1. Constraint-Based Perspective and Tools

In this lecture, we introduce the underlying ideas of the constraints-based approach to the metabolic design problem in which we seek to maximize a metabolic network's performance (in some sense) subject to physical or economic constraints. In this lecture, we'll focus on reaction networks occurring in cells, but later show how we can adapt these tools to consider cell-free systems.

In particular, we'll discuss:

- Intracellular species material balance equations (in the continuum limit)
- Constraint-Based tools such as Flux Balance Analysis (FBA)
- The stoichiometric matrix and convex network decomposition methods

Additional resources (optional):

- Systems Biology: Constraint-based Reconstruction and Analysis, Bernhard Ø. Palsson, Cambridge University Press, 2015
- Lectures from SYSTEMS BIOLOGY: Constraint-based Reconstruction and Analysis (on YouTube by Palsson)

Who is Bernhard Palsson?

- Google Scholar Page for Bernhard Ø. Palsson

2. Dynamic species material balances in the continuum limit

Let's consider the general case in which we have $\mathcal{R}$ chemical reactions and $\mathcal{M}$ metabolites (chemical species) operating in some well-mixed physical (or logical) control volume Ω. In the general continuum well-mixed limit (we ignore random fluctuations, at least for now), the species mole balance around each chemical species (metabolite) i is given by:

$$\frac{d}{dt}(C_i\beta) = \sum_{s=1}^{\mathcal{S}} d_s\dot{n}_s + \left(\sum_{j=1}^{\mathcal{R}}\sigma_{ij}\hat{r}_j\right)\beta \qquad i = 1,2,\dots,\mathcal{M}$$

where C_i denotes the concentration (or number density) of species i calculated with respect to a *system basis* β.

The first set of terms on the right-hand side denotes the rate of physical or *logical* transport (units: mol or number per unit time) of chemical species i into or from control volume Ω, where d_s denotes a direction parameter (in = +1, out = -1) for stream s.

The second set of terms describes the rate of chemical reactions that are occurring in the physical or logical control volume Ω. The $\hat{r}_j$ terms describe the net rate of production (consumption) per unit β of species i by enzyme-catalyzed chemical reaction j, while σ_{ij} denotes the stoichiometric coefficient relating species i with reaction j:

- A stoichiometric coefficient σ_{ij} > 0 implies that metabolite i is **produced** by reaction j
- A stoichiometric coefficient σ_{ij} = 0 implies that metabolite i is **not connected** to reaction j
- A stoichiometric coefficient σ_{ij} < 0 implies that metabolite i is **consumed** by reaction j

2.1 General intracellular species material balances

Let's use the balance equations above, to develop intracellular material balances governing the concentration of metabolite i. In particular, let's expand the accumulation term (chain rule):

$$\frac{d}{dt}(C_i\beta) = \beta\frac{dC_i}{dt} + C_i\frac{d\beta}{dt} \qquad i = 1,\dots,\mathcal{M}$$

which can be substituted into the general balance equation to give:

$$\frac{dC_i}{dt} = \frac{1}{\beta}\left(\sum_{s=1}^{\mathcal{S}} d_s\dot{n}_{is}\right) + \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \frac{C_i}{\beta}\frac{d\beta}{dt} \qquad i = 1,\dots,\mathcal{M}$$

When building models of the *intracellular* state of a cell, there are no physical transport terms (material is not flowing or from the cell). Intracellular balances in industrial metabolic engineering applications for liquid cultures are typically written in *biomass-specific units*, e.g., something per gram dry weight (gDW) of cells in the reactor; concentrations are given in units of $\star$-mol/gDW. In this case, the abstract volume basis β is given by:

$$\beta = XV_R$$

where X denotes the *dry* cell mass per liter in the culture (units: gDW/L) and V_R denotes the liquid volume of the culture. Substituting β into general balance equation gives:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \frac{C_i}{XV_R}\left(X\frac{dV_R}{dt} + V_R\frac{dX}{dt}\right) \qquad i = 1,\dots,\mathcal{N}$$

Expanding the dilution terms gives:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \frac{C_i}{V_R}\frac{dV_R}{dt} - \frac{C_i}{X}\frac{dX}{dt} \qquad i = 1,\dots,\mathcal{M}$$

2.2 Continuous culture

In continuous culture, material stream(s) are introduced in the reactor as some volumetric rate F (units: L/hr) and removed at the same volumetric rate. Hence, there is no overall volume change in the reactor (volume flow rate in = volume flow rate out in). The cell mass balance equation for a continuous culture (assuming a sterile feed stream) is given by:

$$\frac{dX}{dt} = (\mu - D)X$$

where μ denotes the specific growth rate of the cells (units: hr^{-1}), and D denotes the *dilution rate* (units: hr^{-1}) which is defined as $D{\equiv}F/V_R$ (inverse of the residence time in the reactor). Substituting the cell mass time rate of change into the general species material balance gives:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - C_i(\mu - D) \qquad i = 1, \ldots, \mathcal{M}$$

At steady-state, all the time-derivatives vanish, and $\mu{=}D$. Thus the *intracellular metabolic state* of the cell is governed by:

$$\left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) = 0 \qquad i = 1, \ldots, \mathcal{M}$$

In other words, what comes into the cell *must* come out in some form. If we make metabolite *A* (a node in the network), we must consume *A* somehow (chemical reaction, or export *A* out of the cell).

2.3 Batch culture

In a batch culture, there is no liquid volume change (dV_R/dt = 0). However, cell mass and products accumulate, and starting materials (substrates) decrease over time in the culture. Thus, the dilution terms become:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \mu C_i \qquad i = 1, \dots, \mathcal{M}$$

where μ denotes the *specific growth rate* of the cells in the batch culture $\mu = X^{-1}\dot{X}$ (units: hr^{-1}). There is never an *extracellular* steady-state in a batch culture, however, inside the cell in certain time windows (typically when $\mu \simeq \mu_g^{max}$) there is a pseudo-steady-state called *balanced growth* in which all intracellular quantities are time-invariant leading to the condition:

$$\left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \mu C_i = 0 \qquad i = 1, \dots, \mathcal{M}$$

Additional resources:

- Fishov I, Zaritsky A, Grover NB. On microbial states of growth. Mol Microbiol. 1995 Mar;15(5):789-94. doi: 10.1111/j.1365-2958.1995.tb02349.x. PMID: 7596281.

2.4 Fed-batch culture

The volume is not constant in a fed-batch reactor because there is an input stream(s) but no output stream. Thus, the culture volume increases over time. Suppose we have a sterile feed stream (no cells coming into the reactor), and the reactor is fed at the volumetric flow rate F (units: L/hr). Then the cell mass balance is given by:

$$\frac{dX}{dt} = (\mu - D)X$$

which at first glance appears to be the same as the continuous culture case, however unlike a chemostat (continuous culture), the dilution rate D in a fed-batch reactor is a function of time, and no external steady-state is possible. Substituting the cell mass time rate of change into the general species material balance gives:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - DC_i - C_i(\mu - D) \qquad i = 1, \dots, \mathcal{M}$$

which after combining the dilution terms terms becomes:

$$\frac{dC_i}{dt} = \left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \mu C_i \qquad i = 1, \dots, \mathcal{M}$$

While no *extracelluar* steady-state is possible, under the *balanced growth* hypothesis the intracellular state is governed by:

$$\left(\sum_{j=1}^{\mathcal{R}} \sigma_{ij}\hat{r}_j\right) - \mu C_i = 0 \qquad i = 1, \dots, \mathcal{M}$$

which is the same form as the *batch* culture.

Additional resources:

- Fishov I, Zaritsky A, Grover NB. On microbial states of growth. Mol Microbiol. 1995 Mar;15(5):789-94. doi: 10.1111/j.1365-2958.1995.tb02349.x. PMID: 7596281.

3. The constraint based perspective

The constraint-based family of mathematical modeling tools is widely used to understand the biosynthetic capabilities of organisms and to redesign their metabolic networks.

Constraint based review:

- O'Brien EJ, Monk JM, Palsson BO. Using Genome-scale Models to Predict Biological Capabilities. Cell. 2015 May 21;161(5):971-987. doi: 10.1016/j.cell.2015.05.019. PMID: 26000478; PMCID: PMC4451052.

3.1 Logical and Physical Control Volumes

One of the critical ideas of the constraint-based approach is the decomposition of (potentially) dynamic problems into a sequence of steady-state or pseudo-steady-state subproblems, which give snapshots of the flow through a metabolic network (in this world called *flux*).

The mental model that makes this decomposition possible relies on partitioning variables into physical and *logical* control volumes. In the constraint-based world, systems inside a logical control volume are at a steady state. Let's take a look at an example to make this clear.

Consider a batch reactor where we supply cells X, and the starting materials A_{1x} and A_{2x} in the media. In this case, the physical control could be the volume of the culture in the reactor, and the logical control volume could be the intracellular compartment of the cells in the reactor.

Under this model, $A_{\star x}$, P_x and C_x would be *extracellular* variables governed by *dynamic* material balances, while all the intracellular species would be at steady-state. However, this obvious partition illustrates a significant *perceived* shortcoming of the constraint-based approach; constraint-based tools can only be used to model/design systems at steady-state. **However, this is not true**.

Let's consider a different scenario. Suppose the physical control volume is a cell with some volume basis β, while the logical control volume was some local section of the metabolic network in the cell. Under this scenario, $A_{\star x}$, P_x and C_x would be governed by *dynamic* material balances (as before), but $A_\star$, B, C, x and y become *hypothetical local logical variables* that are at steady-state. The *logical* transport terms describe transport into/from the logical control volume. For example, suppose we partition the total amount of $A_{1,T}$ (what we would measure in the cell) into A_{1x} and A_1. Let's write balances around A_{1x} and A_1:

$$\frac{dA_{1x}}{dt} = -\frac{\dot{n}_4}{\beta} - \frac{A_{1x}}{\beta}\frac{d\beta}{dt}$$
$$\frac{dA_1}{dt} = \frac{\dot{n}_4}{\beta} - \hat{r}_1 - \frac{A_1}{\beta}\frac{d\beta}{dt}$$

Adding the A_{1x} and A_1 balances (to compute the time rate of change of the total amount to A_1) gives:

$$\frac{dA_{1x}}{dt} + \frac{dA_1}{dt} = -\hat{r}_1 - (A_{1x} + A_1)\beta^{-1}\dot{\beta}$$

We know that the total amount of $A_{1,T}$ (what we would actually measure in the cell) is the sum of the two types of A_1: $A_{1,T} = A_{1x} + A_1$. Thus, differentiating $A_{1,T}$ with respect to time gives:

$$\frac{dA_{1,T}}{dt} = \frac{dA_{1x}}{dt} + \frac{dA_1}{dt}$$

Putting everything together gives:

$$\frac{dA_{1,T}}{dt} = -\hat{r}_1 - \frac{A_{1,T}}{\beta}\frac{d\beta}{dt}$$

Thus, constraint-based tools can model dynamics; however, they do so in a sequence of discrete pseudo-steady-state snapshots.

3.2 Flux Balance Analysis (FBA)

Flux balance analysis (FBA) is a mathematical modeling and analysis approach that estimates the *intracellular* reaction rate (metabolic flux) of carbon and energy throughout a metabolic network (units: ⋆mol/gDW-time or ⋆mol/L-time for cell-free networks). FBA arguably the most widely used tool in this space. However, there are alternatives to FBA, such as metabolic flux analysis (MFA), but these alternatives vary more in the solution approach than the structure of the estimation problem.

Let's look at the following reference to understand better the different components of a flux balance analysis problem:

- Orth, J., Thiele, I. & Palsson, B. What is flux balance analysis?. Nat Biotechnol 28, 245–248 (2010). https://doi.org/10.1038/nbt.1614

Computational tools for working with flux balance analysis models:

- Heirendt, Laurent et al. "Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0." Nature protocols vol. 14,3 (2019): 639-702. doi:10.1038/s41596-018-0098-2

3.2.1 Flux balance analysis problem structure

The FBA problem is typically encoded as a linear programming (LP) problem of the form:

$$\max_{v} \sum_{i=1}^{\mathcal{R}} c_i v_i$$

subject to the constraints:

$$\begin{aligned}
\mathbf{S}\mathbf{v} &= \mathbf{0} \\
\mathcal{L} \leq \mathbf{v} &\leq \mathcal{U} \\
\dots &= \dots
\end{aligned}$$

where $\mathbf{S}$ denotes the stoichiometric matrix, c_i denote the objective coefficients, $\mathbf{v}$ denotes the metabolic flux (the unknown that we are trying to estimate), and $\mathcal{L}$ ($\mathcal{U}$) denote the permissible lower (upper) bounds on the *unkown* metabolic flux. The first set of constraints enforces conservation of mass, while the second imparts thermodynamic and kinetic information into the calculation. Finally, there are potentially other types of constraints (both linear and nonlinear) used in this type of problem (we will not cover these here, but these additional constraints may be important in specific applications that we see later).

4. Formulation and properties of the stoichiometric matrix S

The stoichiometric matrix $\mathbf{S}$ is a $\mathcal{M}\times\mathcal{R}$ array holding the coefficients σ_{ij}; each row of $\mathbf{S}$ describes a metabolite, while each column represents a possible chemical reaction. Thus, the stoichiometric matrix is a *digital* representation of the biochemical capabilities of a metabolic network.

When formulating the stoichiometric matrix, what should be included to describe known or unknown desired phenomena? For example, suppose we are interested in producing product P or making sense of a metabolic data set. Should every possible reaction or just some *subset* of all possible reactions be included in the stoichiometric matrix? The practical significance of this question is two-fold:

- Fewer material balances are better when describing the behavior of a metabolic network, and
- Fewer chemical reactions (fluxes) result in less uncertainty when estimating the kinetic rates $\hat{r}_j$ (bounds) or other constraints.

Is there a systematic approach to systematically estimating all possible steady-state pathways through a metabolic network?

4.1 Convex pathway analysis

Convex pathway analysis is an approach to catalog all possible steady-state pathways through a metabolic network. In particular, convex decomposition approaches posit that any potential steady-state flux can be represented as the convex combination of a set of convex independent basis vectors called *extreme pathways* $\mathbf{P}_i$:

$$\mathbf{v}=\sum_{i=1}^{\mathcal{P}}\alpha_{i}\mathbf{P}_{i}$$

where $\mathcal{P}$ denotes the number of basis pathways, and $\mathbf{P}_i$ denotes the $\mathcal{R}\times 1$ extreme pathway basis vector. The term(s) $\alpha_i \geq 0$ denotes the weight of extreme pathway i.

Extreme pathways are *very difficult* to compute but can give super exciting insights into the operation and capabilities of a metabolic network.

Convex pathway analysis approaches:

- Bell SL, Palsson BØ. Expa: a program for calculating extreme pathways in biochemical reaction networks. Bioinformatics. 2005 Apr 15;21(8):1739-40. doi: 10.1093/bioinformatics/bti228. Epub 2004 Dec 21. PMID: 15613397.
- Trinh CT, Wlaschin A, Srienc F. Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular metabolism. Appl Microbiol Biotechnol. 2009;81(5):813-826. doi:10.1007/s00253-008-1770-1
- Bordbar A, Nagarajan H, Lewis NE, et al. Minimal metabolic pathway structure is consistent with associated biomolecular interactions. Mol Syst Biol. 2014;10(7):737. Published 2014 Jul 1. doi:10.15252/msb.20145243
- Song HS, Goldberg N, Mahajan A, Ramkrishna D. Sequential computation of elementary modes and minimal cut sets in genome-scale metabolic networks using alternate integer linear programming. Bioinformatics. 2017 Aug 1;33(15):2345-2353. doi: 10.1093/bioinformatics/btx171. PMID: 28369193.

Fig. 1C reproduced from Bordbar A, Nagarajan H, Lewis NE, et al. Minimal metabolic pathway structure is consistent with associated biomolecular interactions. Mol Syst Biol. 2014;10(7):737. Published 2014 Jul 1. doi:10.15252/msb.20145243

```julia
begin

    # Setup a collection of reaction strings -
    reaction_array = Array{String,1}()

    # encode the reactions -
    push!(reaction_array,"v₁,A+ATP,B+ADP,false")
    push!(reaction_array,"v₂,B+NAD,C+NADH,false")
    push!(reaction_array,"v₃,C+ADP,D+ATP,false")
    push!(reaction_array,"v₄,D+NADH,E+NAD,false")
    push!(reaction_array,"v₅,D+NAD,F+NADH,false")
    push!(reaction_array,"v₆,F+NADH,G+NAD,true")
    push!(reaction_array,"v₇,F,H,true")
    push!(reaction_array,"v₈,G+ATP,I+ADP,true")
    push!(reaction_array,"v₉,H+NAD,I+NADH,true")
    push!(reaction_array,"v₁₀,G,J,true")
    push!(reaction_array,"vₐ,∅,A,false")
    push!(reaction_array,"vₑ,E,∅,false")
    push!(reaction_array,"vₕ,H,∅,true")
    push!(reaction_array,"vⱼ,J,∅,true")
    push!(reaction_array,"vATP,ATP,∅,true")
    push!(reaction_array,"vADP,ADP,∅,true")
    push!(reaction_array,"vNADH,NADH,∅,true")
    push!(reaction_array,"vNAD,NAD,∅,true")

    # compute the stoichiometric matrix -
    (S, species_array, reaction_name_array) =
    build_stoichiometric_matrix(reaction_array);

    # show -
    nothing
end
```

```julia
(ℳ,ℛ) = size(S)
```

```julia
species_array
```

```julia
S
```

```julia
B = S |> binary_stoichiometric_matrix
```

4.2 Metabolite connectivity array (MCA)

The metabolite connectivity array (MCA) gives information about the 'connectedness' of the chemical species (metabolites) in a metabolic network. The metabolite connectivity array (MCA):

$$MCA = BB^T$$

has dimensions of $\mathcal{M}\times\mathcal{M}$, where $\mathbf{B}$ denotes the *binary* stoichiometric matrix (all non-zero values of $\mathbf{S}$ replaced by 1's). The diagonal elements of the MCA give the number of reactions that a particular metabolite participates in (either as a reactant or product). In contrast, the off-diagonal elements provide the number of shared reactions between metabolites i and j.

```
MCA = B*transpose(B)
```

4.3 Reaction connectivity array (RCA)

The reaction connectivity array (RCA) gives information about the 'connectedness' of the reaction space of a metabolic network. The reaction connectivity array (RCA):

$$RCA = B^TB$$

has dimensions of $\mathcal{R}\times\mathcal{R}$, where $\mathbf{B}$ denotes the *binary* stoichiometric matrix (all non-zero values of $\mathbf{S}$ replaced by 1's). The diagonal of the RCA gives the number of metabolites participating in a reaction (either as a reactant or product). In contrast, the off-diagonal elements provide the number of shared metabolites between reactions i and j.

```
RCA = transpose(B)*B
```

```
diag(RCA)
```

```
reaction_name_array[11]
```

5. Summary and conclusions

In this lecture we:

- Derived intracellular species material balance equations (in the continuum limit)
- Introduced constraint-based tools and concepts such as Flux Balance Analysis (FBA)
- Introduced the stoichiometric matrix and convex network decomposition methods

6. Next time

In the next lecture we will:

- Estimate the flux through a metabolic network
- Compare a dynamic simulation with a flux balance analysis computation
- Why is convex pathway analysis helpful (expa calculation and analysis)

7. Julia function library

```
function
binary_stoichiometric_matrix(matrix::Array{Float64,2})::Arra
y{Int64,2}

    # initialize -
    (ℳ,ℛ) = size(matrix)
    B = Array{Int64,2}(undef,ℳ,ℛ)

    for row_index ∈ 1:ℳ
        for col_index ∈ 1:ℛ

            old_value = matrix[row_index,col_index]
            if (old_value == 0.0)
                B[row_index,col_index] = 0
            else
                B[row_index,col_index] = 1
            end
        end
    end

    # return -
    return B
end
```

```
function
build_stoichiometric_matrix(reactions::Array{String,1})::Tup
le{Array{Float64,2},
    Array{String,1}, Array{String,1}}

    # initialize -
    species_array = Array{String,1}()
    reaction_array = Array{String,1}()
    reaction_dictionary_array =
   Array{Dict{String,Float64},1}()

    # first: let's discover the species list -
    for reaction_string ∈ reactions

        # initialize tmp storage -
        tmp_dictionary = Dict{String,Float64}()

        # split the reaction into its components -
        component_array = split(reaction_string,',');

        # reaction name -
        reaction_name = String.(component_array[1]);
        push!(reaction_array, reaction_name);

        # reactant phrase => 2, and product phrase => 3
        reactant_phrase = String.(component_array[2]);
        product_phrase = String.(component_array[3]);
```

```
 product_phrase = String.(component_array[3]);

 # generate species lists for the reactants and
products, then merge -
 merge!(tmp_dictionary,
extract_species_dictionary(reactant_phrase; direction
= -1.0))
 merge!(tmp_dictionary,
extract_species_dictionary(product_phrase; direction
= 1.0))

 # grab the tmp_dictionary for later -
 push!(reaction_dictionary_array, tmp_dictionary)

 # the species that we need to look at are the keys
of the tmp_dictionary -
 tmp_species_list = keys(tmp_dictionary)

 # we need a unique species list, so check to see if
we have already discovered this species -
```

```julia
function extract_species_dictionary(reaction_phrase::String;
    direction::Float64 = -1.0)::Dict{String,Float64}

    # initialize -
    species_symbol_dictionary = Dict{String,Float64}()

    # ok, do we hve a +?
    component_array = split(reaction_phrase,'+');
    for component ∈ component_array

        if (contains(component,'*') == true)

            tmp_array = split(component,'*')
            st_coeff = direction*parse(Float64,tmp_array[1])
            species_symbol = String(tmp_array[2])

            # don't cache the ∅ -
            if (species_symbols != "∅")
                species_symbol_dictionary[species_symbol] =
              st_coeff
            end
        else

            # strip any spaces -
            species_symbol = component |> lstrip |> rstrip

            # don't cache the ∅ -
            if (species_symbol != "∅")
                species_symbol_dictionary[species_symbol] =
              direction*1.0
            end
        end
    end

    # return -
    return species_symbol_dictionary
end
```

8. Pluto notebook setup

```julia
begin

    # import some packages -
    using PlutoUI
    using LinearAlgebra

    # setup paths -
    const _PATH_TO_NOTEBOOK = pwd()
    const _PATH_TO_DATA = joinpath(_PATH_TO_NOTEBOOK,"data")
    const _PATH_TO_FIGS = joinpath(_PATH_TO_NOTEBOOK,"figs")
    const _PATH_TO_SRC = joinpath(_PATH_TO_NOTEBOOK,"src")

    # return -
    nothing
end
```

```julia
TableOfContents(title="📚 Table of Contents", indent=true,
depth=5, aside=true)
```