Flux balancing

Under **steady-state conditions**, the mass balance constraints in a metabolic network can be represented mathematically by the matrix equation:

$$\mathbf{S} \cdot v = 0$$

where

- the matrix **S** is the **stoichiometric matrix** and
- the vector *v* represents all **fluxes** in the metabolic network, including the internal fluxes and transport fluxes.

12.5 Flux Balance Analysis (FBA)

Since the number of metabolites is generally smaller than the number of reactions ($m < n$) the flux-balance equation is typically **underdetermined**.

-> There are generally multiple feasible
flux distributions that satisfy the mass balance constraints.

$$\mathbf{S} \cdot \mathbf{v} = \mathbf{0}$$

The set of solutions is confined to the **nullspace** of matrix **S**.

Null space: space of feasible solutions

Consider

$$\begin{pmatrix} 0 & 2 & 1 \\ 3 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

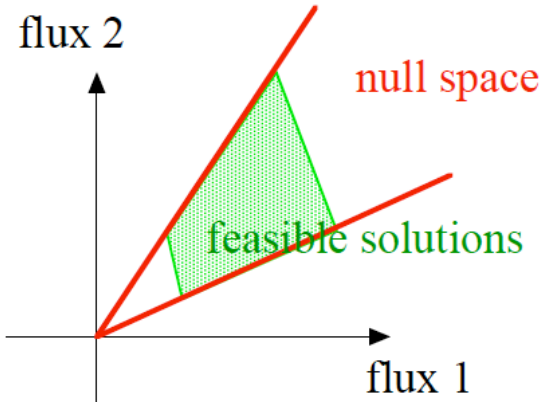
Corresponds to

$$\begin{aligned} 2x_2 + x_3 &= 0 \\ 3x_1 - x_2 + x_3 &= 0 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} 2x_2 &= -x_3 \\ 2x_1 &= -x_3 \end{aligned}$$

=> only one free parameter: x_3

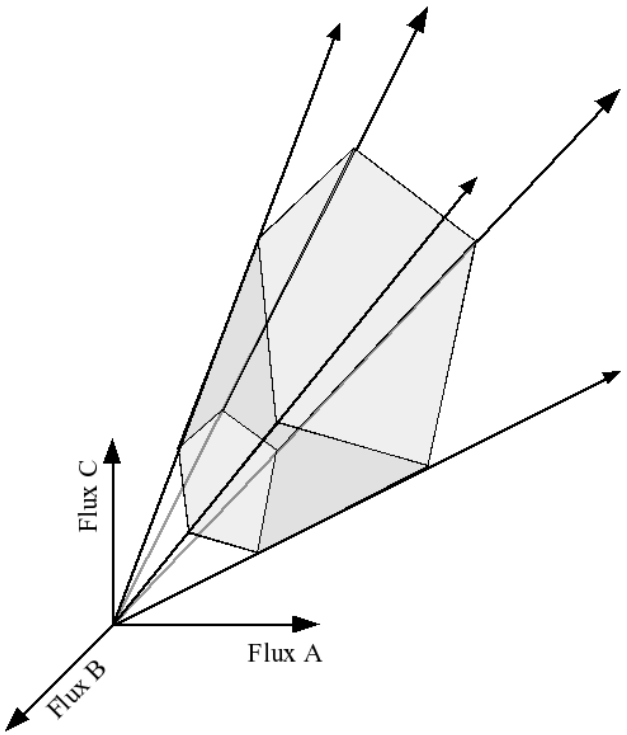
null space: $\vec{x} = \begin{pmatrix} -a \\ -a \\ 2a \end{pmatrix}$

Add inequalities for external fluxes
(here, e.g.: $x_3 \geq 0$)
=> **feasible** solutions for $a \geq 0$



Generally: null space is a cone,
constraints select part of it

Feasible solution set for a metabolic reaction network



The steady-state operation of the metabolic network is restricted to the region within a **pointed cone**, defined as the feasible set.

The feasible set contains all flux vectors that satisfy the physicochemical constraints.

Thus, the feasible set defines the capabilities of the metabolic network.
All feasible metabolic flux distributions lie within the feasible set.

The **extreme pathways** (see V19) are the corner rays of this cone.

Edwards & Palsson PNAS 97, 5528 (2000)

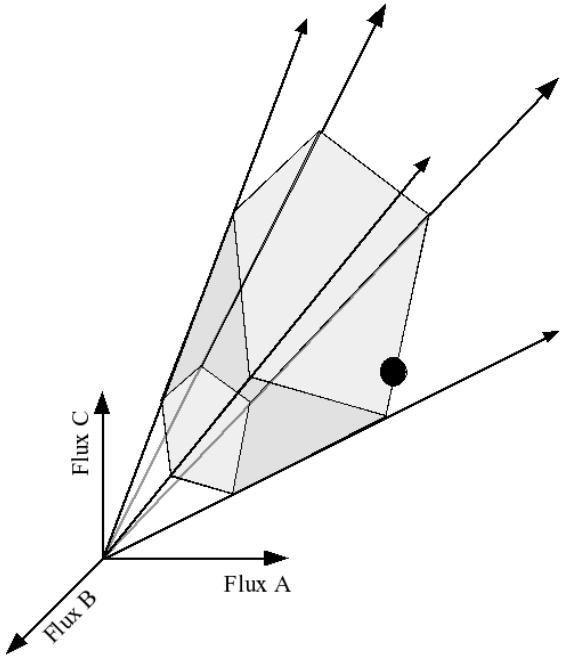
True biological flux

To find the „true“ biological flux in cells (→ e.g. Wittmann / UdS)
one needs additional (experimental) information,
or one may impose constraints

$$\alpha_i \leq v_i \leq \beta_i$$

on the magnitude of each individual metabolic flux.

The intersection of the nullspace and the region defined by those linear inequalities defines a region in flux space = the **feasible set of fluxes**.



In the limiting case, where all constraints on the metabolic network are known, such as the enzyme kinetics and gene regulation, the feasible set may be reduced to a single point. This single point must lie within the feasible set.

E.coli in silico – Flux balance analysis

Define $\alpha_i = 0$ for **irreversible** internal fluxes,

$\alpha_i = -\infty$ for **reversible** internal fluxes (use biochemical literature)

Transport fluxes for PO_4^{2-}, NH_3, CO_2, SO_4^{2-}, K^+, Na^+ were unrestrained.

For other metabolites $0 < v_i < v_i^{max}$ except for those that are able to leave the metabolic network (i.e. acetate, ethanol, lactate, succinate, formate, pyruvate etc.)

Find particular metabolic flux distribution in feasible set by **linear programming**.

LP finds a solution that **minimizes** a particular metabolic **objective** –Z (subject to the imposed constraints) where e.g.

$$Z = \sum c_i \cdot v_i = \langle \mathbf{c} \cdot \mathbf{v} \rangle$$

In FBA, c_i are the (known) coefficients of the optimization goal.

E.coli in silico – Flux balance analysis

In the case of biomass maximization, vector ***c*** is an all-zero vector except for a one (1.0) in the position corresponding to the biomass reaction:

$$Z = \sum c_i \cdot v_i = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} v_{bio} \\ v_1 \\ v_2 \\ \dots \\ v_N \end{pmatrix}$$

What is the **biomass reaction**?

(Montezano et al.) used the mixture on the right that reflects the actual composition of cells of *Mycobacterium tuberculosis*.

$$\begin{aligned} & 0.214\ \text{PROTEIN} \\ + \quad & 0.036\ \text{RNA} \\ + \quad & 0.022\ \text{DNA} \\ + \quad & 0.050\ \text{SMALLMOLECULES} \\ + \quad & 0.006\ \text{PE} \\ + \quad & 0.016\ \text{TAGbio} \\ + \quad & 0.040\ \text{PIMs} \\ + \quad & 0.186\ \text{LAM} \\ + \quad & 0.208\ \text{MAPC} \\ + \quad & 0.035\ \text{P} - \text{L} - \text{GLX} \\ + \quad & 0.007\ \text{CL} \\ + \quad & 0.054\ \text{LM} \\ + \quad & 47\ \text{ATP} \\ = \quad & 1.0\ \text{BIOMASS} + 47\ \text{ADP} + 47\ \text{PI} \end{aligned}$$

Montezano et al (2015) PLoS ONE 10(7): e0134014.

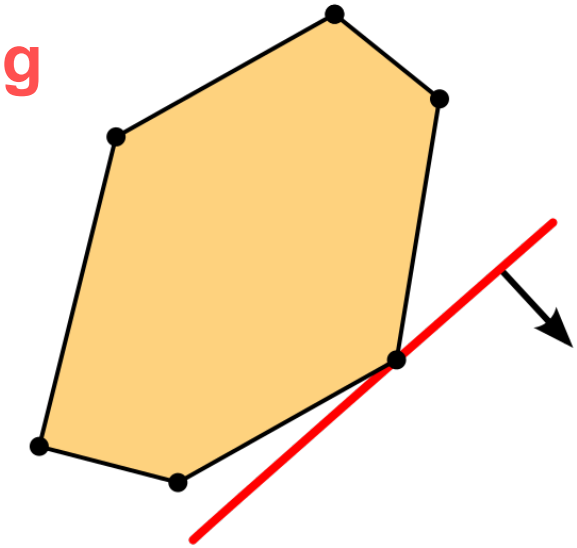
Linear programming

Linear programming is a technique for the **optimization** of a **linear objective function**, subject to linear equality and inequality **constraints**.

Its **feasible region** is a convex polytope, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality.

Its objective function is a real-valued linear function defined on this polyhedron.

A linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.



A pictorial representation of a simple linear program with 2 variables (x and y-axes) and 6 inequalities (borders).
The set of feasible solutions is depicted in yellow and forms a polygon, a 2-dimensional polytope.
The linear **cost function** is represented by the red line and the arrow: The arrow indicates the direction in which we are optimizing.

www.wikipedia.org

Linear programming

Linear programs are problems that can be expressed in canonical form as

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ \text{and} \quad & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where **x** represents the vector of variables (to be determined),
c and **b** are vectors of (known) coefficients,
A is a (known) matrix of coefficients, and $(.)^{\mathrm{T}}$ is the matrix (vector) transpose.

The expression to be maximized or minimized is called the **objective function** ($\mathbf{c}^{\mathrm{T}}\mathbf{x}$ in this case).

The inequalities $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ are the **constraints** which specify a convex polytope over which the objective function is to be optimized.

www.wikipedia.org

Integer linear programming

If all unknown variables are required to be integers, then the problem is called an integer programming (IP) or integer linear programming (ILP) problem.

In contrast to linear programming, which can be solved efficiently,
integer programming problems are in many practical situations NP-hard.

The **branch and bound algorithm** is one type of algorithm to solve ILP problems.

www.wikipedia.org

12.5.1 Gene knock-outs: MOMA algorithm

As just shown, FBA can also predict phenotypes associated with genetic manipulations.

The effects of a gene knockout is realized in FBA calculations by simply setting the entries of the stoichiometric matrix related to the respective protein to zero and then obtaining an optimal flux.

This approach assumes that the mutant bacteria also adopt an optimal metabolic state,

although these artificially generated strains have not been exposed to the typical **evolutionary pressure** that formed the metabolic profile of the wild-type.

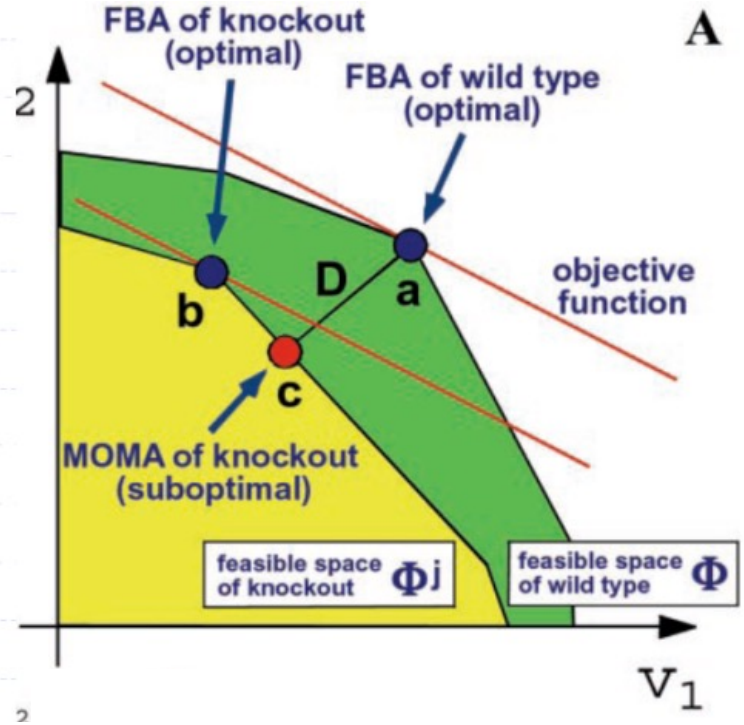
Segre D, Vitkup D, Church GM (2002)
PNAS 99, 15112-15117.

12.5.1 Gene knock-outs: MOMA algorithm

To characterize the flux states of mutants, Church and collagues formulated the method **MOMA** = „**minimization of metabolic adjustment**“.

MOMA applies the same stoichiometric constraints as FBA but does not assume that gene knock-out mutants will show optimal growth flux.

Idea behind MOMA: in the beginning, a mutant will likely possess a suboptimal flux distribution that lies in between the wild-type optimum (**a**) and the mutant optimum (**b**).



MOMA approximates this intermediate suboptimal state by assuming that the flux values in the mutant will initially take on values that match those of the wild-type optimum as closely as possible.

Segre D, Vitkup D, Church GM (2002) *PNAS* 99, 15112-15117.

12.5.1 Gene knock-outs: MOMA algorithm

To predict a metabolic phenotype, MOMA determines a flux vector **v** in the flux space Φ of a mutant with smallest Euclidian distance from a given flux vector **w** for the wild-type organism.

This means that:

$$D(w,x) = \sqrt{\sum_{i=1}^{N}(w_i - v_i)^2} = \sqrt{\sum_{i=1}^{N} w_i^2 - 2w_i v_i + x_i^2}$$

should be minimized. Minimizing *D* is equivalent to minimizing the square of *D.*

Constant terms (the wild-type flux w_i^2) can be left out from the objective function.

Segre D, Vitkup D, Church GM (2002) *PNAS* 99, 15112-15117.

12.5.1 Gene knock-outs: MOMA algorithm

With **Q** as the $n \times n$ unit matrix and **L** set to **–w,** this criterion is equivalent to a quadratic programming problem where the aim is to minimize:

$$f(x) = \mathbf{L} \cdot \mathbf{v} + \frac{1}{2}\mathbf{v}^T \mathbf{Q}\mathbf{v}$$

under a set of linear constraints.

The vector L of length N and the $N \times N$ matrix Q define the linear and quadratic part of the objective function, respectively, and v^T represents the transpose of v.

Thus, the task of minimizing D is reduced to the task of minimizing

$$f(v) = -w \cdot x + \frac{1}{2}x^T x.$$

Flux predictions made by MOMA were

reported to show good correlation to experimental findings.

Segre D, Vitkup D, Church GM (2002)
PNAS 99, 15112-15117.

12.5.1 OptKnock algorithm

In **genetic strain optimization**, the aim of maximizing the yield of a particular chemical compound can also be formulated as a linear programming problem, just like in FBA.

There exist several bi-level strain design approaches that employ **mixed-integer programming** (MIP) to find the mutations required to obtain the **largest synthesis yields of a chemical**.

Such bi-level MIP methods involve an **"outer" problem** and an **"inner" problem**.

In the outer problem, an engineering objective function (selection of **optimal mutant strains**) is optimized.

In the inner problem a cellular objective function is optimized such as **maximizing** the **total flux** via FBA and linear programming.

As one representative of this class of algorithms, we will discuss the **OptKnock** algorithm

Burgard AP, Pharkya P, Maranas CD (2003) *Biotechnology and Bioengineering* 84, 647-57.

12.5.1 OptKnock algorithm

The aim of OptKnock is to over-produce desired chemicals, e.g. in *E. coli*. Given a fixed amount of glucose uptake, the cellular objective can be to **maximize the yield of biomass**.

The effects of gene deletions are modeled by incorporating binary variables y_j that describe whether reaction *j* is active or not into the FBA framework:

$$y_j = \begin{cases} 1 \text{ if reaction flux } v_j \text{ is active} \\ 0 \text{ if reaction flux } v_j \text{ is not active}, \forall j \in M \end{cases}$$

The constraint:

$$v_j^{min} \cdot y_j \leq v_j \leq v_j^{max} \cdot y_j, \forall j \in M$$

guarantees that reaction flux v_j is set to zero only in cases where variable y_j is zero.

When y_j is equal to 1, v_j can adopt values between v_j^{min} and v_j^{max}.

The authors determined v_j^{min} and v_j^{max} by minimizing and subsequently maximizing every reaction flux subject to the constraints from the primal problem.

Burgard AP, Pharkya P, Maranas CD (2003) *Biotechnology and Bioengineering* 84, 647-57.

12.5.1 OptKnock algorithm

The best gene/reaction knockouts are determined by a bilevel optimization.

If biomass formation is the cellular objective, this may be modeled mathematically as the following bilevel mixed-integer optimization task:

$$maximize\ \ v_{chemical} \quad (OptKnock - outer\ problem)$$

$$whereby\ y_j\ is\ subject\ to\ y_j \in \{0,1\}\ \forall j \in M, \sum_{j \in M}(1 - y_j) \leq K\ and$$

$$[maximize\ \ v_{biomass} \quad (Primal - inner\ problem)$$

$$whereby\ v_j\ is\ subject\ to \sum_{j=1}^{M} S_{ij} v_j = 0$$

$$v_{pts} + v_{glk} = v_{glc-uptake}$$

$$v_{atp} \geq v_{atp-main}$$

$$v_{biomass} \geq v_{biomass}^{target}$$

$$v_j^{min} \cdot y_j \leq v_j \leq v_j^{max} \cdot y_j, \forall j \in M]$$

K : maximal number of gene knockouts allowed.

The vector v holds both internal and transport reactions.

v_j : flux of reaction *j* v_{glc_uptake} implements the glucose uptake scenario.

v_{pts} : uptake of glucose through phosphotransferase system , v_{glk} : synthesis of glucose by glucokinase.

v_{atp_main} : lower flux threshold keeping ATP level constant in non-growth-conditions

$v_{biomass}^{target}$: minimum level of biomass production.

12.5.1 OptKnock algorithm

Solving this two-stage optimization problem in a reasonable time can be challenging due to the high dimensionality of the flux space (the system implemented by the authors contained over 700 reactions) and the two nested optimization problems.

To overcome this, the authors turned the linear programming problem into an optimization problem.

Palsson and co-workers applied OptKnock to genome-scale metabolic models of *E. coli* wild-type and mutants followed by adaptive evolution of the engineered strains.

They managed to design bacterial production strains that produced **more lactate** than wild-type *E. coli* (Fong *et al.* 2005).

Burgard AP, Pharkya P, Maranas CD (2003) *Biotechnology and Bioengineering* 84, 647-57.

Compress genome-scale models: Network Reducer

Detailed genome-scale metabolic models contain thousands of metabolites and reactions. Their interpretation and application of the EP method is difficult.

Thus, one wishes to reduce genome scale models to **„core“** models of **lower complexity** but having the **same key elements** and/or key functional features.

One such method is the network reduction algorithm **NetworkReducer**.
It can simplify an input large-scale metabolic network to a smaller subnetwork whereby desired properties of the larger network are kept (Erdrich *et al.* 2015).

As in FBA, one consider vectors **v** of net reaction rates that fulfil $S \cdot v = 0$.

The fluxes **v** satisfying this equation form the null space of S. Its dimensionality may also be termed the number of **degrees of freedom** (dof) and is given by

$$dof = n - rank(S)$$

where *n* is the number of reactions in the system.

Specifications of Network Reducer

(a) PM : set of „**protected metabolites**“ that **must be kept** in the reduced network.

(b) PR : set of „**protected reactions**“ that must be kept in the reduced network.

(c) Protected **functions** (e.g. production of a chemical) and **phenotypes** are characterized by appropriate inequalities.

(d) The reduced network may not have fewer degrees of freedom (dof) than a minimum number: $dof \geq dof_{min}$.

(e) A specified minimal number of reactions must be kept ($n \geq n_{min}$).

A key property of the algorithm is how it treats desired (protected) functions and phenotypes.

Network Reducer

Each protected functionality (there are *s* of them in total) is formulated by a respective set of linear equalities/inequalities,
$D_k v \leq d_k, k = 1 \ldots s$.
The network reduction algorithm first checks the **feasibility** of the protected reactions in the input network.

Then, a loop tries to iteratively discard non-protected reactions unless this violates any of the desired conditions (a) - (e).

To decide on the order of this process, the algorithm computes for each removable (non-protected) reaction *i* the feasible flux ranges.

Let F_i^k denote the **flux range** of reaction *i* under the protected function *k, k* = 1…*s*.

From this, the union F_i of all flux ranges is formed:
$F_i = \bigcup_{k=1}^{s} F_i^k$

Network Reducer

Essential reactions possess an entirely positive or entirely negative flux range F_i^k for any of the desired functionalities *k.*

Such essential reactions are deleted from the list of removable reactions.

From the current set of removable reactions, the next candidate reaction to be discarded is the reaction with overall **smallest flux range** F_i.

It can be safely assumed that a considerable amount of flux variability remains in the network after deleting this reaction.

After discarding a reaction, one needs to test the **feasibility** of the protected functions (condition (c)), protected reactions and of protected metabolites.
If any of these conditions is not fulfilled, then the reaction that was just deleted is reinserted and labeled as non-removable.

Then one continues with the reaction having the second smallest overall range of fluxes F_i.

Network Reducer

After deleting a reaction, the flux ranges are recomputed in the next iteration.

The main loop of network pruning terminates when no additional reaction can be removed without violating any of conditions (a) - (e).

Finally, **unconnected metabolites** in the reduced network that do not participate in any of the remaining reactions are deleted from the network.

In a post-processing step, the network can be (optionally) **compressed** further without loosing degrees of freedom.

For example, reaction sets or enzyme sets belonging to a **linear chain** of reactions can be combined into a single reaction with collapsed stoichiometries.

Compression does not affect protected reactions and metabolites.