

The Chloroplast Electron Transport Chain

Peter Gawthrop (peter.gawthrop@unimelb.edu.au)

November 6, 2019

Contents

1	Introduction	2
2	Photon Energetics	3
3	Redox reactions	4
3.1	Redox potentials	5
4	Bond graph description	6
4.1	Complex PII -- Photosystem II	6
4.2	Complex Cyt -- Cytochrome bf	7
4.3	Complex PI -- Photosystem I	9
4.4	Complex Fer -- Ferredoxin-NADP reductase	10
5	The Electron Transport Chain	12
5.1	Unify species in stoichiometric matrix using <code>stoich.unify()</code>	13
5.2	Physically-plausible model	14
5.3	Unify species in model using <code>modular.unify()</code>	16

Note: this is the Chloroplast.ipynb notebook. The PDF version "The Chloroplast Electron Transport Chain" is available [here](#).

This is a work in progress and needs more explanatory notes.

```
In [1]: ## Some useful imports
import BondGraphTools as bgt
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt

## For reimporting: use imp.reload(module)
import importlib as imp

## Stoichiometric analysis
import stoich as st
##import hstoich as hst

## SVG
import svgBondGraph as sbg

## Display (eg disp.SVG(), disp.
import IPython.display as disp

## Data
import phiData
import redoxData
```

1 Introduction

Photosynthesis within plant chloroplasts is the basis of life on earth (Blankenship, 2015), (Nicholls and Ferguson, 2013).

Like the mitochondrion, the chloroplast has a membrane separating an inner space (lumen) from an outer space (stroma). In the chloroplast, the lumen gains protons and is called the p-space, the stroma loses protons and is called the n-space. Thus geometrically, the lumen corresponds to the mitochondrial matrix and the stroma to the mitochondrial intermembrane space; but electrically the p-space is inside and the n-space outside - the reverse of the mitochondrial situation.

The chloroplast electron transport chain has 4 complexes.

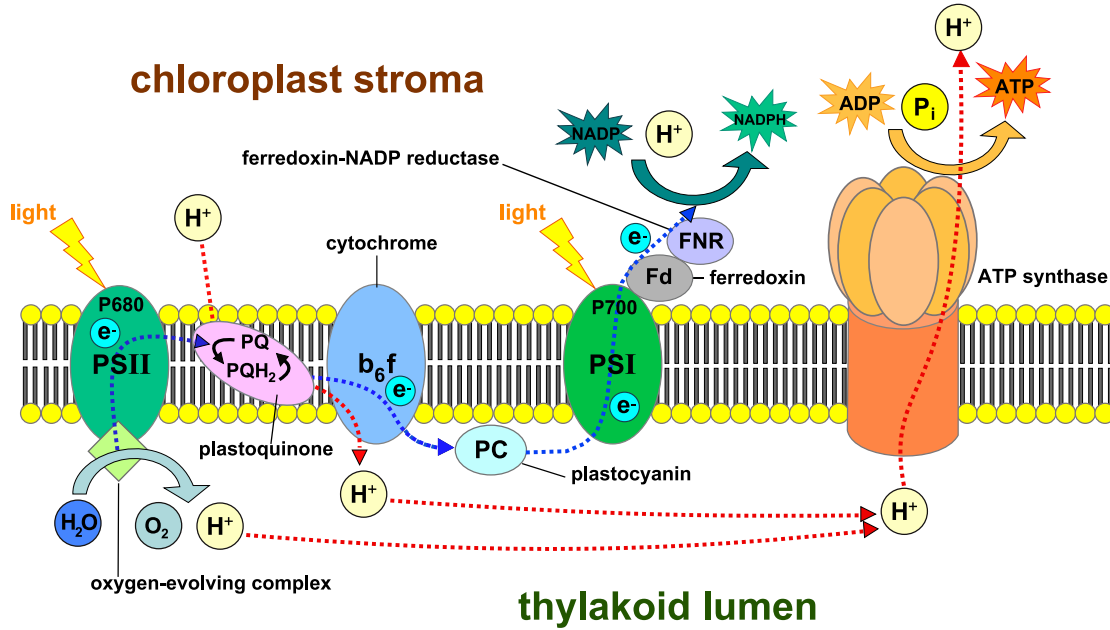
1. Photosystem II (PII) which absorbs photons at 680nm and splits water releasing protons into the p-space and passing electrons to the plastoquinone(PQ)/plastoquinone(PQH₂) couple which absorbs protons from the n-space.
2. Cytochrome bf (Cyt) which passes electrons to the plastoquinone/plastoquinone couple which releases two protons into the p-space. Electrons are passed to the plastocyanine couple (PcOx/PcRed). Two protons are pumped across the membrane.
3. Photosystem I (PI) which absorbs photons at 700nm and transports electrons from the plastocyanine (PcRed/PcOx) couple to the ferredoxin (FdOx/FdRed) couple.

4. Ferredoxin-NADP reductase which transfers electrons from the ferredoxin (FdRed/FdOx) couple to convert NADP to NADPH absorbing a proton from the n-space.

The following figure is: https://commons.wikimedia.org/wiki/File:Thylakoid_membrane_3.svg

In [2]: # https://commons.wikimedia.org/wiki/File:Thylakoid_membrane_3.svg
disp.SVG("Thylakoid_membrane_3.svg")

Out [2]:



2 Photon Energetics

$$\phi_{\text{photon}} = \frac{N_{av}hc}{F\lambda} \quad (1)$$

where N_{av} = Avogadro's number (2)

h = Planck's constant (3)

c = velocity of light (4)

F = Faraday's constant (5)

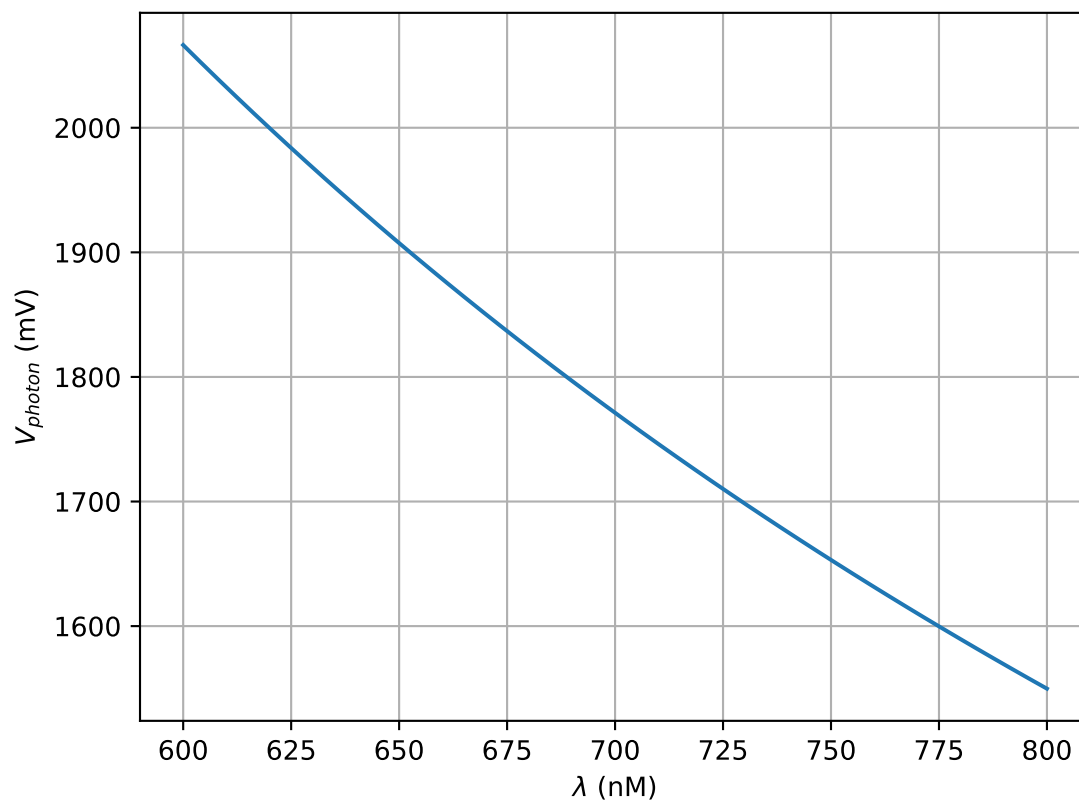
and λ = wavelength (6)

For example:

$$\phi_{\text{photon}} = \begin{cases} 1.82V & \lambda = 680nm \\ 1.77V & \lambda = 700nm \end{cases} \quad (7)$$

In [3]: `disp.SVG('V_photon.svg')`

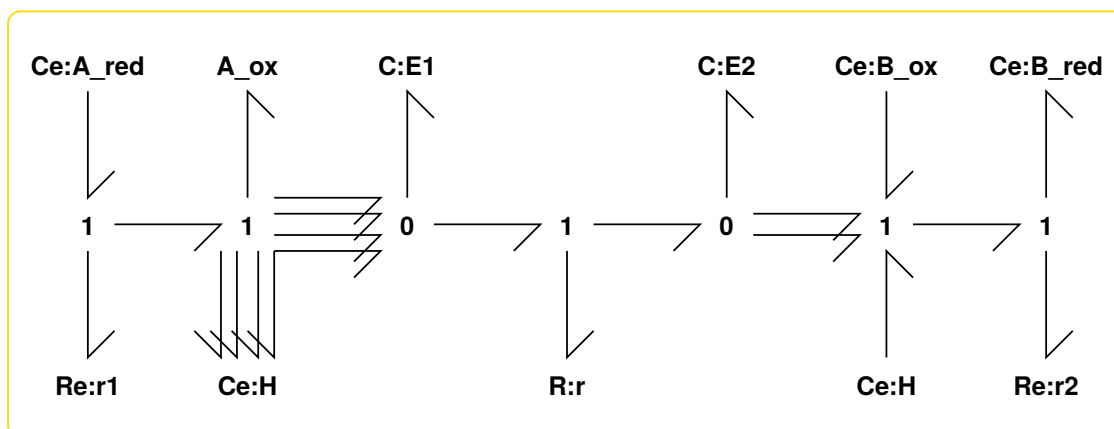
Out[3]:



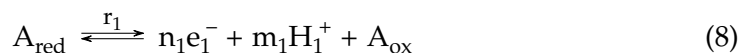
3 Redox reactions

In [4]: `disp.SVG('Redox_abg.svg')`

Out[4]:



Redox reactions can be written as two half-reactions:



3.1 Redox potentials

Note that photon energies have been used for V_700 and V_680; this value is too large as the energy conversion is not direct. A model of this needs to be built. See, for example: (Blankenship and Prince, 1985)

```
In [5]: import redoxData
        imp.reload(redoxData)
        ## pH from BerTymStr 19.3
        pH_p = 4 # pH of p-space
        pH_n = pH_p + 3
        VpH = redoxData.VpH(pH_p - pH_n)
        V680 = redoxData.V_photon(wavelength=680)
        V700 = redoxData.V_photon(wavelength=700)
        print(VpH)
        print('V_pH =', int(1000*VpH), 'mV')
        print('V_680 =', int(1000*V680), 'mV')
        print('V_700 =', int(1000*V700), 'mV')
```

0.18462115653058278

V_pH = 184 mV

V_680 = 1823 mV

V_700 = 1771 mV

```
In [6]: ## Convert redox potentials to species phi
        imp.reload(redoxData)
        phi_redox = redoxData.phi()
        phi_redox['Hn'] = redoxData.VpH(pH_n)
        phi_redox['Hp'] = redoxData.VpH(pH_p)

        ## The correct photon potentials need sorting out
        ## The raw valyes used here are an overestimate. See BlaPri85
        phi_redox['P680'] = V680
        phi_redox['P700'] = V700
        #phi_redox['P680'] = phi_redox['P680+']
        #phi_redox['P700'] = phi_redox['P700+']

        ## The membrane potential is said by some to be zero. (check this)
        phi_redox['dV'] = VpH
```

4 Bond graph description

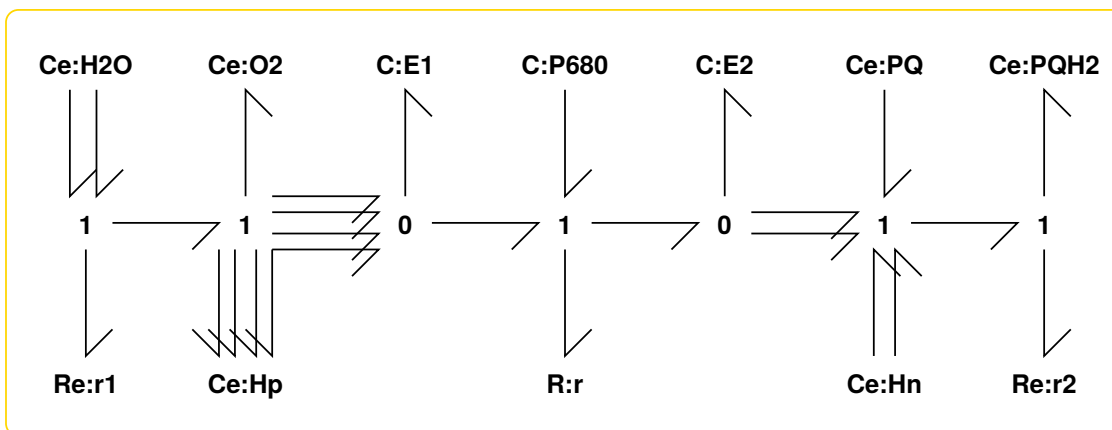
The four complexes are represented by svg graphics which are automatically converted into Bond-GraphTools format.

The stoichiometric toolbox is then used to generate the pathway-reduced equation for the complex.

4.1 Complex PII -- Photosystem II

In [7]: `disp.SVG('PII_abg.svg')`

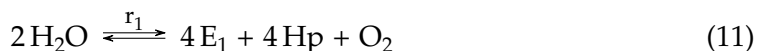
Out [7]:



```
In [8]: imp.reload(sbg)
        sbg.model('PII_abg.svg',convertR=True,convertCe=True,quiet=True)
        import PII_abg
```

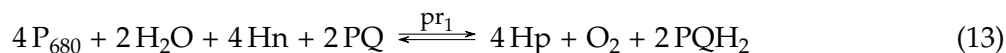
```
In [9]: ## Stoichiometry
        imp.reload(st)
        sPII = st.stoich(PII_abg.model(),quiet=True)
        chemostats = ['H2O','O2','PQ','PQH2','Hn','Hp','P680']
        scPII = st.statify(sPII,chemostats=chemostats)
        spPII = st.path(sPII,scPII)
        ## All reactions
        disp.Latex(st.sprintrl(sPII,chemformula=True))
```

Out [9]:



```
In [10]: ## Pathway reaction
         disp.Latex(st.sprintrl(spPII,chemformula=True))
```

Out[10]:



```
In [11]: ## Compute net redox potential
         RP_PII = (
             - redoxData.EpH('O2/2H2O',pH=pH_p)
             + V680
             + redoxData.EpH('PQ/PQH2',pH=pH_n)
         )

         print(redoxData.EpH('O2/2H2O',pH=pH_p))
         print(redoxData.E('P680+/P680*'))
         print(redoxData.E7('PQ/PQH2'))
         print(redoxData.EpH('PQ/PQH2',pH=pH_n))
         #print(RP_PII)
         print('RP_PII =',int(1000*RP_PII), 'mV')
```

1.0006211565305827

0.8

0

0.0

RP_PII = 822 mV

```
In [12]: ## Compute the reaction potential Phi
         phi = phiData.phi_species(phi_redox,spPII['species'])
         Phi_PII_ = -spPII['N'].T@phi
         Phi_PII = Phi_PII_[0][0]
         print('Phi_PII =',int(1000*Phi_PII), 'mV')
         print('Ratio =',(Phi_PII/RP_PII))
```

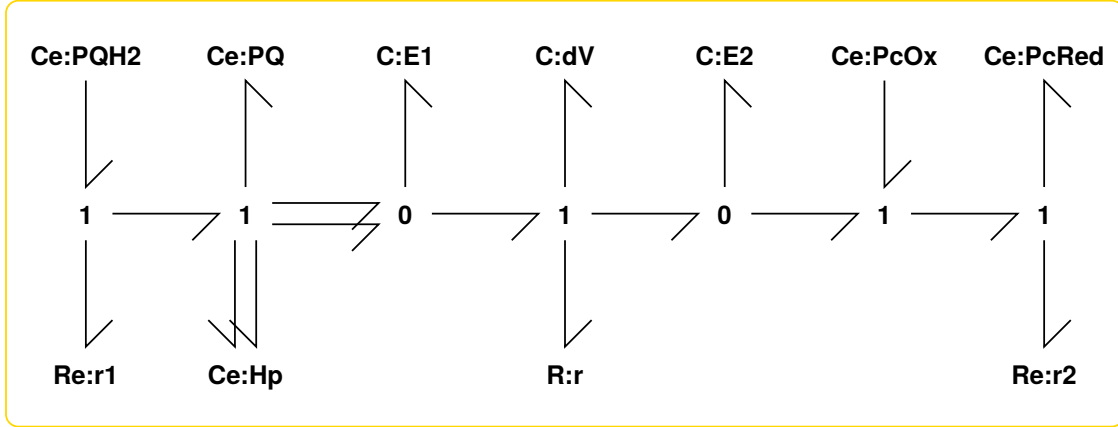
Phi_PII = 3290 mV

Ratio = 4.0

4.2 Complex Cyt -- Cytochrome bf

```
In [13]: disp.SVG('Cyt_abg.svg')
```

Out[13]:

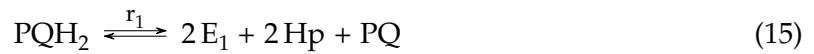


```
In [14]: sbg.model('Cyt_abg.svg',convertR=True,convertCe=True,quiet=True)
import Cyt_abg
imp.reload(Cyt_abg)
```

```
Out[14]: <module 'Cyt_abg' from '/home/peterg/WORK/Research/SystemsBiology/Notes/2019/Chloroplas
```

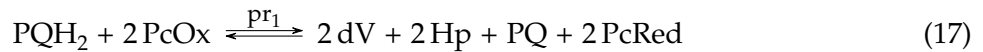
```
In [15]: ## Stoichiometry
sCyt = st.stoich(Cyt_abg.model(),quiet=True)
chemostats = ['PQ','PQH2','PcOx','PcRed','Hp','dV']
scCyt = st.statify(sCyt,chemostats=chemostats)
spCyt = st.path(sCyt,scCyt)
disp.Latex(st.sprintrl(sCyt,chemformula=True))
```

```
Out[15]:
```



```
In [16]: disp.Latex(st.sprintrl(spCyt,chemformula=True))
```

```
Out[16]:
```



```
In [17]: ## Compute net redox potential
RP_Cyt = (- redoxData.EpH('PQ/PQH2',pH=pH_p)
- redoxData.VpH(pH_p - pH_n)
+ redoxData.E('PcOx/PcRed')
)

print(RP_Cyt)
print('RP_Cyt =',redoxData.mV(RP_Cyt), 'mV')
```


0.010757686938834443

RP_Cyt = 11 mV

```
In [18]: ## Compute the reaction potential Phi
phi = phiData.phi_species(phi_redox,spCyt['species'])
Phi_Cyt_ = -spCyt['N'].T@phi
Phi_Cyt = Phi_Cyt_[0][0]
print('Phi_Cyt =',redoxData.mV(Phi_Cyt), 'mV')
print('Ratio =', int(Phi_Cyt/RP_Cyt))
```

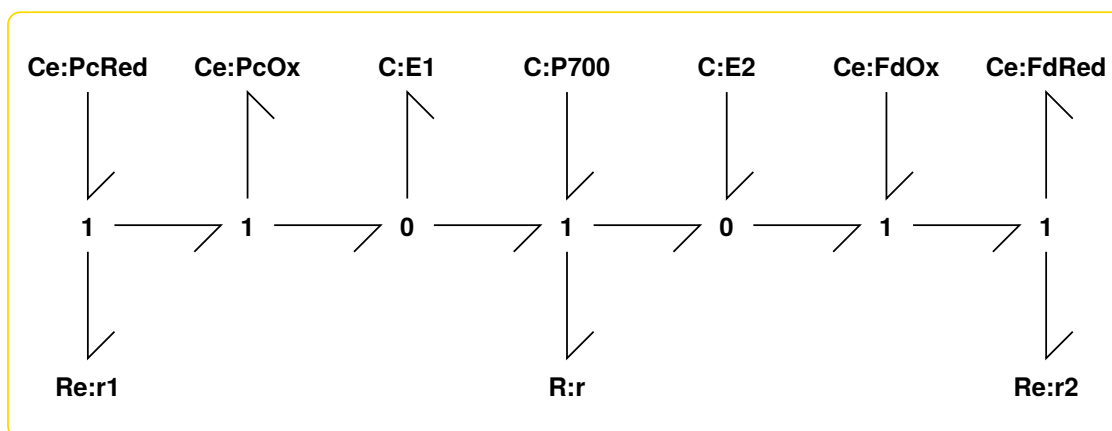
Phi_Cyt = 22 mV

Ratio = 2

4.3 Complex PI -- Photosystem I

```
In [19]: disp.SVG('PI_abg.svg')
```

Out[19]:



```
In [20]: sbg.model('PI_abg.svg',convertR=True,convertCe=True,quiet=True)
import PI_abg
```

```
In [21]: ## Stoichiometry
sPI = st.stoich(PI_abg.model(),quiet=True)
chemostats = ['PcOx','PcRed','FdOx','FdRed','P700']
scPI = st.statify(sPI,chemostats=chemostats)
spPI = st.path(sPI,scPI)
disp.Latex(st.sprintrl(sPI,chemformula=True))
```

Out[21]:



In [22]: `disp.Latex(st.sprintr1(spPI,chemformula=True))`

Out[22]:



In [23]: `## Compute net redox potential`

```
RP_PI = (
    - redoxData.E('PcOx/PcRed')
    + V700
    + redoxData.E('FdOx/FdRed')
)

#print(RP_PI)
print('RP_PI =',redoxData.mV(RP_PI), 'mV')
```

RP_PI = 961 mV

In [24]: `## Compute the reaction potential Phi`

```
phi = phiData.phi_species(phi_redox,spPI['species'])
Phi_PI_ = -spPI['N'].T@phi
Phi_PI = Phi_PI_[0][0]
print('Phi_PI =',redoxData.mV(Phi_PI), 'mV')
print('Ratio =', int(round(Phi_PI/RP_PI)))
```

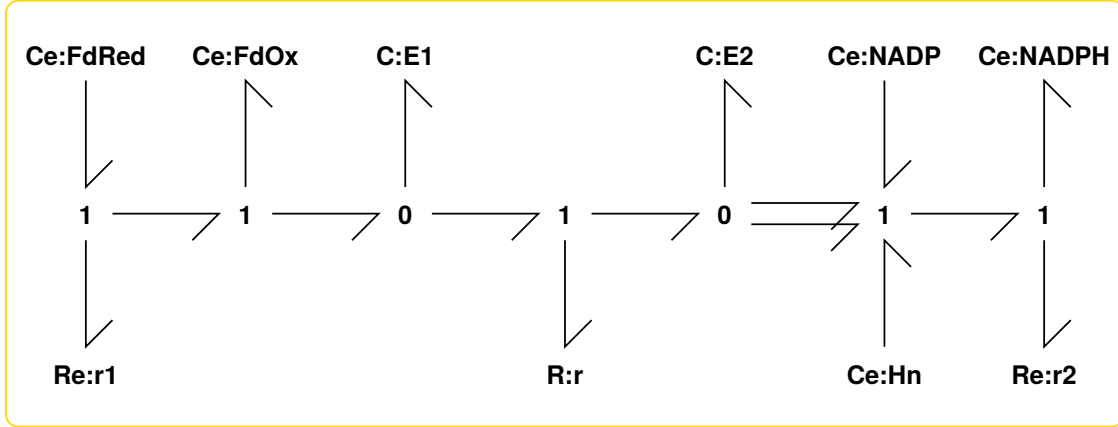
Phi_PI = 961 mV

Ratio = 1

4.4 Complex Fer -- Feredoxin-NADP reductase

In [25]: `disp.SVG('Fer_abg.svg')`

Out[25]:

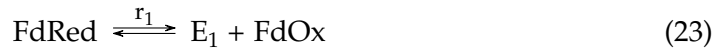


```
In [26]: sbg.model('Fer_abg.svg',convertR=True,convertCe=True,quiet=True)
import Fer_abg
```

```
In [27]: ## Stoichiometry
sFer = st.stoich(Fer_abg.model())
chemostats = ['FdRed', 'FdOx', 'NADP', 'NADPH', 'Hn']
scFer = st.stoich(Fer_abg.model(),chemostats=chemostats)
spFer = st.path(sFer,scFer)
disp.Latex(st.sprintrl(sFer,chemformula=True))
```

Swapping Re:r for two Sf in Fer
Swapping Re:r1 for two Sf in Fer
Swapping Re:r2 for two Sf in Fer
Swapping Re:r for two Sf in Fer
Swapping Re:r1 for two Sf in Fer
Swapping Re:r2 for two Sf in Fer

Out [27] :



```
In [28]: disp.Latex(st.sprintrl(spFer,chemformula=True))
```

Out [28] :



```
In [29]: ## Compute net redox potential
RP_Fer = (
    - redoxData.E('FdOx/FdRed')
    + redoxData.EpH('NADP/NADPH',pH_n)
)

#print(RP_Fer)
print('RP_Fer =',int(1000*RP_Fer), 'mV')

RP_Fer = 105 mV
```

```
In [30]: ## Compute the reaction potential Phi
phi = phiData.phi_species(phi_redox,spFer['species'])
Phi_Fer_ = -spFer['N'].T@phi
Phi_Fer = Phi_Fer_[0][0]
print('Phi_Fer =',int(1000*Phi_Fer), 'mV')
print('Ratio =', int(Phi_Fer/RP_Fer))

Phi_Fer = 212 mV
Ratio = 2
```

5 The Electron Transport Chain

The overall model is described a bond graph tools file:

```
In [31]: ## File ETC_abg.py

import BondGraphTools as bgt
import PII_abg
import Cyt_abg
import PI_abg
import Fer_abg

def model():
    """
    Model of chloroplast electron transport chain
    """

    ETC = bgt.new(name='ETC')    # Create system
    PII = PII_abg.model()
    Cyt = Cyt_abg.model()
    PI = PI_abg.model()
    Fer = Fer_abg.model()
    bgt.add(ETC,PII,Cyt,PI,Fer)

    return ETC
```

5.1 Unify species in stoichiometric matrix using stoich.unify()

In [32]: `import ETC_abg`

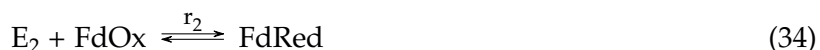
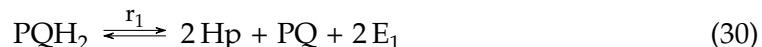
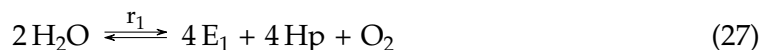
The connections are made by listing the species in common between the four components PII, Cyt, PI and Fer.

```
In [33]: ## Stoichiometry
common = ['PQ', 'PQH2', 'PcOx', 'PcRed', 'FdOx', 'FdRed', 'Hn', 'Hp']
model = ETC_abg.model()
s = st.stoich(model, quiet=True)
st.unify(s, common) # Unify using stoichiometry

## Pathway analysis
chemostats = ['H2O', 'O2', 'NADP', 'NADPH', 'Hp', 'Hn', 'P680', 'P700', 'dV']
sc = st.statify(s, chemostats=chemostats)
sp = st.path(s, sc)
```

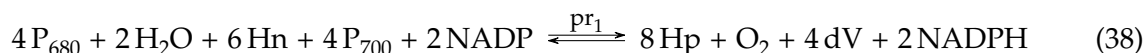
In [34]: `disp.Latex(st.sprintrl(s, chemformula=True))`

Out[34]:



In [35]: `disp.Latex(st.sprintrl(sp, chemformula=True))`

Out[35]:



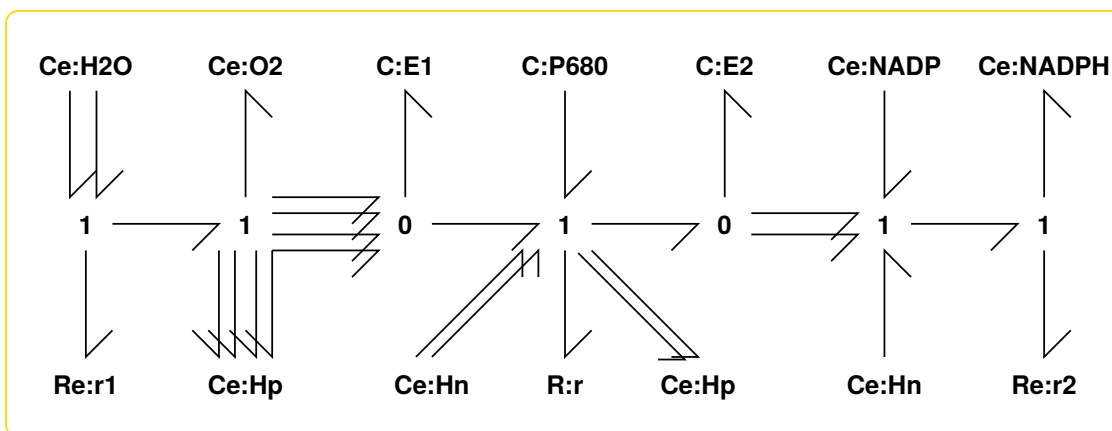
```
In [36]: ## Compute the reaction potential Phi
phi = phiData.phi_species(phi_redox, sp['species'])
Phi_ETC_ = -sp['N'].T@phi
Phi_ETC = Phi_ETC_[0][0]
print('Phi_ETC =', int(1000*Phi_ETC), 'mV')
#print('Ratio =', int(Phi_Fer/RP_Fer))
```

Phi_ETC = 7602 mV

5.2 Physically-plausible model

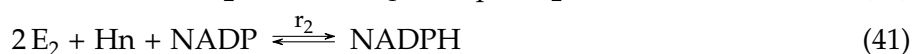
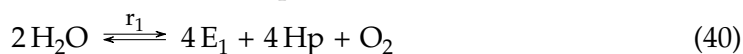
```
In [37]: disp.SVG('ppETC_abg.svg')
```

Out [37]:



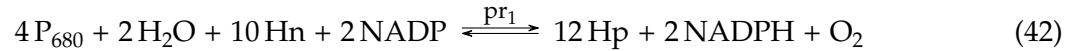
```
In [38]: import ppETC_abg
imp.reload(ppETC_abg)
## Stoichiometry
s = st.stoich(ppETC_abg.model(), quiet=True)
chemostats = ['H2O', 'O2', 'Hn', 'Hp', 'NADP', 'NADPH', 'P680']
sc = st.statify(s, chemostats=chemostats)
sp = st.path(s, sc)
disp.Latex(st.sprintrl(s, chemformula=True))
```

Out [38]:



```
In [39]: disp.Latex(st.sprintrl(sp, chemformula=True))
```

Out [39]:



In [40]: *## Compute net redox potential*

```
RP_PP = (
    - redoxData.EpH('O2/2H2O', pH=pH_p)
    + redoxData.EpH('NADP/NADPH', pH=pH_n)
    + (V680 + V700)
    - 2*(redoxData.VpH(pH_p) - redoxData.VpH(pH_n))
)

print('RP_PP =', int(1000*RP_PP), 'mV')
```

RP_PP = 1900 mV

In [41]: *## Compute the reaction potential Phi*

```
phi = phiData.phi_species(phi_redox, sp['species'])
Phi_PP_ = -sp['N'].T@phi
Phi_PP = Phi_ETC_[0][0]
print('Phi_PP =', int(1000*Phi_PP), 'mV')
print('Ratio =', int(round(Phi_PP/RP_PP)))
```

Phi_PP = 7602 mV

Ratio = 4

In [42]: *## Sanity checks*

```
print(redoxData.mV(redoxData.Phi('NAD/NADH')))
phi = phiData.phi_Std()
print(redoxData.mV(phi_redox['NAD']-phi_redox['NADH']))
print(redoxData.mV(phi['NAD']-phi['NADH']))
```

-209

-209

-220

In [43]: *## Sanity checks*

```
print(redoxData.mV(redoxData.Phi('O2/2H2O')))
print(redoxData.mV(phi_redox['O2']-2*phi_redox['H2O']))
print(redoxData.mV(phi['O2']-2*phi['H2O']))
```

4987

4987

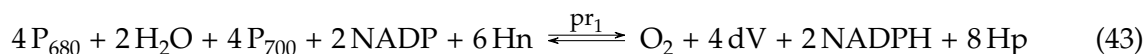
5057

5.3 Unify species in model using modular.unify()

```
In [44]: imp.reload(bgt)
import modular
imp.reload(modular)
imp.reload(st)
model = ETC_abg.model()
common = ['PQ', 'PQH2', 'PcOx', 'PcRed', 'FdOx', 'FdRed', 'Hn', 'Hp']
modular.unify(model, common, quiet=True)
s = st.stoich(model, quiet=True)
#disp.Latex(st.sprintl(s, 'N'))
#disp.Latex(st.sprintrl(s))

In [45]: chemostats = ['H2O', 'O2', 'NADP', 'NADPH', 'Hp', 'Hn', 'P680', 'P700', 'dV']
sc = st.statify(s, chemostats=chemostats)
sp = st.path(s, sc)
disp.Latex(st.sprintrl(sp, chemformula=True))
```

Out[45]:



```
In [46]: ## Compute the reaction potential Phi
phi = phiData.phi_species(phi_redox, sp['species'])
Phi_ = -sp['N'].T@phi
Phi = Phi_[0][0]
print('Phi =', redoxData.mV(Phi), 'mV')
```

Phi = 7603 mV

References

- Robert E. Blankenship. *Molecular mechanisms of photosynthesis*. Wiley Blackwell, Oxford, 2015.
- Robert E. Blankenship and Roger C. Prince. Excited-state redox potentials and the z scheme of photosynthesis. *Trends in Biochemical Sciences*, 10(10):382 – 383, 1985. ISSN 0968-0004. doi:[10.1016/0968-0004\(85\)90059-3](https://doi.org/10.1016/0968-0004(85)90059-3).
- David G Nicholls and Stuart Ferguson. *Bioenergetics 4*. Academic Press, Amsterdam, 2013.