2021

AMN Module 3

FLUX BALANCE ANALYSIS & METABOLIC FLUX ANALYSIS

ALJOSCHA WAHL AND TIMMY PAEZ WATSON

Contents

1 Introduction

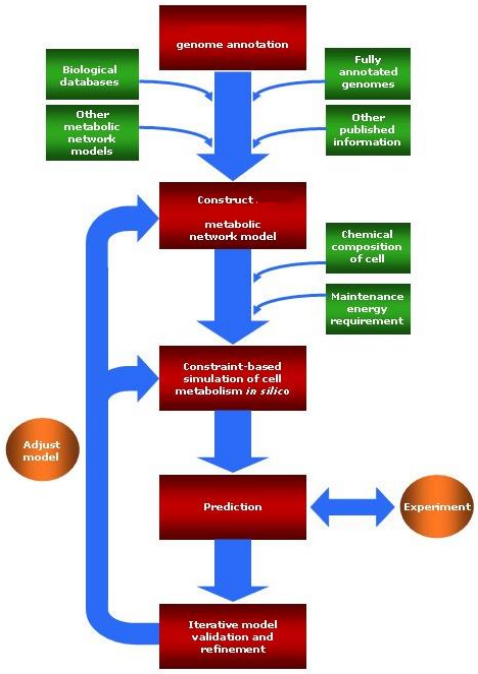
Enzymes catalyse the conversion of carbon and electron source(s) towards biomass and products. Metabolic Flux Analysis (MFA) is used for a quantitative calculation of intracellular carbon fluxes from extracellular measurements. From the quantitative analysis we can:

- Characterize the current metabolic state of a system(flux phenotype),
- See which fluxes are highly active,
- Understand the interactions in the network – e.g. calculate where cofactors (ATP, NADH, etc.) are produced and consumed,
- Derive metabolic engineering targets – e.g. elimination of by-product forming pathways and increase precursor availability.

The metabolic model can be used for further applications like

- calculation of theoretical yields
- prediction of gene knock-out phenotypes.

Clearly, the use of the model depends on its completeness and its correctness. Nowadays many organisms (~ 1600) have been sequenced and annotated opening the possibility for genome scale metabolic networks (Figure 1-1). **Genome-scale models** are available for the most important research strains – *S. cerevisiae*, *E. coli*, CHO and others.



Organism	Genes	Genes in	Metabolites	Reactions
BACTERIA				
Bacillus subtilis	4,114	844	988	1020
Bacillus subtilis	4,114	534	456	563
Corynebacterium glutamicum	3,002	227	423	502
Escherichia coli	4,405	1260	1039	2077
Helicobacter pylori	1,632	291	340	388
Helicobacter pylori	1,632	341	485	476
Mycobacterium tuberculosis	4,402	661	828	939
Mycoplasma genitalium	521	189	274	262
ARCHAEA				
Methanosarcina barkeri	5,072	692	558	619
Halobacterium salinarum	2,867	490	557	711
EUKARYOTES				
Aspergillus nidulans	9,451	666	732	794
Homo sapiens	28,783	1,496	2,766	3,311
Mus musculus	28,287	473	872	1,220
Saccharomyces cerevisiae	6,183	904	713	1,412

Figure 1-1: Genome scale metabolic modelling is based on the genome sequence, annotation (e.g. BLAST search against annotated and curated genomes) and iterative refinement of the reaction network by prediction and experimental validation.

While small metabolic networks can still be handled 'by hand', genome scale models consist of up to 3300 reactions. Even mid-sized metabolic models (> 30 reactions) are tricky to calculate 'by hand', so in this module you will learn the basic computational tools to analyse metabolic networks. Throughout this syllabus, you will learn the following analyses:

2. Identify intracellular balances of a metabolic map.
3. Translate a metabolic map into a 'Stoichiometric' matrix.
4. Basic properties of the metabolic network:
    a. Degree of freedom and type of networks
    b. Blocked reactions
    c. Non identifiable fluxes (internal cycles)
    d. Conserved moieties
5. Calculation of intracellular fluxes from measurements
6. Flux Balance Analysis (FBA): optimization techniques to maximize or minimize fluxes.

2 Metabolic flux analysis (MFA) – Intracellular balances

The modelling for MFA starts with balancing all intracellular metabolite pools. An example network is shown in Figure 2-1.

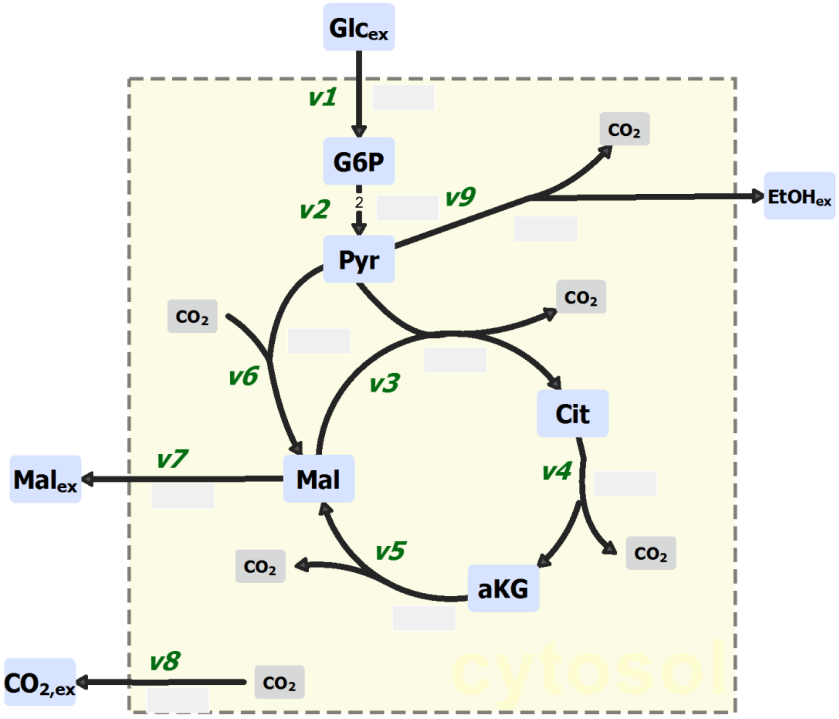


Figure 2-1: Example model of a metabolic map. Metabolites are connected by fluxes (v_i).

In the extracellular space we find Glc_{ex}, $EtOH_{ex}$, Mal_{ex} and $CO_{2,ex}$ that are in the liquid phase (we won't consider CO_2 in the gas phase for now). The mass balances of the extracellular pools can be obtained based on the theory that you learnt during Module 1 (Black box modelling). A fresh reminder, these balances look like this:

$$\begin{aligned}
\frac{d\left(VC_{Glcex}\right)}{dt} &= q_S C_X V + \phi_{in} C_{Glc,ex} - \phi_{out} C_{Glc,ex} \\
\frac{d\left(VC_{EtOH}\right)}{dt} &= q_{EtOH} C_X V - \phi_{out} C_{EtOH} \\
\frac{d\left(VC_{Mal,ex}\right)}{dt} &= q_{Mal,ex} C_X V - \phi_{out} C_{Mal,ex} \\
\frac{d\left(VC_{CO2}\right)}{dt} &= q_{CO2} C_X V - \phi_{out} C_{CO2,ex}
\end{aligned} \tag{1}$$

These balances together with measurements of the process allow us to calculate all respective rates (q_i). However, this was already done in Module 1.

Now, in the intracellular space we find G6P, Pyr, Cit, Mal, aKG and CO_2. The balance of the intracellular pools can be derived from the reaction network:

$$\begin{aligned}
\frac{d\,\mathrm{G6P}}{dt} &= v_1 - v_2 \\
\frac{d\,\mathrm{Pyr}}{dt} &= 2\,v_2 - v_3 - v_6 - v_9 \\
\frac{d\,\mathrm{Cit}}{dt} &= v_3 - v_4 \\
\frac{d\,\mathrm{aKG}}{dt} &= v_4 - v_5 \\
\frac{d\,\mathrm{Mal}}{dt} &= v_5 + v_6 - v_3 - v_7 \\
\frac{d\,\mathrm{CO_2}}{dt} &= v_4 + v_5 + v_9 - v_6 - v_8
\end{aligned} \tag{2}$$

Note that the balance for *Pyr* includes 2* V_2 as indicated in the metabolic map. The left hand side of these balances can be expressed by a vector dMet/dt:

$$\frac{dMet}{dt} = \left(\frac{dG6P}{dt}, \frac{dPyr}{dt}, \frac{dCit}{dt}, \frac{daKG}{dt}, \frac{dMal}{dt}, \frac{dCO_2}{dt} \right)^T \tag{3}$$

The metabolism is now driven to steady-state conditions, e.g. by cultivation in a chemostat. In steady-state, concentrations and associated fluxes are not changing and Eq. (2) changes from a set of differential equations into a set of linear equations for reaction rates:

$$\begin{aligned}
0 &= v_1 - v_2 \\
0 &= 2\,v_2 - v_3 - v_6 - v_9 \\
0 &= v_3 - v_4 \\
0 &= v_4 - v_5 \\
0 &= v_5 + v_6 - v_3 - v_7 \\
0 &= v_4 + v_5 + v_9 - v_6 - v_8
\end{aligned} \tag{4}$$

Independent of the type of reaction and network such linear equation systems are always obtained. These can be formulated conveniently using matrix algebra. Therefore, we define a vector containing the unknowns (fluxes) **v**:

$$\mathbf{v} = \left(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9 \right)^T \tag{5}$$

3 Metabolic flux analysis (MFA) – Define a 'Stoichiometric' matrix

Having defined these vectors, the equations in the balances (2) can be rewritten using matrix calculus. This is also called a Stoichiometric matrix (mainly represented with the symbol **S**). This matrix shows the relations between metabolites and fluxes. For the example in (2), the corresponding **S** matrix would look like this:

$$\mathbf{S} = \begin{array}{c} \\ \mathrm{G6P} \\ \mathrm{Pyr} \\ \mathrm{Cit} \\ \mathrm{aKG} \\ \mathrm{Mal} \\ \mathrm{CO_2} \end{array} \begin{array}{c} \begin{array}{ccccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_S & v_9 \end{array} \\ \begin{bmatrix} 1 & -1 & & & & & & & \\ & 2 & -1 & & & -1 & & & -1 \\ & & 1 & -1 & & & & & \\ & & & 1 & -1 & & & & \\ & & -1 & & 1 & 1 & -1 & & \\ & & 1 & 1 & 1 & & & -1 & 1 \end{bmatrix} \end{array} \tag{6}$$

The rows in the matrix S represent the balances of intracellular metabolites, the columns represent the reactions. The values represent the stoichiometric coefficients of the particular reaction. The matrix has to be written in the order of the defined vectors **v** and d**x**/dt. Having this matrix the equation system in Eq. (2) can now be rewritten as:

$$\frac{d\mathbf{x}}{dt} = \mathbf{S}\,\mathbf{v} \tag{7}$$

At steady state, this simplifies to:

$$\mathbf{0} = \mathbf{S}\,\mathbf{v} \tag{8}$$

Equation (8) is the basic constraint of these types of models. For this reason, most metabolic models that consider the steady state assumption are also called 'constraint based models'.

To analyse the properties of this system, different calculations should be performed. This is the topic of the following subchapter.

4 Basic properties of a metabolic model

Now you know how to build a basic stoichiometric matrix by balancing the intracellular metabolites of a metabolic network together with the reactions (sometimes referred to as fluxes). This can be done with any type of reaction network, however it is very important to analyse the properties of the system before moving into more complicated analyses.

4.1 Degrees of freedom and type of system

The **degree of freedom** of the system reflects the amount of 'free' variables that can be changed **independently**. A low number (e.g. 1 or 2) indicates that the network is very 'rigid' – two fluxes are enough to define the whole network function.

The degrees of freedom (n_{df}) are calculated from the number of fluxes to be determined (size of **v**) and the number of equations without redundancies or dependencies (number of metabolites that are independent). The independent metabolites (rows of S) can be calculated by analysing the rank of the matrix (rank(**S**)).

$$n_{df} = \dim(\mathbf{v}) - \mathrm{rank}(\mathbf{S}) \tag{9}$$

In the example presented in equation (6), a total of 9 fluxes are present and the rank is 6 (because all metabolites are independent). Thus, there are 3 degrees of freedom. This number is also very relevant for the calculation of fluxes – to determine the network fluxes at least n_{df} measurements are needed. You will learn how to exactly determine dependencies between fluxes or metabolites in the following sections (redundant systems and conserved moieties).

Based on the degrees of freedom and available measurements, metabolic networks can be classified (Klamt et al. 2002).

- **Underdetermined**: measured rates < n_{df} ; there are not enough linearly independent constraints for computing all rates of v, uniquely.
  Determined: measured rates = n_{df} ; there are enough linearly independent constraints for computing all rates of v uniquely.
- **Redundant**: measured rates > n_{df} ; There are more measurements than degrees of freedom which would directly solve the system. This type of cases does not appear with the kind of systems in (6), but appears when multiple metabolite balances are dependent on each other. These kind of systems are not possible to be solved.

Although this classification seems straightforward, there are multiple special cases that might lead to combinations of the types of systems. These special cases arise when there are subsystems in the network that lead to either a redundancy or an underdetermined system, while the overall system can be classified differently. For example, there are cases of networks that can be underdetermined as a whole but redundant within an internal cycle at the same time.

4.2 Blocked reactions

Suppose we have the following metabolic network:

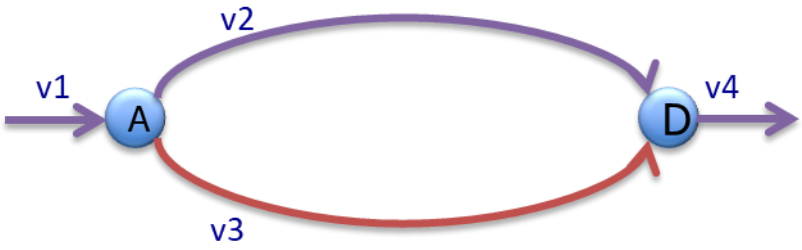


Figure 4-1: Example model of simple metabolic network that is redundant and underdetermined.

This reaction network has two observable problems. First, the system is underdetermined for v2 and v3. If we measure 2 rates (say v1 and v4), the information does not allow us to calculate a solution on the flux distribution to v2 or v3. These can basically take any value.

Additionally, the system is redundant for v1 and v4. The flux of v1 should be equal to the flux of v4, so measuring both rates would be of no use to solve the system.

For the system on Figure 4-1, we can study the so called 'spanning vectors', which indicate the possible combination of solutions of the network. The spanning vectors are obtained by doing a null space analysis of the stoichiometric matrix (**S**). In the specific case, we would obtain the following:

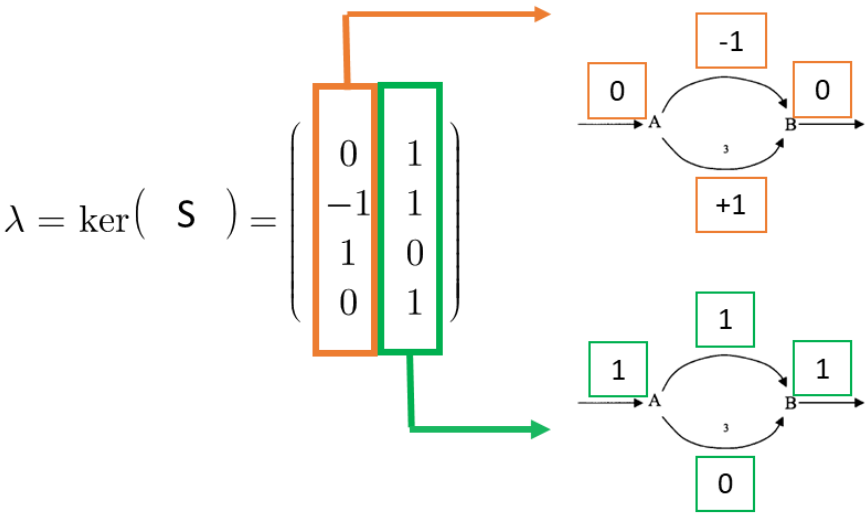
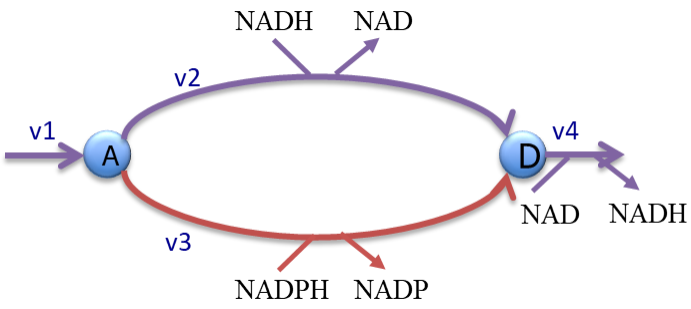


Figure 4-2: Spanning vectors obtained from the null space analysis of metabolic network in Figure 4-1. The resulting vectors show the possible combinations of solutions, and those solutions are further represented to the right.

In the example, all the rows (representing fluxes) of the null space analysis have all at least one non-zero entry. This indicates that all the fluxes could potentially contribute to the solution of the system of equations.

Now, we will expand the information on the previous network by adding cofactors involved in the reactions. When this information is added, the resulting null space indicates only one combination of fluxes leading to a solution.



$$\lambda = \ker\left(\quad S \quad \right) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Figure 4-3: (Left) Metabolic network including cofactor use. (Right) Spanning vector resulting from the null space analysis of the stoichiometric matrix.

Now that more information has been added to the system, the null space analysis indicates only one possible combination of fluxes that could exist as part of the solution. In this case, the 3rd row has a 0 entry, which indicates that this flux will always be 0 regardless of the values of the other fluxes. The specific reason for v3 to be a blocked reaction in this case is that it uses a metabolite (NADPH) and produces another one (NADP) that are not produced or consumed by any other reactions of the network. Taking into account equation (8) assuming steady state, this flux would not be possible.

This is a blocked reaction, and the way to identify them is by applying the null space analysis of **S** and searching for zero entries. Below you can find the Python code used to calculate blocked reactions.

Work with the jupyter notebook file

You can work with this example in Python (Jupyter notebook format) on Brightspace (M2_basic_properties).

```
'''Metabolic network from syllabus Figure 4-3. Identifying blocked reactions'''

lab_flx = [ 'v1', 'v2', 'v3', 'v4' ]     # Labels of fluxes
lab_met = [ 'A', 'B', 'NADPH', 'NADP' ]  # Labeles of metabolites

S = np.array( [[ 1, -1, -1,  0 ],    # A
               [ 0,  1,  1, -1 ],    # B
               [ 0,  0,  1,  0 ],    # NADPH
               [ 0,  0,  -1, 0 ]])   # NADP

SV = null_space( S )   # Calculate spanning vector
# check for fluxes that are NOT present in the nullspace
for ii in range( SV.shape[0] ):
    # find indices of the zero entries in ii-th row:
    if all( abs( SV[ii,:] ) < 1e-12 ) :
        print( 'Reaction {:s} is blocked'.format( lab_flx[ ii ] ) )

```

```
Reaction v3 is blocked
```

This script has been compiled as a function called **check_blocked_reacations**(S, v_lab); and will be uploaded to the functions file on brightspace.

4.3 Non-identifiable fluxes

Another important property of a metabolic network is the presence of fluxes that are non-identifiable considering the current measured rates. This is very common when dealing with internal cycles that can run at any flux independently from the external fluxes of the system. It does not mean that the reactions of the cycle are correct or wrong, but it means that the fluxes obtained from a solution will not be unique. Identifying these fluxes can guide us to perform additional measurements to gain insights on these measurements.

Let us consider the example of the network from Figure 4-1 (without involvement of cofactors). The system is composed of 4 variables (rates) and 2 metabolites. Thus, it is assumed that by measuring two fluxes we can solve the system. And as we analysed previously, flux v_1 should be equal to flux v_4.

To define the internal cycles, we will add a measurement of v1 to be equals to 0 by extending the **S** matrix with an additional row (also called matrix **C**) indicating the measured fluxes as follows:

$$\underbrace{\begin{pmatrix} 1 & -1 & -1 & \\ & 1 & 1 & -1 \end{pmatrix}}_{S} \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}}_{v} = 0 \xrightarrow{(v_1 = 0)} \underbrace{\begin{pmatrix} 1 & -1 & -1 & \\ & 1 & 1 & -1 \\ 1 & & & \end{pmatrix}}_{\begin{pmatrix} S \\ C \end{pmatrix}} \underbrace{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}}_{v} = 0 \Rightarrow \begin{aligned} 0 &= v_1 - v_2 - v_3 \\ 0 &= v_2 + v_3 - v_4 \\ 0 &= v_1 \end{aligned} \tag{10}$$

The system of equations from (10) represents a network in which no external input is given (in this case v_1 and v_4 are 0). Similarly as done with the blocked reactions analysis, we now perform a null space analysis of the (S; C) matrix and analyse the obtained spanning vectors.

$$\lambda = \ker \begin{pmatrix} \mathbf{S} \\ \mathbf{C} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} \tag{11}$$

The resulting vector shows that there is a solution of the system where not all fluxes are 0. This is strange, because the system doesn't receive any input, but it indicates that the fluxes v_2 and v_3 form the internal cycle that was mentioned earlier. Thus, these are fluxes that cannot be identified by measuring external fluxes v_1 or v_4.

The way to find non-identifiable reactions is by applying the null space analysis of (**S; C**) and searching for non-zero entries. Below you can find the Python code used to identify these reactions.

Work with the jupyter notebook file

You can work with this example in Python (Jupyter notebook format) on Brightspace (M2_basic_properties).

```python
'''Metabolic network from Syllabus Figure 4-1.
Checking if there are any non-identifyable fluxes '''


lab_flx = [ 'v1', 'v2', 'v3', 'v4' ]  # Labels of fluxes
lab_met = [ 'A', 'B' ]                # Labels of metabolites

S = np.array( [[ 1, -1, -1,  0 ],     # A
               [ 0,  1,  1, -1 ]])    # B

# uptake flux is measured (index 0)
C = np.zeros( (1, S.shape[1]) )
C[0,0] = 1

SC = np.vstack( ( S, C) )
NS = null_space( SC )

# are there fluxes in the nullspace?
for ii in range( NS.shape[0] ):
    # flux in the nullspace?
    if any( abs( NS[ ii, : ] ) > 1e-12 ):
        print( 'flux ', lab_flx[ii], ' cannot be determined' )

```

```
flux  v2  cannot be determined
flux  v3  cannot be determined
```

In practice, the networks to be studied are much larger than the one used in this example. Hence, the method needs to be adjusted to constrain all external fluxes to be 0 (not just v1 as in the example). This can be done easily by identifying metabolites in the **S** matrix that contain either only one producing reaction or only one consuming reaction. This can be done as indicated in the Python code below:

```python
def internal_cycles(S, v_lab):
    '''Function used to identify internal fluxes (non identifiable fluxes) of a network
    The function will identify all exchange fluxes and constrain them to be 0.

    Parameters
    ----------
    S    : Stoichiometric matrix of the metabolic network
    v_lab: Vector containing the flux names.

    Returns:
        Vector with the positions of fluxes that cannot be identified
        '''

    # Check for internal cycles, which cannot be determined from extracellular measurements
    # Approach: Use nullspace analysis to find
    # all solution vectors, add constraints representing the in- and outputs of the network
    print('-'*12,'Internal cycles','-'*12)
    # Constrain extracellular fluxes
    # extracellular fluxes only have (one) input or output, not both at the same time
    n_ext_flx = np.where(np.sum(np.absolute(S), axis=0) < 1.1)
    n_ext_flx = n_ext_flx[0]

    # construct a eye matrix, select the extracellular lines:
    C_tmp = np.eye(S.shape[1])
    C = np.copy(C_tmp[n_ext_flx,:])

    NS = Matrix(np.concatenate((S,C), axis=0)).nullspace()
    NS = np.array(NS).T
    n_undef = []
    # are there fluxes in the nullspace?
    for i in range(NS.shape[0]):
        # flux in the nullspace?
        if any(np.absolute(NS[i,:]) > 0):
            print('flux %s cannot be determined' % (v_lab[i]))
            n_undef = np.append(n_undef, i)

    if len(n_undef) == 0:
        print('There are no internal cycles in this metabolic network')

    return n_undef

n_indef = internal_cycles(S, lab_flx)
```

```
------------ Internal cycles ------------
flux v2 cannot be determined
flux v3 cannot be determined
```

This script has been compiled as a function called **internal_cycles**(S, v_lab); and will be uploaded to the functions file on brightspace.

4.4 Conserved moieties

A moiety is defined as each of two parts into which something is composed of. In our case, a conserved moiety is easily observed when dealing with 'cofactor' pairs, such as NADH and NAD^+, NADPH and $NADP^+$, ATP and ADP, etc. These cofactors are not always conserved moieties and identifying when such a case occurs is important.

The presence of conserved moieties in an **S** matrix does not lead to problems of non-solvability, blocked reactions or internal cycles. It is only redundant information that is not useful for finding a solution and can be removed. Using the example from Figure 4-3, but with only NADH (not NADPH) to solve the blocked reactions problem, we obtain the following network:

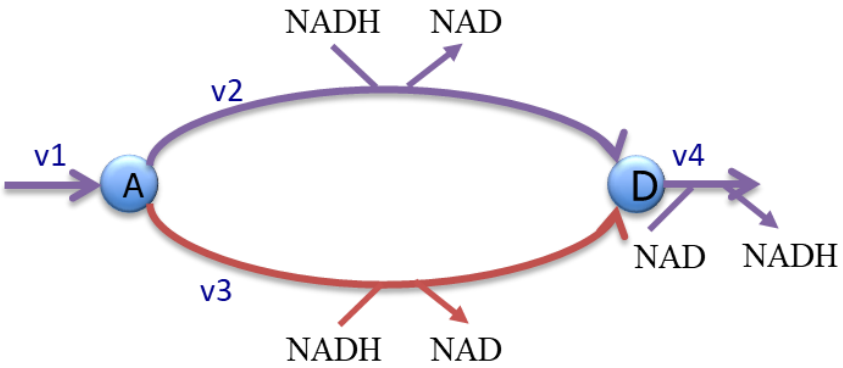


Figure 4-4: Metabolic network including cofactor use of NADH.

From observing Figure 4-4, you can directly identify the relationship between NADH being consumed and NAD being produced. That is a conserved moiety, because an **S** matrix containing the information of both metabolites would not add any new information. But are there more conserved moieties in this network?

Similarly as what was done to identify blocked reactions, we now need to analyse the null space of the **S** matrix, but the transposed version ($\mathbf{S^T}$). This is done because by analysing the null space, we can identify linear dependencies between columns of $\mathbf{S^T}$ (also known as metabolite balances or rows of **S**). The resulting null space matrix contains the combinations of balances (of metabolites) that are linearly dependent. For the example network:

$$\begin{array}{c} \\ v1 \\ v2 \\ v3 \\ v4 \end{array}\begin{array}{c} \begin{array}{cccc} A & D & NADH & NAD \end{array} \\ \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ 0 & -1 & 1 & -1 \end{pmatrix}}_{S^T} \end{array} \underbrace{\begin{pmatrix} eq_1 \\ eq_2 \\ eq_3 \\ eq_4 \end{pmatrix}}_{eq} = 0$$

$$\ker\,(S^T) = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

1*(D) + 1*(NADH) = 0

-1*(D) + 1*(NAD) = 0

Figure 4-5: (Left) Transposed stoichiometric matrix and balances. (Right) Null space analysis of the transposed S matrix (S^T) gives information on the two linear dependencies observed in the network.

From the analysis in Figure 4-5, we identify two conserved moieties in the system. Surprisingly, the balance for metabolite 'D', forms a conserved moiety with NADH (1*D + 1*NADH = 0) and also with NAD (-1*D + 1*NAD = 0). Based on this information, the system would remain unchanged if we remove the row for metabolite D in the **S** matrix.

In short, the method used to identify conserved moieties is by applying the null space analysis of ($\mathbf{S}^T$) and searching for dependencies. Below you can find the Python code used to identify these moieties.

Work with the jupyter notebook file

You can work with this example in Python (Jupyter notebook format) on Brightspace (M2_basic_properties).

```
'''Metabolic network from Syllabus Figure 4-4.
Checking if there are any conserved moieties'''

lab_flx = [ 'v1', 'v2', 'v3', 'v4' ]
lab_met = [ 'A', 'D', 'NADH', 'NAD' ]

S = np.array( [[ 1, -1, -1, 0  ],    # A
               [ 0,  1,  1, -1 ],    # D
               [ 0,  1,  1, -1 ],    # NADH
               [ 0, -1, -1, 1  ]])   # NAD

SV = null_space( S.T )

# check for balances that are redundant in the nullspace
for ii in range( SV.shape[1] ):
    # find indices of the non-zero entries in ii-th row:
    i_row = np.where( abs( SV[:,ii] ) > 1e-12 )[0]
    if i_row.size > 0:
        print( 'dependent balance:')
        for k in i_row:
            print( '+ {:3.2f} * (d{:s}/dt) '.format( SV[k,ii], lab_met[k] ), end='' )
        print( ' = 0' )

```

```
dependent balance:
+ -0.82 * (dD/dt) + 0.41 * (dNADH/dt) + -0.41 * (dNAD/dt)  = 0
dependent balance:
+ 0.71 * (dNADH/dt) + 0.71 * (dNAD/dt)  = 0
```

At this point, you can remove the row matrix for the metabolite balance that is redundant. The presence of this balance will not interfere with solving the system, but it is redundant information and removing it makes the analysis easier and in some times computationally faster.

We also provide a set of functions in python named **conserved_moieties**(S, m_lab) to check for conserved moieties. You can find this function in the functions file on Brighstpase.

4.5 Elementary Flux Modes Analysis (EFMA)

Elementary Flux Modes Analysis (EFMA) is a promising tool in metabolic flux analysis. It allows to decompose a given metabolic network into minimal *functional units*. In other words, it allows the calculation of all possible *unique* solutions to a network while maintaining steady state assumption (S.v = 0). Each Elementary Flux Mode (EFM) resulting from this analysis is unique in the sense that:

- All the reactions in the solution are essential (*i.e.* cannot be removed without disrupting this solution).

- Each individual EFM **cannot** be written as the superposition of other EFMs.

An example of EFMA can be seen in Figure 4-6.

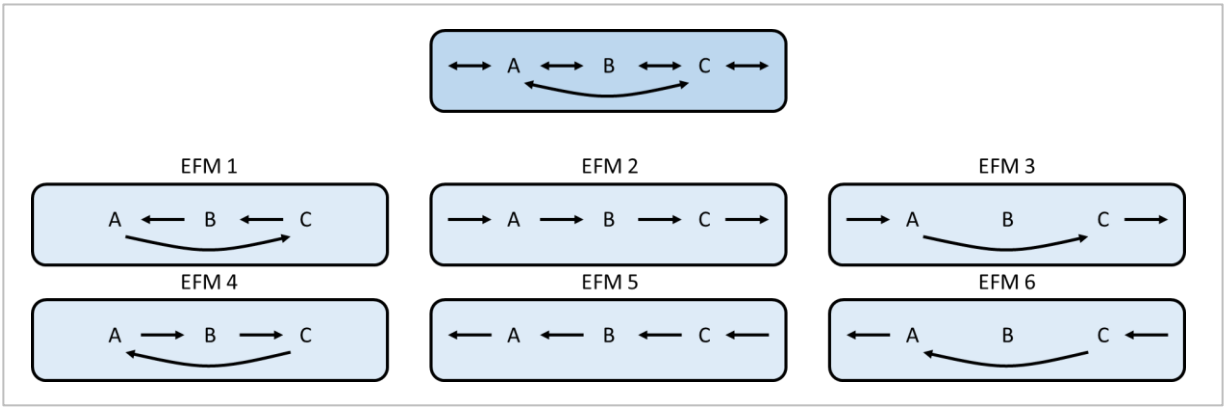


Figure 4-6: EFMA of a 'toy' metabolic model consisting of 3 metabolites interconnected with 5 reversible reactions.

How can we identify all the EFMs of a metabolic model? Well, it turns out that this calculation is computationally demanding, especially considering the combinatorial effect that each new reaction will add to the potential solutions. It can be calculated manually though; by systematically eliminating reactions from the network until the steady state condition breaks, you arrive to one EFM. Repetition of this elimination covering all possible reactions will yield the final result. Of course, this calculation is impossible with large scale models. Just as an example, a medium-scale metabolic model for central carbon metabolism (containing around 100 reactions) contains up to **272 million EFMs**!

For relatively smaller-scale models, EFMA can be performed and could give important information on a biological system. Because any given phenotype (observed experimental rates) will be a consequence of one EFM or the superposition of several EFMs, similarly the metabolic potential of a microorganism can be explored *in silico*, and aid with the identification of metabolic engineering strategies.

Different tools are available to perform EFMA on metabolic models. The most widely used of these algorithms is called "efmtool" developed by Terzer, M., Stelling, J. (2008)[1].

[1] *Large scale computation of elementary flux models with bit pattern trees. Bioinformatics 2008. 24, 2229-2235.*

5 Calculation of the intracellular fluxes from measurements

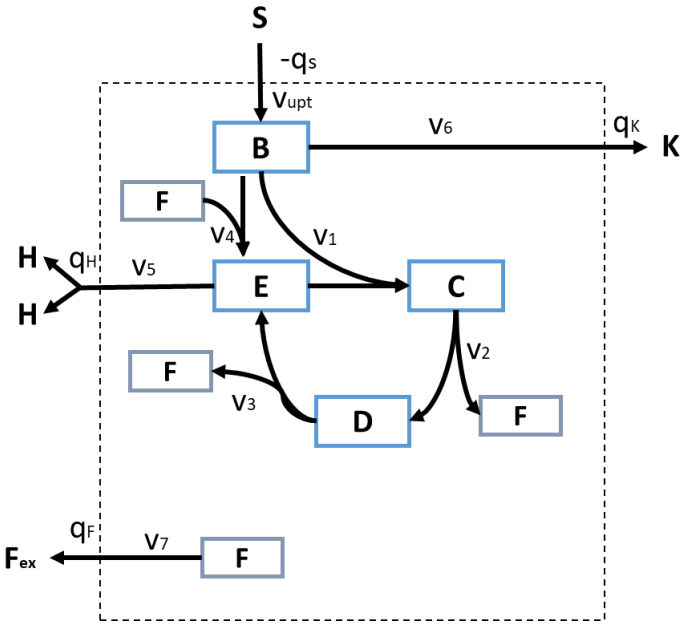


Figure 5-1: Example model used for MFA and FBA calculations. Fluxes v1 and v4 are reversible reactions.

Equation (8) (Sv = 0) represents the stoichiometry of the network and its solution(s) valid flux distribution(s) of the intracellular reaction network considering that all metabolites are in steady state.

*Please note that all examples are now based on the network in **Figure 5-1**.*

To estimate the unknown intracellular fluxes from available measurement data these have to 'enter' the variables in Eq. (8). Thus there has to be a 'coupling' between the **q**-rates and the intracellular fluxes. Again, first of all, a vector (**q**) containing all measurements is defined. For the example network a set of 3 fluxes was measured.

$$\mathbf{q} = \left(q_S, q_H, q_F\right)^T \tag{12}$$

Now, a coupling is defined by adding the equations needed, which are:

$$\begin{aligned} v_{upt} &= -q_S \\ 2v_5 &= q_H \\ v_7 &= q_F \end{aligned} \tag{13}$$

This information is now added to the equation system (8) by adding the equations. To achieve this a measurement matrix **S** is constructed mapping the intracellular fluxes to the available measurements **q**. For the current example, this gets:

$$\begin{matrix} q_S \\ q_H \\ q_F \\ \mathbf{q} \end{matrix} = \underbrace{\begin{pmatrix} -1 & & \\ & 2 & \\ & & 1 \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} v_{upt} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix}_{\mathbf{v}} \tag{14}$$

The stoichiometric relations (Eq. (8)) and the measurement equations are now combined:

$$\begin{pmatrix} 0 \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} \mathbf{S} \\ \mathbf{M} \end{pmatrix} \mathbf{v} \tag{15}$$

If the measured fluxes were non-redundant now a quadratic matrix with full rank (rank(**S M**)^T=8) should be obtained. If this is not the case, following distinctions can be made:

rank(**S M**)^T = dim(**v**): determined system, everything ok.

rank(**S M**)^T < dim(**v**): under-determined system,

No calculations possible – there are redundancies in the measurements

rank(**S M**)^T > dim(**v**): over-determined system

You have more measurements than needed

Now, taking into consideration the metabolic network from Figure 5-1, we will introduce examples when available measured rates result in a determined, over-determined and under-determined systems. Before we deep into each example, we need to define the model (build S matrix, define fluxes and metabolites).

Work with the jupyter notebook file

You can work with this example in Python (Jupyter notebook format) on Brightspace (M2_Flux_Analysis).

```python
# Define the stoichiometric matrix

# fluxes    u  1  2  3  4  5  6  7
S = np.array(
        [ [ 1,-1, 0, 0,-1, 0,-1, 0 ],    # B
          [ 0, 1,-1, 0, 0, 0, 0, 0 ],    # C
          [ 0, 0, 1,-1, 0, 0, 0, 0 ],    # D
          [ 0,-1, 0, 1, 1,-1, 0, 0 ],    # E
          [ 0, 0, 1, 1,-1, 0, 0,-1 ] ] ) # F

# Define vectors containing the labels for metabolites and fluxes
v_lab = ['upt', 'v1', 'v2', 'v3', 'v4', 'v5', 'v6', 'v7']
m_lab = [ 'B', 'C', 'D', 'E', 'F' ]

# Use the AMN_functions:
print_reactions(S, v_lab, m_lab)
```

```
Reactions of the network:
1 upt:    --->  1 B
2 v1:    1 B + 1 E  --->  1 C
3 v2:    1 C  --->  1 D + 1 F
4 v3:    1 D  --->  1 E + 1 F
5 v4:    1 B + 1 F  --->  1 E
6 v5:    1 E  --->
7 v6:    1 B  --->
8 v7:    1 F  --->

------------ Degrees of freedom? ------------
Number of fluxes: 8
Number of balances: 5
Number of independent balances: 5
This metabolic network has 3 degrees of freedom
```

5.1 Determined case

Because a system of full rank is available, the inverse calculation of Eq. (14) simply gets:

$$\mathbf{v} = \begin{pmatrix} \mathbf{S} \\ \mathbf{M} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbf{q} \end{pmatrix} \tag{16}$$

From this equation the intracellular fluxes **v** are obtained from the uptake and production rates (vector **q**).

```
# measurements available
qS = 1
qH = 0.3
qF = 0.2

q = np.array( [-qS, qH, qF ] )

# measurement matrix

# fluxes:    u  1  2  3  4  5  6  7
M = np.array(
        [ [ -1, 0, 0, 0, 0, 0, 0, 0 ],
          [  0, 0, 0, 0, 0, 2, 0, 0 ],
          [  0, 0, 0, 0, 0, 0, 0, 1 ] ] )

# calculate the soultion:

# Stack S and M into one matrix (SM)
SM = np.vstack( (S,M) )
# Stack vector (0; q)
yq = np.concatenate( (np.zeros(S.shape[0]), q ) )
# Perform the calculation (S; M)^-1 * (0; q)
v = linalg.inv(SM) @ yq

# Print the resulting fluces
for i in range( v.size ):
    print( '{:s} = {:g} '.format( v_lab[i], v[i] ) )
```

```
upt = 1
v1 = 0.175
v2 = 0.175
v3 = 0.175
v4 = 0.15
v5 = 0.15
v6 = 0.675
v7 = 0.2
```

5.2 Error propagation

A closer look at (16) tells much more. Besides the value also their sensitivity – that is the influence of experimental errors in **q** on the estimation of **v**. Deriving a linear equation with respect to the variable (here **q**) gives its sensitivity. For a linear system like here the sensitivity is just $\begin{pmatrix}\mathbf{S}\\ \mathbf{M}\end{pmatrix}^{-1}$. This is very important information as you can read which measurements are sensitive – thus which measurements contain information on fluxes and which measurements do not.

Biological measurements usually contain errors – the **q** rates are determined with a certain accuracy which is mostly quantified using standard deviations. But, how do these errors influence the accuracy of the estimated fluxes? What is the standard deviation of a calculated intracellular flux?

With the sensitivity matrix in hand this is a straightforward step using **error propagation**. For this, the measurement errors are described in a covariance matrix. This matrix contains on the diagonal the variance (square of the std. dev.) and covariances on the off-diagonals. If the measurement methods are independent (thus do not influence each other) which is often the case, the covariance matrix simplifies to a diagonal matrix with the variances on the diagonal. For the running example:

$$\mathrm{cov}(\mathbf{q}) = \begin{pmatrix} \sigma_{q_S}^2 & & \\ & \sigma_{q_H}^2 & \\ & & \sigma_{q_F}^2 \end{pmatrix} \tag{17}$$

The std. dev. can then be calculated using error propagation:

$$\mathrm{cov}(\mathbf{v}) = \begin{pmatrix}\mathbf{S}\\ \mathbf{M}\end{pmatrix}^{-1} \mathrm{cov}\begin{pmatrix}\mathbf{0}\\ \mathbf{q}\end{pmatrix}\left(\begin{pmatrix}\mathbf{S}\\ \mathbf{M}\end{pmatrix}^{-1}\right)^{T} \qquad \boldsymbol{\sigma}_{\mathbf{v}} = \sqrt{diag\left\{\mathrm{cov}(\mathbf{v})\right\}} \tag{18}$$

Note, that the stoichiometric relations are determined without any error, thus for each balance, a zero on the diagonal of the covariance matrix has to be added before the variance of **q** are placed (cov(0; q)). In the running example, a standard deviation of 0.1 from the measured q rates is assumed:

```python
# Start by defining a matrix of 0s the size of SM
cov_q = np.zeros(SM.shape)
# Assuming a standard deviation of 0.1 for q rates
var = 0.1 ** 2
# Add variance in the lower diagonal (corresponding to the M matrix)
cov_q[5,5] = var
cov_q[6,6] = var
cov_q[7,7] = var

# Calculate the inverse of SM and perform error propagation
J = linalg.inv( SM )
cov_v = J @ cov_q @ J.T
sigma_v = np.diag( cov_v ) ** 0.5

for i in range( v.size ):
    print( '{:s} = {:g} +/- {:g}'.format( v_lab[i], v[i], sigma_v[i] ) )
```

```
upt = 1 +/- 0.1
v1 = 0.175 +/- 0.0559017
v2 = 0.175 +/- 0.0559017
v3 = 0.175 +/- 0.0559017
v4 = 0.15 +/- 0.05
v5 = 0.15 +/- 0.05
v6 = 0.675 +/- 0.134629
v7 = 0.2 +/- 0.1
```

With the information obtained, we can make the following observations:

1. All given rates are matched (v_upt=-qS=-1, v5=0.5 qH=0.15, v7 = qF = 0.2)
2. There is production of K (v6 = 0.675)
3. The standard deviation of v5 is half the standard deviation of qH (stoichiometry 2 v5 == qH)
4. The off-diagonal elements of cov_v are not 0 – because of the network connections, correlation is observed – the intracellular fluxes depend on all measurements thus there is cross-influence.
5. v6 is most inaccurate estimated, looking at the absolute std. deviation (σ=0.135).
6. v4 and v5 most accurately (direct coupling v4=v5)
7. v1, v2, v3 a bit less as their value also depends on the other measurements

With v6 being badly determined it might be better to measure this one. Maybe qS, qH, qF was not the best choice of measurements.

In the next calculation the same flux distribution is present but instead of qH, qK is measured:

```
# measurements
qS = 1;
qH = 0.3;
qK = 0.675;

q = np.array( [-qS, qH, qK ] )

# measurement matrix
#        u  1  2  3  4  5  6  7

M = np.array(
    [ [ -1, 0, 0, 0, 0, 0,-0, 0 ],
      [  0, 0, 0, 0, 0, 2, 0, 0 ],
      [  0, 0, 0, 0, 0, 0, 1, 0 ] ] )

# calculate the soultion:

# Stack S and M into one matrix (SM)
SM = np.vstack( (S,M) )
# Stack vector (0; q)
yq = np.concatenate( (np.zeros(S.shape[0]), q ) )
# Perform the calculation (S; M)^-1 * (0; q)
v = linalg.inv(SM) @ yq

# error propagation:
cov_q = np.zeros( (SM.shape[0], SM.shape[0]) )
cov_q[5,5] = 0.1 ** 2
cov_q[6,6] = 0.1 ** 2
cov_q[7,7] = 0.1 ** 2

J = linalg.inv( SM )
cov_v = J @ cov_q @ J.T
sigma_v = np.diag( cov_v ) ** 0.5

for i in range( v.size ):
    print( '{:s} = {:g} +/- {:g}'.format( v_lab[i], v[i], sigma_v[i] ) )


```

```
upt = 1 +/- 0.1
v1 = 0.175 +/- 0.15
v2 = 0.175 +/- 0.15
v3 = 0.175 +/- 0.15
v4 = 0.15 +/- 0.05
v5 = 0.15 +/- 0.05
v6 = 0.675 +/- 0.1
v7 = 0.2 +/- 0.320156
```

Now v5 is better determined (σ=0.1), but v7 got much worse (σ=0.32). Thus each set of measurements seems to have its weakness. Therefore it would be of advantage if all available measurements, instead of a subset of only 3 (independent) ones would be used.

Adding a fourth measurement leads to a redundant equation system (9 equations, but only 8 variables).

5.3 Redundant information

Solving a system with more measurements is different. Because the measurements are error-prone it is not possible to exactly reproduce the measurements. A compromise has to be found on which measurements to trust more and find a flux distribution that is as close as possible to ALL measurements. The criteria 'as close as possible' is usually expressed using the sum of squares – this is the sum of the squared difference between model calculation (v) and the measurements (w). Because measurements accuracy can vary strongly for different metabolites (e.g. glucose can be measured very precisely while the CO_2 content in the off gas is more error prone) it is necessary to 'weight' the difference with the accuracy. A deviation from a precise measurement (σ = 0.1) should have a bigger value in the sum of squares than the same deviation from an imprecise (σ=1) measurement. This is achieved by weighting with respect to the standard deviation:

$$SSQ = \chi^2 = \sum_{i=i}^{m} \left(\frac{\mathbf{q}_i - \mathbf{q}_{m,i}}{\sigma_i} \right)^2 \tag{19}$$

With the covariance matrix (Eq. (17)), Eq. (19) can be rewritten as:

$$\chi^2 = \left(\mathbf{q} - \mathbf{q}\right)^T \mathrm{cov}^{-1}(\mathbf{q}) \left(\mathbf{q} - \mathbf{q}\right) \tag{20}$$

Now, a good estimation of fluxes has to be found. The closest solution is given by reaching the minimum of χ^2. The fluxes fulfilling these criteria are the best estimations $\hat{\mathbf{v}}$. At the same time, the stoichiometry of the network has to be fulfilled. Thus the optimization problem is (minimize χ^2 by modifying the fluxes v and fulfilling the stoichiometry **S v**=**0**):

$$\hat{\mathbf{v}} = \arg\min_{v}(\chi^2), \quad \text{subject to:} \mathbf{S}\,\mathbf{v} = \mathbf{0} \tag{21}$$

This is similar to the solution of equations from module 1. One set has to have an exact solution, the other one the least-square solution. We can apply the Lagrange approach:

$$\begin{matrix} \mathbf{S}\,\mathbf{v} = \mathbf{0} \\ \mathbf{M}\,\mathbf{v} = \mathbf{q} \pm \sigma_{\mathbf{q}} \end{matrix} \xrightarrow{\textit{Lagrange multipliers}} \begin{pmatrix} \mathbf{M}^T\mathbf{q}_m \\ \mathbf{0} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{M}^T\mathbf{M} & \mathbf{E}^T \\ \mathbf{E} & \mathbf{0} \end{pmatrix}}_{=\mathbf{L}} \begin{pmatrix} \mathbf{q} \\ \lambda \end{pmatrix} \tag{22}$$

With this reduction, the least-squares solution is obtained by:

$$\begin{pmatrix} \mathbf{v} \\ \lambda \end{pmatrix} = \mathbf{L}^{-1} \begin{pmatrix} \mathbf{M}^T \\ \mathbf{0} \end{pmatrix} \tag{23}$$

With now the complete set of measurements taken into account:

```
# measurements available
qS = 1;
qH = 0.3;
qF = 0.2;
qK = 0.675;
q = np.array( [-qS, qH, qK, qF ] )

# measurement matrix
# fluxes:    u  1  2  3  4  5  6  7
M = np.array(
        [ [ -1, 0, 0, 0, 0, 0, 0, 0 ],     # qS
          [  0, 0, 0, 0, 0, 2, 0, 0 ],     # qH
          [  0, 0, 0, 0, 0, 0, 1, 0 ],     # qK
          [  0, 0, 0, 0, 0, 0, 0, 1 ] ] )  # qF

# calculate fluxes & standard deviation (using AMN function lagrange_solve)
v = lagrange_solve( S, M, q )

for i in range( v.size ):
    print( '{:s} = {:g}'.format( v_lab[i], v[i] ) )

```

```
upt = 1
v1 = 0.175
v2 = 0.175
v3 = 0.175
v4 = 0.15
v5 = 0.15
v6 = 0.675
v7 = 0.2
```

Including also the error propagation:

```
# include measurement accuracy with stdev = 0.1
std_q = np.array( [0.1, 0.1, 0.1, 0.1] )

(v, std_v) = lagrange_solve_w( S, M, q, std_q )

for i in range( v.size ):
    print( '{:s} = {:g} +/- {:g}'.format( v_lab[i], v[i], std_v[i] ) )

```

```
upt = 1 +/- 0.0802773
v1 = 0.175 +/- 0.0494413
v2 = 0.175 +/- 0.0494413
v3 = 0.175 +/- 0.0494413
v4 = 0.15 +/- 0.0447214
v5 = 0.15 +/- 0.0447214
v6 = 0.675 +/- 0.0802773
v7 = 0.2 +/- 0.0954521
```

The use of all available measurements now leads to an estimation with smaller standard deviations than the actual measurements. That increase in accuracy is reached from the redundancy.

5.4 Underdetermined case: q-rates give insufficient information

In case that less rates than necessary for the degrees of freedom are measured or the system contains intracellular cycles that cannot be observed from the extracellular measurements, additional constraints have to be generated. This can be done by i) adding more measurement information by ^{13}C tracer experiments. ii) Assume certain reactions to be inactive, iii) impose an optimization criteria.

Approach i) is most complex and will not be treated here, ii) is trivial (reducing the stoichiometric matrix columns by eliminating fluxes) leaving us with iii) optimization.

The basic idea is to fill missing constraints by defining a goal of metabolism and finding the optimum flux distribution to fulfil this goal (within the given information).

Different criteria can be defined (see Table 3-1). Commonly it is assumed that the organism tries to grow as efficient as possible – thus, the biomass yield is optimal.

Mathematically, optimization is solved by applying linear programming. For this calculation, it is important to define equality constraints (for example S.v = 0, M.v = q) and **inequality constraints**. This term is new, and in metabolic flux analysis it is mainly used to define reactions that only run in one direction (e.g. irreversible reactions). For example, models typically include a maintenance reaction consuming ATP. If this reaction runs in the opposite direction, it would generate free ATP out of nothing (against the second law of thermodynamics), hence this flux must always be > 0. We can define these reactions by using a vector indicating irreversible reactions (v_{irr}). The optimization can therefore be formulated as:

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}} \left(\mathbf{c}^T \mathbf{v} \right) \quad \text{subject to} \begin{cases} \mathbf{S}\,\mathbf{v} = \mathbf{0} \\ \mathbf{M}\,\mathbf{v} = \mathrm{q} \\ \mathbf{v}_{\mathbf{irr}} \geq \mathbf{0} \end{cases} \tag{24}$$

The vector multiplication $\mathbf{c}^T\,\mathbf{v}$ is called target function (e.g. optimal flux of producing a metabolite of interest as a linear combination of **v**). Assuming that our metabolic model contains a reaction to produce biomass, when optimizing for μ, the 'c' vector will contain one entry '1' at the position of μ in the flux vector similar to what was done with the matrix M.

Because the uptake is usually known, when we optimize for specific fluxes in a metabolic model (say ethanol (q_p) or biomass production (μ)), we are indirectly optimizing for the yield (Y_{ps} and Y_{xs} respectively) and not for the rates.

This approach solves the problem of non-identifiably using an elegant mathematical trick. Nevertheless, care has to be taken with these results. The criteria sound logical, but it is not known whether the analysed organism under the analysed condition really follows this criteria. Certain fluxes might be unrealistically high. You could perform further experimentation (13C label experiments) to prove your obtained solutions. Another approach (we will learn this in Module 3) is to apply thermodynamic analyses to the fluxes under specific conditions.

Table 3-1: Commonly used objective (target) functions for setting optimization criteria when not enough constraints are available.

Table III Objective functions implemented in constraint-based FBA

Objective function[a]	Mathematical definition	Explanation	Rationale
Max biomass[b]	$\max \frac{v_{biomass}}{v_{glucose}}$	Maximization of biomass yield	Evolution drives selection for maximal biomass yield ($Y_{X/S}$)
Max ATP	$\max \frac{v_{ATP}}{v_{glucose}}$	Maximization of ATP yield	Evolution drives maximal energetic efficiency ($Y_{ATP/S}$)
$\text{Min}\sum v^{2c}$	$\min \sum_{i=1}^{n} v_i^2$	Minimization of the overall intracellular flux	Postulates maximal enzymatic efficiency for cellular growth (analogous to minimization of the Euclidean norm)
Max ATP per flux unit[c]	$\max \frac{v_{ATP}}{\sum_{i=1}^{n} v_i^2}$	Maximization of ATP yield per flux unit	Cells operate to maximize ATP yield while minimizing enzyme usage
Max biomass per flux unit[c]	$\max \frac{v_{biomass}}{\sum_{i=1}^{n} v_i^2}$	Maximization of biomass yield per flux unit	Cells operate to maximize biomass yield while minimizing enzyme usage
Min glucose	$\min \frac{v_{glucose}}{v_{biomass}}$	Minimization of glucose consumption	Evolution drives selection for most efficient usage of substrate
Min reaction steps[c]	$\min \sum_{i=1}^{n} y_i^2, y_i \in \{0, 1\}$	Minimization of reaction steps	Cells minimizes number of reaction steps to produce biomass
Max ATP per reaction step[c]	$\min \frac{v_{ATP}}{\sum_{i=1}^{n} y_i^2}, y_i \in \{0, 1\}$	Maximization of ATP yield per reaction step	Cells operate to maximize ATP yield per reaction step
Min redox potential[d,e]	$\min \frac{\sum_{n} v_{NADH}}{v_{glucose}}$	Minimization of redox potential[f]	Cells decrease number of oxidizing reactions thus conserving their energy or using their energy in the most efficient way possible
Min ATP production[d,e]	$\min \frac{\sum_{n} v_{ATP}}{v_{glucose}}$	Minimization of ATP producing fluxes[g]	Cells grow while using the minimal amount of energy, thus conserving energy
Max ATP production[d,e]	$\max \frac{\sum_{n} v_{ATP}}{v_{glucose}}$	Maximization of ATP producing fluxes[h]	Cells produce as much ATP as possible

6 Flux Balance Analysis (FBA) - theoretical yields

Linear programming is most often applied to calculate theoretical yields based on a given network. Therefore, the production flux is optimized:

$$\mathbf{v}^{\text{opt}} = \arg\max_{\mathbf{v}} \left(\mathbf{c}^T \mathbf{v} \right) \quad \text{subject to} \begin{cases} \mathbf{S}\,\mathbf{v} = \mathbf{0} \\ \mathbf{M}\,\mathbf{v} = \mathbf{w} \\ \mathbf{v}_{\text{irr}} \geq \mathbf{0} \end{cases} \tag{25}$$

The term **M v** = **w** reduces to one line constraining the uptake flux (usually normalized, v_{upt} = 1). Linear programming is part of the scipy.optimize library– looking at the output of help linprog:

scipy.optimize.linprog

scipy.optimize.linprog(*c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=None, method='interior-point', callback=None, options=None, x0=None*) [source]

Linear programming: minimize a linear objective function subject to linear equality and inequality constraints.

Linear programming solves problems of the following form:

$$\begin{aligned} \min_x \ & c^T x \\ \text{such that } & A_{ub} x \leq b_{ub}, \\ & A_{eq} x = b_{eq}, \\ & l \leq x \leq u, \end{aligned}$$

where x is a vector of decision variables; c, b_{ub}, b_{eq}, l, and u are vectors; and A_{ub} and A_{eq} are matrices.

Informally, that's:

minimize:

```
c @ x
```

such that:

```
A_ub @ x <= b_ub
A_eq @ x == b_eq
lb <= x <= ub
```

From the documentation of linprog we see that all constraints – equality as well as inequality – can be specified. Nevertheless, instead of maximization linprog minimizes. Thus, the maximization has to be reformulated as minimization problem while keeping the system linear. This is achieved by multiplying the objective flux with -1.

As an example, let's maximize the flux to H (v5) in the example network:

Work with the jupyter notebook file

You can work with this example in Python (Jupyter notebook format) on Brightspace (M2_FBA).

```
'''Linear programming for optimization using linprog'''

# Define the M and qmatrix
qS = 1
q = np.array( [qS] )
M = np.zeros( (1, 8) )
M[0,0] = 1;

# Stack S and M, and 0s and qs
A_eq = np.vstack( (S, M ) )
b_eq = np.concatenate( (np.zeros(S.shape[0]), q ) )


# Irriversible: Linprog takes the arguments A_ub (matrix of positions) and b_ub (upper bounds of fluxes)
# Define the irreversibility matrix (in this case, fluxes v6 v7)
v_irr = np.array([ 6, 7 ])
# A_ub is the matrix of positions for irreversible rcts
A_ub = np.zeros( (np.size(v_irr), 8) )
for i, irr in enumerate(v_irr):
    A_ub[i, irr] = -1
# b_ub includes values that irriversible fluxes cannot excede (note A_ub[pos] = -1)
b_ub = np.zeros( (np.size(v_irr), 1) )


# Define a 'c' vector with target of optimization
c = np.zeros( 8 )
c[5] = -1      # Optimize for flux 'v5'    (min(-c.v))



# Perform optimization with linprog
# bounds = (-1000, 1000): Typically all fluxes are added an upper and lower limit of 1000.
opt_res = linprog( c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds = (-1000, 1000) )

# opt.res contains all relevant information as a structure
v_opt = opt_res.x

for i in range( v_opt.size ):
    print( '{:s} = {:g} '.format( v_lab[i], v_opt[i] ) )

```

```
upt = 1
v1 = 0.333333
v2 = 0.333333
v3 = 0.333333
v4 = 0.666666
v5 = 0.666666
v6 = 2.25057e-08
v7 = 6.56831e-09
```

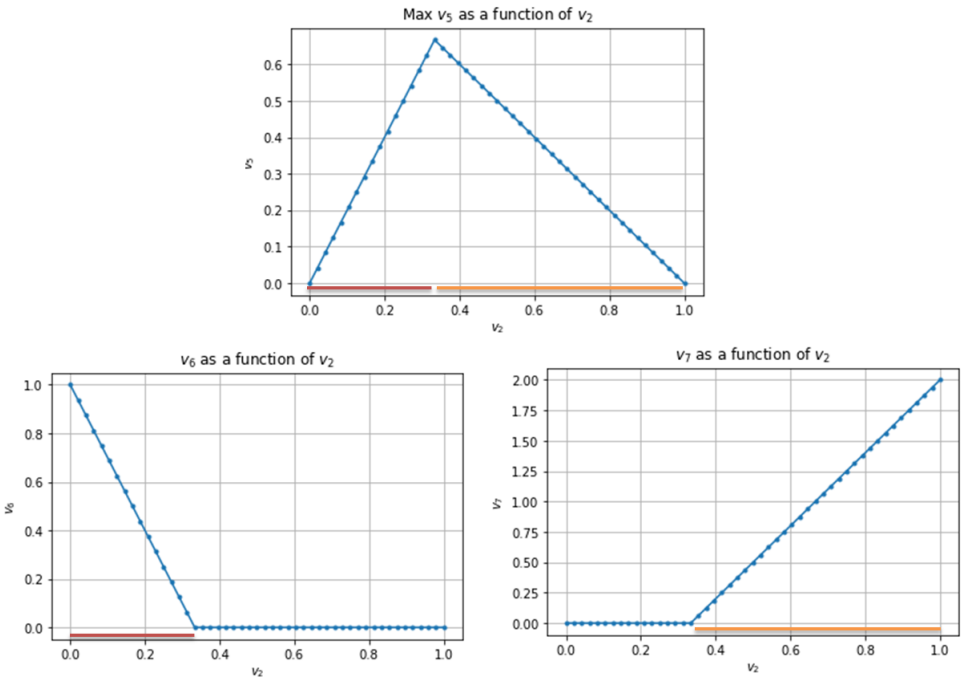
It is seen, that a maximal yield of 0.666 can be achieved. No by-products (K, F) are formed at the optimal flux distribution.

6.1 Including additional constraints

Frequently, to study the impact of certain reactions in the network, variation studies are performed – most of the times called "parameter sweeps", i.e. analysing the influence of one parameter on the outcome of the calculation (here optimization).

Let's assume, we want to analyse if v2 of the current network is a putative bottleneck. We vary this flux between v2 = 0 .. 1 and observe v5 as a function of this flux.

```
''' Defining vuptake and loop v2 for optimization of v5'''

# Define the M and q matrix
qS = 1
M = np.zeros( (2, 8) )
M[0,0] = 1;
M[1,2] = 1;

# Stack S and M
A_eq = np.vstack( (S, M ) )

# Define a range for v2
v2 = np.linspace( 0, 1, 49 ) # generation 50 values between 0 and 1

v_all = np.zeros( (v2.size, S.shape[1]) )   # Storage matrix

for i, qv2 in enumerate(v2):
    q = np.array( [qS, qv2] )    # Changing value for v2
    b_eq = np.concatenate( (np.zeros(S.shape[0]), q ) )   # b_eq stacking 0s and qs
    opt_res = linprog( c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds = (-1000, 1000) )
    v_all[i,:] = opt_res.x

```



Flux v5 is clearly dependent on the intracellular flux v2. The optimum of v5 = 0.66 can only be obtained when the intracellular flux v2 is 0.33.

In the range from v2 = 0 to 0.33, the slope is 2, which is easily explained by the requirement of v4 for metabolite F. F is produced in the cycle v2,v3 generating 2 F, clearly observed with flux v6 as a function of v2.

When going beyond 0.33, the bottleneck is the split between v1 and v4 – the precursor B can be used for making F (via v1, v2, v3), or H via v4 (which needs F). One B less, makes one H less, explaining the slope of -1. Surplus F is then excreted via v7.

7 Literature

van der Heijden, R.; Heijnen, J.; Hellinga, C.; Romein, B. & Luyben, K. Linear constraint relations in biochemical reaction systems: I. Classification of the calculability and the balanceability of conversion rates, *Biotechnology and Bioengineering,* **1994**, *43*, 3-10

Klamt, S.; Schuster, S. & Gilles, E. Calculability analysis in underdetermined metabolic networks illustrated by a model of the central metabolism in purple nonsulfur bacteria, *Biotechnology and Bioengineering,* **2002**, *77*, 734-751

Palsson, B. Ø. (**2006**) Systems Biology: PROPERTIES OF RECONSTRUCTED NETWORKS, University of California, San Diego

Wahl, S. A.; Takors, R. & Wiechert, W. Interpretation of metabolic flux maps by limitation potentials and constrained limitation sensitivities. *Biotechnol Bioeng,* **2006**, *94*, 263-272

Stephanopoulos, G.; Aristidou, A. A.; Nielsen, J. H. & Nielsen, J. *(ed.)* Metabolic engineering: principles and methodologies, *Academic Press,* **1998**