of the matrix to the cytosol [11]. The thermodynamic driving force for translocation of hydrogen ions (H^+) across the IMM has two components: the difference in electrostatic potential across the membrane, $\Delta\Psi$ (V), and the difference in H^+ concentration (or activity) between the media on either side of the membrane, $\Delta\mathrm{pH}$, that is

$$\begin{aligned}\Delta G_{\mathrm{H}} &= -F\Delta\Psi + R\mathrm{T}\ln\left([\mathrm{H}^+]_x/[\mathrm{H}^+]_c\right)\\ &= -F\Delta\Psi - 2.3R\mathrm{T}\,\Delta\mathrm{pH},\end{aligned} \tag{3}$$

where the subscripts x and c indicate matrix and external (cytosol) spaces. $\Delta\Psi$ is defined as the cytosolic potential minus matrix potential, yielding a negative change in free energy for a positive potential. Membrane potential in respiring mitochondria is approximately $150\text{-}200\ \mathrm{mV}$, yielding a contribution to ΔG_{H} on the order of $15\text{-}20\ \mathrm{kJ\ mol^{-1}}$ [15]. Under in vitro conditions, $\Delta\mathrm{pH}$ between the matrix and external buffer is on the order of $0.1\ \mathrm{pH}$ units [15]. Thus, the contribution to proton motive force from a pH difference is less than $1\ \mathrm{kJ\ mol^{-1}}$ and substantially smaller than that from $\Delta\Psi$.

Thermodynamics of ATP synthesis/hydrolysis

Under physiological conditions the ATP hydrolysis reaction

$$\mathrm{ATP}^{4-} + \mathrm{H_2O} \rightleftharpoons \mathrm{ADP}^{3-} + \mathrm{HPO_4^{2-}} + \mathrm{H}^+ \tag{4}$$

is thermodynamically favored to proceed from the left-to-right direction. The Gibbs energy associated with turnover of this reaction is

$$\Delta_r G_{\mathrm{ATP}} = \Delta_r G^o_{\mathrm{ATP}} + R\mathrm{T}\ln\left(\frac{[\mathrm{ADP}^{3-}][\mathrm{HPO_4^{2-}}][\mathrm{H}^+]}{[\mathrm{ATP}^{4-}]}\right), \tag{5}$$

where the Gibbs energy for ATP hydrolysis under physiological conditions is approximately $\Delta_r G^o_{\mathrm{ATP}} = 4.99\ \mathrm{kJ\ mol^{-1}}$ [7].

Calculation of the ATP hydrolysis potential

Equation (5) expresses the Gibbs energy of chemical Equation (4) in terms of its *chemical species*. In practice, biochemistry typically deals with biochemical *reactants*, which are comprised of sums of rapidly interconverting chemical species. We calculate the total ATP concentration, $[\Sigma\mathrm{ATP}]$, in terms of its bound and unbound species, that is,

$$\begin{aligned}[\Sigma\mathrm{ATP}] &= [\mathrm{ATP}^{4-}] + [\mathrm{MgATP}^{2-}] + [\mathrm{HATP}^{3-}] + [\mathrm{KATP}^{3-}]\\ &= [\mathrm{ATP}^{4-}] + \frac{[\mathrm{Mg}^{2+}][\mathrm{ATP}^{4-}]}{K_{\mathrm{MgATP}}} + \frac{[\mathrm{H}^+][\mathrm{ATP}^{4-}]}{K_{\mathrm{HATP}}} + \frac{[\mathrm{K}^+][\mathrm{ATP}^{4-}]}{K_{\mathrm{KATP}}}\\ &= [\mathrm{ATP}^{4-}]\left(1 + \frac{[\mathrm{Mg}^{2+}]}{K_{\mathrm{MgATP}}} + \frac{[\mathrm{H}^+]}{K_{\mathrm{HATP}}} + \frac{[\mathrm{K}^+]}{K_{\mathrm{KATP}}}\right)\\ &= [\mathrm{ATP}^{4-}]P_{\mathrm{ATP}},\end{aligned} \tag{6}$$

where P_{ATP} is a *binding polynomial*. Here, we we account for only the single cation-bound species. (Free H^+ in solution associates with water to form $\mathrm{H_3O^+}$. Here we use $[\mathrm{H}^+]$ to indicate hydrogen ion activity, which is equal to $10^{-\mathrm{pH}}$.) Table 1 lists the dissociation constants used in this study from [7]. Similarly, total ADP, $[\Sigma\mathrm{ADP}]$, and inorganic phosphate, $[\Sigma\mathrm{Pi}]$, concentrations are

$$\begin{aligned}[\Sigma\mathrm{ADP}] &= [\mathrm{ADP}^{3-}]\left(1 + \frac{[\mathrm{Mg}^{2+}]}{K_{\mathrm{MgADP}}} + \frac{[\mathrm{H}^+]}{K_{\mathrm{HADP}}} + \frac{[\mathrm{K}^+]}{K_{\mathrm{KADP}}}\right)\\ &= [\mathrm{ADP}^{3-}]P_{\mathrm{ADP}}\end{aligned} \tag{7}$$

and

$$\begin{aligned}[\Sigma\mathrm{Pi}] &= [\mathrm{HPO_4^{2-}}]\left(1 + \frac{[\mathrm{Mg}^{2+}]}{K_{\mathrm{MgPi}}} + \frac{[\mathrm{H}^+]}{K_{\mathrm{HPi}}} + \frac{[\mathrm{K}^+]}{K_{\mathrm{KPi}}}\right)\\ &= [\mathrm{HPO_4^{2-}}]P_{\mathrm{Pi}},\end{aligned} \tag{8}$$

for binding polynomials P_{ADP} and P_{Pi}.

Expressing the Gibbs energy of ATP hydrolysis in Equation (4) in terms of biochemical reactant concentrations, we obtain

(9)

$$
\begin{aligned}
\Delta_r G_{\mathrm{ATP}} &= \Delta_r G^o_{\mathrm{ATP}} + RT\ln\left(\frac{[\Sigma\mathrm{ADP}][\Sigma\mathrm{Pi}]}{[\Sigma\mathrm{ATP}]} \cdot \frac{[\mathrm{H}^+]P_{\mathrm{ATP}}}{P_{\mathrm{ADP}}P_{\mathrm{Pi}}}\right) \\
&= \Delta_r G^o_{\mathrm{ATP}} + RT\ln\left(\frac{[\mathrm{H}^+]P_{\mathrm{ATP}}}{P_{\mathrm{ADP}}P_{\mathrm{Pi}}}\right) + RT\ln\left(\frac{[\Sigma\mathrm{ADP}][\Sigma\mathrm{Pi}]}{[\Sigma\mathrm{ATP}]}\right) \\
&= \Delta_r G'^o_{\mathrm{ATP}} + RT\ln\left(\frac{[\Sigma\mathrm{ADP}][\Sigma\mathrm{Pi}]}{[\Sigma\mathrm{ATP}]}\right)
\end{aligned}
$$

where $\Delta_r G'^o_{\mathrm{ATP}}$ is a transformed, or *apparent*, reference Gibbs energy for the reaction.

	Ligand (L)		
	Mg^{2+}	**H^+**	**K^+**
$K_{L-\mathrm{ATP}}$	$10^{-3.88}$	$10^{-6.33}$	$10^{-1.02}$
$K_{L-\mathrm{ADP}}$	$10^{-3.00}$	$10^{-6.26}$	$10^{-0.89}$
$K_{L-\mathrm{Pi}}$	$10^{-1.66}$	$10^{-6.62}$	$10^{-0.42}$

Table 1 Dissociation constants given as $10^{-\mathrm{p}K_a}$.

The following code computes the apparent Gibbs energy with $\mathrm{pH} = 7$, $[\mathrm{K}^+] = 150$ mM, and $[\mathrm{Mg}^{2+}] = 1$ mM. Biochemical reactant concentrations are set such that the total adenine nucleotide (TAN) pool inside the mitochondrion is 10 mM, $[\Sigma\mathrm{ATP}] = 0.5$ mM, $[\Sigma\mathrm{ADP}] = 9.5$ mM, and $[\Sigma\mathrm{Pi}] = 1$ mM. Here, we obtain a value of approximately -45 kJ mol^{-1}.

```
# Import numpy package for calculations
import numpy as np

# Dissociation constants
K_MgATP = 10**(-3.88)
K_MgADP = 10**(-3.00)
K_MgPi  = 10**(-1.66)
K_HATP  = 10**(-6.33)
K_HADP  = 10**(-6.26)
K_HPi   = 10**(-6.62)
K_KATP  = 10**(-1.02)
K_KADP  = 10**(-0.89)
K_KPi   = 10**(-0.42)

#  Gibbs energy under physiological conditions(J mol^(-1))
DrGo_ATP = 4990

# Thermochemical constants
R = 8.314           # J (mol * K)**(-1)
T = 310.15          # K
F = 96485           # C mol**(-1)

# Environment concentrations
pH = 7
H  = 10**(-pH)      # Molar
K  = 150e-3         # Molar
Mg = 1e-3           # Molar

# Binding polynomials
P_ATP = 1 + H/K_HATP + K/K_KATP + Mg/K_MgATP # equation 6
P_ADP = 1 + H/K_HADP + K/K_KADP + Mg/K_MgADP # equation 7
P_Pi  = 1 + H/K_HPi  + K/K_KPi  + Mg/K_MgPi  # equation 8

# Total concentrations
sumATP = 0.5e-3         # Molar
sumADP = 9.5e-3         # Molar
sumPi  = 1.0e-3         # Molar

# Reaction:
# ATP4- + H2O ⇌ ADP3- + HPO2-4 + H+

# Use equation 9 to calcuate apparent Gibbs energy
DrG_ATP_apparent = DrGo_ATP + R * T * np.log(H * P_ATP / (P_ADP * P_Pi))

# Use equation 9 to calculate actual Gibbs energy
DrG_ATP = DrG_ATP_apparent + R * T * np.log((sumADP * sumPi / sumATP))

print('Gibbs energy of ATP hydrolysis (kJ mol^(-1))')
print(DrG_ATP / 1000)
```

```
Gibbs energy of ATP hydrolysis (kJ mol^(-1))
-45.47130666167594
```

The reactant concentrations used in the above calculation represent reasonable values for concentrations in the mitochondrial matrix. In the cytosol, the ATP/ADP ratio is on the order of 100:1, yielding a $\Delta_r G_{\mathrm{ATP}}$ of approximately -64 kJ mol^{-1}.

ATP synthesis in the mitochondrial matrix

The $\mathrm{F_0F_1}$ ATP synthase catalyzes the synthesis of ATP from ADP and Pi by coupling to the translocation of $n_{\mathrm{F}} = 8/3$ protons from the cytosol to the matrix via the combined reaction

$$(\mathrm{ADP}^{3-})_x + (\mathrm{HPO_4}^{2-})_x + (\mathrm{H}^+)_x + n_{\mathrm{F}}(\mathrm{H}^+)_c \rightleftharpoons (\mathrm{ATP})_x^{4-} + \mathrm{H_2O} + n_{\mathrm{F}}(\mathrm{H}^+)_x \,. \tag{10}$$

Using the Gibbs energy of the reaction of Equation (9) and the proton motive force in Equation (3), the overall Gibbs energy for the coupled process of ATP synthesis and proton transport via the $\mathrm{F_0F_1}$ ATP synthase is

$$\begin{aligned} \Delta G_{\mathrm{F}} =& -\Delta_r G_{\mathrm{ATP}} + n_{\mathrm{F}} \Delta G_{\mathrm{H}} \\ =& -\Delta_r G'^{o}_{\mathrm{ATP}} - R\mathrm{T} \ln\left(\frac{[\Sigma\mathrm{ADP}]_x [\Sigma\mathrm{Pi}]_x}{[\Sigma\mathrm{ATP}]_x} \right) - n_{\mathrm{F}} F \Delta\Psi + R\mathrm{T} \ln\left(\frac{[\mathrm{H}^+]_x}{[\mathrm{H}^+]_c} \right)^{n_{\mathrm{F}}}. \end{aligned} \tag{11}$$

Note that the negative before $\Delta_r G_{\mathrm{ATP}}$ indicates that the reaction of Equation (4) is reversed in Equation (10). The equilibrium concentration ratio occurs when $\Delta G_{\mathrm{F}} = 0$. Solving for the second term in Equation (11), we calculate the apparent equilibrium constant for ATP synthesis as

$$K'_{eq,\mathrm{F}} = \left(\frac{[\Sigma\mathrm{ATP}]_x}{[\Sigma\mathrm{ADP}]_x [\Sigma\mathrm{Pi}]_x} \right)_{eq} = \exp\left\{ \frac{\Delta_r G'^{o}_{\mathrm{ATP}} + n_{\mathrm{F}} F \Delta\Psi}{R\mathrm{T}} \right\} \left(\frac{[\mathrm{H}^+]_c}{[\mathrm{H}^+]_x} \right)^{n_{\mathrm{F}}}.$$

Mathematical modeling ATP synthesis

A simple model of ATP synthesis kinetics can be constructed using the apparent equilibrium constant and mass-action kinetics in the form

$$J_{\mathrm{F}} = X_{\mathrm{F}} (K'_{eq,\mathrm{F}} [\Sigma\mathrm{ADP}]_x [\Sigma\mathrm{Pi}]_x - [\Sigma\mathrm{ATP}]_x),$$

where $X_{\mathrm{F}} = 1000$ mol s^{-1} (L mito)$^{-1}$ is a rate constant set to an arbitrarily high value that maintains the reaction in equilibrium in model simulations. To simulate ATP synthesis at a given membrane potential, matrix pH, cytosolic pH, and cation concentrations, we have

$$\begin{cases} \dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_x}{\mathrm{d}t} &= J_{\mathrm{F}}/W_x \\ \dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_x}{\mathrm{d}t} &= -J_{\mathrm{F}}/W_x \\ \dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_x}{\mathrm{d}t} &= -J_{\mathrm{F}}/W_x, \end{cases} \tag{12}$$

where W_x ((L matrix water) (L mito)$^{-1}$) is the fraction of water volume in the mitochondrial matrix to total volume of the mitochondrion. Dissociation constants are listed in Table 1 and all other parameters are listed in Table 2.

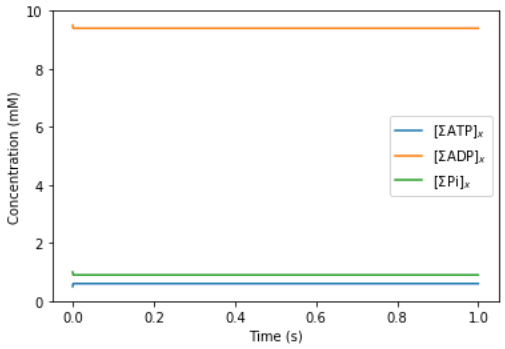
Symbol	Units	Description	Value	Source
F_0F_1 ATP synthase constants				
n_{F}		Protons translocated	8/3	[11]
X_{F}	mol s^{-1} (L mito)$^{-1}$	Rate constant	1000	
$\Delta_r G^{\circ}_{\mathrm{ATP}}$	kJ mol^{-1}	Reference Gibbs energy	4.99	[7]
Biophysical constants				
R	J mol^{-1} K^{-1}	Gas constant	8.314	
T	K	Temperature	310.15	
F	C mol^{-1}	Faraday's constant	96485	
C_m	mol V^{-1} (L mito)$^{-1}$	IMM capacitance	3.1e-3	[1]
Volume ratios				
V_c	(L cyto) (L cell)$^{-1}$	Cyto to cell ratio	0.6601	[15]
V_m	(L mito) (L cell)$^{-1}$	Mito to cell ratio	0.2882	[15]
V_{m2c}	(L mito) (L cyto)$^{-1}$	Mito to cyto ratio	V_m/V_c	
W_c	(L cyto water) (L cyto)$^{-1}$	Cyto water space ratio	0.8425	[15]
W_m	(L mito water) (L mito)$^{-1}$	Mito water space ratio	0.7238	[15]
W_x	(L matrix water) (L mito)$^{-1}$	Mito matrix water space ratio	0.9 W_m	[15]
W_i	(L IM water) (L mito)$^{-1}$	IMS water space ratio	0.1 W_m	[15]

Table 2 Parameters for ATP synthesis in vitro.

The following code simulates steady state ATP, ADP, and Pi concentrations for $\Delta\Psi = 175$ mV. Here, a pH gradient is fixed across the IMM such that the pH in the matrix is slightly more basic than the cytosol, $\mathrm{pH}_x = 7.4$ and $\mathrm{pH}_c = 7.2$. All other conditions remain unchanged.

```python
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
import numpy as np

# Define system of ordinary differential equations from equation (12)
def dXdt(t, X, DPsi, pH_c):
    # Unpack X state variable
    sumATP, sumADP, sumPi = X

    # Biophysical constants
    R   = 8.314          # J (mol * K)**(-1)
    T   = 310.15         # K
    F   = 96485          # C mol**(-1)
    C_m = 3.1e-3         # mol (V * L mito)**(-1)

    # F0F1 constants
    n_F = 8/3
    X_F = 1000            # mol (s * L mito)**(-1)

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_MgADP = 10**(-3.00)
    K_MgPi  = 10**(-1.66)
    K_HATP  = 10**(-6.33)
    K_HADP  = 10**(-6.26)
    K_HPi   = 10**(-6.62)
    K_KATP  = 10**(-1.02)
    K_KADP  = 10**(-0.89)
    K_KPi   = 10**(-0.42)

    # Environment concentrations
    pH_x = 7.4          # pH in matrix
    H_x  = 10**(-pH_x)  # M
    H_c  = 10**(-pH_c)  # M
    K_x  = 150e-3       # M
    Mg_x = 1e-3         # M

    # Binding polynomials
    P_ATP = 1 + H_x/K_HATP + K_x/K_KATP + Mg_x/K_MgATP # equation 6
    P_ADP = 1 + H_x/K_HADP + K_x/K_KADP + Mg_x/K_MgADP # equation 7
    P_Pi  = 1 + H_x/K_HPi  + K_x/K_KPi  + Mg_x/K_MgPi  # equation 8

    # Volume ratios
    W_m = 0.7238         # (L mito water) (L mito)**(-1)
    W_x = 0.9 * W_m      # (L matrix water) (L mito)**(-1)

    # Gibbs energy (equation 9)
    DrGo_F   = 4990      # (J mol**(-1))
    DrGapp_F = DrGo_F + R * T * np.log(H_x * P_ATP / (P_ADP * P_Pi))

    # Apparent equilibrium constant
    Kapp_F = np.exp((DrGapp_F + n_F * F * DPsi)/ (R * T)) * (H_c / H_x) ** n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP * sumPi - sumATP)

    ###### Differential equations (equation 12) ######
    dATP = J_F / W_x
    dADP = -J_F / W_x
    dPi  = -J_F / W_x

    dX = (dATP, dADP, dPi)
    return dX


# Simple steady state simulation at 175 mV membrane potential

# Initial conditions (M)
sumATP_0 = 0.5e-3
sumADP_0 = 9.5e-3
sumPi_0  = 1e-3

X_0 = np.array([sumATP_0, sumADP_0, sumPi_0])

# Inputs
DPsi = 175e-3 # Constant membrane potential (V)
pH_c = 7.2    # IMS/buffer pH

solutions = solve_ivp(dXdt, [0, 1], X_0, method = 'Radau', args = (DPsi,pH_c))
t = solutions.t
results = solutions.y
results = results * 1000

# Plot figure
plt.figure()
plt.plot(t, results[0,:], label = '[$\Sigma$ATP]$_x$')
plt.plot(t, results[1,:], label = '[$\Sigma$ADP]$_x$')
```

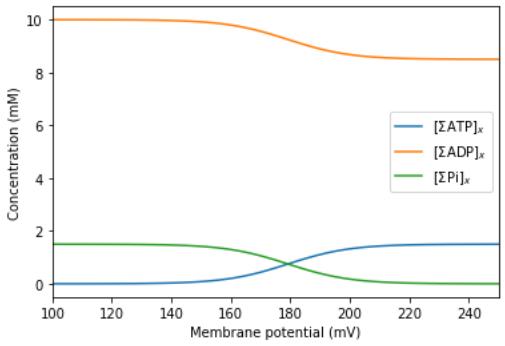
```
plt.plot(t, results[2,:], label = '[$\Sigma$Pi]$_x$')
plt.legend()
plt.xlabel('Time (s)')
plt.ylabel('Concentration (mM)')
plt.ylim(0, 10)
plt.show()
```



The above simulation shows that under the clamped pH and $\Delta\Psi$ conditions simulated here, the model is nearly in equilibrium at the initial conditions of ATP, ADP, and Pi concentrations. Most of the adenine nucleotide remains in the form of ADP and the ATP/ADP ratio in the matrix is approximately $1{:}20$, with the inorganic phosphate concentration of approximately 1 mM.

To explore how the equilibrium changes with membrane potential, the following code computes the predicted equilibrium steady-state over a ranges of $\Delta\Psi$ from 100 to 250 mV.

```
### Simulate over a range of Membrane potential from 100 mV to 250 mV ###

# Define array to iterate over
membrane_potential = np.linspace(100,250)    # mV

# Constant external pH
pH_c = 7.2 # IMS/buffer pH

# Define arrays to store steady state results
ATP_steady_DPsi = np.zeros(len(membrane_potential))
ADP_steady_DPsi = np.zeros(len(membrane_potential))
Pi_steady_DPsi  = np.zeros(len(membrane_potential))

# Iterate through range of membrane potentials
for i in range(len(membrane_potential)):
    DPsi = membrane_potential[i] / 1000      # convert to V
    temp_results = solve_ivp(dXdt, [0, 5], X_0, method = 'Radau', args = (DPsi,
pH_c,)).y*1000  # Concentration in mM
    ATP_steady_DPsi[i] = temp_results[0,-1]
    ADP_steady_DPsi[i] = temp_results[1,-1]
    Pi_steady_DPsi[i] = temp_results[2,-1]

# Concentration vs DPsi
plt.figure()
plt.plot(membrane_potential, ATP_steady_DPsi, label = '[$\Sigma$ATP]$_x$')
plt.plot(membrane_potential, ADP_steady_DPsi, label = '[$\Sigma$ADP]$_x$')
plt.plot(membrane_potential, Pi_steady_DPsi, label = '[$\Sigma$Pi]$_x$')
plt.legend()
plt.xlabel('Membrane potential (mV)')
plt.ylabel('Concentration (mM)')
plt.xlim([100, 250])
plt.show()
```



The above simulations show that under physiological levels of ΔpH, matrix ATP concentrations become essentially zero for values of the membrane potential less than approximately $150\ \mathrm{mV}$. At high levels of $\Delta\Psi$ all of the available phosphate is used to phosphorylate ADP to ATP. Since the initial $[\mathrm{Pi}]$ and $[\mathrm{ATP}]$ are $1\ \mathrm{mM}$ and $0.5\ \mathrm{mM}$, respectively, the maximum ATP obtained at the maximal $\Delta\Psi$ is $1.5\ \mathrm{mM}$.

Building a model of oxidative ATP synthesis from energetic components

Simulations in the preceding section illustrate how matrix ATP and ADP concentrations are governed by the contributors to the proton motive force. They also show how the matrix ATP/ADP ratio must typically be less than 1, in contrast to the cytosolic ATP/ADP ratio, which is on the order of 100. To understand the dependence of ATP synthesis and transport on the proton motive force, the kinetics of the processes that generate it, and the interplay of these processes, we can assemble models of the adenine nucleotide translocase (ANT), mitochondrial phosphate transport, and complexes I, III, and IV of the electron transport chain (ETC) to generate a core model of mitochondrial oxidative ATP synthesis.

Adenine nucleotide translocase

Following synthesis of ATP from ADP and Pi in the matrix, the final step in delivering ATP to the cytosol at physiological free energy levels is the electrically driven exchange of a matrix ATP^{4-} for a cytosolic ADP^{3-}. This exchange process,

$$(\mathrm{ATP}^{4-})_x + (\mathrm{ADP}^{3-})_c \rightleftharpoons (\mathrm{ATP}^{4-})_c + (\mathrm{ADP}^{3-})_x\ ,$$

is catalyzed by the ANT. Here, we assume rapid transport of species between the cytosol and the IMS, and therefore, equate IMS and cytosol species concentrations.

To simulate the kinetics of this process, we use the Metelkin et al. model [8], which accounts for pH and electrochemical dependencies. (Kinetic parameter value estimates for this model were updated by Wu et al. [3].) The steady-state flux of ANT is expressed

$$J_{\mathrm{ANT}} = E_{\mathrm{ANT}} \frac{\dfrac{k_2^{\mathrm{ANT}} q}{K_o^D}[\mathrm{ATP}^{4-}]_x[\mathrm{ADP}^{3-}]_c - \dfrac{k_3^{\mathrm{ANT}}}{K_o^T}[\mathrm{ADP}^{3-}]_x[\mathrm{ATP}^{4-}]_c}{\left(1 + \dfrac{[\mathrm{ATP}^{4-}]_c}{K_o^T} + \dfrac{[\mathrm{ADP}^{3-}]_c}{K_o^D}\right)([\mathrm{ADP}^{3-}]_x + [\mathrm{ATP}^{4-}]_x q)},$$

where $E_{\mathrm{ANT}}\ (\mathrm{mol\ (L\ mito)}^{-1})$ is the total ANT content of the mitochondria and

$$\begin{aligned} k_2^{\mathrm{ANT}} &= k_{2,o}^{\mathrm{ANT}} e^{(-3A-4B+C)\phi}, \\ k_3^{\mathrm{ANT}} &= k_{3,o}^{\mathrm{ANT}} e^{(-4A-3B+C)\phi}, \\ K_o^D &= K_o^{D,0} e^{3\delta_D \phi}, \\ K_o^T &= K_o^{T,0} e^{4\delta_T \phi}, \\ q &= \frac{k_3^{\mathrm{ANT}} K_o^D}{k_2^{\mathrm{ANT}} K_o^T} e^{\phi}, \quad \text{and} \\ \phi &= F\Delta\Psi/R\mathrm{T}. \end{aligned} \tag{13}$$

All parameter values and units can be found in Table 3, reproduced from [15].

Parameter	Units	Description	Value
E_{ANT}	mol (L mito)$^{-1}$	ANT activity	0.325
δ_D		ADP displacement binding constant	0.0167
δ_T		ATP displacement binding constant	0.0699
$k_{2,o}^{\mathrm{ANT}}$	s^{-1}	Forward translocation rate	0.159
$k_{3,o}^{\mathrm{ANT}}$	s^{-1}	Reverse translocation rate	0.501
$K_o^{D,0}$	μmol (L cyto water)$^{-1}$	ADP binding constant	38.89
$K_o^{T,0}$	μmol (L cyto water)$^{-1}$	ATP binding constant	56.05
A		Translocation displacement constant	0.2829
B		Translocation displacement constant	−0.2086
C		Translocation displacement constant	0.2372

Table 3 Adenine nucleotide translocase (ANT) parameters.

To simulate ANT and F_0F_1 ATP synthase activity simultaneously, we extend the system of Equation (12) by adding states for cytosolic species $[\Sigma\mathrm{ATP}]_c$ and $[\Sigma\mathrm{ADP}]_c$, yielding

$$\begin{cases} \dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_x}{\mathrm{d}t} = (J_{\mathrm{F}} - J_{\mathrm{ANT}})/W_x, & \dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_c}{\mathrm{d}t} = (V_{m2c}J_{\mathrm{ANT}})/W_c, \\ \dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_x}{\mathrm{d}t} = (-J_{\mathrm{F}} + J_{\mathrm{ANT}})/W_x, & \dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_c}{\mathrm{d}t} = (-V_{m2c}J_{\mathrm{ANT}})/W_c, \\ \dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_x}{\mathrm{d}t} = 0 \end{cases} \tag{14}$$

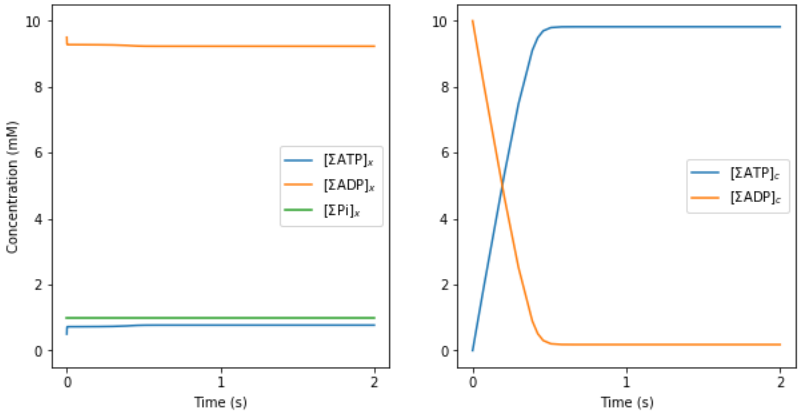
where V_{m2c} (L mito) (L cyto)$^{-1}$ is the fraction of the volume of mitochondria per volume cytosol and W_c (L cyto water) (L cyto)$^{-1}$ is the fraction of water volume in the cytoplasm to the total volume of the cytoplasm (Table 2).

Here, we clamp the matrix phosphate concentration at a constant value since the equation (14) does not account for phosphate transport between the matrix and the cytosol.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 0.6601       # cytosol volume fraction       # L cyto (L cell)**(-1)
V_m = 0.2882       # mitochondrial volume fraction # L mito (L cell)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cuvette)**(-1)
W_c = 0.8425      # cytosol water space           # L cyto water (L cyto)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Membrane capacitance
Cm   = 3.1e-3

# Membrane potential
DPsi = 175/1000

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3     # mol (L matrix water)**(-1)
K_c  = 140e-3     # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3     # mol (L matrix water)**(-1)
Mg_c = 1.0e-3     # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25          # mmHg

###### Parameter vector ######
X_F   = 1000      # Synthase activity
E_ANT = 0.325     # Nucleotide transporter activity
activity_array = np.array([X_F, E_ANT])     # Note: This array will be larger in the
future parts

###### Initial Conditions ######
# Matrix species
sumATP_x_0 = 0.5e-3  # mol (L matrix water)**(-1)
sumADP_x_0 = 9.5e-3  # mol (L matrix water)**(-1)
sumPi_x_0  = 1e-3  # mol (L matrix water)**(-1)

# Cytoplasmic species
sumATP_c_0 = 0 #9.95e-3        # mol (L cyto water)**(-1)
sumADP_c_0 = 10e-3 #0.05e-3        # mol (L cyto water)**(-1)

X_0 = np.array([sumATP_x_0, sumADP_x_0, sumPi_x_0, sumATP_c_0, sumADP_c_0])

def dXdt(t, X, activity_array):
    # Unpack variables
    sumATP_x, sumADP_x, sumPi_x, sumATP_c, sumADP_c  = X
    X_F, E_ANT = activity_array

    # Hydrogen ion concentration
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)

    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
    K_KADP  = 10**(-0.89)
    K_MgPi  = 10**(-1.66)
    K_HPi   = 10**(-6.62)
    K_KPi   = 10**(-0.42)
```

```
    ## Binding polynomials
    # Matrix species # mol (L mito water)**(-1)
    PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
    PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
    PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

    # Cytosol species # mol (L cuvette water)**(-1)
    PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
    PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
    PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

    ## Unbound species
    # Matrix species
    ATP_x = sumATP_x / PATP_x # [ATP4-]_x
    ADP_x = sumADP_x / PADP_x # [ADP3-]_x
    Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

    # Cytosol species
    ATP_c = sumATP_c / PATP_c # [ATP4-]_c
    ADP_c = sumADP_c / PADP_c # [ADP3-]_c

    ###### F0F1-ATPase ######
    # ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

    # Gibbs energy (J mol**(-1))
    DrGo_F   = 4990
    DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))

    # Apparent equilibrium constant
    Kapp_F = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)

    ###### ANT ######
    # ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

    #Constants
    del_D   = 0.0167
    del_T   = 0.0699
    k2o_ANT = 9.54/60      # s**(-1)
    k3o_ANT = 30.05/60     # s**(-1)
    K0o_D   = 38.89e-6     # mol (L cuvette water)**(-1)
    K0o_T   = 56.05e-6     # mol (L cuvette water)**(-1)
    A       = +0.2829
    B       = -0.2086
    C       = +0.2372

    phi = F * DPsi / (R * T)

    # Reaction rates (s**(-1))
    k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
    k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

    # Dissociation constants (M)
    K0_D = K0o_D * np.exp(3*del_D*phi)
    K0_T = K0o_T * np.exp(4*del_T*phi)

    q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
    term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
    term2 = k3_ANT * ADP_x * ATP_c / K0_T
    num   = term1 - term2
    den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

    # Flux (mol (s * L mito)**(-1))
    J_ANT = E_ANT * num / den

    ###### Differential equations (equation 14) ######
    # Matrix species
    dATP_x = (J_F  - J_ANT) / W_x
    dADP_x = (-J_F + J_ANT) / W_x
    dPi_x  = 0 #(-J_F ) / W_x

    # Cytosol species
    dATP_c = ( V_m2c * J_ANT) / W_c
    dADP_c = (-V_m2c * J_ANT) / W_c

    dX = [dATP_x, dADP_x, dPi_x, dATP_c, dADP_c]
    return dX

# Solve ODE
results = solve_ivp(dXdt, [0, 2], X_0, method = 'Radau', args=(activity_array,))
t = results.t
sumATP_x, sumADP_x, sumPi_x, sumATP_c, sumADP_c = results.y

# Plot figures
fig, ax = plt.subplots(1,2, figsize = (10,5))
```

```
ax[0].plot(t, sumATP_x*1000, label = '[$\Sigma$ATP]$_x$')
ax[0].plot(t, sumADP_x*1000, label = '[$\Sigma$ADP]$_x$')
ax[0].plot(t, sumPi_x*1000, label = '[$\Sigma$Pi]$_x$')
ax[0].legend(loc="right")
ax[0].set_ylim((-.5,10.5))
ax[0].set_xlabel('Time (s)')
ax[0].set_xticks([0,1,2])
ax[0].set_ylabel('Concentration (mM)')

ax[1].plot(t, sumATP_c*1000, label = '[$\Sigma$ATP]$_c$')
ax[1].plot(t, sumADP_c*1000, label = '[$\Sigma$ADP]$_c$')
ax[1].set_ylim((-0.5,10.5))
ax[1].set_xticks([0,1,2])
ax[1].legend(loc="right")
ax[1].set_xlabel('Time (s)')
plt.show()
```



The above simulations of the system of Equation (14) show how the electrogenic nature of the ANT transport results in markedly different ATP/ADP ratios in the cytosol compared to the matrix. As we saw in the previous chapter, the ATP hydrolysis potential in the matrix is approximately -45 kJ mol^{-1}. The roughly 50:1 ratio of ATP to ADP in the cytosol is associated with a hydrolysis potential of approximately -65 kJ mol^{-1}. The difference of 20 kJ mol^{-1} between the matrix and the cytosolic space is driven primarily by the membrane potential, which is equivalent to 20 kJ mol^{-1}.

Inorganic phosphate transport

During active ATP synthesis, mitochondrial Pi is replenished via the activity of the phosphate-proton cotransporter (PiC), catalyzing the electroneutral cotransport of protonated inorganic phosphate, $\mathrm{H_2PO_4^-}$, and $\mathrm{H^+}$ across the membrane. Again, we assume rapid transport between the cytoplasm and intermembrane space, and hence, we have

$$(\mathrm{H_2PO_4^-})_x + (\mathrm{H^+})_x \rightleftharpoons (\mathrm{H_2PO_4^-})_c + (\mathrm{H^+})_c.$$

Adopting the flux equation from Bazil et al. [15], we have

$$J_{\mathrm{PiC}} = E_{\mathrm{PiC}} \frac{[\mathrm{H^+}]_c[\mathrm{H_2PO_4^-}]_c - [\mathrm{H^+}]_x[\mathrm{H_2PO_4^-}]_x}{[\mathrm{H_2PO_4^-}]_c + k_{\mathrm{PiC}}},$$

where E_{PiC} (L matrix water) s^{-1} (L mito)$^{-1}$ is the PiC activity rate and $k_{\mathrm{PiC}} = 1.61$ mM is an effective Michaelis-Menten constant. The $\mathrm{H_2PO_4^-}$ concentrations in the matrix and cytosol are computed via the relationship

$$[\mathrm{H_2PO_4^-}] = [\Sigma\mathrm{Pi}] \left([\mathrm{H^+}]/K_{\mathrm{HPi}}\right)/P_{\mathrm{Pi}}.$$

To incorporate PiC into Equation (14), we add a new state $[\Sigma\mathrm{Pi}]_c$ such that at given membrane potential, matrix and cytosolic pH, and cation concentrations, we obtain

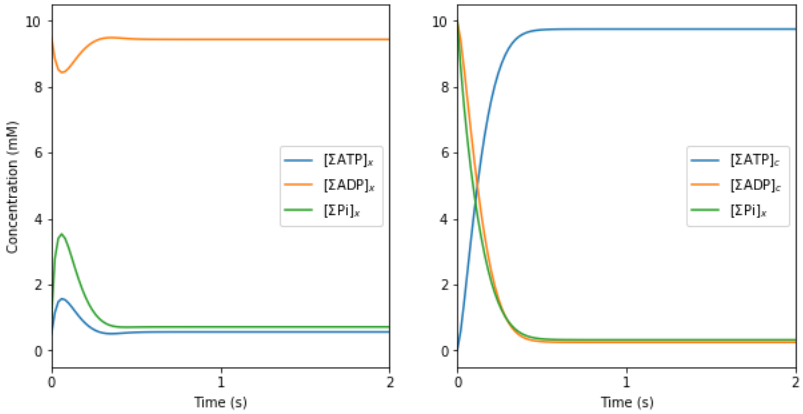
$$\begin{cases} \dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_x}{\mathrm{d}t} = (J_{\mathrm{F}} - J_{\mathrm{ANT}})/W_x, & \dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_c}{\mathrm{d}t} = (V_{m2c}J_{\mathrm{ANT}})/W_c \\ \dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_x}{\mathrm{d}t} = (-J_{\mathrm{F}} + J_{\mathrm{ANT}})/W_x, & \dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_c}{\mathrm{d}t} = (-V_{m2c}J_{\mathrm{ANT}})/W_c, \\ \dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_x}{\mathrm{d}t} = (-J_{\mathrm{F}} + J_{\mathrm{PiC}})/W_x, & \dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_c}{\mathrm{d}t} = (-V_{m2c}J_{\mathrm{PiC}})/W_c, \end{cases} \tag{15}$$

The following code simulates the synthesis of ATP from ADP and Pi and their translocation across the IMM under physiological conditions.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 0.6601       # cytosol volume fraction       # L cyto (L cell)**(-1)
V_m = 0.2882       # mitochondrial volume fraction # L mito (L cell)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cuvette)**(-1)
W_c = 0.8425      # cytosol water space           # L cyto water (L cyto)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Membrane capacitance
Cm = 3.1e-3

# Membrane potential
DPsi = 175/1000

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3      # mol (L matrix water)**(-1)
K_c  = 140e-3      # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3        # mol (L matrix water)**(-1)
Mg_c = 1.0e-3        # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25 # mmHg

###### Parameter vector ######
X_F   = 100        # Synthase activity
E_ANT = 0.325      # Nucleotide transporter activity
E_PiC = 5.0e6      # Phosphate transporter activity

activity_array = np.array([X_F, E_ANT, E_PiC])

###### Initial Conditions ######
# Matrix species
sumATP_x_0 = 0.5e-3  # mol (L matrix water)**(-1)
sumADP_x_0 = 9.5e-3  # mol (L matrix water)**(-1)
sumPi_x_0  = 1e-3  # mol (L matrix water)**(-1)

# Cytosolic species
sumATP_c_0 = 0       # mol (L cyto water)**(-1)
sumADP_c_0 = 10e-3   # mol (L cyto water)**(-1)
sumPi_c_0  = 10e-3   # mol (L cyto water)**(-1)

X_0 = np.array([sumATP_x_0, sumADP_x_0, sumPi_x_0, sumATP_c_0, sumADP_c_0, sumPi_c_0])

def dXdt(t, X, activity_array):
    # Unpack variables
    sumATP_x, sumADP_x, sumPi_x, sumATP_c, sumADP_c, sumPi_c  = X
    X_F, E_ANT, E_PiC = activity_array

    # Hydrogen ion concentration
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)

    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3
    n_C1 = 4
    n_C3 = 2
    n_C4 = 4

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
```

```
K_KADP  = 10**(-0.89)
K_MgPi  = 10**(-1.66)
K_HPi   = 10**(-6.62)
K_KPi   = 10**(-0.42)

## Binding polynomials
# Matrix species # mol (L mito water)**(-1)
PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

# Cytosol species # mol (L cuvette water)**(-1)
PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

## Unbound species
# Matrix species
ATP_x = sumATP_x / PATP_x # [ATP4-]_x
ADP_x = sumADP_x / PADP_x # [ADP3-]_x
Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

# Cytosol species
ATP_c = sumATP_c / PATP_c # [ATP4-]_c
ADP_c = sumADP_c / PADP_c # [ADP3-]_c
Pi_c  = sumPi_c  / PPi_c  # [HPO42-]_c

###### H+-PI2 cotransporter ######
# H2PO42-_x + H+_x = H2PO42-_c + H+_c

# Constant
k_PiC = 1.61e-3  # mol (L cuvette)**(-1)

# H2P04- species
HPi_c = Pi_c * (H_c / K_HPi)
HPi_x = Pi_x * (H_x / K_HPi)

# Flux (mol (s * L mito)**(-1))
J_PiC = E_PiC * (H_c * HPi_c - H_x * HPi_x) / (k_PiC + HPi_c)

###### F0F1-ATPase ######
# ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

# Gibbs energy (J mol**(-1))
DrGo_F = 4990
DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))

# Apparent equilibrium constant
Kapp_F   = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

# Flux (mol (s * L mito)**(-1))
J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)

###### ANT ######
# ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

# Constants
del_D   = 0.0167
del_T   = 0.0699
k2o_ANT = 9.54/60      # s**(-1)
k3o_ANT = 30.05/60     # s**(-1)
K0o_D   = 38.89e-6     # mol (L cuvette water)**(-1)
K0o_T   = 56.05e-6     # mol (L cuvette water)**(-1)
A       = +0.2829
B       = -0.2086
C       = +0.2372

phi = F * DPsi / (R * T)

# Reaction rates (s**(-1))
k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

# Dissociation constants (M)
K0_D = K0o_D * np.exp(3*del_D*phi)
K0_T = K0o_T * np.exp(4*del_T*phi)

q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
term2 = k3_ANT * ADP_x * ATP_c / K0_T
num   = term1 - term2
den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

# Flux (mol (s * L mito)**(-1))
J_ANT = E_ANT * num / den

###### Differential equations (equation 15) ######
```

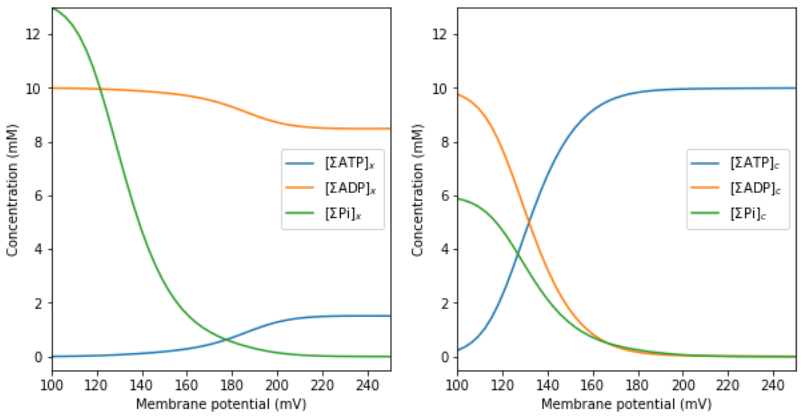
```
    # Matrix species
    dATP_x  = (J_F  - J_ANT) / W_x
    dADP_x  = (-J_F + J_ANT) / W_x
    dPi_x   = (-J_F + J_PiC) / W_x

    # Buffer species
    dATP_c = ( V_m2c * J_ANT) / W_c
    dADP_c = (-V_m2c * J_ANT) / W_c
    dPi_c  = (-V_m2c * J_PiC) / W_c

    dX = [dATP_x, dADP_x, dPi_x, dATP_c, dADP_c, dPi_c]
    return dX

# Solve ODE
t = np.linspace(0,2,100)
results = solve_ivp(dXdt, [0, 2], X_0, method = 'Radau', t_eval = t, args=
(activity_array,))
sumATP_x, sumADP_x, sumPi_x, sumATP_c, sumADP_c, sumPi_c = results.y

# Plot figures
fig, ax = plt.subplots(1,2, figsize = (10,5))
ax[0].plot(t, sumATP_x*1000, label = '[$\Sigma$ATP]$_x$')
ax[0].plot(t, sumADP_x*1000, label = '[$\Sigma$ADP]$_x$')
ax[0].plot(t, sumPi_x*1000, label = '[$\Sigma$Pi]$_x$')
ax[0].legend(loc="right")
ax[0].set_ylim((-.5,10.5))
ax[0].set_xlim((0,2))
ax[0].set_xticks([0,1,2])
ax[0].set_xlabel('Time (s)')
ax[0].set_ylabel('Concentration (mM)')

ax[1].plot(t, sumATP_c*1000, label = '[$\Sigma$ATP]$_c$')
ax[1].plot(t, sumADP_c*1000, label = '[$\Sigma$ADP]$_c$')
ax[1].plot(t, sumPi_c*1000, label = '[$\Sigma$Pi]$_x$')
ax[1].set_ylim((-0.5,10.5))
ax[1].set_xlim((0,2))
ax[1].set_xticks([0,1,2])
ax[1].legend(loc="right")
ax[1].set_xlabel('Time (s)')
plt.show()
```



For the above simulations, cytosolic inorganic phosphate is set to 10 mM initially, and all other initial conditions remain unchanged. Driven by $\Delta\mathrm{pH}$, a gradient in phosphate concentration is established, with a steady-state ratio of matrix-to-cytosol concentration of approximately 2.2. As seen in the previous section, with a constant membrane potential of 175 mV, the ATP/ADP ratio is maintained at a much higher level in the cytosol than in the matrix.

The final matrix and cytosol ATP and ADP concentrations depend not only on the membrane potential, but also on the total amount of exchangeable phosphate in the system. Here these simulations start with $[\mathrm{Pi}]_c = 10$ mM and $[\mathrm{Pi}]_x = 1$ mM. The initial 10 mM of ADP in the cytosol becomes almost entirely phosphorylated to ATP, leaving 0.32 mM of inorganic phosphate in the cytosol in the final steady state. To explore how these steady states depend on $\Delta\Psi$, the following code simulates the steady-state behavior of this system for a range of $\Delta\Psi$ from 100 to 200 mV.

```
### Simulate over a range of Membrane potential from 100 mV to 250 mV ###

# Define array to iterate over
membrane_potential = np.linspace(100,250)    # mV

# Constant external pH
pH_c = 7.2 # IMS/buffer pH

# Define arrays to store steady state results
ATP_x_steady = np.zeros(len(membrane_potential))
ADP_x_steady = np.zeros(len(membrane_potential))
Pi_x_steady  = np.zeros(len(membrane_potential))

ATP_c_steady = np.zeros(len(membrane_potential))
ADP_c_steady = np.zeros(len(membrane_potential))
Pi_c_steady  = np.zeros(len(membrane_potential))

# Iterate through range of membrane potentials
for i in range(len(membrane_potential)):
    DPsi = membrane_potential[i] / 1000      # convert to V
    temp_results = solve_ivp(dXdt, [0, 200], X_0, method = 'Radau', args=
(activity_array,)).y*1000  # Concentration in mM
    ATP_x_steady[i] = temp_results[0,-1]
    ADP_x_steady[i] = temp_results[1,-1]
    Pi_x_steady[i]  = temp_results[2,-1]
    ATP_c_steady[i] = temp_results[3,-1]
    ADP_c_steady[i] = temp_results[4,-1]
    Pi_c_steady[i]  = temp_results[5,-1]

# Plot figures
fig, ax = plt.subplots(1,2, figsize = (10,5))
ax[0].plot(membrane_potential, ATP_x_steady, label = '[$\Sigma$ATP]$_x$')
ax[0].plot(membrane_potential, ADP_x_steady, label = '[$\Sigma$ADP]$_x$')
ax[0].plot(membrane_potential, Pi_x_steady, label = '[$\Sigma$Pi]$_x$')
ax[0].legend(loc = "right")
ax[0].set_xlabel('Membrane potential (mV)')
ax[0].set_ylabel('Concentration (mM)')
ax[0].set_xlim([100, 250])
ax[0].set_ylim([-0.5,13])

ax[1].plot(membrane_potential, ATP_c_steady, label = '[$\Sigma$ATP]$_c$')
ax[1].plot(membrane_potential, ADP_c_steady, label = '[$\Sigma$ADP]$_c$')
ax[1].plot(membrane_potential, Pi_c_steady, label = '[$\Sigma$Pi]$_c$')
ax[1].legend(loc = "right")
ax[1].set_xlabel('Membrane potential (mV)')
ax[1].set_ylabel('Concentration (mM)')
ax[1].set_xlim([100, 250])
ax[1].set_ylim([-0.5,13])

plt.show()
```



Simulation of this system reinforces the fact that ATP cannot be synthesized at physiological free energy levels for mitochondrial membrane potential of less than approximately $150\ \mathrm{mV}$.

Respiratory complexes and NADH synthesis

The previous sections have assumed a constant membrane potential. To account for the processes that generate the membrane potential, we model proton pumping associated with the respiratory complexes I, III, and IV of the ETC (Fig. 1).

ETC complex I

Coupled with the translocation of $n_{C1} = 4$ protons across the IMM against the electrochemical gradient, electrons are transferred from NADH to ubiquinone (Q) at complex I of the ETC via the reaction

$$(\mathrm{NADH^{2-}})_x + (\mathrm{H^+})_x + (\mathrm{Q})_x + n_{C1}(\mathrm{H^+})_x \rightleftharpoons (\mathrm{NAD^-})_x + (\mathrm{QH_2})_x + \mathrm{H_2O} + n_{C1}(\mathrm{H^+})_c. \tag{16}$$

Since protons move against the gradient when the reaction proceeds in the left-to-right direction, the overall Gibbs energy for the reaction of Equation (16) is

$$\begin{aligned}\Delta G_{C1} &= \Delta_r G_{C1} - n_{C1}\Delta G_{\mathrm{H}} \\ &= \Delta_r G^\circ_{C1} + RT\ln\left(\frac{[\mathrm{NAD^-}]_x[\mathrm{QH_2}]_x}{[\mathrm{NADH^{2-}}]_x[\mathrm{Q}]_x}\cdot\frac{1}{[\mathrm{H^+}]_x}\right) + n_{C1}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C1}} \\ &= \Delta_r G'^\circ_{C1} + RT\ln\left(\frac{[\mathrm{NAD^-}]_x[\mathrm{QH_2}]_x}{[\mathrm{NADH^{2-}}]_x[\mathrm{Q}]_x}\right) + n_{C1}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C1}},\end{aligned} \tag{17}$$

where

$$\Delta_r G'^\circ_{C1} = \Delta_r G^\circ_{C1} - RT\ln([\mathrm{H^+}]_x)$$

is the apparent Gibbs energy for Equation (16). The apparent equilibrium constant is

$$K'_{eq,C1} = \left(\frac{[\mathrm{NAD^-}]_x[\mathrm{QH_2}]_x}{[\mathrm{NADH^{2-}}]_x[\mathrm{Q}]_x}\right)_{eq} = \exp\left\{\frac{-(\Delta_r G'^\circ_{C1} + n_{C1}F\Delta\Psi)}{RT}\right\}\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C1}}.$$

To simulate the flux of complex I, J_{C1} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$, across the IMM by mass-action kinetics, we have

$$J_{C1} = X_{C1}(K'_{eq,C1}[\mathrm{NADH^{2-}}]_x[\mathrm{Q}]_x - [\mathrm{NAD^-}]_x[\mathrm{QH_2}]_x), \tag{18}$$

for X_{C1} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$ the a rate constant. Table 4 lists the constants for complex I.

ETC complex III

The reaction catalyzed by complex III reduces two cytochrome c proteins for every $\mathrm{QH_2}$ oxidized

$$(\mathrm{QH_2})_x + 2\ (\mathrm{c}^{3+}_{ox})_i + n_{C3}(\mathrm{H^+})_x \rightleftharpoons (\mathrm{Q})_x + 2\ (\mathrm{c}^{2+}_{red})_i + 2\ (\mathrm{H^+})_c + n_{C3}(\mathrm{H^+})_c, \tag{19}$$

where c^{3+}_{ox} and c^{2+}_{red} are the oxidized and reduced cytochrome c species and the subscript *i* indicates that cytochrome c is confined to the IMS. This reaction is coupled with the transport of $n_{C3} = 2$ protons from the matrix to the cytosol against the electrochemical gradient. Thus, the Gibbs energy for the overall reaction given in Equation (19) is

$$\begin{aligned}\Delta G_{C3} &= \Delta_r G_{C3} - n_{C3}\Delta G_{\mathrm{\backslash rm\ H}} \\ &= \Delta_r G^\circ_{C3} + RT\ln\left(\frac{[\mathrm{Q}]_x[\mathrm{c}^{2+}_{red}]_i^2}{[\mathrm{QH_2}]_x[\mathrm{c}^{3+}_{ox}]_i^2}\cdot[\mathrm{H^+}]_c^2\right) + n_{C3}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C3}} \\ &= \Delta_r G'^\circ_{C3} + RT\ln\left(\frac{[\mathrm{Q}]_x[\mathrm{c}^{2+}_{red}]_i^2}{[\mathrm{QH_2}]_x[\mathrm{c}^{3+}_{ox}]_i^2}\right) + n_{C3}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C3}},\end{aligned} \tag{20}$$

where

$$\Delta_r G'^\circ_{C3} = \Delta_r G^\circ_{C3} + 2RT\ln([\mathrm{H^+}]_c)$$

is the apparent Gibbs energy for complex III. The apparent equilibrium constant is

$$K'_{eq,C3} = \left(\frac{[\mathrm{Q}]_x[\mathrm{c}^{2+}_{red}]_i^2}{[\mathrm{QH_2}]_x[\mathrm{c}^{3+}_{ox}]_i^2}\right)_{eq} = \exp\left\{\frac{-(\Delta_r G'^\circ_{C3} + n_{C3}F\Delta\Psi)}{RT}\right\}\left(\frac{[\mathrm{H^+}]_x}{[\mathrm{H^+}]_c}\right)^{n_{C3}},$$

To simulate the flux of complex III, J_{C3} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$, by mass-action kinetics, we have

$$J_{C3} = X_{C3}(K'_{eq,C3}[\mathrm{QH_2}]_x[\mathrm{c}^{3+}_{ox}]_i^2 - [\mathrm{Q}]_x[\mathrm{c}^{2+}_{red}]_i^2),$$

where X_{C3} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$ is the rate constant.

ETC complex IV

In the final step of the ECT catalzyed by complex IV, electrons are transferred from cytochrome c to oxygen, forming water

(21)

$$2\,(\mathrm{c}^{2+}_{red})_i + \frac{1}{2}(\mathrm{O}_2)_x + 2\,(\mathrm{H}^+)_c + n_{\mathrm{C4}}([\mathrm{H}^+])_x \rightleftharpoons 2\,(\mathrm{c}^{3+}_{ox})_i + \mathrm{H_2O} + n_{\mathrm{C4}}([\mathrm{H}^+])_c,$$

coupled with the translocation of $n_{\mathrm{C4}} = 4$ protons across the IMM against against the electrochemical gradient. The Gibbs energy of the reaction in Equation (21) is

$$\begin{aligned}\Delta G_{\mathrm{C4}} &= \Delta_r G_{\mathrm{C4}} - n_{\mathrm{C4}}\Delta G_{\mathrm{H}} \\ &= \Delta_r G^o_{\mathrm{C4}} + RT\ln\left(\frac{[\mathrm{c}^{3+}_{ox}]_i^2}{[\mathrm{c}^{2+}_{red}]_i^2[\mathrm{O}_2]_x^{1/2}}\cdot\frac{1}{[\mathrm{H}^+]_c^2}\right) + n_{\mathrm{C4}}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H}^+]_x}{[\mathrm{H}^+]_c}\right)^{n_{\mathrm{C4}}} \\ &= \Delta_r G'^{\circ}_{\mathrm{C4}} + RT\ln\left(\frac{[\mathrm{c}^{3+}_{ox}]_i^2}{[\mathrm{c}^{2+}_{red}]_i^2[\mathrm{O}_2]_x^{1/2}}\right) + n_{\mathrm{C4}}F\Delta\Psi - RT\ln\left(\frac{[\mathrm{H}^+]_x}{[\mathrm{H}^+]_c}\right)^{n_{\mathrm{C4}}},\end{aligned} \tag{22}$$

where

$$\Delta_r G'^{\circ}_{\mathrm{C4}} = \Delta_r G^{\circ}_{\mathrm{C4}} - 2RT\ln([\mathrm{H}^+]_c)$$

is the apparent Gibbs energy for complex IV. The apparent equilibrium constant is

$$K'_{eq,\mathrm{C4}} = \left(\frac{[\mathrm{c}^{3+}_{ox}]_i^2}{[\mathrm{c}^{2+}_{red}]_i^2[\mathrm{O}_2]_x^{1/2}}\right)_{eq} = \exp\left\{\frac{-(\Delta_r G'^{\circ}_{\mathrm{C4}} + n_{\mathrm{C4}}F\Delta\Psi)}{RT}\right\}\left(\frac{[\mathrm{H}^+]_x}{[\mathrm{H}^+]_c}\right)^{n_{\mathrm{C4}}}.$$

To simulate the flux of complex IV, J_{C4} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$, we use mass-action kinetics and account for binding of oxygen to complex IV as

$$J_{\mathrm{C4}} = X_{\mathrm{C4}}\left(\frac{1}{1+\frac{k_{\mathrm{O}_2}}{[\mathrm{O}_2]}}\right)\left[\left(K'_{eq,\mathrm{C4}}\right)^{1/2}[\mathrm{c}^{2+}_{red}]_i[\mathrm{O}_2]_x^{1/4} - [\mathrm{c}^{3+}_{ox}]_i\right],$$

where X_{C4} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$ is the rate constant and k_{O_2} is the O_2 binding constant (Table 4). For this study, we assume a partial pressure of O_2 at 25 mmHg.

Dehydrogenase activity

We do not explicitly model the full TCA cycle but rather the combined action of NADH-producing reactions, that is,

$$(\mathrm{NAD^-})_x \rightleftharpoons (\mathrm{NADH^{2-}})_x + (\mathrm{H^+})_x$$

From Beard [1], the flux of NADH dehydrogenase is assumed to be phosphate-dependent

$$J_{\mathrm{DH}} = X_{\mathrm{DH}}(r[\mathrm{NAD^-}] - [\mathrm{NADH^{2-}}])\left(\frac{1+[\Sigma\mathrm{Pi}]_x/k_{\mathrm{Pi},1}}{1+[\Sigma\mathrm{Pi}]_x/k_{\mathrm{Pi},2}}\right),$$

where X_{DH} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$ is the dehydrogenase activity and r (dimensionless), $k_{\mathrm{Pi},1}$ $(\mathrm{mol\ (L\ matrix\ water)^{-1}})$, and $k_{\mathrm{Pi},2}$ $(\mathrm{mol\ (L\ matrix\ water)^{-1}})$ are constants. Parameter values can be found in Table 4.

Proton leak

To simulate proton leakage across the IMM, we adopt the Goldman-Hodgkins-Katz formulation from Wu et al. [3],

$$J_{\mathrm{H}} = X_{\mathrm{H}}([\mathrm{H}^+]_c\ e^{\phi/2} - [\mathrm{H}^+]_x\ e^{-\phi/2})$$

where X_{H} $(\mathrm{mol\ s^{-1}\ (L\ mito)^{-1}})$ is the proton leak activity and ϕ is given in Equation (13).

Parameter	Units	Description	Value	Source
n_{C1}		Protons translocated by complex I	4	[11]
n_{C3}		Protons translocated by complex III	2	[11]
n_{C4}		Protons translocated by complex IV	4	[11]
X_{C1}	mol s^{-1} (L mito)$^{-1}$	Complex I rate constant	1e4	
X_{C3}	mol s^{-1} (L mito)$^{-1}$	Complex III rate constant	1e6	
X_{C4}	mol s^{-1} (L mito)$^{-1}$	Complex IV rate constant	0.0125	
X_{DH}	mol s^{-1} (L mito)$^{-1}$	NADH dehydrogenase rate constant	0.1732	
X_{H}	mol s^{-1} (L mito)$^{-1}$	Proton leak activity	1e3	
r		Dehydrogenase parameter	6.8385	
$k_{Pi,1}$	mmol (L matrix water)$^{-1}$	Dehydrogenase parameter	0.466	
$k_{Pi,2}$	mmol (L matrix water)$^{-1}$	Dehydrogenase parameter	0.658	
k_{PiC}	mmol (L cell)$^{-1}$	PiC constant	1.61	[15]
k_{O_2}	μmol (L matrix water)$^{-1}$	O_2 binding constant	120	[4]
$\Delta_r G^o_{C1}$	kJ mol^{-1}	Gibbs energy of reaction for complex I	−109.7	[7]
$\Delta_r G^o_{C3}$	kJ mol^{-1}	Gibbs energy of reaction for complex III	46.7	[7]
$\Delta_r G^o_{C4}$	kJ mol^{-1}	Gibbs energy of reaction for complex IV	−202.2	[7]
$[NAD]_{tot}$	mmol (L matrix water)$^{-1}$	Total NAD pool in the matrix	2.97	[4]
$[Q]_{tot}$	mmol (L matrix water)$^{-1}$	Total Q pool in the matrix	1.35	[4]
$[c]_{tot}$	mmol (L IM water)$^{-1}$	Total cytochrome c pool in the IMS	2.70	[4]

Table 4 Respiratory complex and inorganic phosphate transport parameters

Simulating ATP synthesis in vitro

The flux expressions developed above may be used to simulate mitochondrial ATP synthesis in vitro, governed by the system of equations

$$\begin{cases} \dfrac{d\Delta\Psi}{dt} = (n_{C1}J_{C1} + n_{C3}J_{C3} + n_{C4}J_{C4} - n_F J_F - J_{ANT} - J_H)/C_m \\ \hline \dfrac{d[\Sigma ATP]_x}{dt} = (J_F - J_{ANT})/W_x \\ \dfrac{d[\Sigma ADP]_x}{dt} = (-J_F + J_{ANT})/W_x \\ \dfrac{d[\Sigma Pi]_x}{dt} = (-J_F + J_{PiC})/W_x \quad \text{matrix species} \\ \dfrac{d[NADH^{2-}]_x}{dt} = (J_{DH} - J_{C1})/W_x \\ \dfrac{d[QH_2]_x}{dt} = (J_{C1} - J_{C3})/W_x \\ \hline \dfrac{d[c^{2+}_{red}]_i}{dt} = 2(J_{C3} - J_{C4})/W_i \quad \text{intermembrane space species} \\ \hline \dfrac{d[\Sigma ATP]_c}{dt} = (V_{m2c}J_{ANT} - J_{AtC})/W_c \\ \dfrac{d[\Sigma ADP]_c}{dt} = (-V_{m2c}J_{ANT} + J_{AtC})/W_c \quad \text{cytoplasm species} \\ \dfrac{d[\Sigma Pi]_c}{dt} = (-V_{m2c}J_{PiC} + J_{AtC})/W_c, \end{cases} \tag{23}$$

Here, we incorporate a constant ATP consumption flux, J_{AtC} (mol s^{-1} (L cyto)$^{-1}$), that is

$$J_{AtC} = X_{AtC}/V_c$$

where V_c is the ratio of the volume of cytosol per L cell. X_{AtC} is the ATP consumption rate expressed in units of mmol s^{-1} (L cell)$^{-1}$. Equation (23) does not explicitly treat matrix or external pH, K^+, Mg^{2+}, or O_2 as variables. Reasonable clamped concentrations for these variables are $pH_x = 7.4$, $pH_c = 7.2$, $[Mg^{2+}]_x = 1$ mmol (L matrix water)$^{-1}$, $[Mg^{2+}]_c = 1$ mmol (L cyto water)$^{-1}$, $[K^+]_x = 100$ mmol (L matrix water)$^{-1}$, and $[K^+]_c = 140$ mmol (L cyto water)$^{-1}$, and O_2 partial pressure of 25 mmHg. Respiratory reactants are determined from a total concentration of metabolites within the mitochondrion, that is, the total pools for NAD, cytochrome c, and Q species are

$$\begin{aligned} [NAD]_{tot} &= [NAD^-]_x + [NADH^{2-}]_x \\ [c]_{tot} &= [c^{2+}_{red}]_i + [c^{3+}_{ox}]_i, \quad \text{and} \\ [Q]_{tot} &= [Q]_x + [QH_2]_x. \end{aligned}$$

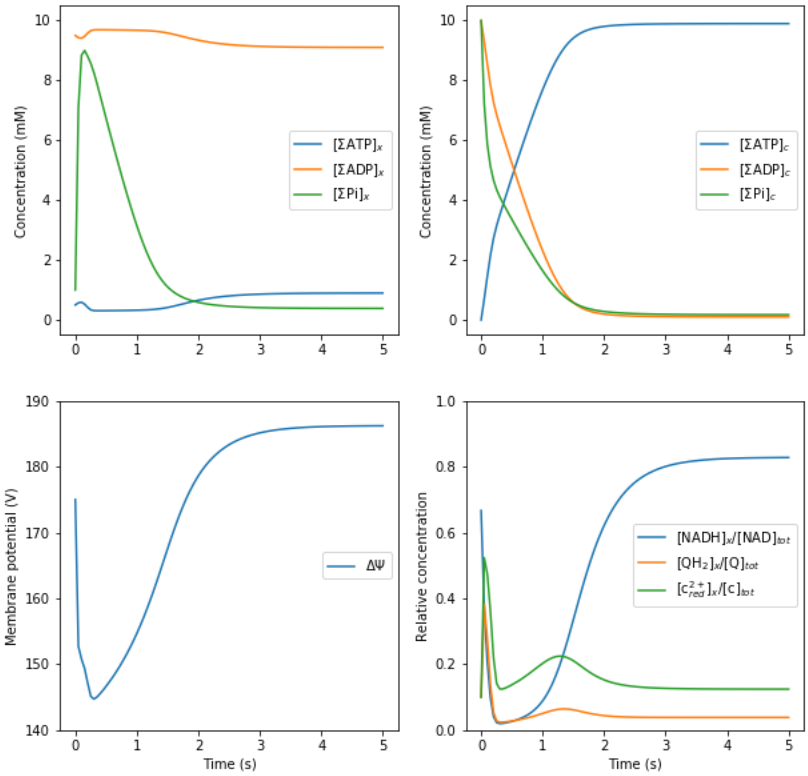
The pools are $[NAD]_{tot} = 2.97$ mmol (L matrix water)$^{-1}$, $[c]_{tot} = 2.7$ mmol (L IMS water)$^{-1}$, and $[Q]_{tot} = 1.35$ mmol (L matrix water)$^{-1}$. Initial conditions are set under the assumption that the TAN for both the matrix and cytosol is 10 mM, but the ATP/ADP ratio is <1 in the matrix and ~ 100 in the cytosol. The following code simulates in vitro mitochondrial function without ATP consumption in the external (cytosolic space).

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 0.6601       # cytosol volume fraction       # L cyto (L cell)**(-1)
V_m = 0.2882       # mitochondrial volume fraction # L mito (L cell)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cuvette)**(-1)
W_c = 0.8425      # cytosol water space           # L cyto water (L cyto)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Total pool concentrations
NAD_tot = 2.97e-3  # NAD+ and NADH conc            # mol (L matrix water)**(-1)
Q_tot   = 1.35e-3  # Q and QH2 conc                # mol (L matrix water)**(-1)
c_tot   = 2.7e-3   # cytochrome c ox and red conc  # mol (L IM water)**(-1)

# Membrane capacitance
Cm = 3.1e-3

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3      # mol (L matrix water)**(-1)
K_c  = 140e-3      # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3        # mol (L matrix water)**(-1)
Mg_c = 1.0e-3        # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25 # mmHg

###### Parameter vector ######
X_DH  = 0.1732
X_C1  = 1.0e4
X_C3  = 1.0e6
X_C4  = 0.0125
X_F   = 1.0e3
E_ANT = 0.325
E_PiC = 5.0e6
X_H   = 1.0e3
X_AtC = 0

activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC])

###### Initial Conditions ######
# Membrane Potential
DPsi_0 = 175/1000      # V

# Matrix species
sumATP_x_0 = 0.5e-3         # mol (L matrix water)**(-1)
sumADP_x_0 = 9.5e-3         # mol (L matrix water)**(-1)
sumPi_x_0  = 1.0e-3         # mol (L matrix water)**(-1)
NADH_x_0   = 2/3 * NAD_tot  # mol (L matrix water)**(-1)
QH2_x_0    = 0.1 * Q_tot    # mol (L matrix water)**(-1)

# IMS species
cred_i_0 = 0.1 * c_tot # mol (L IMS water)**(-1)

# Cytosolic species
sumATP_c_0 = 0       # mol (L cyto water)**(-1)
sumADP_c_0 = 10e-3   # mol (L cyto water)**(-1)
sumPi_c_0  = 10e-3   # mol (L cyto water)**(-1)

X_0 = np.array([DPsi_0, sumATP_x_0, sumADP_x_0, sumPi_x_0, NADH_x_0, QH2_x_0,
cred_i_0, sumATP_c_0, sumADP_c_0, sumPi_c_0])

def dXdt(t, X, activity_array, solve_ode):
    # Unpack variables
    DPsi, sumATP_x,sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c,
sumPi_c = X
    X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC = activity_array

    # Hydrogen ion concentration
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)
```

```
    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3
    n_C1 = 4
    n_C3 = 2
    n_C4 = 4

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
    K_KADP  = 10**(-0.89)
    K_MgPi  = 10**(-1.66)
    K_HPi   = 10**(-6.62)
    K_KPi   = 10**(-0.42)

    # Other concentrations computed from the state variables:
    NAD_x = NAD_tot - NADH_x  ## mol (L matrix water)**(-1)
    Q_x   = Q_tot - QH2_x     ## mol (L matrix water)**(-1)
    cox_i = c_tot - cred_i    ## mol (L matrix water)**(-1)

    ## Binding polynomials
    # Matrix species # mol (L mito water)**(-1)
    PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
    PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
    PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

    # Cytosol species # mol (L cuvette water)**(-1)
    PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
    PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
    PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

    ## Unbound species
    # Matrix species
    ATP_x = sumATP_x / PATP_x # [ATP4-]_x
    ADP_x = sumADP_x / PADP_x # [ADP3-]_x
    Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

    # Cytosolic species
    ATP_c = sumATP_c / PATP_c # [ATP4-]_c
    ADP_c = sumADP_c / PADP_c # [ADP3-]_c
    Pi_c  = sumPi_c  / PPi_c  # [HPO42-]_c

    ###### NADH Dehydrogenase ######

    # Constants
    r      = 6.8385
    k_Pi1  = 4.659e-4    # mol (L matrix water)**(-1)
    k_Pi2  = 6.578e-4    # mol (L matrix water)**(-1)

    # Flux
    J_DH = X_DH * (r * NAD_x - NADH_x) * ((1 + sumPi_x / k_Pi1) / (1+sumPi_x / k_Pi2))

    ###### Complex I ######
    # NADH_x + Q_x + 5H+_x <-> NAD+_x + QH2_x + 4H+_i + 4DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C1 = -109680
    DrGapp_C1 = DrGo_C1 - R * T * np.log(H_x)

    # Apparent equilibrium constant
    Kapp_C1   = np.exp( -(DrGapp_C1 + n_C1 * F * DPsi) / (R * T)) * ((H_x /
H_c)**n_C1)

    # Flux (mol (s * L mito)**(-1))
    J_C1 = X_C1 * (Kapp_C1 * NADH_x * Q_x - NAD_x * QH2_x)

    ###### Complex III ######
    # QH2_x + 2cuvetteC(ox)3+_i + 2H+_x <-> Q_x + 2cuvetteC(red)2+_i + 4H+_i + 2DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C3 = 46690
    DrGapp_C3 = DrGo_C3 + 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C3   = np.exp(-(DrGapp_C3 + n_C3 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C3

    # Flux (mol (s * L mito)**(-1))
    J_C3 = X_C3 * (Kapp_C3 * cox_i**2 * QH2_x - cred_i**2 * Q_x)
```

```
    ###### Complex IV ######
    # 2 cytoC(red)2+_i + 0.5O2_x + 4H+_x <-> cytoC(ox)3+_x + H2O_x + 2H+_i +2DPsi

    # Constant
    k_O2 = 1.2e-4      # mol (L matrix water)**(-1)

    # Gibbs energy (J mol**(-1))
    DrGo_C4 = -202160  # J mol**(-1)
    DrGapp_C4 = DrGo_C4 - 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C4   = np.exp(-(DrGapp_C4 + n_C4 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C4

    # Flux (mol (s * L mito)**(-1))
    J_C4 = X_C4 *(Kapp_C4**0.5 * cred_i * O2_x**0.25 - cox_i) * (1 / (1 + k_O2 /
O2_x))

    ###### F1F0-ATPase ######
    # ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

    # Gibbs energy (J mol**(-1))
    DrGo_F = 4990
    DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))

    # Apparent equilibrium constant
    Kapp_F   = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)

    ###### ANT ######
    # ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

    # Constants
    del_D   = 0.0167
    del_T   = 0.0699
    k2o_ANT = 9.54/60      # s**(-1)
    k3o_ANT = 30.05/60     # s**(-1)
    K0o_D   = 38.89e-6     # mol (L cuvette water)**(-1)
    K0o_T   = 56.05e-6     # mol (L cuvette water)**(-1)
    A       = +0.2829
    B       = -0.2086
    C       = +0.2372

    phi = F * DPsi / (R * T)

    # Reaction rates
    k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
    k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

    # Dissociation constants
    K0_D = K0o_D * np.exp(3*del_D*phi)
    K0_T = K0o_T * np.exp(4*del_T*phi)

    q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
    term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
    term2 = k3_ANT * ADP_x * ATP_c / K0_T
    num   = term1 - term2
    den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

    # Flux (mol (s * L mito)**(-1))
    J_ANT = E_ANT * num / den

    ###### H+-PI2 cotransporter ######
    # H2PO42-_x + H+_x = H2PO42-_c + H+_c

    # Constant
    k_PiC = 1.61e-3  # mol (L cuvette)**(-1)

    # H2P04- species
    HPi_c = Pi_c * (H_c / K_HPi)
    HPi_x = Pi_x * (H_x / K_HPi)

    # Flux (mol (s * L mito)**(-1))
    J_PiC = E_PiC * (H_c * HPi_c - H_x * HPi_x) / (k_PiC + HPi_c)

    ###### H+ leak ######

    # Flux (mol (s * L mito)**(-1))
    J_H = X_H * (H_c * np.exp(phi/2) - H_x * np.exp(-phi/2))

    ###### ATPase ######
    # ATP4- + H2O = ADP3- + PI2- + H+

    #Flux (mol (s * L cyto)**(-1))
    J_AtC = X_AtC / V_c
```

```
    ###### Differential equations (equation 23) ######
    # Membrane potential
    dDPsi = (n_C1 * J_C1 + n_C3 * J_C3 + n_C4 * J_C4 - n_F * J_F - J_ANT - J_H) / Cm

    # Matrix species
    dATP_x  = (J_F  - J_ANT) / W_x
    dADP_x  = (-J_F + J_ANT) / W_x
    dPi_x   = (-J_F + J_PiC) / W_x
    dNADH_x = (J_DH  - J_C1)  / W_x
    dQH2_x  = (J_C1  - J_C3)  / W_x

    # IMS species
    dcred_i = 2 * (J_C3 - J_C4) / W_i

    # Buffer species
    dATP_c = ( V_m2c * J_ANT - J_AtC ) / W_c
    dADP_c = (-V_m2c * J_ANT + J_AtC ) / W_c
    dPi_c  = (-V_m2c * J_PiC + J_AtC) / W_c

    dX = [dDPsi, dATP_x, dADP_x, dPi_x, dNADH_x, dQH2_x, dcred_i, dATP_c, dADP_c,
dPi_c]

    # Need to be able to calculate fluxes after the fact
    if solve_ode == 1:
        return dX
    else:
        J = np.array([PATP_x, PADP_x, PPi_x, PATP_c, PADP_c, PPi_c, J_DH, J_C1, J_C3,
J_C4, J_F, J_ANT, J_PiC, DrGapp_F])
        return dX, J


# Time vector
t = np.linspace(0,5,100)

# Solve ODE
results = solve_ivp(dXdt, [0, 5], X_0, method = 'Radau', t_eval=t, args=
(activity_array,1))
DPsi, sumATP_x,sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c, sumPi_c =
results.y

# Plot figures
fig, ax = plt.subplots(2,2, figsize = (10,10))
ax[0,0].plot(t, sumATP_x*1000, label = '[$\Sigma$ATP]$_x$')
ax[0,0].plot(t, sumADP_x*1000, label = '[$\Sigma$ADP]$_x$')
ax[0,0].plot(t, sumPi_x*1000, label = '[$\Sigma$Pi]$_x$')
ax[0,0].legend(loc="right")
ax[0,0].set_ylabel('Concentration (mM)')
ax[0,0].set_ylim((-.5,10.5))

ax[0,1].plot(t, sumATP_c*1000, label = '[$\Sigma$ATP]$_c$')
ax[0,1].plot(t, sumADP_c*1000, label = '[$\Sigma$ADP]$_c$')
ax[0,1].plot(t, sumPi_c*1000, label = '[$\Sigma$Pi]$_c$')
ax[0,1].legend(loc="right")
ax[0,1].set_ylabel('Concentration (mM)')
ax[0,1].set_ylim((-.5,10.5))

ax[1,0].plot(t, DPsi*1000, label = '$\Delta\Psi$')
ax[1,0].set_ylim((140, 190))
ax[1,0].legend(loc="right")
ax[1,0].set_xlabel('Time (s)')
ax[1,0].set_ylabel('Membrane potential (V)')

ax[1,1].plot(t, NADH_x/NAD_tot, label = '[NADH]$_x$/[NAD]$_{tot}$')
ax[1,1].plot(t, QH2_x/Q_tot, label = '[QH$_2$]$_x$/[Q]$_{tot}$')
ax[1,1].plot(t, cred_i/c_tot, label = '[c$^{2+}_{red}$]$_x$/[c]$_{tot}$')
ax[1,1].legend(loc="right")
ax[1,1].set_xlabel('Time (s)')
ax[1,1].set_ylabel('Relative concentration')
ax[1,1].set_ylim((0, 1))

plt.show()
```



The above simulations reach a final steady state where the phophate metabolite concentrations are $[\mathrm{Pi}]_c = 0.2$ mM and $[\mathrm{Pi}]_x = 0.4$ mM, and the membrane potential is 186 mV. This state represents a *resting* energetic state with no ATP hydrolyis in the cytosol. The Gibbs energy of ATP hydrolysis associated with this predicted state is $\Delta G_{\mathrm{ATP}} = $ -51 kJ mol^{-1}, as calculated below.

Dan - Did you want to include the ratios for NAD, Q, and c?

```
### Find Gibbs energy of ATP hydrolysis ###
dX, J_new = dXdt(t[-1],results.y[:,-1],activity_array,0)
DrGapp_F  = J_new[-1]

DrG_ATP = DrGapp_F + 8.314 * 310.15 * np.log((sumADP_x[-1] * sumPi_x[-1] /
sumATP_x[-1]))

print('Gibbs energy of ATP hydrolysis (kJ mol^(-1))')
print(DrG_ATP/1000)
```

```
Gibbs energy of ATP hydrolysis (kJ mol^(-1))
-51.07498014437706
```

Simulation of respiratory control in vitro

The integrated model developed in the previous section can be adapted to simulate in vitro experiments performed using suspensions of mitochondria purified from primary tissues. To simulate in vitro experiments, the major change that we make to the model is to replace the cytosolic components of the model with variables and associated volumes of distribution representing experimental buffers used in specific experiments. Here, we set $V_c = 1$, $V_m = $ 5e-4, and $W_c = 1$ to represent mitochondria suspended in a buffer solution (i.e., in this virtual experiment, the mitochondria take up 0.05% of the volume of the experimental media).

The following code simulates an experiment in which the non-energized mitochondrial state (state 1), is followed by a leak state (state 2), followed by oxidative phosphorylation state (state 3), followed by another leak state (state 4) [10]. The virtual experiment represents a real experiment in which a purified suspension of mitochondria is introduced to a buffer containing no fuel substrates at time $t = 0$. This non-energized state is simulated by setting the dehydrogenase activity in the model to $X_{\mathrm{DH}} = 0$ for $0 \leq t < 25$ s. State 2 is achieved by setting X_{DH} to 0.1732 for $25 \leq t < 75$ s to represent the addision of substrates (such as pyruvate and malate) to fuel NADH production. At time $t = 75$ s, 0.375 μM of ADP is introduced into the respiration buffer to initiate the active oxidative phosphorylation state, or state 3. State 4 occurs when the ADP is nearly fully phosphorylated at approximately $t = 170$ s, and the system returns to a leak state. Note that this experiment is conducted under high external inorganic phosphate conditions (i.e., $[\mathrm{Pi}]_c = 5$ mM).

Parameter or Condition	State 1	State 2	State 3 and 4
X_{DH}	0	0.1732	–
$[\mathrm{NADH}]_x$	0	–	–
$[\mathrm{ATP}]_c$	0	–	–
$[\mathrm{ADP}]_c$	0	–	+375 μM
$[\mathrm{Pi}]_c$	5	–	–

Table 5 Isolated mitochondria in silico experiment.

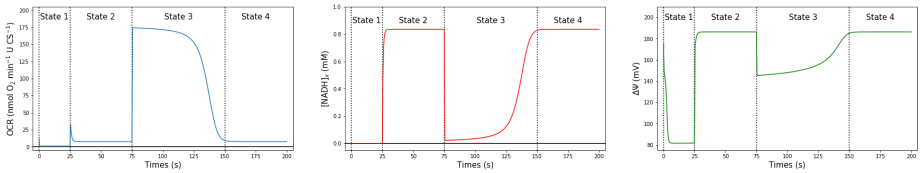
```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 1.0          # buffer volume fraction        # L buffer (L cuvette)**(-1)
V_m = 0.0005       # mitochondrial volume fraction # L mito (L cuvette)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cuvette)**(-1)
W_c = 1.0          # buffer water space            # L buffer water (L buffer)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Total pool concentrations
NAD_tot = 2.97e-3  # NAD+ and NADH conc            # mol (L matrix water)**(-1)
Q_tot   = 1.35e-3  # Q and QH2 conc                # mol (L matrix water)**(-1)
c_tot   = 2.7e-3   # cytochrome c ox and red conc  # mol (L IM water)**(-1)

# Membrane capacitance
Cm = 3.1e-3

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3      # mol (L matrix water)**(-1)
K_c  = 140e-3      # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3        # mol (L matrix water)**(-1)
Mg_c = 1.0e-3        # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25 # mmHg

###### Parameter vector ######
X_DH  = 0.1732
X_C1  = 1.0e4
X_C3  = 1.0e6
X_C4  = 0.0125
X_F   = 1.0e3
E_ANT = 0.325
E_PiC = 5.0e6
X_H   = 1.0e3
X_AtC = 0

activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC])

###### Initial Conditions ######
# Membrane Potential
DPsi_0 = 175/1000      # V

# Matrix species
sumATP_x_0 = 0.5e-3        # mol (L matrix water)**(-1)
sumADP_x_0 = 9.5e-3        # mol (L matrix water)**(-1)
sumPi_x_0  = 0.3e-3        # mol (L matrix water)**(-1)
NADH_x_0   = 0             # mol (L matrix water)**(-1)
QH2_x_0    = 0.1 * Q_tot   # mol (L matrix water)**(-1)

# IMS species
cred_i_0 = 0.1 * c_tot # mol (L IMS water)**(-1)

# Cytosolic species
sumATP_c_0 = 0       # mol (L cyto water)^(-1)
sumADP_c_0 = 0       # mol (L cyto water)^(-1)
sumPi_c_0  = 5.0e-3  # mol (L cyto water)^(-1)

X_0 = np.array([DPsi_0, sumATP_x_0, sumADP_x_0, sumPi_x_0, NADH_x_0, QH2_x_0,
cred_i_0, sumATP_c_0, sumADP_c_0, sumPi_c_0])

def dXdt(t, X, activity_array, solve_ode):
    # Unpack variables
    DPsi, sumATP_x,sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c,
sumPi_c = X
    X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC = activity_array

    # Hydrogen ion concentration
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)
```

```
    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3
    n_C1 = 4
    n_C3 = 2
    n_C4 = 4

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
    K_KADP  = 10**(-0.89)
    K_MgPi  = 10**(-1.66)
    K_HPi   = 10**(-6.62)
    K_KPi   = 10**(-0.42)

    # Other concentrations computed from the state variables
    NAD_x = NAD_tot - NADH_x  ## mol (L matrix water)**(-1)
    Q_x   = Q_tot - QH2_x     ## mol (L matrix water)**(-1)
    cox_i = c_tot - cred_i    ## mol (L matrix water)**(-1)

    ## Binding polynomials
    # Matrix species # mol (L mito water)**(-1)
    PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
    PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
    PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

    # Cytosol species # mol (L cuvette water)**(-1)
    PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
    PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
    PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

    ## Unbound species
    # Matrix species
    ATP_x = sumATP_x / PATP_x # [ATP4-]_x
    ADP_x = sumADP_x / PADP_x # [ADP3-]_x
    Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

    # Cytosolic species
    ATP_c = sumATP_c / PATP_c # [ATP4-]_c
    ADP_c = sumADP_c / PADP_c # [ADP3-]_c
    Pi_c  = sumPi_c  / PPi_c  # [HPO42-]_c


    ###### NADH Dehydrogenase ######

    # Constants
    r      = 6.8385
    k_Pi1  = 4.659e-4    # mol (L matrix water)**(-1)
    k_Pi2  = 6.578e-4    # mol (L matrix water)**(-1)

    # Flux (mol (s * L mito)**(-1))
    J_DH = X_DH * (r * NAD_x - NADH_x) * ((1 + sumPi_x / k_Pi1) / (1+sumPi_x / k_Pi2))

    ###### Complex I ######
    # NADH_x + Q_x + 5H+_x <-> NAD+_x + QH2_x + 4H+_i + 4DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C1 = -109680
    DrGapp_C1 = DrGo_C1 - R * T * np.log(H_x)

    # Apparent equilibrium constant
    Kapp_C1   = np.exp( -(DrGapp_C1 + n_C1 * F * DPsi) / (R * T)) * ((H_x /
H_c)**n_C1)

    # Flux (mol (s * L mito)**(-1))
    J_C1 = X_C1 * (Kapp_C1 * NADH_x * Q_x - NAD_x * QH2_x)

    ###### Complex III ######
    # QH2_x + 2cuvetteC(ox)3+_i + 2H+_x <-> Q_x + 2cuvetteC(red)2+_i + 4H+_i + 2DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C3 = 46690
    DrGapp_C3 = DrGo_C3 + 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C3   = np.exp(-(DrGapp_C3 + n_C3 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C3

    # Flux (mol (s * L mito)**(-1))
    J_C3 = X_C3 * (Kapp_C3 * cox_i**2 * QH2_x - cred_i**2 * Q_x)
```

```
    ###### Complex IV ######
    # 2 cytoC(red)2+_i + 0.5O2_x + 4H+_x <-> cytoC(ox)3+_x + H2O_x + 2H+_i +2DPsi

    # Constant
    k_O2 = 1.2e-4      # mol (L matrix water)**(-1)

    # Gibbs energy (J mol**(-1))
    DrGo_C4 = -202160  # J mol**(-1)
    DrGapp_C4 = DrGo_C4 - 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C4   = np.exp(-(DrGapp_C4 + n_C4 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C4

    # Flux (mol (s * L mito)**(-1))
    J_C4 = X_C4 *(Kapp_C4**0.5 * cred_i * O2_x**0.25 - cox_i) * (1 / (1 + k_O2 /
O2_x))

    ###### F0F1-ATPase ######
    # ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

    # Gibbs energy (J mol**(-1))
    DrGo_F = 4990
    DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))

    # Apparent equilibrium constant
    Kapp_F   = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)

    ###### ANT ######
    # ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

    # Constants
    del_D   = 0.0167
    del_T   = 0.0699
    k2o_ANT = 9.54/60      # s**(-1)
    k3o_ANT = 30.05/60     # s**(-1)
    K0o_D   = 38.89e-6     # mol (L cuvette water)**(-1)
    K0o_T   = 56.05e-6     # mol (L cuvette water)**(-1)
    A       = +0.2829
    B       = -0.2086
    C       = +0.2372

    phi = F * DPsi / (R * T)

    # Reaction rates
    k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
    k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

    # Dissociation constants
    K0_D = K0o_D * np.exp(3*del_D*phi)
    K0_T = K0o_T * np.exp(4*del_T*phi)

    q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
    term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
    term2 = k3_ANT * ADP_x * ATP_c / K0_T
    num   = term1 - term2
    den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

    # Flux (mol (s * L mito)**(-1))
    J_ANT = E_ANT * num / den

    ###### H+-PI2 cotransporter ######
    # H2PO42-_x + H+_x = H2PO42-_c + H+_c

    # Constant
    k_PiC = 1.61e-3  # mol (L cuvette)**(-1)

    # H2P04- species
    HPi_c = Pi_c * (H_c / K_HPi)
    HPi_x = Pi_x * (H_x / K_HPi)

    # Flux (mol (s * L mito)**(-1))
    J_PiC = E_PiC * (H_c * HPi_c - H_x * HPi_x) / (k_PiC + HPi_c)

    ###### H+ leak ######

    # Flux (mol (s * L mito)**(-1))
    J_H = X_H * (H_c * np.exp(phi/2) - H_x * np.exp(-phi/2))

    ###### ATPase ######
    # ATP4- + H2O = ADP3- + PI2- + H+

    #Flux (mol (s * L cyto)**(-1))
    J_AtC = X_AtC / V_c
```

```
    ###### Differential equations (equation 23) ######
    # Membrane potential
    dDPsi = (n_C1 * J_C1 + n_C3 * J_C3 + n_C4 * J_C4 - n_F * J_F - J_ANT - J_H) / Cm

    # Matrix species
    dATP_x  = (J_F  - J_ANT) / W_x
    dADP_x  = (-J_F + J_ANT) / W_x
    dPi_x   = (-J_F + J_PiC) / W_x
    dNADH_x = (J_DH  - J_C1)  / W_x
    dQH2_x  = (J_C1  - J_C3)  / W_x

    # IMS species
    dcred_i = 2 * (J_C3 - J_C4) / W_i

    # Buffer species
    dATP_c = ( V_m2c * J_ANT - J_AtC ) / W_c
    dADP_c = (-V_m2c * J_ANT + J_AtC ) / W_c
    dPi_c  = (-V_m2c * J_PiC + J_AtC) / W_c

    dX = [dDPsi, dATP_x, dADP_x, dPi_x, dNADH_x, dQH2_x, dcred_i, dATP_c, dADP_c,
dPi_c]

    # Need to be able to calculate fluxes after the fact
    if solve_ode == 1:
        return dX
    else:
        J = np.array([PATP_x, PADP_x, PPi_x, PATP_c, PADP_c, PPi_c, J_DH, J_C1, J_C3,
J_C4, J_F, J_ANT, J_PiC])
        return dX, J

### Four State Model ###
# State 1 - no substrates
t_1 = np.linspace(0,25,25*2)
X_AtC = 0.
X_DH = 0.001 # Kept non-zero for solver stability
activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC])
state_1_results = solve_ivp(dXdt, [0,25], X_0, method = 'Radau', t_eval = t_1, args=
(activity_array,1,))

# State 2 - Add substrate (i.e. turn on X_DH)
t_2 = np.linspace(25,75,(75 - 25)*2)
X_DH = 0.0866 * 2
activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC])
state_2_results = solve_ivp(dXdt, [25,75], state_1_results.y[:,-1], method = 'Radau',
t_eval = t_2, args=(activity_array,1,))

# State 3 and 4 - Add ADP
t_3 = np.linspace(75,200,(200 - 75)*2)
state_2_results.y[8, -1] = 0.375e-3 # Molar
state_3_results = solve_ivp(dXdt, [75,200], state_2_results.y[:,-1], method = 'Radau',
t_eval = t_3, args=(activity_array,1,))

# Concatenate Results
# Note: need to prevent duplicating points
all_results = np.hstack((state_1_results.y[:,0:-1],
state_2_results.y[:,0:-1],state_3_results.y))
t = np.concatenate((state_1_results.t[0:-1], state_2_results.t[0:-1],
state_3_results.t))

DPsi, sumATP_x, sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c, sumPi_c
= all_results


# Calculate complex IV Flux
J_C4 = np.zeros(len(t))
for i in range(len(t)):
    dX, J = dXdt(t[i], all_results[:,i], activity_array, 0)
    J_C4[i] = J[9]

# Convert complex IV flux to oxygen flux in nmol O2 / U Citrate Synthase
JO2 = J_C4/2 * 60 * 1e9 * 0.0000012232
```

```
fig, ax = plt.subplots(1,3, figsize = (30,5))

ax[0].plot((-5,205),(0,0),'k')
ax[0].plot((0,0),(-5,205),'k:', linewidth = 1.75)
ax[0].plot((25,25),(-5,205),'k:', linewidth = 1.75)
ax[0].plot((75,75),(-5,205),'k:', linewidth = 1.75)
ax[0].plot((150,150),(-5,205),'k:', linewidth = 1.75)
ax[0].plot(t, JO2)
ax[0].set_ylim([-5,205])
ax[0].set_xlim([-5,205])
ax[0].text(12.5, 190, 'State 1', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[0].text(50, 190, 'State 2', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[0].text(112.5, 190, 'State 3', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[0].text(175, 190, 'State 4', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[0].set_xlabel('Times (s)', fontsize = 15)
ax[0].set_ylabel('OCR (nmol O$_2$ min$^{-1}$ U CS$^{-1}$)', fontsize = 15)

ax[1].plot((-5,205),(0,0),'k')
ax[1].plot((0,0),(-0.05,1),'k:', linewidth = 1.75)
ax[1].plot((25,25),(-0.05,1),'k:', linewidth = 1.75)
ax[1].plot((75,75),(-0.05,1),'k:', linewidth = 1.75)
ax[1].plot((150,150),(-0.05,1),'k:', linewidth = 1.75)
ax[1].plot(t, NADH_x/NAD_tot,'r')
ax[1].set_ylim([-0.05,1])
ax[1].set_xlim([-5,205])
ax[1].text(12.5, 0.9, 'State 1', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[1].text(50, .9, 'State 2', horizontalalignment='center', verticalalignment='center',
fontsize = 15)
ax[1].text(112.5, .9, 'State 3', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[1].text(175, .9, 'State 4', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[1].set_xlabel('Times (s)', fontsize = 15)
ax[1].set_ylabel('[NADH]$_x$ (mM)', fontsize = 15)

ax[2].plot((0,0),(75,210),'k:', linewidth = 1.75)
ax[2].plot((25,25),(75,210),'k:', linewidth = 1.75)
ax[2].plot((75,75),(75,210),'k:', linewidth = 1.75)
ax[2].plot((150,150),(75,210),'k:', linewidth = 1.75)
ax[2].plot(t, DPsi*1000,'g')
ax[2].set_ylim([75,210])
ax[2].set_xlim([-5,205])
ax[2].text(12.5, 200, 'State 1', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[2].text(50, 200, 'State 2', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[2].text(112.5, 200, 'State 3', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[2].text(175, 200, 'State 4', horizontalalignment='center',
verticalalignment='center', fontsize = 15)
ax[2].set_xlabel('Times (s)', fontsize = 15)
ax[2].set_ylabel('$\Delta \Psi$ (mV)', fontsize = 15)

plt.show()
```



During the leak states, NADH and $\Delta\Psi$ are maintained at a maximal values while the OCR represents the oxidative flux necessary to balance proton leak across the IMM. During state 3 (oxidative phosphorylation), the OCR markedly increases while the magnitudes of NADH and $\Delta\Psi$ are temporarily decreased. This simple model overestimates the degree to which the NADH becomes depleted during state 3.

Predictions of the in vitro model may also be compared to quasi-steady state data on OCRs, phosphate metabolite levels, $\Delta\Psi$, NADH, and cytochrome c redox state, analyzed by Bazil et al. [15]. In their experiments, several steady state ATP demand levels were achieved by titrating ATP hydrolyzing enzymes into the respiration buffer. We simulate this experiment with our model by varying the rate of the external ATP consuming process, J_{AtC}. We vary J_{AtC} by adusting the ATP consumption rate X_{AtC} from 0 to $60\ \mathrm{mol\ s^{-1}\ (L\ cell)^{-1}}$. Experiments were conducted under conditions of two different approximately constant external phosphate concentrations (1 and 5 mM) and with external $\mathrm{TAN} = 5\ \mathrm{mM}$. The following code computes the steady state of the in vitro model at different levels of J_{AtC} and compares the measured data on OCR, $\Delta\Psi$, NADH, cytochrome c redox state, and buffer ADP concentration to data.

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 1.0          # buffer volume fraction        # L buffer (L cuvette)**(-1)
V_m = 0.0005       # mitochondrial volume fraction # L mito (L cuvette)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cuvette)**(-1)
W_c = 1.0          # buffer water space            # L buffer water (L buffer)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Total pool concentrations
NAD_tot = 2.97e-3  # NAD+ and NADH conc            # mol (L matrix water)**(-1)
Q_tot   = 1.35e-3  # Q and QH2 conc                # mol (L matrix water)**(-1)
c_tot   = 2.7e-3   # cytochrome c ox and red conc  # mol (L IM water)**(-1)

# Membrane capacitance
Cm = 3.1e-3

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3      # mol (L matrix water)**(-1)
K_c  = 140e-3      # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3        # mol (L matrix water)**(-1)
Mg_c = 1.0e-3        # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25 # mmHg

conc = np.array([pH_x, pH_c, K_x, K_c, Mg_x, Mg_c, PO2])

###### Parameter vector ######
X_DH  = 0.1732
X_C1  = 1.0e4
X_C3  = 1.0e6
X_C4  = 0.0125
X_F   = 1.0e3
E_ANT = 0.325
E_PiC = 5.0e6
X_H   = 1.0e3
X_AtC = 0

activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC])

###### Initial Conditions ######
# Membrane Potential
Psi_0 = 175/1000      # Volts

# Total Pool Sizes
NAD_tot = 2.97e-3  # NAD+ and NADH conc            # mol (L matrix water)^(-1)
Q_tot   = 1.35e-3  # Q and QH2 conc               # mol (L matrix water)^(-1)
c_tot   = 2.7e-3

# Matrix species
sumATP_x_0 = 0.5e-3  # mol (L matrix water)**(-1)
sumADP_x_0 = 9.5e-3  # mol (L matrix water)**(-1)
sumPi_x_0  = 0.3e-3  # mol (L matrix water)**(-1)
NADH_x_0   = 0.1 * NAD_tot  # mol (L matrix water)**(-1)
QH2_x_0    = 0.1 * Q_tot    # mol (L matrix water)**(-1)

# IMS species
cred_i_0 = 0.1 * c_tot # mol (L IMS water)**(-1)

# Cytosoo=lic species
sumATP_c_0 = 0       # mol (L cyto water)^(-1)
sumADP_c_0 = 0       # mol (L cyto water)^(-1)
sumPi_c_0  = 5.0e-3  # mol (L cyto water)^(-1)

X_0 = np.array([Psi_0, sumATP_x_0, sumADP_x_0, sumPi_x_0, NADH_x_0, QH2_x_0, cred_i_0,
sumATP_c_0, sumADP_c_0, sumPi_c_0])

def dXdt(t, X, activity_array, solve_ode):
    # Unpack variables
    DPsi, sumATP_x,sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c,
sumPi_c = X
    X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_AtC = activity_array

    # Hydrogen ion concentration
```

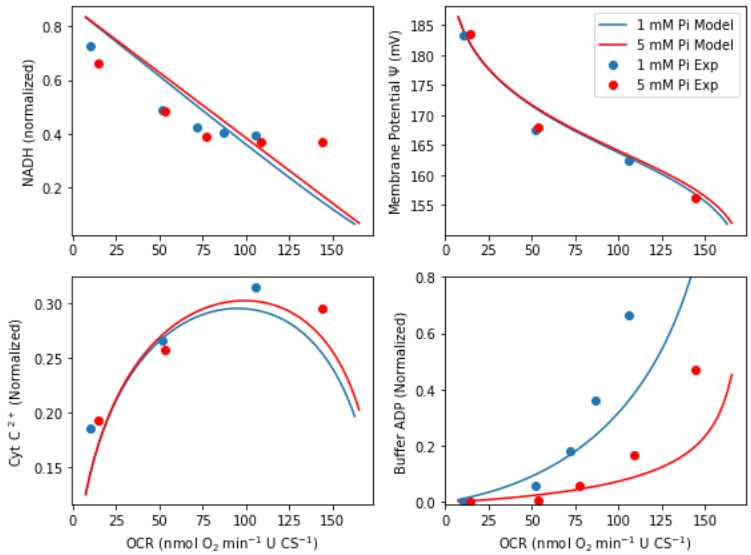
```
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)

    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3
    n_C1 = 4
    n_C3 = 2
    n_C4 = 4

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
    K_KADP  = 10**(-0.89)
    K_MgPi  = 10**(-1.66)
    K_HPi   = 10**(-6.62)
    K_KPi   = 10**(-0.42)

    ## Other concentrations computed from the state variables:
    NAD_x = NAD_tot - NADH_x  ## mol (L matrix water)**(-1)
    Q_x   = Q_tot - QH2_x     ## mol (L matrix water)**(-1)
    cox_i = c_tot - cred_i    ## mol (L matrix water)**(-1)

    ## Binding polynomials
    # Matrix species # mol (L mito water)**(-1)
    PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
    PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
    PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

    # Cytosol species # mol (L cuvette water)**(-1)
    PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
    PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
    PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

    ## Unbound species
    # Matrix species
    ATP_x = sumATP_x / PATP_x # [ATP4-]_x
    ADP_x = sumADP_x / PADP_x # [ADP3-]_x
    Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

    # Cytosol species
    ATP_c = sumATP_c / PATP_c # [ATP4-]_c
    ADP_c = sumADP_c / PADP_c # [ADP3-]_c
    Pi_c  = sumPi_c  / PPi_c  # [HPO42-]_c


    ###### NADH Dehydrogenase ######

    # Constants
    r      = 6.8385
    k_Pi1  = 4.659e-4    # mol (L matrix water)**(-1)
    k_Pi2  = 6.578e-4    # mol (L matrix water)**(-1)

    # Flux
    J_DH = X_DH * (r * NAD_x - NADH_x) * ((1 + sumPi_x / k_Pi1) / (1+sumPi_x / k_Pi2))

    ###### Complex I ######
    # NADH_x + Q_x + 5H+_x <-> NAD+_x + QH2_x + 4H+_i + 4DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C1 = -109680
    DrGapp_C1 = DrGo_C1 - R * T * np.log(H_x)

    # Apparent equilibrium constant
    Kapp_C1   = np.exp( -(DrGapp_C1 + n_C1 * F * DPsi) / (R * T)) * ((H_x /
H_c)**n_C1)

    # Flux (mol (s * L mito)**(-1))
    J_C1 = X_C1 * (Kapp_C1 * NADH_x * Q_x - NAD_x * QH2_x)

    ###### Complex III ######
    # QH2_x + 2cuvetteC(ox)3+_i + 2H+_x <-> Q_x + 2cuvetteC(red)2+_i + 4H+_i + 2DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C3 = 46690
```

```
    DrGapp_C3 = DrGo_C3 + 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C3   = np.exp(-(DrGapp_C3 + n_C3 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C3

    # Flux (mol (s * L mito)**(-1))
    J_C3 = X_C3 * (Kapp_C3 * cox_i**2 * QH2_x - cred_i**2 * Q_x)

    ###### Complex IV ######
    # 2 cytoC(red)2+_i + 0.5O2_x + 4H+_x <-> cytoC(ox)3+_x + H2O_x + 2H+_i + 2DPsi

    # Constants
    k_O2 = 1.2e-4      # mol (L matrix water)**(-1)

    # Gibbs energy (J mol**(-1))
    DrGo_C4 = -202160  # J mol**(-1)
    DrGapp_C4 = DrGo_C4 - 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C4   = np.exp(-(DrGapp_C4 + n_C4 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C4

    # Flux (mol (s * L mito)**(-1))
    J_C4 = X_C4 *(Kapp_C4**0.5 * cred_i * O2_x**0.25 - cox_i) * (1 / (1 + k_O2 /
O2_x))

    ###### F0F1-ATPase ######
    # ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

    # Gibbs energy (J mol**(-1))
    DrGo_F = 4990
    DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))

    # Apparent equilibrium constant
    Kapp_F   = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)

    ###### ANT ######
    # ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

    # Constants
    del_D   = 0.0167
    del_T   = 0.0699
    k2o_ANT = 9.54/60      # s**(-1)
    k3o_ANT = 30.05/60     # s**(-1)
    K0o_D   = 38.89e-6     # mol (L cuvette water)**(-1)
    K0o_T   = 56.05e-6     # mol (L cuvette water)**(-1)
    A       = +0.2829
    B       = -0.2086
    C       = +0.2372

    phi = F * DPsi / (R * T)

    # Reaction rates
    k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
    k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

    # Dissociation constants
    K0_D = K0o_D * np.exp(3*del_D*phi)
    K0_T = K0o_T * np.exp(4*del_T*phi)

    q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
    term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
    term2 = k3_ANT * ADP_x * ATP_c / K0_T
    num   = term1 - term2
    den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

    # Flux (mol (s * L mito)**(-1))
    J_ANT = E_ANT * num / den

    ###### H+-PI2 cotransporter ######
    # H2PO42-_x + H+_x = H2PO42-_c + H+_c

    # Constant
    k_PiC = 1.61e-3  # mol (L cuvette)**(-1)

    # H2P04- species
    HPi_c = Pi_c * (H_c / K_HPi)
    HPi_x = Pi_x * (H_x / K_HPi)

    # Flux (mol (s * L mito)**(-1))
    J_PiC = E_PiC * (H_c * HPi_c - H_x * HPi_x) / (k_PiC + HPi_c)

    ###### H+ leak ######

    # Flux (mol (s * L mito)**(-1))
```

```
    J_H = X_H * (H_c * np.exp(phi/2) - H_x * np.exp(-phi/2))

    ###### ATPase ######
    # ATP4- + H2O = ADP3- + PI2- + H+

    #Flux (mol (s * L cyto)**(-1))
    J_AtC = X_AtC / V_c

    ###### Differential equations (equation 23) ######
    # Membrane potential
    dDPsi = (n_C1 * J_C1 + n_C3 * J_C3 + n_C4 * J_C4 - n_F * J_F - J_ANT - J_H) / Cm

    # Matrix species
    dATP_x  = (J_F  - J_ANT) / W_x
    dADP_x  = (-J_F + J_ANT) / W_x
    dPi_x   = (-J_F + J_PiC) / W_x
    dNADH_x = (J_DH  - J_C1)  / W_x
    dQH2_x  = (J_C1  - J_C3)  / W_x

    # IMS species
    dcred_i = 2 * (J_C3 - J_C4) / W_i

    # Buffer species
    dATP_c = ( V_m2c * J_ANT - J_AtC ) / W_c
    dADP_c = (-V_m2c * J_ANT + J_AtC ) / W_c
    dPi_c  = (-V_m2c * J_PiC + J_AtC) / W_c

    dX = [dDPsi, dATP_x, dADP_x, dPi_x, dNADH_x, dQH2_x, dcred_i, dATP_c, dADP_c,
dPi_c]
    if solve_ode == 1:
        return dX
    else:
        J = np.array([PATP_x, PADP_x, PPi_x, PATP_c, PADP_c, PPi_c, J_DH, J_C1, J_C3,
J_C4, J_F, J_ANT, J_PiC])
        return dX, J


###### Run Low Pi experiments ######

# Membrane potential
DPsi_0 = 175*1e-3

# Matrix species
ATP_x_0  = 0.5e-3
ADP_x_0  = 9.5e-3
Pi_x_0   = 0.3e-3
NADH_x_0 = 0.1 * NAD_tot
QH2_x_0  = 0.1 * Q_tot

# IMS species
cred_i_0 = 0.1 * c_tot

# Cytosol species
ATP_c_0 = 5.0e-3
ADP_c_0 = 0.0e-3
Pi_c_0  = 1.0e-3

X_0 = np.array([DPsi_0, ATP_x_0, ADP_x_0, Pi_x_0, NADH_x_0, QH2_x_0, cred_i_0,
ATP_c_0, ADP_c_0, Pi_c_0 ])

# range of ATP consumption rates
X_AtC = np.linspace(0,6e-6, 60)   # Increase max hydrolysis to find apparent Km.
steady_state = np.zeros((len(X_AtC), len(X_0)))
JO2 = np.zeros(len(X_AtC))

# looping through different ATP consumptions states
for i in range(len(X_AtC)):
   activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H,
X_AtC[i]])
   # run for long time to acheive steady-state
   steady_state_temp_results = solve_ivp(dXdt, [0,3000], X_0, method = 'Radau', args=
(activity_array,1,)).y[:,-1]
   steady_state[i] = steady_state_temp_results
   f,J = dXdt(30,steady_state_temp_results, activity_array, 0)
   J_C4 = J[9] # oxygen flux in mol O / sec / (L mito)

   # convert to units of nmol / min / UCS
   # using the conversion factor 0.0012232 mL of mito per UCS
   JO2[i] = J_C4/2 * 60 * 1e9 * 0.0000012232

DPsi_low_pi, ATP_x_low_pi, ADP_x_low_pi, Pi_x_low_pi, NADH_x_low_pi, QH2_x_low_pi,
cred_i_low_pi, ATP_c_low_pi, ADP_c_low_pi, Pi_c_low_pi = steady_state.T
fig, ax = plt.subplots(2,2, figsize = (8,6))

# Low Pi Plotting  Plotting
ax[0,0].plot(JO2, NADH_x_low_pi/NAD_tot, 'C0')   # NADH
ax[0,1].plot(JO2, DPsi_low_pi * 1000, 'C0', label = '1 mM Pi Model')      # Membrane
```

```
Potential
ax[1,0].plot(JO2, cred_i_low_pi/c_tot,'C0')      # Cytochrome C
ax[1,1].plot(JO2, ADP_c_low_pi * 1000, 'C0')     # ADP

###### Running High Pi experiments ######

# Membrane potential
DPsi_0 = 175*1e-3

# Matrix species
ATP_x_0  = 0.5e-3
ADP_x_0  = 9.5e-3
Pi_x_0   = 0.3e-3
NADH_x_0 = 0.1 * NAD_tot
QH2_x_0  = 0.1 * Q_tot

# IM species
cred_i_0 = 0.1 * c_tot

# Cytosol species
ATP_c_0 = 5.0e-3
ADP_c_0 = 0.0e-3
Pi_c_0  = 5.0e-3

X_0 = np.array([DPsi_0, ATP_x_0, ADP_x_0, Pi_x_0, NADH_x_0, QH2_x_0, cred_i_0,
ATP_c_0, ADP_c_0, Pi_c_0 ])

# range of ATP consumption rates
X_AtC = np.linspace(0,6e-6, 60)   # Increase max hydrolysis to find apparent Km.
steady_state = np.zeros((len(X_AtC), len(X_0)))
JO2 = np.zeros(len(X_AtC))

# looping through different ATP consumptions states
for i in range(len(X_AtC)):
   activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H,
X_AtC[i]])
   # run for long time to acheive steady-state
   steady_state_temp_results = solve_ivp(dXdt, [0,3000], X_0, method = 'Radau', args=
(activity_array,1,)).y[:,-1]
   steady_state[i] = steady_state_temp_results
   f,J = dXdt(30,steady_state_temp_results, activity_array, 0)
   J_C4 = J[9] # oxygen flux in mol O / sec / (L mito)

   # convert to units of nmol / min / UCS
   # using the conversion factor 0.0012232 mL of mito per UCS
   JO2[i] = J_C4/2 * 60 * 1e9 * 0.0000012232

DPsi_hi_pi, ATP_x_hi_pi, ADP_x_hi_pi, Pi_x_hi_pi, NADH_x_hi_pi, QH2_x_hi_pi,
cred_i_hi_pi, ATP_c_hi_pi, ADP_c_hi_pi, Pi_c_hi_pi = steady_state.T

# High Pi Plotting  Plotting
ax[0,0].plot(JO2, NADH_x_hi_pi/NAD_tot, 'red')   # NADH
ax[0,1].plot(JO2, DPsi_hi_pi * 1000, 'red', label = '5 mM Pi Model')      # Membrane
Potential
ax[1,0].plot(JO2, cred_i_hi_pi/c_tot,'red')      # Cytochrome C
ax[1,1].plot(JO2, ADP_c_hi_pi * 1000,  'red')     # ADP
ax[1,1].set_ylim([-0.01,0.8])

# Plot experimental data

# NADH data
data_NADH = [10.494,      0.7284,               14.509, 0.6642,
       51.927,        0.4879,               53.625, 0.4821,
       71.967,        0.4241,               77.07,    0.3901,
       86.95,          0.4063,               108.57, 0.3688,
       105.62,        0.3964,               144.36, 0.3695]
data_NADH = np.reshape(data_NADH,(5,4))
ax[0,0].plot(data_NADH[:,0], data_NADH[:,1], 'o')
ax[0,0].plot(data_NADH[:,2], data_NADH[:,3], 'o', color = 'red')
ax[0,0].set_ylabel('NADH (normalized)')


# Membrane potential data
data_DPsi = [10.494, 183.2317,                        14.509, 183.4706,
        51.927,       167.5529,                        53.625, 167.9643,
        105.62,       162.3817,                        144.36, 156.2841]

data_DPsi = np.reshape(data_DPsi,(3,4))
ax[0,1].plot(data_DPsi[:,0], data_DPsi[:,1], 'o', label = '1 mM Pi Exp')
ax[0,1].plot(data_DPsi[:,2], data_DPsi[:,3], 'o', color = 'red', label = '5 mM Pi
Exp')
ax[0,1].set_ylabel('Membrane Potential $\Psi$ (mV)')


# Cytochrome C
data_cred = [10.494, 0.1862,               14.509, 0.1931,
        51.927,       0.2658,               53.625, 0.2576,
```

```
        105.62,       0.3147,               144.36, 0.2959];

data_cred = np.reshape(data_cred,(3,4))
ax[1,0].plot(data_cred[:,0], data_cred[:,1], 'o')
ax[1,0].plot(data_cred[:,2], data_cred[:,3], 'o', color = 'red')
ax[1,0].set_xlabel('OCR (nmol O$_2$ min$^{-1}$ U CS$^{-1}$)')
ax[1,0].set_ylabel('Cyt C $^{2+}$ (Normalized)')

# Buffer ADP
data_ADP = [10.494, 0, 14.509, 0,
            51.927, 0.0561, 53.625, 0.0049,
            71.967, 0.1827, 77.07,    0.0578,
            86.95, 0.3589, 108.57, 0.1648,
            105.62, 0.6622, 144.36, 0.4716]

data_ADP = np.reshape(data_ADP, (5, 4))
ax[1, 1].plot(data_ADP[:, 0], data_ADP[:, 1],  'o')
ax[1, 1].plot(data_ADP[:, 2], data_ADP[:, 3],  'o', color='red')
ax[1, 1].set_xlabel('OCR (nmol O$_2$ min$^{-1}$ U CS$^{-1}$)')
ax[1, 1].set_ylabel('Buffer ADP (Normalized)')
plt.tight_layout()
ax[0,1].legend()
plt.show()
```



The match to the data is remarkably good considering how much simpler the model developed here is compared to the model of Bazil et al. [15]. The Bazil et al. model accounts for the mechanisms underlying oxidative phosphorylation and associated side reactions at a deeper level of detail, and thus, finds broader applications. Yet the basic frameworks of our models are equivalent.

Simulation of cardiac energetics in vivo

In this final section we apply the developed model, which is shown to match data on the relationships between ATP synthesis and phosphate metabolite levels in vivo predicting the nature of those relationships in vivo.

Creatine kinase

To use the mitochondrial model developed above to simulate energetics in muscle in vivo, we must account for the cytosolic creatine kinase reaction

$$(\mathrm{ADP}^{3-})_c + (\mathrm{CrP}^{2-})_c + (\mathrm{H}^{+})_c \rightleftharpoons (\mathrm{ATP}^{4-})_c + (\mathrm{Cr})_c, \tag{24}$$

where Cr denotes creatine and CrP creatine phosphate. The total cytosolic creatine pool, $[\mathrm{Cr}]_{tot}$, is conserved, that is,

$$[\mathrm{Cr}]_{tot} = [\mathrm{Cr}]_c + [\mathrm{CrP}]_c.$$

To determine the effective Gibbs energy of the creatine kinase reaction in terms of measurable biochemical reactants, we substitute Equations (6) and (7) obtaining

$$\begin{aligned}\Delta_r G_{\mathrm{CK}} &= \Delta_r G^{\circ}_{\mathrm{CK}} + RT\ln\left(\frac{[\Sigma\mathrm{ATP}]_c[\mathrm{Cr}]_c}{[\Sigma\mathrm{ADP}]_c[\mathrm{CrP}^{2-}]_c}\cdot\frac{P_{\mathrm{ADP}}}{P_{\mathrm{ATP}}[\mathrm{H}^+]_c}\right)\\ &= \Delta_r G'^{\circ}_{\mathrm{CK}} + RT\ln\left(\frac{[\Sigma\mathrm{ATP}]_c[\mathrm{Cr}]_c}{[\Sigma\mathrm{ADP}]_c[\mathrm{CrP}^{2-}]_c}\right),\end{aligned}$$

where

$$\Delta_r G'^{\circ}_{\mathrm{CK}} = \Delta_r G^{\circ}_{\mathrm{CK}} + RT\ln\left(\frac{P_{\mathrm{ADP}}}{P_{\mathrm{ATP}}[\mathrm{H}^+]_c}\right).$$

Here, $\Delta_r G^{\circ}_{\mathrm{CK}} = -RT\ln(K_{eq,\mathrm{CK}})$ for experimental equilibrium constant $K_{eq,\mathrm{CK}} = 3.5\times10^8$ [2]. The apparent equilibrium of Equation (24) is

$$K'_{eq,\mathrm{CK}} = \left(\frac{[\Sigma\mathrm{ATP}]_c[\mathrm{Cr}]_c}{[\Sigma\mathrm{ADP}]_c[\mathrm{CrP}^{2-}]_c}\right)_{eq} = \exp\left\{\frac{-\Delta_r G'^{\circ}_{\mathrm{CK}}}{RT}\right\}.$$

We simulate creatine kinase flux, J_{CK} (mol s^{-1} (L mito)$^{-1}$), via mass-action kinetics as

$$J_{\mathrm{CK}} = X_{\mathrm{CK}}(K'_{eq,\mathrm{CK}}[\Sigma\mathrm{ADP}]_c[\mathrm{CrP}]_c - [\mathrm{ATP}]_c[\mathrm{Cr}]_c) \tag{25}$$

where X_{CK} (mol s^{-1} (L cyto)$^{-1}$) is the creatine kinase activity.

To simulate cardiac energetics and in vivo experiments, we incorporate the creatine kinase module and obtain the following system:

$$\begin{cases}
\dfrac{\mathrm{d}\Delta\Psi}{\mathrm{d}t} = (n_{\mathrm{C1}}J_{\mathrm{C1}} + n_{\mathrm{C3}}J_{\mathrm{C3}} + n_{\mathrm{C4}}J_{\mathrm{C4}} - n_{\mathrm{F}}J_{\mathrm{F}} - J_{\mathrm{ANT}} - J_{\mathrm{H}})/C_m \\
\hline
\dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_x}{\mathrm{d}t} = (J_{\mathrm{F}} - J_{\mathrm{ANT}})/W_x \\
\dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_x}{\mathrm{d}t} = (-J_{\mathrm{F}} + J_{\mathrm{ANT}})/W_x \\
\dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_x}{\mathrm{d}t} = (-J_{\mathrm{F}} + J_{\mathrm{PiC}})/W_x \quad \text{matrix species} \\
\dfrac{\mathrm{d}[\mathrm{NADH}^{2-}]_x}{\mathrm{d}t} = (J_{\mathrm{DH}} - J_{\mathrm{C1}})/W_x \\
\dfrac{\mathrm{d}[\mathrm{QH}_2]_x}{\mathrm{d}t} = (J_{\mathrm{C1}} - J_{\mathrm{C3}})/W_x \\
\hline
\dfrac{\mathrm{d}[\mathrm{c}^{2+}_{red}]_i}{\mathrm{d}t} = 2(J_{\mathrm{C3}} - J_{\mathrm{C4}})/W_i \quad \text{intermembrane space species} \\
\hline
\dfrac{\mathrm{d}[\Sigma\mathrm{ATP}]_c}{\mathrm{d}t} = (V_{m2c}J_{\mathrm{ANT}} - J_{\mathrm{AtC}} + J_{\mathrm{CK}})/W_c \\
\dfrac{\mathrm{d}[\Sigma\mathrm{ADP}]_c}{\mathrm{d}t} = (-V_{m2c}J_{\mathrm{ANT}} + J_{\mathrm{AtC}} - J_{\mathrm{CK}})/W_c \quad \text{cytoplasm species} \\
\dfrac{\mathrm{d}[\Sigma\mathrm{Pi}]_c}{\mathrm{d}t} = (-V_{m2c}J_{\mathrm{PiC}} + J_{\mathrm{AtC}})/W_c \\
\dfrac{\mathrm{d}[\mathrm{CrP}]}{\mathrm{d}t} = -J_{\mathrm{CK}}/W_c.
\end{cases} \tag{26}$$

In addition to the incorporation of the creatine kinase reaction, the in vivo model is adapted from the in vitro model by adjusting the mitochondrial volume. In the in vitro model, the volume fraction of the experimental system taken up by mitochondria is 0.0005 corresponding to a dilute suspension of purified mitochondria (Table 2). In the in vivo model, the volume fraction of a cardiomyocyte taken up by mitochondria is 0.2882 (Table 6).

Symbol	Units	Description	Value	Source
X_{CK}	mol s^{-1} (L cyto)$^{-1}$	Creatine kinase activity	10^7	[15]
$[\mathrm{Cr}]_{tot}$	mmol (L cell)$^{-1}$	Total creatine pool in the cell	40	[9]

Table 6 Parameters for ATP synthesis in vivo.

Simulation of respiratory control in vivo

Previous investigations ([9][5]) have revealed that the certain cytosolic metabolite pools influence the phosphate metabolite levels in the myocardium in vivo. These metabolite pools are the total adeninine nucleotide (TAN, $\text{mmol (L cell)}^{-1}$), total exchangeable phosphate (TEP, $\text{mmol (L cell)}^{-1}$), and total creatine ($[\text{Cr}]_{tot}$, $\text{mmol (L cell)}^{-1}$) pool, which may be computed from our model variables via

$$\begin{aligned}
\text{TAN} &= (V_c W_c + V_m W_i)([\text{ATP}]_c + [\text{ADP}]_c) + V_m W_x([\text{ATP}]_x + [\text{ADP}]_x), \\
\text{TEP} &= (V_c W_c + V_m W_i)(2[\text{ATP}]_c + [\text{ADP}]_c + [\text{Pi}]_c + [\text{CrP}]_c) + V_m W_x([\text{ATP}]_x + [\text{ADP}]_x + [\text{Pi}]_x), \quad \text{and} \\
\text{Cr}_{tot} &= V_c W_c([\text{Cr}]_c + [\text{CrP}]_c).
\end{aligned}$$

To simulate healthy normal conditions, these pools are set as $\text{TAN} = 7.6$, $\text{TEP} = 27.5$, and $[\text{Cr}]_{tot} = 40\ \text{mM (L cell)}^{-1}$. The levels of these metabolite pools have been shown to decrease in heart failure compared to normal physiological conditions. In the simulations below we explore the predicted effects of altering these metabolite pool levels.

The code below computes the steady-state behavior of the in vivo model over a range of ATP consumption rates, representing myocardial ATP demand levels associated with resting and exercise conditions. The resting state is associated with a myocardial ATP consumption rate of approximately $0.4\ \text{mmol s}^{-1}\ \text{(L cell)}^{-1}$ while under vigorous exercise conditions the ATP consumption rate is approximately $1.2\ \text{mmol s}^{-1}\ \text{(L cell)}^{-1}$ [6]. Here the myocardial ATP consumption rate is varied over the range of 0.4 to $1.2\ \text{mmol s}^{-1}\ \text{(L cell)}^{-1}$, corresponding to a range of myocardial oxygen consumption rate of approximately 25 to $60\ \text{nmol min}^{-1}\ \text{(U CS)}^{-1}$.

Symbol	Description	Units	Healthy	Heart failure [9]
TAN	Total adenine nucleotide	mM	7.6	6.976
TEP	Total exchangeable phosphate	mM	27.5	24.11
$[\text{Cr}_{tot}]$	Total creatine	mM	40	23.03

Table 7 Metabolite pool concentrations in mmol (L cell)$^{-1}$.

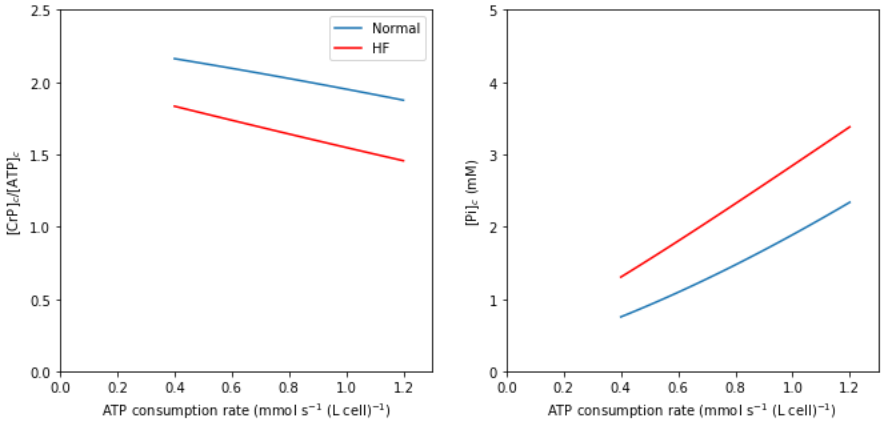
```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

###### Constants defining metabolite pools ######
# Volume fractions and water space fractions
V_c = 0.6601       # cytosol volume fraction       # L cyto (L cell)**(-1)
V_m = 0.2882       # mitochondrial volume fraction # L mito (L cell)**(-1)
V_m2c = V_m / V_c  # mito to cyto volume ratio     # L mito (L cyto)**(-1)
W_c = 0.8425      # cytosol water space           # L cyto water (L cyto)**(-1)
W_m = 0.7238       # mitochondrial water space     # L mito water (L mito)**(-1)
W_x = 0.9*W_m      # matrix water space            # L matrix water (L mito)**(-1)
W_i = 0.1*W_m      # intermembrane water space     # L IM water (L mito)**(-1)

# Total pool concentrations
NAD_tot = 2.97e-3  # NAD+ and NADH conc            # mol (L matrix water)**(-1)
Q_tot   = 1.35e-3  # Q and QH2 conc                # mol (L matrix water)**(-1)
c_tot   = 2.7e-3   # cytochrome c ox and red conc  # mol (L IM water)**(-1)

# Membrane capacitance
Cm = 3.1e-3    # mol (V * L mito)^(-1)

###### Set fixed pH, cation concentrations, and O2 partial pressure ######
# pH
pH_x = 7.40
pH_c = 7.20

# K+ concentrations
K_x  = 100e-3      # mol (L matrix water)**(-1)
K_c  = 140e-3      # mol (L cyto water)**(-1)

# Mg2+ concentrations
Mg_x = 1.0e-3        # mol (L matrix water)**(-1)
Mg_c = 1.0e-3        # mol (L cyto water)**(-1)

# Oxygen partial pressure
PO2 = 25 # mmHg

conc = np.array([pH_x, pH_c, K_x, K_c, Mg_x, Mg_c, PO2])

###### Parameter vector ######
X_DH  = 0.1732
X_C1  = 1.0e4
X_C3  = 1.0e6
X_C4  = 0.0125
X_F   = 1.0e3
E_ANT = 0.325
E_PiC = 5.0e6
X_H   = 1.0e3
X_CK  = 1e7
X_AtC = 0.5e-3

activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_CK,
X_AtC])

def dXdt(t, X, activity_array, solve_ode):
    # Unpack variables
    DPsi, sumATP_x,sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c,
sumPi_c, CrP_c = X
    X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_CK, X_AtC = activity_array

    # Hydrogen ion concentration
    H_x = 10**(-pH_x) # mol (L matrix water)**(-1)
    H_c = 10**(-pH_c) # mol (L cuvette water)**(-1)

    # Oxygen concentration
    a_3  = 1.74e-6   # oxygen solubility in cuvette   # mol (L matrix water * mmHg)**
(-1)
    O2_x = a_3*PO2   # mol (L matrix water)**(-1)

    # Thermochemical constants
    R = 8.314          # J (mol K)**(-1)
    T = 37 + 273.15    # K
    F = 96485          # C mol**(-1)

    # Proton motive force parameters (dimensionless)
    n_F  = 8/3
    n_C1 = 4
    n_C3 = 2
    n_C4 = 4

    # Dissociation constants
    K_MgATP = 10**(-3.88)
    K_HATP  = 10**(-6.33)
    K_KATP  = 10**(-1.02)
    K_MgADP = 10**(-3.00)
    K_HADP  = 10**(-6.26)
```

```
    K_KADP  = 10**(-0.89)
    K_MgPi  = 10**(-1.66)
    K_HPi   = 10**(-6.62)
    K_KPi   = 10**(-0.42)

    ## Other concentrations computed from the state variables
    NAD_x = NAD_tot - NADH_x  ## mol (L matrix water)**(-1)
    Q_x   = Q_tot - QH2_x     ## mol (L matrix water)**(-1)
    cox_i = c_tot - cred_i    ## mol (L matrix water)**(-1)
    Cr_c  = Cr_tot_c - CrP_c  ## mol (L cyto water)**(-1)

    ## Binding polynomials
    # Matrix species # mol (L mito water)**(-1)
    PATP_x = 1 + H_x/K_HATP + Mg_x/K_MgATP + K_x/K_KATP
    PADP_x = 1 + H_x/K_HADP + Mg_x/K_MgADP + K_x/K_KADP
    PPi_x  = 1 + H_x/K_HPi  + Mg_x/K_MgPi  + K_x/K_KPi

    # Cytosol species # mol (L cyto water)**(-1)
    PATP_c = 1 + H_c/K_HATP + Mg_c/K_MgATP + K_c/K_KATP
    PADP_c = 1 + H_c/K_HADP + Mg_c/K_MgADP + K_c/K_KADP
    PPi_c  = 1 + H_c/K_HPi  + Mg_c/K_MgPi  + K_c/K_KPi

    ## Unbound species
    # Matrix species
    ATP_x = sumATP_x / PATP_x # [ATP4-]_x
    ADP_x = sumADP_x / PADP_x # [ADP3-]_x
    Pi_x  = sumPi_x  / PPi_x  # [HPO42-]_x

    # Cytosol species
    ATP_c = sumATP_c / PATP_c # [ATP4-]_c
    ADP_c = sumADP_c / PADP_c # [ADP3-]_c
    Pi_c  = sumPi_c  / PPi_c  # [HPO42-]_c


    ###### NADH Dehydrogenase ######

    # Constants
    r      = 6.8385
    k_Pi1  = 4.659e-4    # mol (L matrix water)**(-1)
    k_Pi2  = 6.578e-4 # mol (L matrix water)**(-1)

    # Flux (mol (s * L mito)^(-1))
    J_DH = X_DH * (r * NAD_x - NADH_x) * ((1 + sumPi_x / k_Pi1) / (1+sumPi_x / k_Pi2))

    ###### Complex I ######
    # NADH_x + Q_x + 5H+_x <-> NAD+_x + QH2_x + 4H+_i + 4dPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C1 = -109680
    DrGapp_C1 = DrGo_C1 - R * T * np.log(H_x)

    # Apparent equilibrium constant
    Kapp_C1   = np.exp( -(DrGapp_C1 + n_C1 * F * DPsi) / (R * T)) * ((H_x /
H_c)**n_C1)

    # Flux (mol (s * L mito)**(-1))
    J_C1 = X_C1 * (Kapp_C1 * NADH_x * Q_x - NAD_x * QH2_x)

    ###### Complex III ######
    # QH2_x + 2cuvetteC(ox)3+_i + 2H+_x <-> Q_x + 2cuvetteC(red)2+_i + 4H+_i + 2DPsi

    # Gibbs energy (J mol**(-1))
    DrGo_C3 = 46690
    DrGapp_C3 = DrGo_C3 + 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C3   = np.exp(-(DrGapp_C3 + n_C3 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C3

    # Flux (mol (s * L mito)**(-1))
    J_C3 = X_C3 * (Kapp_C3 * cox_i**2 * QH2_x - cred_i**2 * Q_x)


    ###### Complex IV ######
    # 2 cytoC(red)2+_i + 0.5O2_x + 4H+_x <-> cytoC(ox)3+_x + H2O_x + 2H+_i + 2DPsi

    # Constant
    k_O2 = 1.2e-4      # mol (L matrix water)**(-1)

    # Gibbs energy (J mol**(-1))
    DrGo_C4 = -202160  # J mol**(-1)
    DrGapp_C4 = DrGo_C4 - 2 * R * T * np.log(H_c)

    # Apparent equilibrium constant
    Kapp_C4   = np.exp(-(DrGapp_C4 + n_C4 * F * DPsi) / (R * T)) * (H_x / H_c)**n_C4

    # Flux (mol (s * L mito)**(-1))
    J_C4 = X_C4 *(Kapp_C4**0.5 * cred_i * O2_x**0.25 - cox_i) * (1 / (1 + k_O2 /
```

```
O2_x))

    ###### F0F1-ATPase ######
    # ADP3-_x + HPO42-_x + H+_x + n_A*H+_i <-> ATP4- + H2O + n_A*H+_x

    # Gibbs energy (J mol**(-1))
    DrGo_F = 4990

    DrGapp_F = DrGo_F + R * T * np.log( H_x * PATP_x / (PADP_x * PPi_x))
    Kapp_F   = np.exp( (DrGapp_F + n_F * F * DPsi ) / (R * T)) * (H_c / H_x)**n_F

    # Flux (mol (s * L mito)**(-1))
    J_F = X_F * (Kapp_F * sumADP_x * sumPi_x - sumATP_x)


    ###### ANT ######
    # ATP4-_x + ADP3-_i <-> ATP4-_i + ADP3-_x

    # Constants
    del_D   = 0.0167
    del_T   = 0.0699
    k2o_ANT = 9.54/60      # s**(-1)
    k3o_ANT = 30.05/60     # s**(-1)
    K0o_D   = 38.89e-6     # mol (L cyto water)**(-1)
    K0o_T   = 56.05e-6     # mol (L cyto water)**(-1)
    A       = +0.2829
    B       = -0.2086
    C       = +0.2372

    phi = F * DPsi / (R * T)

    # Reaction rates
    k2_ANT = k2o_ANT * np.exp((A*(-3) + B*(-4) + C)*phi)
    k3_ANT = k3o_ANT * np.exp((A*(-4) + B*(-3) + C)*phi)

    # Dissociation constants
    K0_D = K0o_D * np.exp(3*del_D*phi)
    K0_T = K0o_T * np.exp(4*del_T*phi)

    q     = k3_ANT * K0_D * np.exp(phi) / (k2_ANT * K0_T)
    term1 = k2_ANT * ATP_x * ADP_c * q / K0_D
    term2 = k3_ANT * ADP_x * ATP_c / K0_T
    num   = term1 - term2
    den   = (1 + ATP_c/K0_T + ADP_c/K0_D) * (ADP_x + ATP_x * q)

    # Flux
    J_ANT = E_ANT * num / den

    ###### H+-PI2 cotransporter ######
    # H2PO42-_x + H+_x = H2PO42-_c + H+_c

    # Constant
    k_PiC = 1.61e-3  # mol (L cell)**(-1)

    # H2P04- species
    HPi_c = Pi_c * (H_c / K_HPi)
    HPi_x = Pi_x * (H_x / K_HPi)

    # Flux (mol (s * L mito)**(-1))
    J_PiC = E_PiC * (H_c * HPi_c - H_x * HPi_x) / (k_PiC + HPi_c)

    ###### H+ leak ######

    # Flux (mol (s * L mito)**(-1))
    J_H = X_H * (H_c * np.exp(phi/2) - H_x * np.exp(-phi/2))

    ###### ATPase ######
    # ATP4- + H2O = ADP3- + PI2- + H+

    #Flux (mol (s * L cyto)**(-1))
    J_AtC = X_AtC / V_c

    ###### Creatine kinase reaction ######
    # ADP3- + CrP2- + H+ = ATP4- + Cr

    # Equilibrium constant (dimensionless)
    Keq_CK  = 3.5e8

    # Gibbs energy (J mol**(-1))
    DrGo_CK = - R * T * np.log(Keq_CK)
    DrGapp_CK = DrGo_CK + R * T * np.log(PADP_c / (PATP_c * H_c))

    # Apparent equilibrium constant
    Kapp_CK   = np.exp(-DrGapp_CK / (R * T))

    # Flux (mol (s * L cyto)^(-1))
    J_CK = X_CK * (Kapp_CK * ADP_c * CrP_c - ATP_c * Cr_c)
```

```
    ###### Differential equations (equation 26) ######
    # Membrane potential
    dDPsi = (n_C1 * J_C1 + n_C3 * J_C3 + n_C4 * J_C4 - n_F * J_F - J_ANT - J_H) / Cm

    # Matrix species
    dATP_x  = (J_F  - J_ANT) / W_x
    dADP_x  = (-J_F + J_ANT) / W_x
    dPi_x   = (-J_F + J_PiC) / W_x
    dNADH_x = (J_DH - J_C1)  / W_x
    dQH2_x  = (J_C1 - J_C3)  / W_x

    # IM space species
    dcred_i = 2 * (J_C3 - J_C4) / W_i

    # Cytosol species
    dATP_c = ( V_m2c * J_ANT - J_AtC + J_CK ) / W_c
    dADP_c = (-V_m2c * J_ANT + J_AtC - J_CK ) / W_c
    dPi_c  = (-V_m2c * J_PiC + J_AtC) / W_c
    dCrP_c = -J_CK / W_c

    dX = [dDPsi, dATP_x, dADP_x, dPi_x, dNADH_x, dQH2_x, dcred_i, dATP_c, dADP_c,
dPi_c, dCrP_c]
    if solve_ode == 1:
        return dX
    else:
        J = np.array([PATP_x, PADP_x, PPi_x, PATP_c, PADP_c, PPi_c, J_DH, J_C1, J_C3,
J_C4, J_F, J_ANT, J_PiC, J_CK])
        return dX, J

###### Initial Conditions ######
# Membrane Potential
Psi_0 = 175/1000      # Volts

# Matrix species
sumATP_x_0 = 0.5e-3  # mol (L matrix water)^(-1)
sumADP_x_0 = 9.5e-3  # mol (L matrix water)^(-1)
sumPi_x_0 = 0.3e-3   # mol (L matrix water)^(-1)
NADH_x_0 = 2/3 * NAD_tot  # mol (L matrix water)^(-1)
QH2_x_0 = 0.1 * Q_tot   # mol (L matrix water)^(-1)

# IMS species
cred_i_0 = 0.1 * c_tot

# Cytoplasmic species
#sumATP_c_0 = 9.95e-3  # mol (L cyto water)^(-1)
sumADP_c_0 = 0.05e-3  # mol s(L cyto water)^(-1)
#sumPi_c_0 = 5.0e-3  # mol (L cyto water)^(-1)

###### Healthy normal case ######
TAN = 0.0076 #(M per liter cell)
TEP = 0.0275  #(M per liter cell)
Cr_tot  = 0.040    #(M per liter cell)

sumATP_c_0 = (TAN - V_m*W_x*(sumATP_x_0 + sumADP_x_0))/(V_c*W_c+V_m*W_i) - sumADP_c_0
Cr_tot_c   = Cr_tot / (V_c * W_c) # convert to mol (L cyto water)^(-1)
CrP_c_0    = .3 * Cr_tot_c  # mol (L cyto water)^(-1)
sumPi_c_0  = (TEP-V_m*W_x*(sumATP_x_0 + sumADP_x_0 + sumPi_x_0 ))/(V_c*W_c+V_m*W_i) -
2*sumATP_c_0 - sumADP_c_0 - CrP_c_0

X_0 = np.array([Psi_0, sumATP_x_0, sumADP_x_0, sumPi_x_0, NADH_x_0, QH2_x_0, cred_i_0,
sumATP_c_0, sumADP_c_0, sumPi_c_0, CrP_c_0])

# range of ATP consumption rates
X_AtC = np.linspace(0.4e-3,1.2e-3, 60)   # Increase max hydrolysis to find apparent Km.
steady_state = np.zeros((len(X_AtC), len(X_0)))
JO2 = np.zeros(len(X_AtC))
tspan = np.array([0,10])
# looping through different ATP consumptions states
for i in range(len(X_AtC)):
   activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_CK,
X_AtC[i]])
   # run for long time to acheive steady-state
   steady_state_temp_results = solve_ivp(dXdt, tspan, X_0, method = 'Radau', args=
(activity_array,1,),max_step = 0.1).y[:,-1]
   steady_state[i] = steady_state_temp_results

DPsi, sumATP_x, sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c, sumPi_c,
CrP_c = steady_state.T

fig, ax = plt.subplots(1,2, figsize = (11,5))

## Plot normal case
# CrP/ATP ratio
ax[0].plot(X_AtC * 1000, CrP_c * (V_c * W_c)/(sumATP_x * V_m * W_x+ sumATP_c *(V_c *
W_c + V_m * W_i)),label = 'Normal')      #
```

```
# Pi_c
ax[1].plot(X_AtC * 1000, sumPi_c * 1000,label = 'Normal')     # Pi_c


###### Heart Failure (HF/TAC) case ######
#Mean TAC pools
TAN = 0.006976 #(M per liter cell)
TEP = 0.02411  #(M per liter cell)
Cr_tot =  0.02303 #(M per liter cell)

sumATP_c_0 = (TAN - V_m*W_x*(sumATP_x_0 + sumADP_x_0))/(V_c*W_c+V_m*W_i) - sumADP_c_0
Cr_tot_c   = Cr_tot / (V_c * W_c) # convert to mol (L cyto water)^(-1)
CrP_c_0    = .3 * Cr_tot_c  # mol (L cyto water)^(-1)
sumPi_c_0  = (TEP-V_m*W_x*(sumATP_x_0 + sumADP_x_0 + sumPi_x_0 ))/(V_c*W_c+V_m*W_i) -
2*sumATP_c_0 - sumADP_c_0 - CrP_c_0

X_0 = np.array([Psi_0, sumATP_x_0, sumADP_x_0, sumPi_x_0, NADH_x_0, QH2_x_0, cred_i_0,
sumATP_c_0, sumADP_c_0, sumPi_c_0, CrP_c_0])

# range of ATP consumption rates
steady_stateHF = np.zeros((len(X_AtC), len(X_0)))
# looping through different ATP consumptions states
for i in range(len(X_AtC)):
   activity_array = np.array([X_DH, X_C1, X_C3, X_C4, X_F, E_ANT, E_PiC, X_H, X_CK,
X_AtC[i]])
   # run for long time to acheive steady-state
   steady_state_temp_resultsHF = solve_ivp(dXdt, tspan, X_0, method = 'Radau', args=
(activity_array,1,),max_step = 0.1).y[:,-1]
   steady_stateHF[i] = steady_state_temp_resultsHF

DPsi, sumATP_x, sumADP_x, sumPi_x, NADH_x, QH2_x, cred_i, sumATP_c, sumADP_c, sumPi_c,
CrP_c = steady_stateHF.T

## Plot figures
# CrP/ATP ratio
ax[0].plot(X_AtC * 1000, CrP_c * (V_c * W_c)/(sumATP_x * V_m * W_x+ sumATP_c *(V_c *
W_c + V_m * W_i)),'red',label = 'HF')
ax[0].set_ylabel('[CrP]$_c$/[ATP]$_c$')
ax[0].set_xlabel('ATP consumption rate (mmol s$^{-1}$ (L cell)$^{-1}$)')
ax[0].set_xlim([0,1.3])
ax[0].set_ylim([0.0,2.5])
ax[0].legend()

# Pi_c
ax[1].plot(X_AtC * 1000, sumPi_c * 1000, 'red',label = 'HF')
ax[1].set_ylabel('[Pi]$_c$ (mM)')
ax[1].set_xlabel('ATP consumption rate (mmol s$^{-1}$ (L cell)$^{-1}$)')
ax[1].set_xlim([0,1.3])
ax[1].set_ylim([0,5])
plt.show()
```



Simulations of the normal case (blue lines) show that over the physiological range of ATP demand and oxygen consumption, the CrP/ATP ratio in the myocardium decreases from a value of 2.2 at rest to 2.0 in exercise, while the cytosolic phosphate concentration increases from approximately 0.75 mM at rest to 2.3 mM in exercise. These model predictions are remarkably close to experiment observations, given the relative simplicity of this model. Consistent with previous analyses, when the metabolite pool levels are changed to represent heart failure conditions (red lines), the CrP/ATP ratio decreases compared and the inorganic phosphate concentration is predicted to increase compared to physiological levels.

Summary and conclusions

The collection of calculations and simulations presented here represents a framework for analyzing and simulating mitochondrial function using basic thermodynamic and kinetic principles. This framework largely uses simple mass-action representations of kinetic processes, rather than models accounting for detailed catalytic mechanisms. It is perhaps remarkable that this simple approach is able effectively capture the systems-level function of ATP synthesis and respiratory control both in vitro and in vivo. Since these models and simulations are grounded in a rigorous treatment of the thermodynamics, we conclude that—with respect to the phenomena addressed in the simulations presented here—the thermodynamics of many of the underlying chemical processes is more important than the kinetics. However, in contrast to that general conclusion, the adenine nucleotide translocase, in particular, is represented with a model to account for the kinetics of its catalytic mechanism. Since the adenine nucleotide translocase catalyzes the final step in oxidative ATP synthesis, its kinetic mechanism is predicted to have the most direct influence of all the steps in governing cellular energetics and respiratory control in vivo. Moreover, the conclusion, for example, that the kinetics of complex I is effectively represented by the simple mass-action model of Equation (18), is dependent on context. While the model works for the purposes used here, it does not provide insights into phenomena that depend on a more detailed representation of a catalytic mechanism, as may be investigated using more complex models [15].

By embedding all of the computer codes for the calculations and simulations that we have presented, our intention is to facilitate an easy adoption and modification of the concepts developed here. Armed with these concepts and codes, it is hoped that readers will reuse, repurpose, update, modify, test, and extend these models. Moreover, we hope that users and developers of these concepts will adopt publication vehicles such as this one, for sharing data, codes, and associated calculations with the community.

References

[1]
D.A. Beard. A biophysical model of the mitochondrial respiratory system and oxidative phosphorylation. *PLoS Comp Biol*, 1:e36, 2005. doi:10.1371/journal.pcbi.0010036.

[2]
D.A. Beard. *Biosimulation: Simulation of Living Systems*. Cambridge University Press, 2012.

[3]
F. Wu. E.Y. Zhang. J. Zhang. R.J. Bache. D.A. Beard. Phosphate metabolite concentrations and atp hydrolysis potential in normal and ischaemic hearts. *J Physiol*, 586:4193–4208, 2008. doi:10.1113/jphysiol.2008.154732.

[4]
F. Wu. F. Yang. K.C. Vinnakota. D.A. Beard. Computer modeling of mitochondrial tricarboxylic acid cycle, oxidative phosphorylation, metabolite transport, and electrophysiology. *J Biol Chem*, 282:24525–24537, 2007. doi:10.1074/jbc.M701024200.

[5]
F. Wu. J. Zhang. D.A. Beard. Phosphate metabolite concentrations and atp hydrolysis potential in normal and ischaemic hearts. *Proc Natl Acad Sci USA*, 106:7143–7148, 2009. doi:0.1073/pnas.0812768106.

[6]
X. Gao. D.G. Jakovljevic. D.A. Beard. Cardiac metabolic limitations contribute to diminished performance of the heart in aging. *Biophys J*, 117:2295–2302, 2019. doi:10.1016/j.bpj.2019.06.026.

[7]
X. Li. F. Wu. F. Qi. D.A. Beard. A database of thermodynamic properties of the reactions of glycolysis, the tricarboxylic acid cycle, and the pentose phosphate pathway. *Database*, 2011:bar005, 2011. doi:10.1093/database/bar005.

[8]
E. Metelkin. I. Goryanin. O. Demin. Mathematical modeling of mitochondrial adenine nucleotide translocase. *Biophys J*, 90:423–432, 2006. doi:10.1529/biophysj.105.061986.

[9]
R. Lopez. B. Marzban. X. Gao. E. Lauinger. F. Van den Bergh. S.E. Whitesall. K. Converso-Baran. C.F. Burant. D.E. Michele. D.A. Beard. Impaired myocardial energetics causes mechanical dysfunction in decompensated failing hearts. *Function*, 1:zqaa018, 2020. doi:10.1093/function/zqaa018.

[10]
E. Gnaiger et al. Mitochondrial physiology. *Bioenerg Commun*, 2020. doi:10.26124/mitofit:190001.

[11]
D.G. Nicholls. S.J. Ferguson. *Bioenergetics*. Academic Press, Elsevier, Amsterdam, The Netherlands, 2013. ISBN 9780123884312.

[12]
P. Mitchell. Coupling of phosphorylation to electron and hydrogen transfer by a chemi-osmotic type of mechanism. *Nature*, 191:144–148, 1961. doi:10.1038/191144a0.

[13]
J.R. Friedman. J. Nunnari. Mitochondrial form and function. *Nature*, 505(7483):335–343, 2014. doi:10.1038/nature12985.

[14]
B. Chance. S. Leigh Jr. B.J. Clark. J. Maris. J. Kent. S. Nioka. D. Smith. Control of oxidative metabolism and oxygen delivery in human skeletal muscle: a steady-state analysis of the work/energy cost transfer function. *Proc Natl Acad Sci USA*, 82:8384–8388, 1985.

[15]
J.N. Bazil. D.A. Beard. K.C. Vinnakota. Catalytic coupling of oxidative phosphorylation, atp demand, and reactive oxygen species generation. *Biophys J*, 110(4):962–971, 2016. doi:10.1016/j.bpj.2015.09.036.

[16]
B. Chance. G.R. Williams. Respiratory enzymes in oxidative phosphorylation. i. kinetics of oxygen utilization. *J Biol Chem*, 217:383–935, 1955.

By E. Benjamin Randall, Marcus Hock, Rachel Lopez, Bahador Marzban, Collin Marshall, Daniel A. Beard