rba.core
Building constraint matrices from RBA data.

Biosys - MAIAGE

September 30, 2018

Contents

1. RBA constraints

In order to build an RBA system, we need to build three sets of constraints.

(C_1) Mass conservation.

(C_2) Capacity constraints for enzymes and, e.g., ribosomes.

(C_3) Density constraints.

All these constraints are linear (in)equalities. We will write them in matrix form, as it is the best way to summarize all information needed to build a constraint.

2. Important concepts

Production/degradation vector for macromolecules In the following, each macromolecule is described by its **production vector**. It is a column vector containing the metabolites necessary to build it, with a **minus** sign for metabolites consumed and a **plus** sign for byproducts generated. Similarly, we define **degradation vectors**.

Concentration to flux conversion Suppose you have a metabolite at concentration $C = n/V$. At growth-rate μ, we have $dV \simeq \mu V$. The variation of concentration due to dilution is

$$dC = d(n/V) = -n/V^2 dV = -\mu C$$

In other words, in order to keep the concentration constant, we need an input metabolite flux of

$$\nu = \mu C$$

When a variable is expressed as a concentration, it may be necessary to convert it to a flux by using this relationship. When necessary, we indicate this conversion by inserting a conversion matrix between the main matrix and the variable vector.

Variable vector representation Variables are represented as a row vector below matrices. This clarifies how colums and variables are associated. Note the transpose at the end of the row vector if you are worried about mathematical purity.

3. Matrix variables

- Fluxes through metabolic reactions $R = (\nu_1, \dots, \nu_R)$.
- Enzyme concentrations $E = (e_1, \dots, e_E)$.
- Process machinery concentrations $P = (p_1, \dots, p_P)$.

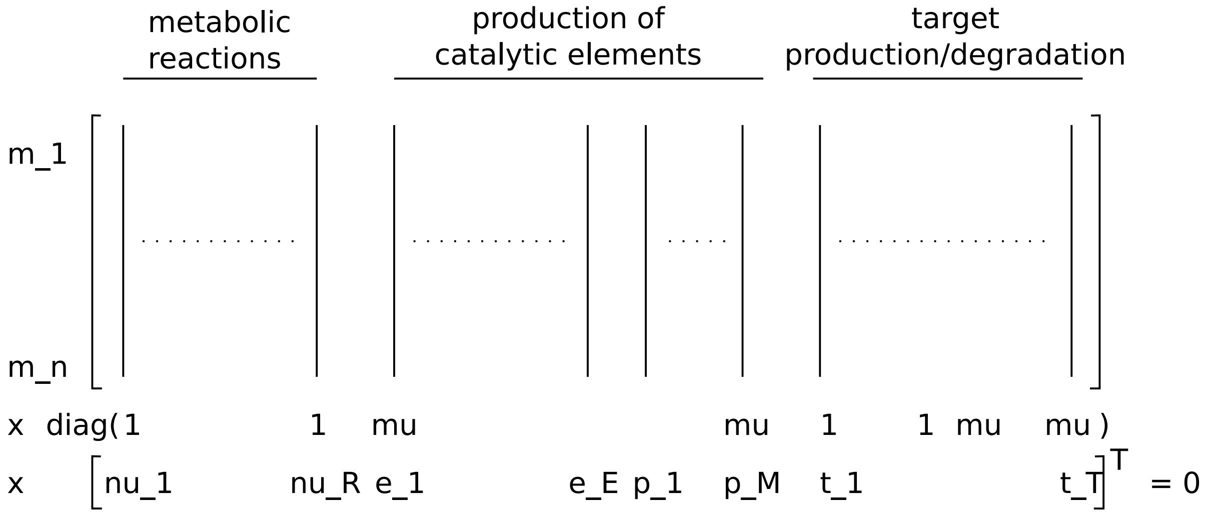

metabolic reactions
production of catalytic elements
target production/degradation
m_1
m_n
x diag(1 1 mu mu 1 1 mu mu)
x [nu_1 nu_R e_1 e_E p_1 p_M t_1 t_T] T = 0

Figure 1: Mass conservation constraint. Bars in the main matrix are metabolic reaction or production/degradation vectors associated with each variable.

- Target values $T = (t_1, \dots, t_T)$. Targets may be expressed as fluxes or concentrations. We note target fluxes TF and target concentrations TC. When necessary, we assume that flux targets are all listed first, i.e., $T = (TF, TC)$.

Targets are reactions that are necessary for the cell to be functional (e.g. keeping key metabolites at some given concentration). Note that target values may actually be predetermined values, while all others are true variables that must be optimized. We will see later how predetermined values may be eliminated.

4. Constraint matrices

4.1. Mass conservation (C_1)

The mass conservation constraint is represented by a matrix where rows are metabolites, labelled $M = (m_1, \dots, m_M)$. Metabolite fluxes must cancel out in order to achive mass conservation. Note that numerous variables are expressed as concentrations and must be converted to fluxes as explained previously.

Figure 1 shows how mass conservation is expressed in matrix formalism. The main matrix contains the metabolic reactions, the production vectors for catalytic elements, the production/degradation vectors for target elements. This matrix is growth-rate independent. The second matrix converts variables that are expressed as concentrations to fluxes. This can also be seen as multiplying the appropriate columns in the main matrix by the growth-rate.

4.2. Capacity constraints (C_2)

The capacity constraintes are represented by a matrix where rows are enzymes or process machineries. Currently, we use two different formalisms for enzymes and process machineries.

Enzymes are associated to exactly one reaction that may be reversible. For every enzyme we have the following constraint:

$$\nu \leq k_{forward} e_i$$

where ν is the flux through the reaction catalyzed by e_i. If the reaction is irreversible, we have the additional constraint

$$\nu \geq -k_{backward} e_i$$

In order to write all constraints, we need a reaction to enzyme mapping. This is represented by a matrix where we have one row per constraint. Each row has a 1 on the column corresponding to the reaction catalyzed, and 0s everywhere else.

Process machineries participate in the synthesis/degradation of several macromolecules (enzymes, machineries and targets). For every target, we have a constraint of the form

$$[machinery\_cost].[E, P, T]^T \leq k_{machinery} p_i$$

Every macromolecule has a set of machinery costs associated with it. It tells how much a machinery is used in order to produce/degrade the macromolecule (the cost is often 0).

The final matrices are very sparse (Fig. 2). A first matrix contains the reaction to enzyme mapping and the machinery costs. This matrix is growth-rate independent. A second matrix contains efficiencies, that may depend on growth-rate. Note that we did not need to convert concentrations to fluxes here.

4.3. Density constraints (C_3)

Density constraints are represented by matrices where rows are compartments, labelled $C = (c_1, \dots, c_C)$.

Every variable that represents a concentration participates to this constraint. For every macromolecule, we define a weight vector that defines how much volume one molecule occupies in every compartment (in user defined units). By putting these vectors together we get a weight matrix.

The user also defines a vector of maximal weights for every compartment, yielding a simple set of constraints (Fig. 3). Only the right-hand part may contain growth-rate dependent values. Again, no conversion from concentration to flux is needed here.

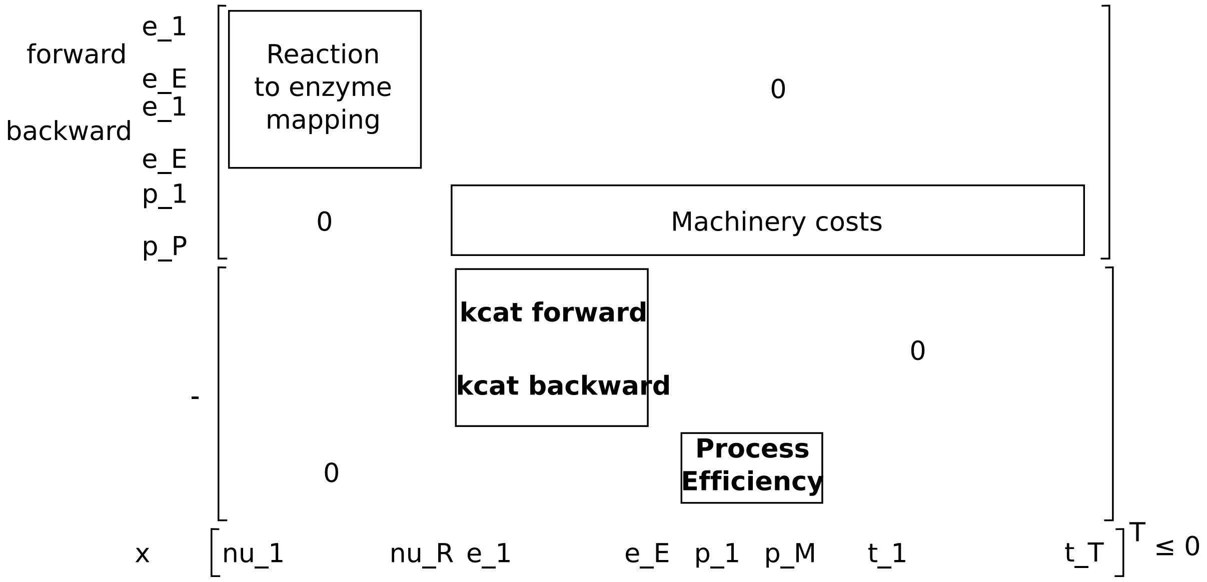


Figure 2: Capacity constraints.

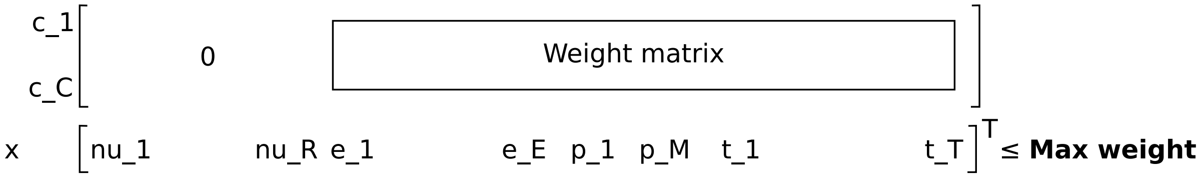
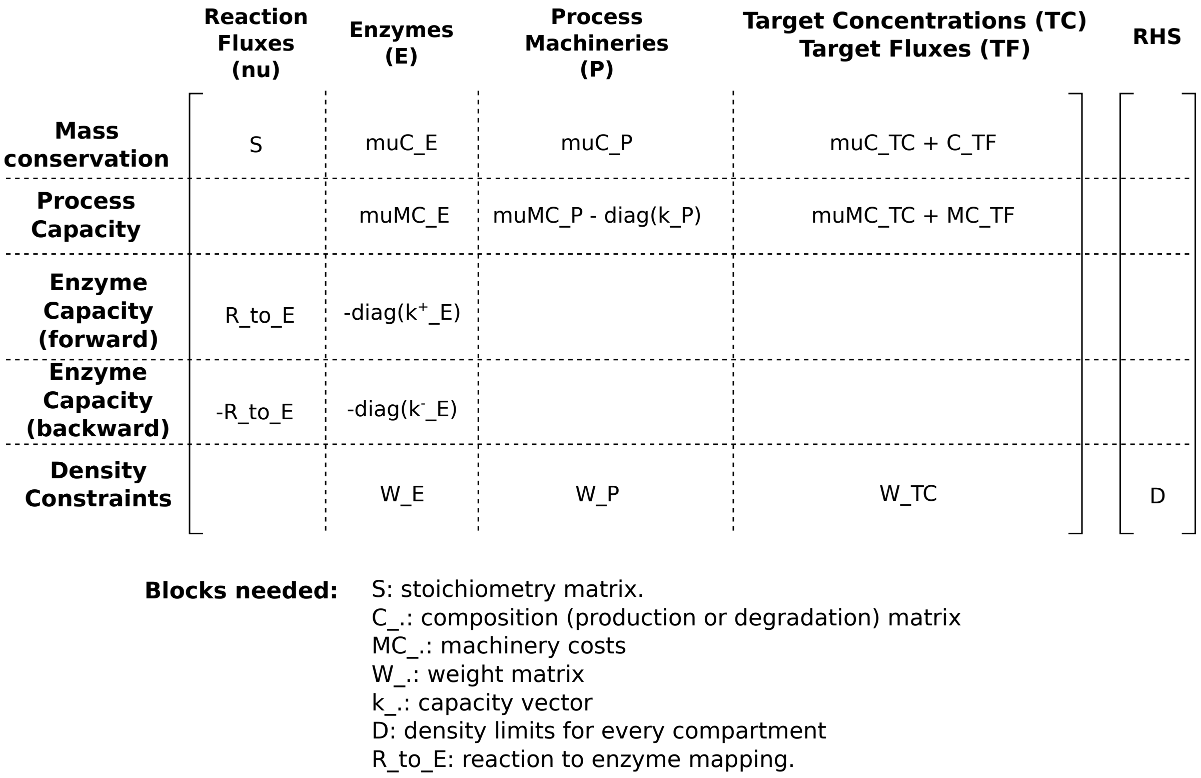


Figure 3: Density constraints.



	Reaction Fluxes (nu)	Enzymes (E)	Process Machineries (P)	Target Concentrations (TC) Target Fluxes (TF)	RHS
Mass conservation	S	muC_E	muC_P	muC_TC + C_TF	
Process Capacity		muMC_E	muMC_P - diag(k_P)	muMC_TC + MC_TF	
Enzyme Capacity (forward)	R_to_E	-diag(k^+_E)			
Enzyme Capacity (backward)	-R_to_E	-diag(k^-_E)			
Density Constraints		W_E	W_P	W_TC	D

Blocks needed:
S: stoichiometry matrix.
C_.: composition (production or degradation) matrix
MC_.: machinery costs
W_.: weight matrix
k_.: capacity vector
D: density limits for every compartment
R_to_E: reaction to enzyme mapping.

Figure 4: Blocks that need to be assembled. There is also a vector of constraint signs that is omitted here (E for equality, L for lower than, G for greater than).

5. Building matrices from XML files

In this file, we briefly describe how matrices are built from files.

Building blocks Figure 4 shows the blocks that are used to build the final matrices.

Stoichiometry matrix The stoichiometry matrix is built from metabolic reactions. External metabolites are removed from the metabolite pool.

Density limits Density limits are simply extracted from RBADensity and assembled into a vector (one coefficient per compartment).

Species matrices Figure 5 shows how macromolecules are broken down into matrices describing their composition, machinery cost and weight. In the end, they are merged into a single matrix describing composition, machinery cost and weight of all metabolites and macromolecules.

Enzyme and machinery matrices Figure 6 shows how enzyme and process machinery matrices are built.

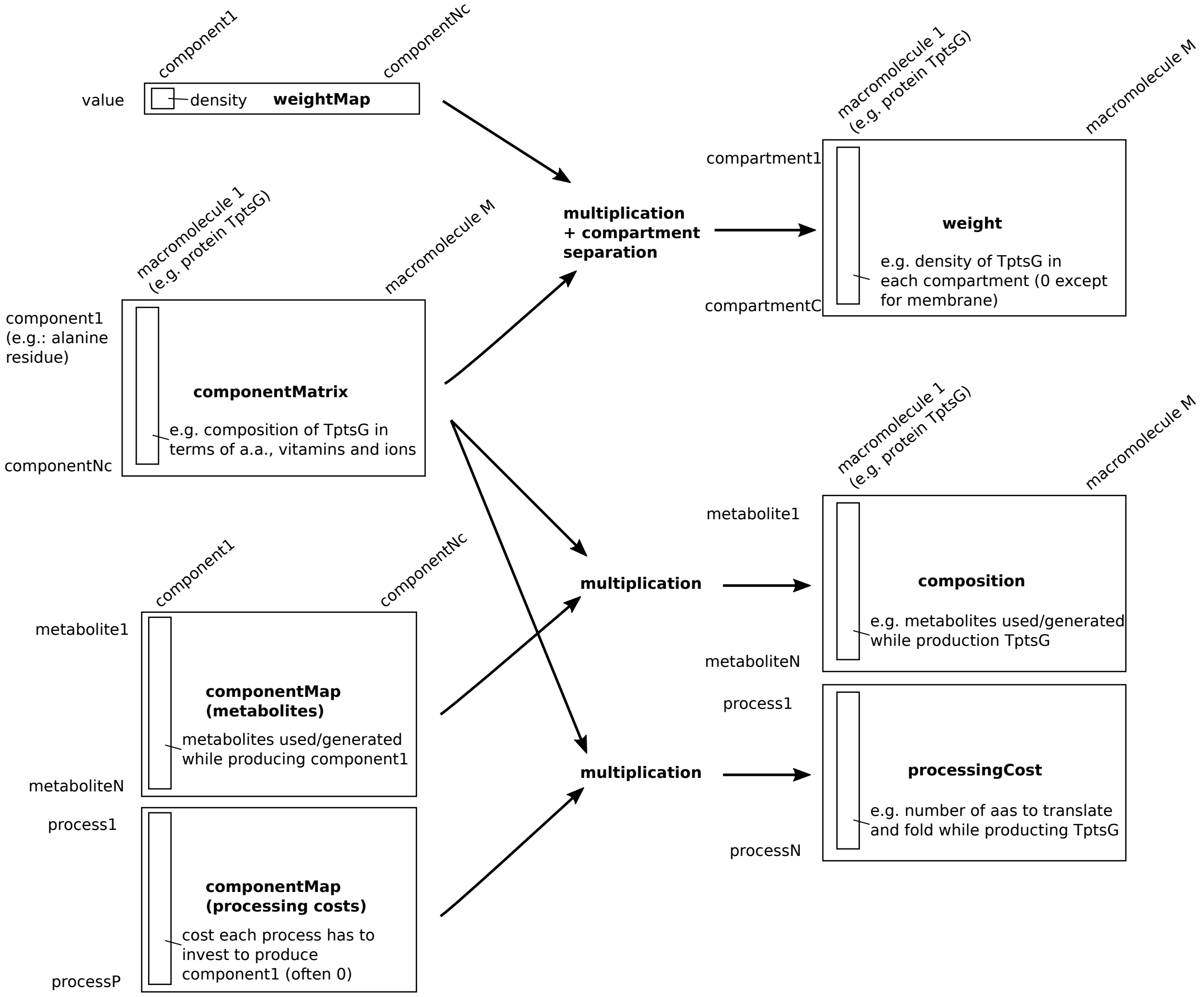


Figure 5: Matrices extracted from macromolecule information. An example is given with proteins but in the end, they contain all macromolecules and all internal metabolites.

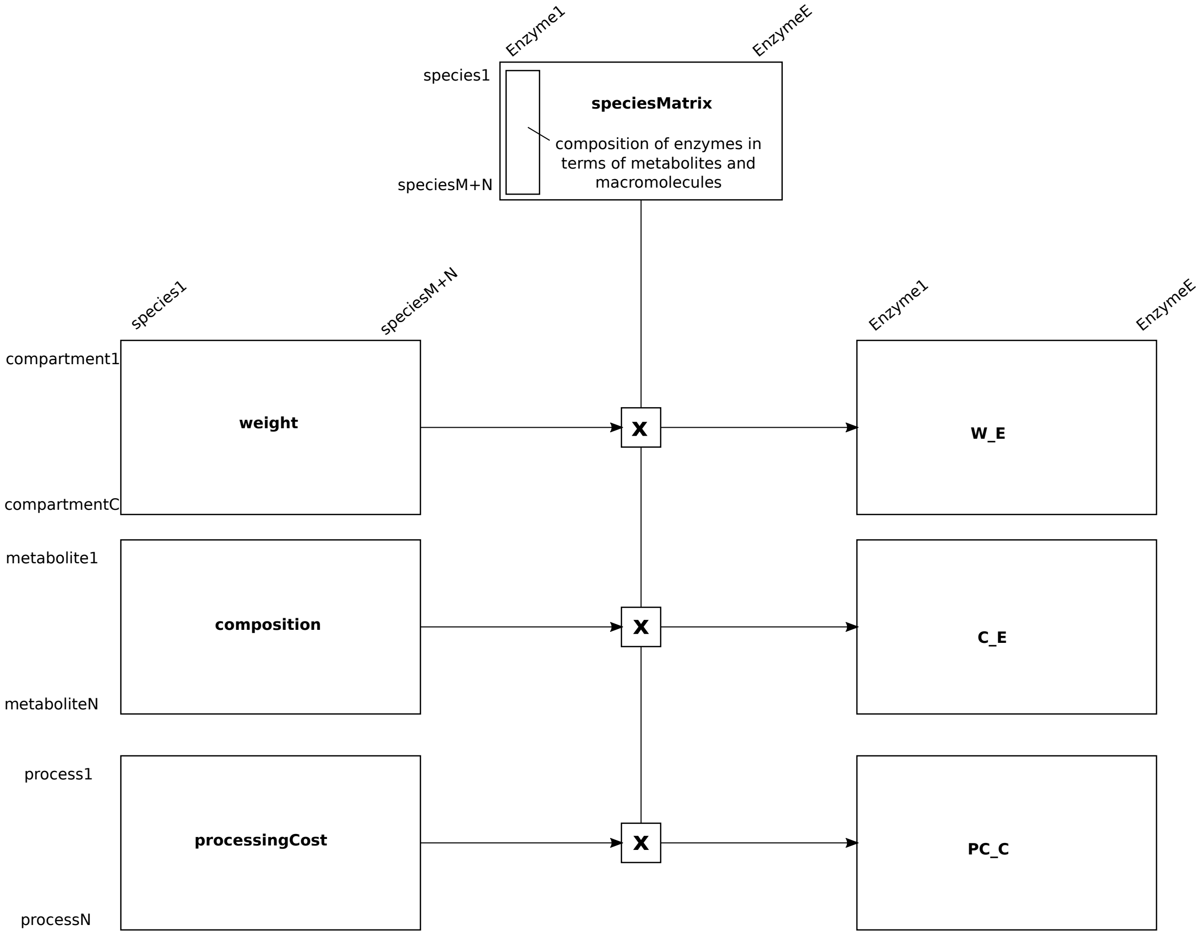


Figure 6: Every machinery can be described by a reaction matrix. Reactants are species (metabolites or macromolecules) needed to build the machinery and products are byproducts of the assembly process. Through matrix multiplication with the species matrices, we can deduce its composition, weight and machinery cost.

Target matrices Targets are either metabolites or macromolecules. Their composition, machinery cost and weight can be extracted as columns from the species matrices.

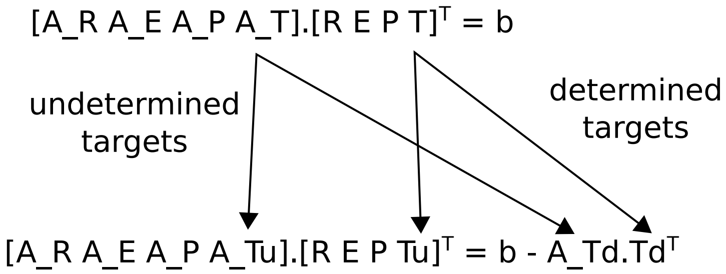

[A_R A_E A_P A_T].[R E P T]^T = b
undetermined targets
determined targets
[A_R A_E A_P A_Tu].[R E P Tu]^T = b - A_Td.Td^T

Figure 7: Procedure to reduce matrix size by eliminating targets with predefined values. This yield a smaller A matrix and a slightly more complicated b matrix.

A. Eliminating target variables

In the previous sections, we included all targets as variables. In practice, most variables are predetermined, e.g., the target for concentrations for metabolites is known, it does not need to be optimized. Optimization algorithms will typically eliminate all such variables in a presolving step, but we might want to eliminate them manually in order to reduce matrix sizes.

The procedure is fairly simple: we extract the submatrix corresponding to targets with known values, multiply it by the value vector, and move it to the right-hand side (Fig. 7).

B. Work in progress

B.1. Harmonize enzymes and process machineries

Currently every reaction must be catalyzed by a different enzyme. In comparison, process machineries, such as ribosomes, may catalyze multiple production reactions. We may use a similar system for enzymes where one enzyme may catalyze several reactions.

In this case, the reaction to enzyme mapping becomes a little more complicated and must take into account

- Different k_cat values for different reactions.
- Distinguish forward and backward reactions.

A side document explains how enzyme capacity constraints may be rewritten in order to be closer to process capacity constraints.

Implementing the new mapping yields smaller matrices, as we need less constraints overall.

B.2. Generalize density constraints

Instead of having one density constraint per compartment, it would be nice to have constraints for linear combinations of compartments. This would enable to define global density constraints, i.e, the sum of weights in all compartments has to be lower or equal to some limit.

B.3. Matrix conditioning and objective function

Performance is extremely variable because our matrix is not well conditioned. Sometimes, adding zero rows and columns drastically improves convergence (not sure why, cplex eliminates these rows and columns during presolving, but somehow converges in fewer iterations, maybe a bigger matrix increases convergence tolerance?). Finding a good matrix scaling and objective function may affect performance significantly.