

Additional material for: Network Thermodynamics of Biological Systems: A Bond Graph Approach

Peter Gawthrop and Michael Pan

April 26, 2022

Contents

1	Introduction	3
1.1	Some useful imports	3
1.2	Photosynthesis	4
2	Photon Energetics	5
3	Redox reactions	6
3.1	2 port Re version	6
3.2	1 port Re version	7
3.3	Redox formulation	8
3.3.1	Stoichiometry	8
3.3.2	Pathway analysis	8
4	Chemical reaction network (CRN) representation	9
4.1	Stoichiometry	10
4.2	Pathway analysis	10
5	Redox potentials and equivalent species potentials ϕ	11
6	Bond graph description of ETC complexes	13
6.1	Complex PII – Photosystem II	13
6.1.1	Stoichiometry	13
6.1.2	Pathway analysis with redox and pathway potentials	14
6.2	Complex Cyt – Cytochrome bf	15
6.2.1	Stoichiometry	15
6.2.2	Pathway analysis with redox and pathway potentials	15
6.3	Complex PI – Photosystem I	16
6.3.1	Stoichiometry	16
6.3.2	Pathway analysis with redox and pathway potentials	17
6.4	Complex Fer – Ferredoxin-NADP reductase	17
6.4.1	Stoichiometry	18
6.4.2	Pathway analysis with redox and pathway potentials	18
7	The Electron Transport Chain	19
7.1	Graphical description	19
7.1.1	Stoichiometry	19
7.1.2	Pathway analysis with redox and pathway potentials	20

7.2	Computational description	21
7.2.1	Unify species in model using <code>mbg.unify()</code>	21
8	Chloroplast	22
8.1	Modular model	22
8.2	Generate equations	23
8.3	Pathway analysis	23
8.3.1	Pathway reaction	23
8.4	Efficiency from estimated ϕ	24
8.5	Efficiency by direct calculation	25

1 Introduction

- This document contains additional material for the paper: *Network Thermodynamics of Biological Systems: A Bond Graph Approach* by Peter Gawthrop and Michael Pan.
- It illustrates how the Python package [BondGraphTools](#) can be used to create and analyse chemical reaction networks.
- It provides background to Section 4.2 *Redox Reactions*, Section 6 *Stoichiometry and Bond Graphs* and Section 7 *Example: Photosynthesis*.
- This document is **Chloroplast.pdf** (see also **Reactions.pdf**).
- The document is available as the Jupyter notebook **Chloroplast.ipynb**.

1.1 Some useful imports

```
[1]: ## Some useful imports
import BondGraphTools as bgt
print('Using BondGraphTools', bgt.version)
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt

## Stoichiometric analysis
import stoich as st

## SVG bg representation conversion
import svgBondGraph as sbg

## Stoich to BG
import stoichBondGraph as stbg

## Modular bond graphs
import modularBondGraph as mbg

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

## Data
import phiData
import redoxData

## Copy
import copy

import importlib as imp

quiet = True
chemformula = True
savefig = False
```

Using BondGraphTools 0.3.7

```
[2]: def SaveFig(savefig,name):
      if savefig:
          print('Saving',name)
          plt.savefig(name)
      else:
          print('Not saving',name)
```

```
[3]: def mV(E):
      return int(round(1000*E))
```

1.2 Photosynthesis

Photosynthesis within plant chloroplasts is the basis of life on earth Blankenship (2021), 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.

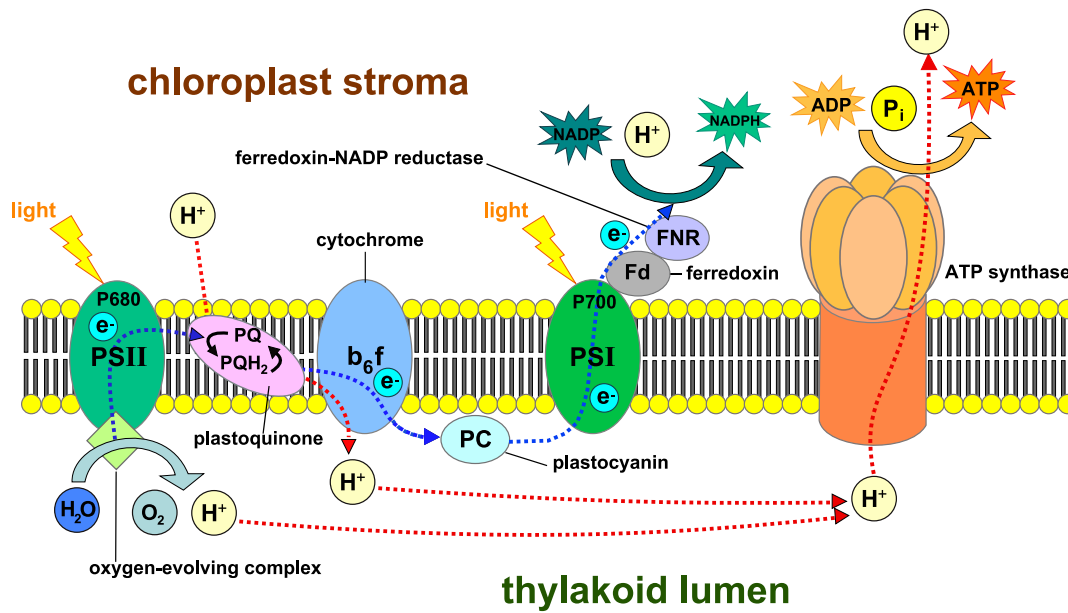
1. Photosystem II (P_{II}) which absorbs photons at 680nm and splits water releasing protons into the p-space and passing electrons to the plastoquinone(PQ)/plastoquinone(PQH₂) couple which absorbs protons from the n-space.
2. Cytochrome bf (Cyt) which passes electrons to the plastoquinone/plastoquinone 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 (P_I) 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

- See section 7 of paper.

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

```
[4]:
```



2 Photon Energetics

- See section 7.2 of paper.

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

$$\text{where } N_{av} = \text{Avogadro's number} \quad (2)$$

$$h = \text{Planck's constant} \quad (3)$$

$$c = \text{velocity of light} \quad (4)$$

$$F = \text{Faraday's constant} \quad (5)$$

$$\text{and } \lambda = \text{wavelength} \quad (6)$$

For example:

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

```
[5]: ## Plot
wavelength = np.linspace(600,800,100)
phi_680 = redoxData.V_photon(680)
phi_700 = redoxData.V_photon(700)
print(f'phi_680 = {phi_680:0.2f}')
print(f'phi_700 = {phi_700:0.2f}')
phi_wave = [phi_680,phi_700]
wave = [680,700]
one = np.ones(2)

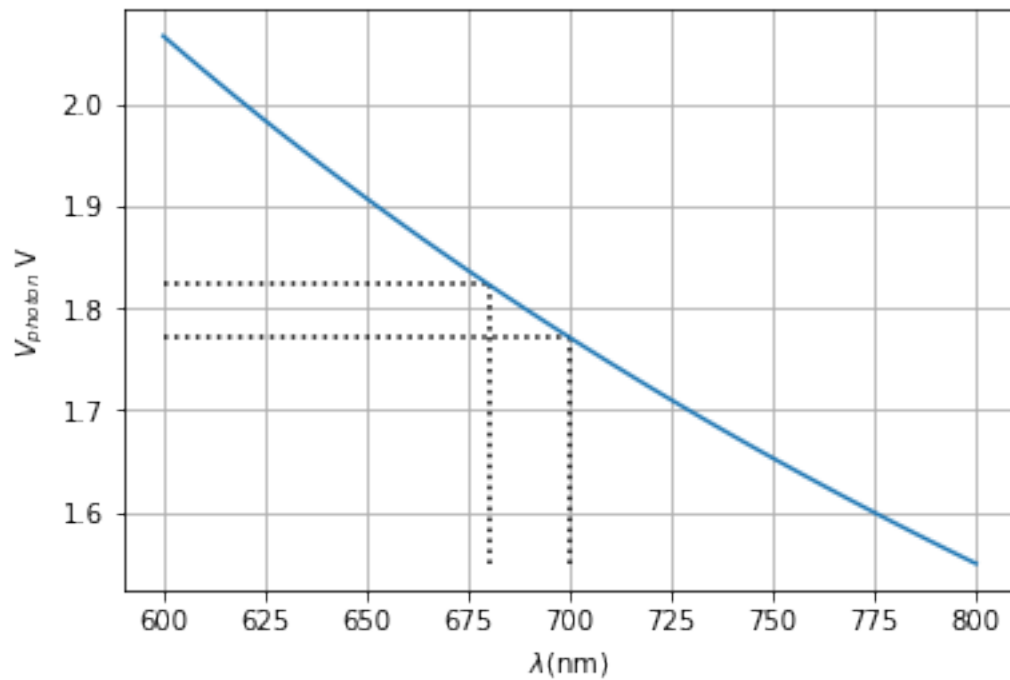
PHI = redoxData.V_photon(wavelength)
plt.plot(wavelength,PHI)
```

```
plt.hlines(phi_wave,min(wavelength)*one,wave,linestyles='dotted')
plt.vlines(wave,min(PHI)*one,phi_wave,linestyles='dotted')
plt.grid()
plt.xlabel('$\lambda$(nm)')
plt.ylabel('$V_{\text{photon}}$ V')
SaveFig(savefig,'Figs/V_photon.pdf')
```

phi_680 = 1.82

phi_700 = 1.77

Not saving Figs/V_photon.pdf



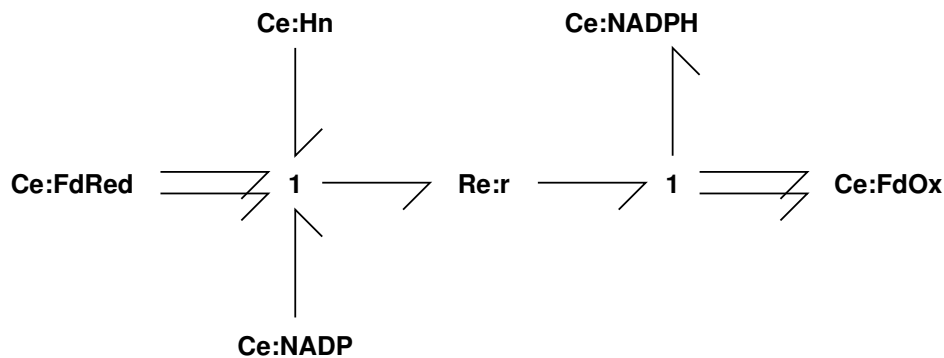
3 Redox reactions

3.1 2 port Re version

- See section 3.5 of paper.

```
[6]: ## 2 port Re version
disp.SVG('Fer0_abg.svg')
```

[6]:

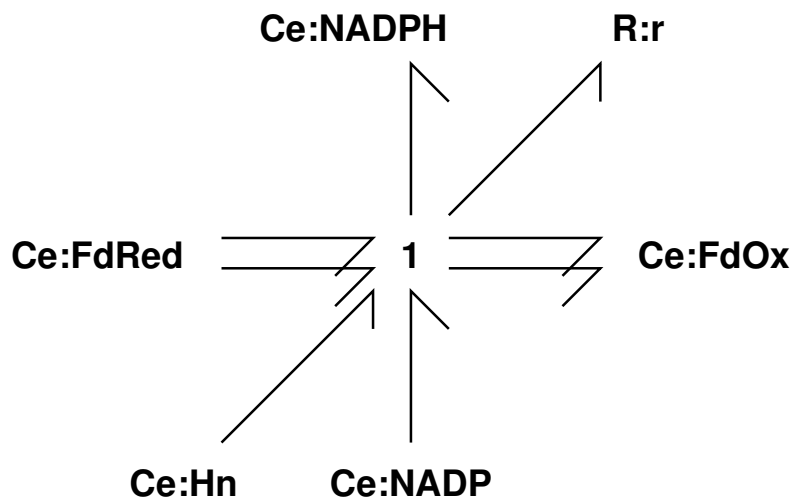


3.2 1 port Re version

- See section 3.6 of paper.

```
[7]: ## One-port Re version
disp.SVG('Fer1_abg.svg')
```

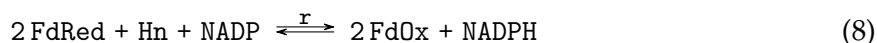
[7]:



```
[8]: ## Convert to BGtools
sbg.model('Fer1_abg.svg', convertCe=True, convertR=True, quiet=quiet)
import Fer1_abg
```

```
[9]: ## Stoichiometry
sFer1= st.stoich(Fer1_abg.model(), quiet=True)
disp.Latex(st.sprintrl(sFer1, chemformula=chemformula))
```

[9]:

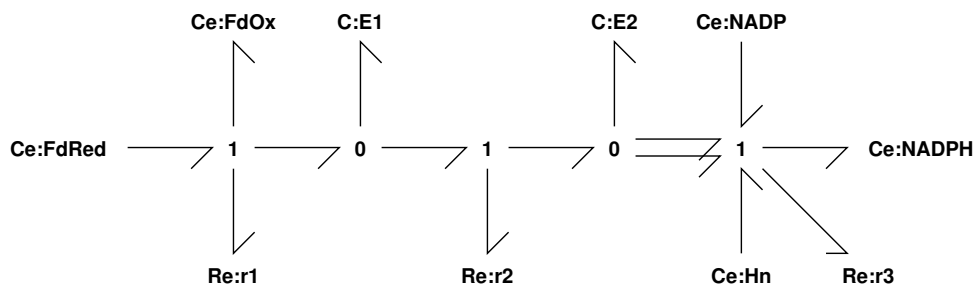


3.3 Redox formulation

This *redox reaction* can be written as two half-reactions with the following BG representation:

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

[10]:

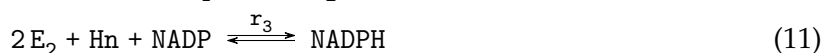


```
[11]: sbg.model('Fer_abg.svg', convertCe=True, convertR=True, quiet=quiet)
import Fer_abg
# imp.reload(Fer_abg)
```

3.3.1 Stoichiometry

```
[12]: ## Stoichiometry
sFer= st.stoich(Fer_abg.model(), quiet=True)
disp.Latex(st.sprintrl(sFer, chemformula=chemformula))
```

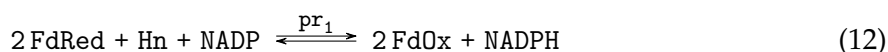
[12]:



3.3.2 Pathway analysis

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

[13]:



4 Chemical reaction network (CRN) representation

- See section 6.4 of paper.

```
[14]: disp.Latex(st.sprintl(sFer, 'species'))
```

[14]:

$$X = \begin{pmatrix} X_{E1} \\ X_{E2} \\ X_{FdOx} \\ X_{FdRed} \\ X_{Hn} \\ X_{NADP} \\ X_{NADPH} \end{pmatrix} \quad (13)$$

```
[15]: disp.Latex(st.sprintl(sFer, 'N'))
```

[15]:

$$N = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -2 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix} \quad (14)$$

```
[16]: disp.Latex(st.sprintl(sFer, 'Z'))
```

[16]:

$$Z = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (15)$$

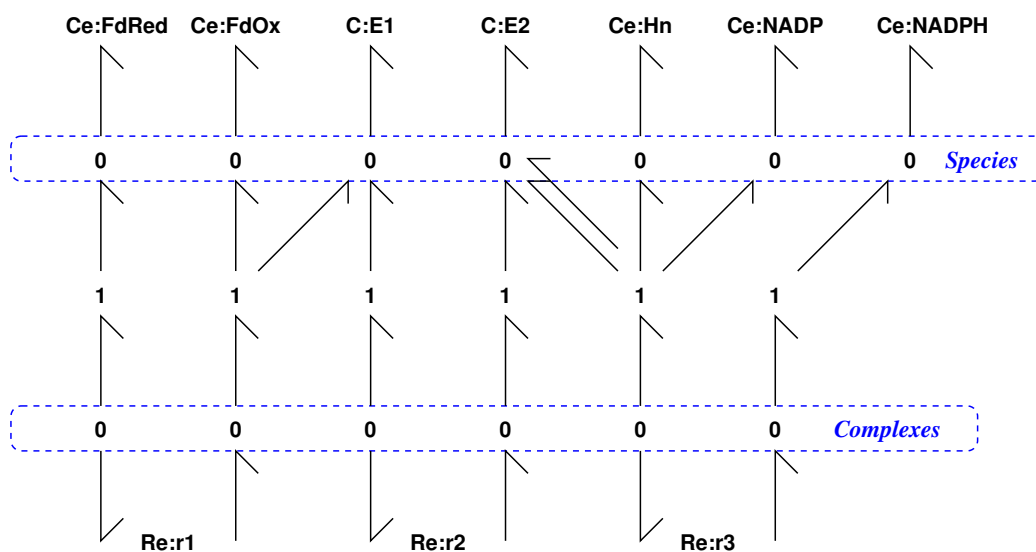
```
[17]: disp.Latex(st.sprintl(sFer, 'D'))
```

[17]:

$$D = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (16)$$

```
[18]: disp.SVG('FerCRN_abg.svg')
```

[18]:

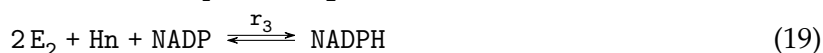


```
[19]: sbg.model('FerCRN_abg.svg', quiet=quiet)
import FerCRN_abg
# imp.reload(FerCRN_abg)
```

4.1 Stoichiometry

```
[20]: ## Stoichiometry
imp.reload(st)
sFerCRN = st.stoich(FerCRN_abg.model(), quiet=quiet)
disp.Latex(st.sprintrl(sFerCRN, chemformula=chemformula))
```

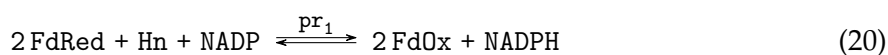
[20]:



4.2 Pathway analysis

```
[21]: ## Pathway analysis
chemostats = ['FdRed', 'FdOx', 'NADP', 'NADPH', 'Hn']
scFerCRN = st.stoich(FerCRN_abg.model(), chemostats=chemostats, quiet=True)
spFerCRN = st.path(sFerCRN, scFerCRN)
disp.Latex(st.sprintrl(spFerCRN, chemformula=chemformula))
```

[21]:



5 Redox potentials and equivalent species potentials ϕ

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 \(1985\)](#)

According to [Blankenship \(2021\)](#) (sec. 8.3) the *electrical* part of the membrane potential is small in isolated chloroplasts.

```
[22]: #disp.Latex(redoxData.table())
```

```
[23]: import 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')
```

```
0.18462122057715774
```

```
V_pH = 184 mV
```

```
V_680 = 1823 mV
```

```
V_700 = 1771 mV
```

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

      ## The electrical membrane potential is close to zero ()
      phi_redox['dV'] = VpH

      ## Put in my computed values for photon potentials
      phi_redox['P680'] = phi_680
      phi_redox['P700'] = phi_700
```

```
[25]: ## phi for ATP etc from Phi for hydrolysis
      # Phi_Hyd = 0.424
      Phi_Hyd = 35e3/st.F()
      print(f'Phi_hyd = {Phi_Hyd:0.2f}')
      phi_H2O = phi_redox['H2O']
      phi_H = phi_redox['Hn']
      Phi_A = Phi_Hyd - phi_H2O + phi_H
      print(f'Phi_A: {Phi_A:0.2f}')

      phi_ATP = Phi_A/2
      phi_ADP = phi_Pi = -Phi_A/4
```

```

phi_redox['ATP'] = phi_ATP
phi_redox['ADP'] = phi_ADP
phi_redox['Pi'] = phi_Pi

## Sanity check
Phi_Hyd_check = phi_ATP + phi_H2O -(phi_ADP + phi_Pi + phi_H)
print(f'{Phi_Hyd_check-Phi_Hyd:0.2f}')

```

```

Phi_hyd = 0.36
Phi_A: 1.18
0.00

```

```

[26]: # Print values
for spec in phi_redox.keys():
    if not (('*' in spec) or ('E' in spec) or ('+' in spec)):
        print(f'phi_{spec} = {phi_redox[spec]:0.2f} V')

```

```

phi_NADP = -0.11 V
phi_NADPH = 0.11 V
phi_NAD = -0.10 V
phi_NADH = 0.10 V
phi_O2 = 2.49 V
phi_H2O = -1.25 V
phi_P700 = 1.77 V
phi_P870 = 0.45 V
phi_P680 = 1.82 V
phi_PQ = 0.43 V
phi_PQH2 = -0.43 V
phi_PcOx = 0.19 V
phi_PcRed = -0.19 V
phi_FdOx = -0.21 V
phi_FdRed = 0.21 V
phi_Hn = -0.43 V
phi_Hp = -0.25 V
phi_dV = 0.18 V
phi_ATP = 0.59 V
phi_ADP = -0.29 V
phi_Pi = -0.29 V

```

```

[27]: ## Compatibility with Chloroplast model (but makes no difference)
phi_redox['etc_FdRed'] = phi_redox['FdRed']
phi_redox['etc_PcRed'] = phi_redox['PcRed']
phi_redox['etc_PQH2'] = phi_redox['PQH2']

```

```

[28]: ## Redox potentials with zero photon contribution
phi_redox_0 = copy.copy(phi_redox)
phi_redox_0['P680'] = phi_redox_0['P700'] = 0

```

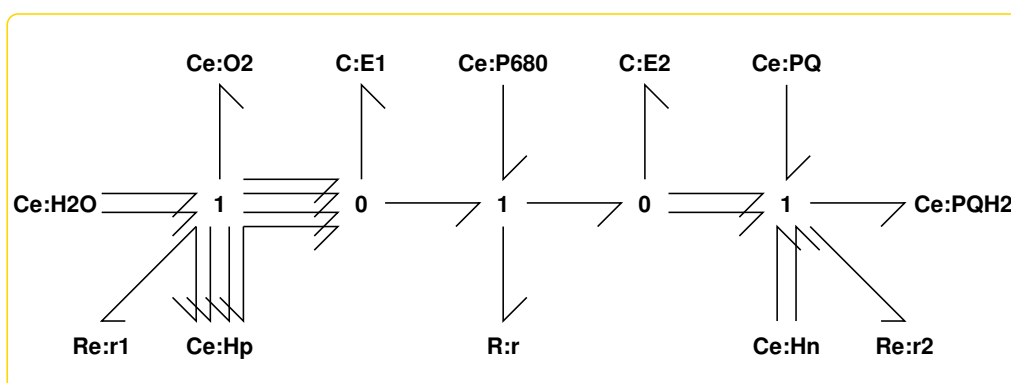
6 Bond graph description of ETC complexes

- The four ETC complexes are represented by SVG graphics which are automatically converted into BondGraphTools format.
- The stoichiometric toolbox is then used to generate the pathway-reduced equation for the complex.
- See section 7 of paper.

6.1 Complex PII – Photosystem II

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

[29]:

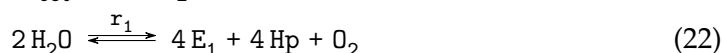


```
[30]: imp.reload(sbg)
sbg.model('PII_abg.svg',convertCe=True,convertR=True,quiet=quiet)
import PII_abg
# imp.reload(PII_abg)
```

6.1.1 Stoichiometry

```
[31]: ## Stoichiometry
sPII = st.stoich(PII_abg.model(),quiet=quiet)
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=chemformula))
```

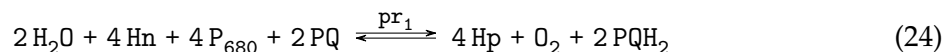
[31]:



6.1.2 Pathway analysis with redox and pathway potentials

```
[32]: ## Pathway analysis
disp.Latex(st.sprintrl(spPII,chemformula=chemformula))
```

[32]:



```
[33]: ## 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')
```

```
1.0006212205771576
0.8
0
0.0
RP_PII = 822 mV
```

```
[34]: ## 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(f'Phi_PII = {int(1000*Phi_PII)} mV')
print(f'Ratio = {(Phi_PII/RP_PII):.2f}')
```

```
Phi_PII = 3290 mV
Ratio = 4.00
```

```
[35]: ## Photon contribution
n_phot=4
print(f"Photon contribution = {mV(n_phot*redoxData.E('P680+/P680*'))} mV")
```

```
Photon contribution = 3200 mV
```

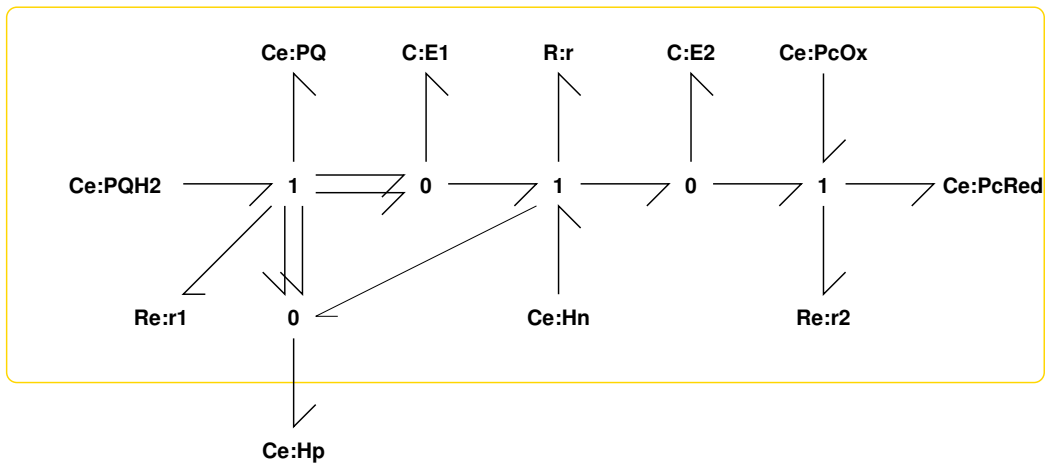
```
[36]: ## Potential per proton
n_prot = 4
print(f'Potential per proton = {mV(Phi_PII/n_prot)}')
```

```
Potential per proton = 823
```

6.2 Complex Cyt – Cytochrome bf

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

[37]:

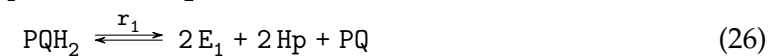


```
[38]: sbg.model('Cyt_abg.svg', convertR=True, convertCe=True, quiet=quiet)
import Cyt_abg
# imp.reload(Cyt_abg)
```

6.2.1 Stoichiometry

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

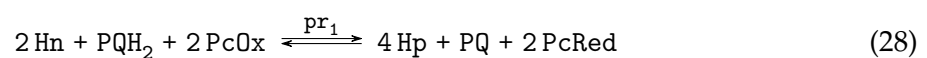
[39]:



6.2.2 Pathway analysis with redox and pathway potentials

```
[40]: disp.Latex(st.sprintrl(spCyt, chemformula=chemformula))
```

[40]:



```
[41]: ## 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')

```

RP_Cyt = 11 mV

```

[42]: ## 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(f'Ratio = {(Phi_Cyt/RP_Cyt):0.2f}')

```

Phi_Cyt = 22 mV

Ratio = 2.00

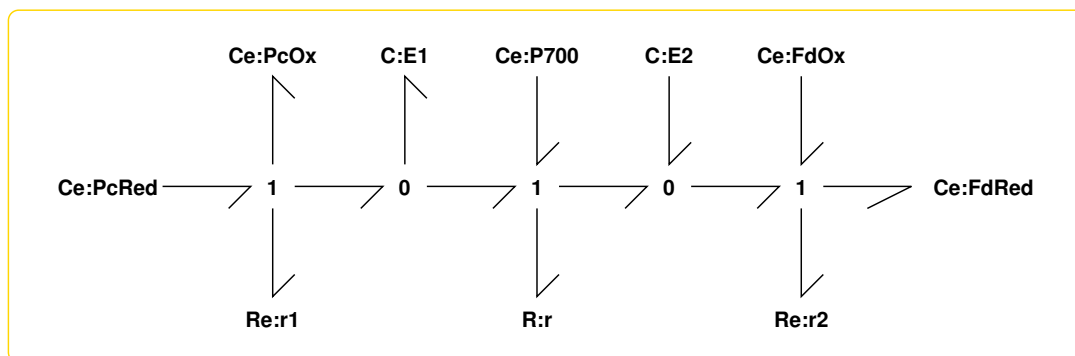
6.3 Complex PI – Photosystem I

```

[43]: disp.SVG('PI_abg.svg')

```

[43]:



```

[44]: sbg.model('PI_abg.svg',convertR=True,convertCe=True,quiet=True)
import PI_abg
# imp.reload(PI_abg)

```

6.3.1 Stoichiometry

```

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

```

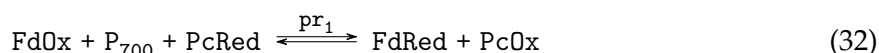
[45]:



6.3.2 Pathway analysis with redox and pathway potentials

```
[46]: disp.Latex(st.sprintrl(spPI,chemformula=chemformula))
```

[46]:



```
[47]: ## 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

```
[48]: ## 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(f'Ratio = {(Phi_PI/RP_PI):0.2f}')
```

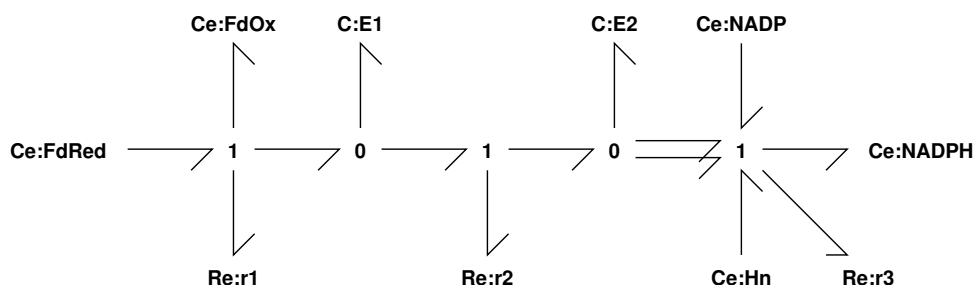
Phi_PI = 961 mV

Ratio = 1.00

6.4 Complex Fer – Feredoxin-NADP reductase

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

[49]:



```
[50]: sbg.model('Fer_abg.svg',convertR=True,convertCe=True,quiet=quiet)
import Fer_abg
# imp.reload(Fer_abg)
```

6.4.1 Stoichiometry

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

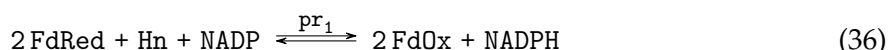
[51]:



6.4.2 Pathway analysis with redox and pathway potentials

```
[52]: disp.Latex(st.sprintrl(spFer,chemformula=chemformula))
```

[52]:



```
[53]: ## 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

```
[54]: ## 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 =',mV(Phi_Fer), 'mV')
print(f'Ratio = {(Phi_Fer/RP_Fer):0.2f}')
```

Phi_Fer = 212 mV

Ratio = 2.00

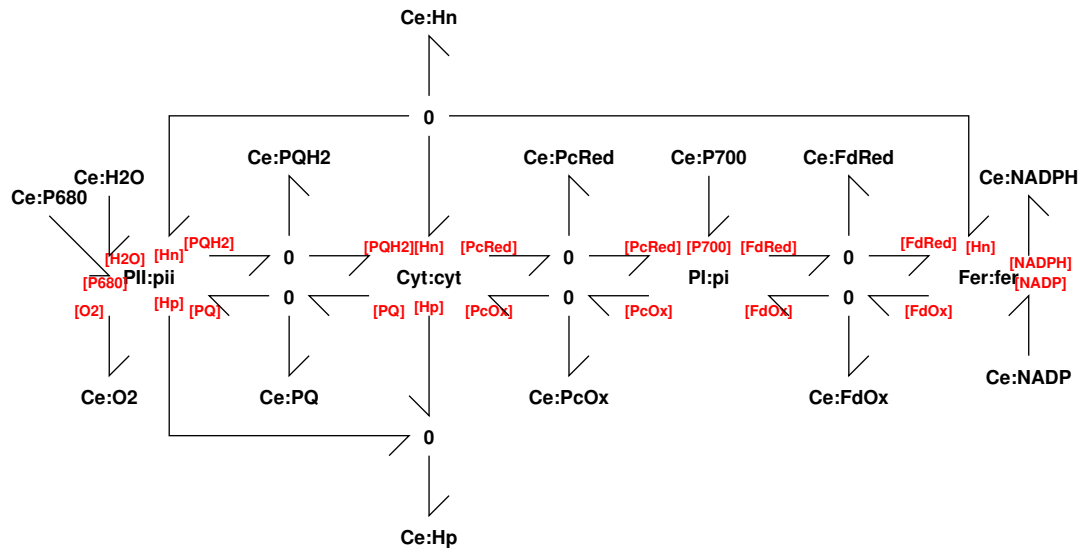
7 The Electron Transport Chain

- See section 7.1 of paper.

7.1 Graphical description

```
[55]: disp.SVG('ETCg_abg.svg')
```

[55]:



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

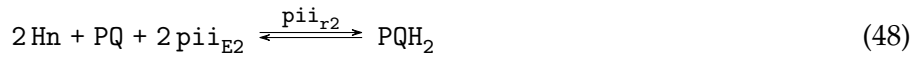
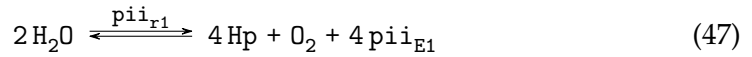
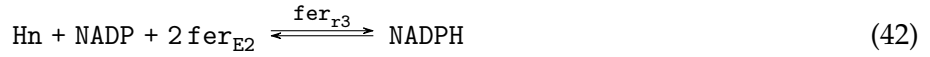
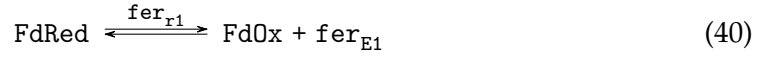
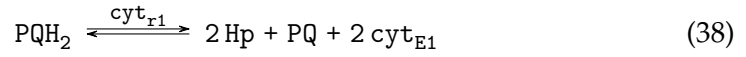
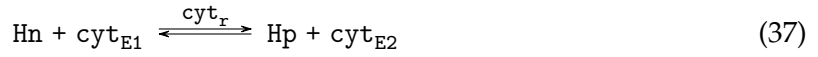
```
Creating subsystem: Cyt:cyt
Creating subsystem: Fer:fer
Creating subsystem: PI:pi
Creating subsystem: PII:pii
```

```
[56]: <module 'ETCg_abg' from '/home/peterg/WORK/Dissemination/Edmund/SpecialIssue/
↳ Exa
mples/Chloroplast/ETCg_abg.py'>
```

7.1.1 Stoichiometry

```
[57]: ## Stoichiometry
sETCg = st.stoich(ETCg_abg.model(), quiet=True)
disp.Latex(st.sprintrl(sETCg, chemformula=chemformula))
```

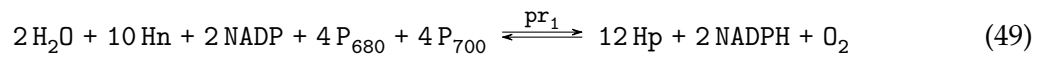
[57]:



7.1.2 Pathway analysis with redox and pathway potentials

```
[58]: chemostats = ['P680', 'P700', 'H2O', 'O2', 'Hn', 'Hp', 'NADP', 'NADPH']
      scETCg = st.stoich(ETCg_abg.model(), chemostats=chemostats, quiet=True)
      spETCg = st.path(sETCg, scETCg)
      disp.Latex(st.sprintrl(spETCg, chemformula=chemformula, all=True))
```

[58]:



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

```
['FdRed', 'H2O', 'Hn', 'Hp', 'NADP', 'NADPH', 'O2', 'P680', 'P700', 'PQH2',
 'PcRed']
```

Phi = 7603 mV

```
[60]: ## Flatten
      stbg.model(sETCg, filename='ETCg_abg')
      import ETCg_abg
      imp.reload(ETCg_abg)
```

```
[60]: <module 'ETCg_abg' from '/home/peterg/WORK/Dissemination/Edmund/SpecialIssue/
      ↪mples/Chloroplast/ETCg_abg.py'>
```

7.2 Computational description

The overall model is described a bond graph tools file:

```
[61]: ## File ETC_abg.py

## Import the modules
import PII_abg
import Cyt_abg
import PI_abg
import Fer_abg

## Create the model using BondGraphTools
def model():
    """
    Model of chloroplast electron transport chain
    """

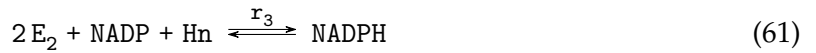
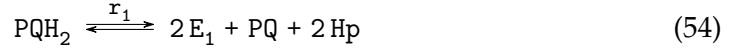
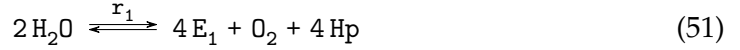
    ETC = bgt.new(name='ETC')    # Create system
    PII = PII_abg.model()
    Cyt = Cyt_abg.model()
    PI = PI_abg.model()
    Fer = Fer_abg.model()
    bgt.add(ETC,PII,Cyt,PI,Fer)

    return ETC
```

7.2.1 Unify species in model using mbg.unify()

```
[62]: # import ETC_abg
ETC = model()
common = ['PQ','PQH2','PcOx','PcRed','FdOx','FdRed','Hn','Hp']
mbg.unify(ETC,common,quiet=quiet)
ss = st.stoich(ETC,quiet=quiet)
disp.Latex(st.sprintrl(ss,chemformula=chemformula))
```

```
[62]:
```



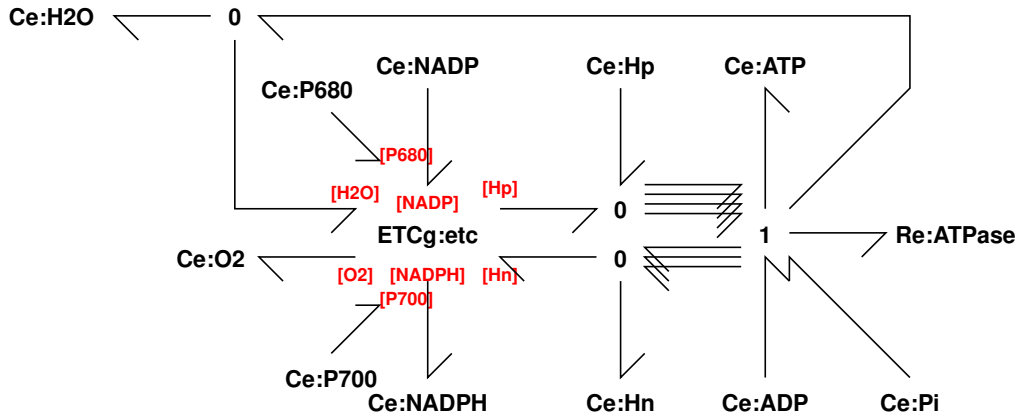
8 Chloroplast

- See section 7.1 of paper.

8.1 Modular model

```
[63]: disp.SVG('Chloroplast_abg.svg')
```

[63]:



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

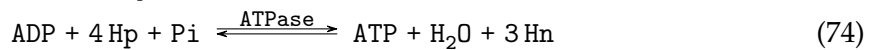
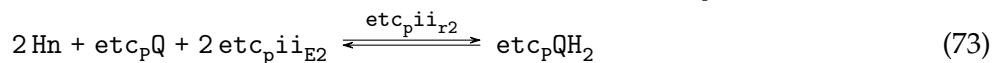
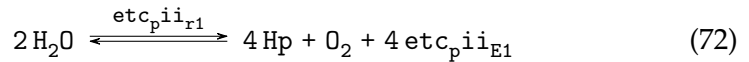
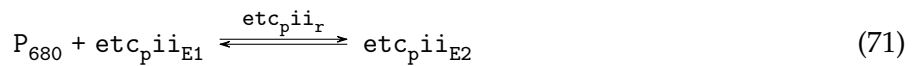
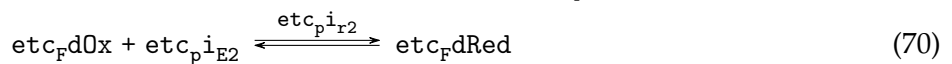
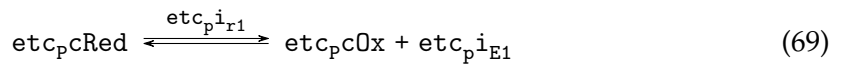
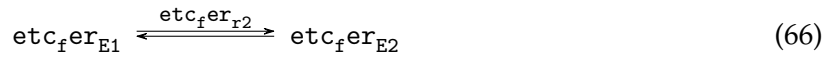
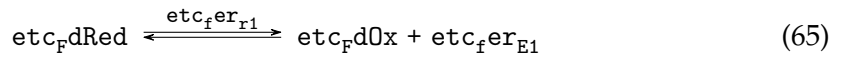
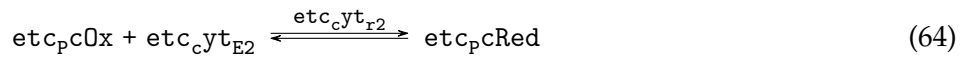
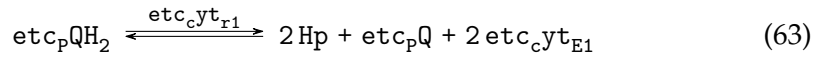
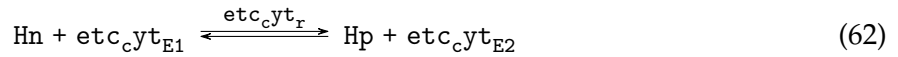
Creating subsystem: ETCg:etc

```
[64]: <module 'Chloroplast_abg' from '/home/peterg/WORK/Dissemination/Edmund/
      ↪SpecialIs
      sue/Examples/Chloroplast/Chloroplast_abg.py'>
```

8.2 Generate equations

```
[65]: ## Stoichiometry
sChloroplast = st.stoich(Chloroplast_abg.model(), quiet=True)
disp.Latex(st.sprintrl(sChloroplast, chemformula=chemformula))
```

[65]:



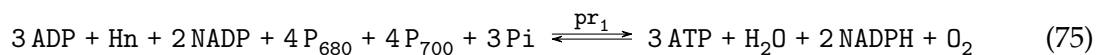
8.3 Pathway analysis

```
[66]: # imp.reload(st)
chemostats = ['P680', 'P700', 'H2O', 'O2', 'Hn', 'NADP', 'NADPH', 'ATP', 'ADP', 'Pi']
scChloroplast = st.stoich(Chloroplast_abg.
      ↪model(), chemostats=chemostats, quiet=True)
spChloroplast = st.path(sChloroplast, scChloroplast)
```

8.3.1 Pathway reaction

```
[67]: disp.Latex(st.sprintrl(spChloroplast, chemformula=chemformula, all=True))
```

[67]:



8.4 Efficiency from estimated ϕ

```
[68]: ## Compute the reaction potential Phi
species = spChloroplast['species']
# species = chemostats
# print(species)

phi = phiData.phi_species(phi_redox,species)

Phi_ = -spChloroplast['N'].T@phi
Phi= Phi_[0][0]
print(f'Phi = {Phi:0.2f} V')

## Photo input
# print(phi_redox['P680'] , phi_redox['P700'])
Phi_phot = 4*(phi_redox['P680'] + phi_redox['P700'])
print(f'Phi_phot = {Phi_phot:0.2f} V')

##Outputs
Phi_out = Phi_phot - Phi
print(f'Phi_out = {Phi_out:0.2f} V')

## Efficiency
efficiency = Phi_out/Phi_phot
print(f'Efficiency = {(efficiency*100):.1f} %')
```

```
Phi = 8.73 V
Phi_phot = 14.38 V
Phi_out = 5.65 V
Efficiency = 39.3 %
```

```
[69]: ## Compute the reaction potential Phi but with no photons
phi = phiData.phi_species(phi_redox_0,species)

Phi_ = -spChloroplast['N'].T@phi
Phi_out = -Phi_[0][0]
print(f'Phi_out = {Phi_out:0.2f} V')

## Photo input
# print(phi_redox['P680'] , phi_redox['P700'])
Phi_phot = 4*(phi_redox['P680'] + phi_redox['P700'])
print(f'Phi_phot = {Phi_phot:0.2f} V')

## Efficiency
efficiency = Phi_out/Phi_phot
print(f'Efficiency = {(efficiency*100):.1f} %')
```

```
Phi_out = 5.65 V
Phi_phot = 14.38 V
Efficiency = 39.3 %
```


8.5 Efficiency by direct calculation

- See section 7.2 of paper.

```
[70]: def getPhi(name):  
    redox_data = redoxData.data()  
    n = redox_data[name]['electrons']  
    E = redox_data[name]['E7']  
    Phi = n*E  
    print(f'{name}:\t E = {E} V;\t n = {n};\t Phi = {Phi} V')  
    return Phi
```

```
[71]: ## Photons  
Phi_phot = 4*(phi_redox['P680'] + phi_redox['P700'])  
  
print(f'Phi_phot = {Phi_phot:0.2f} V')  
## NADPH  
Phi_1 = getPhi('NADP/NADPH')  
  
## O2  
Phi_2 = getPhi('O2/2H2O')  
  
## ATP Hydrolysis  
print(f'Hydrolysis:\t Phi = {Phi_Hyd:0.3f}')
```

```
Phi_phot = 14.38 V  
NADP/NADPH:      E = -0.324 V;    n = 2;   Phi = -0.648 V  
O2/2H2O:         E = 0.816 V;     n = 4;   Phi = 3.264 V  
Hydrolysis:      Phi = 0.363
```

```
[72]: ## All  
Phi_chem = -2*Phi_1 + Phi_2 + 3*Phi_Hyd  
print(f'Phi_chem = {Phi_chem:0.2f}')
```

```
Phi_chem = 5.65
```

```
[73]: ## Efficiency  
efficiency = Phi_chem/Phi_phot  
print(f'Efficiency = {(efficiency*100):.1f} %')
```

```
Efficiency = 39.3 %
```

```
[ ]:
```

References

- Robert E. Blankenship. *Molecular mechanisms of photosynthesis*. Wiley, Chichester, UK, 3rd edition, 2021.
- 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.

David G Nicholls and Stuart Ferguson.
Amsterdam, 2013.

Bioenergetics 4.

Academic Press,