# Artificial Metabolic Networks: enabling neural computation with metabolic networks

Léon Faure[1,2], Bastien Mollet[2,3], Wolfram Liebermeister[1,4], and Jean-Loup Faulon[1,2,5,6*]

[1]University of Paris-Saclay, Saclay, France, [2]MICALIS Institute INRAe, Jouy-en-Josas, France, [3]ENS Lyon, Lyon, France, [4]MaIAGE, INRAe, Jouy-en-Josas, France, [5]LSSB Laboratory, Genoscope, CEA, CNRS, [6]Manchester Institute of Biotechnology, University of Manchester, Manchester, UK.
*Corresponding author: Jean-loup.Faulon@inrae.fr, ORCID 0000-0003-4274-2953

## Abstract

Metabolic networks have largely been exploited as mechanistic tools to predict the behavior of microorganisms with a defined genotype in different environments. However, flux predictions by constraint-based modeling approaches are limited in quality unless labor intensive experiments including the measurement of media intake fluxes, are performed. Using machine learning instead of an optimization of biomass flux – on which most existing constraint-based methods are based – provides ways to improve flux and growth rate predictions. In this paper, we show how Recurrent Neural Networks can surrogate constraint-based modeling and make metabolic networks suitable for backpropagation and consequently be used as an architecture for machine learning. We refer to our hybrid - mechanistic and neural network – models as Artificial Metabolic Networks (AMN). We showcase AMN and illustrate its performance with an experimental dataset of *Escherichia coli* growth rates in 73 different media compositions. We reach a regression coefficient of $R^2$=0.78 on cross-validation sets. We expect AMNs to provide easier discovery of metabolic insights and prompt new biotechnological applications.

**Keywords**: Artificial Neural Network, Metabolic Network, Mechanistic Modeling, Metabolic Flux Analysis, Scientific Machine Learning, Hybrid Modeling, Artificial Metabolic Network.

**Abbreviations**: **AMN**: Artificial Metabolic Networks, **ANN**: Artificial Neural Network, **CBM**: Constraint-Based Modelling, (p)**FBA**: (parsimonious) Flux Balance Analysis, **LP**(QP): Linear (Quadradic) Programming, **MFA**: Metabolic Flux Analysis, **ML**: Machine Learning, **MM**: Mechanistic Modelling, **RNN**: Recurrent Neural Network.

# Introduction

The increasing amounts of data available for biological research brings the challenge of data integration with machine learning (ML) methods. Systems and Synthetic Biology along with Metabolic Engineering are no exception to this trend [1][2][3][4]. Metabolic Engineering relies on models that predict the phenotype of a strain from its genotype and environment. In the past three decades, Constraint-Based Modelling (CBM) has been the main approach to study the relationship between the uptake of nutrients and the metabolic phenotype (*i.e.*, the steady-state flux distribution) of a given organism, *e.g.*, *E. coli*, with a model iteratively refined for 30 years [5]. The main pitfall in CBM is the under-determination of the system. As a result, CBM needs to be constrained to obtain realistic flux distributions. Constraining consists in setting bounds for some fluxes and restraining the possible solutions of the model to the ones compliant with the constraints. To find these constraints, fluxomic datasets can be obtained from different experimental flux determination methods. With isotopic labeling (like 13-C), one can follow the path of a nutrient in the metabolic network [6]. With metabolomics methods, one can derive metabolic fluxes from metabolite concentrations and possibly in a time-resolved approach [7]. With transcriptomics methods, RNA sequencing data is used as input for models estimating fluxes in an indirect fashion, which, recently has even been achieved at the single-cell level [8]. In some cases, data integration of several -omics methods is possible, constituting a state-of-the-art multi-omics data integration for Metabolic Flux Analysis (MFA) [9][10][11]. Naturally, ML approaches were developed to efficiently integrate the data and enhance the predictive power of CBM. However, as described by Sahu *et al*. [12], in MFA, the interplay between CBM and ML is still showing a gap: some approaches use ML as input for CBM, others use CBM as input for ML, but none of them can do both, like we attempt to do in this paper with the Artificial Metabolic Networks (AMNs) hybrid model.

The lack of progress towards integrating mechanistic modelling (MM, to which CBM belongs) and machine learning in fundamental biology studies is at odds with the widespread adoption of ML approaches by the life science communities. MM and ML approaches are based on two different paradigms. While the former is aimed at understanding biological phenomena, some systems are too complex to be modeled with interpretability. Conversely, the latter does a good job predicting the outcomes of complex biological processes using large input/output datasets, but does not provide understanding of the underlying mechanisms. The pros of one are the cons of the other, suggesting that a hybrid approach should be developed towards enabling a symbiotic relationship between both. An emerging field, Scientific Machine Learning (SciML), aims to develop such hybrid models [13]. The

main advantage of hybrid modeling is to bring models that comply well with experimental results via ML, but that also take mechanistic insights from MM. Recently, Nilsson *et al*. [14] showed good performances of such a model, developed for signaling networks, where they used the supposed topology of a signaling network as a platform for learning on experimental data, therefore displaying more insights than a black-box approach. Also, Anantharaman *et al*. [15] showed that an implicit ML architecture, here a reservoir specifically designed for surrogating stiff mechanistic Ordinary Differential Equations (ODEs) used in quantitative systems pharmacology, can function in a more accurate, fast and robust fashion than classical ODE solvers, once these implicit architectures have been trained on mechanistic simulations. Another example of hybrid modeling for biological systems is the work of Lagergren *et al.* [16], who estimated the parameters of a mechanistic model using "biologically-informed neural networks". Finally, Culley *et al*. [17] showed how simulating data with a mechanistic metabolic model could enhance machine learning approaches to characterize *S. cerevisiae* growth and bring more biological insights.

The hybrid AMN model shown here fits in the emerging SciML field. AMNs bridge the gap between ML and CBM by solving linear programming (LP) problems for metabolic flux models with a recurrent neural network (RNN) that has the same topology as the metabolic network itself. By doing so, our model is a mechanistic model, determined by the stoichiometry and any other possible constraint of CBM, but also a ML model, as it can be used as a learning architecture, with any MFA suitable data.

The use of RNNs for solving optimization problems is a long-standing field of research [18] inspired by the pioneering work of Hopfield and Tank [19]. A few years later, RNNs were showcased to perform well for solving linear problems [20]. Simpler and more efficient networks were developed over the years [21][22] integrating both linear and Quadratic Programming (QP) problems to be in the reach of the network. In this study, we were able to use those RNNs for solving the LP optimization problems of CBM .

CBM has a critical limitation for experimentalists: while realistic and condition-dependent bounds on uptake rates would be critical for growth rate predictions, the conversion from extracellular concentrations, *i.e*., the controlled experimental setting, to such bounds on uptake rates, is unknown. The satFBA [23] method exploits kinetic models to predict this conversion, relying on a saturation value for each exchange reaction. Consequently, this MM method relies on assumptions, not necessarily valid, and knowledge, not necessarily accessible, and does not integrate any large-scale regulation phenomena. As a result, classical CBM can predict growth yields but can hardly be used to predict actual growth rates [24]. AMNs are showcased in this study for tackling the same issue:

predicting metabolites uptake rates from their external concentrations. To do so, where satFBA uses another layer of mechanical kinetic modelling, we use a pre-processing dense neural layer, that is accessible for learning thanks to the AMN gradient backpropagation abilities.

AMNs provide a new paradigm for MFA: instead of predicting fluxes and growth rates from an optimality principle (suited for CBM), we do so by learning from known flux distributions. First, we develop models that can learn from flux distributions and show their good performance. The flux distributions used as learning references (the training sets) are produced by parsimonious Flux Balance Analysis (pFBA), maximizing the biomass reaction (*i.e*., the growth rate) and subsequently minimizing all the other fluxes (with the biomass reaction fixed at its optimal value). Once the model has been trained on adequate simulated pFBA data, we immobilized its parameters, resulting in a gradient backpropagation compatible reservoir. This reservoir is then used to tackle the abovementioned issue of unknown uptake rates. A dense neural network layer predicting uptake rates from external metabolite concentrations is then trained to feed uptake rate to the reservoir. That dense layer can be reused by any FBA user to improve the predictive power of a metabolic model, with an adequate experimental set-up.

# Results

We first present the basic design and functioning of two alternative neural computation methods that surrogate CBM, *i.e*., the prediction of intracellular fluxes and growth rate based on predefined bounds on cellular uptake rates. We use simulation data as reference, generated with pFBA, with different models of different sizes and different media compositions. Note that for this part, our methods are simply mimicking predictions by classical CBM with a neural network approach. Importantly, they are not AMNs per se, but are core methods that '*replace*' CBM in our AMNs. A crucial step before establishing neural network approaches on metabolic networks is to ensure that all reactions are unidirectional which has been done by splitting all bi-directional reactions in the model into forward and backward reactions. In the resulting model, all reaction fluxes are positive, and flux cycles within a single reaction are suppressed by the usage of pFBA (see Methods - Making metabolic networks suitable for neural computations).

The first method (Fig. 1) learns a weight matrix from example flux distributions, which represents consensual flux branching ratios found in the training set. Consequently, we assume that a consensual metabolic state exists for most of the flux distributions examples of our training set. As a result, rules for generating the training sets are critical for good performance of this method, those are detailed in Methods – Generation of training sets with COBRApy. A simple toy model network is shown to demonstrate the functioning of this first method in Fig. 1A. With CBM, the flux distribution maximizing the rate of the 'BIOMASS' reaction providing a nutrient uptake through reaction 'TPI' is obtained by solving a linear program (Fig. 1B). At steady state, metabolite production fluxes can be calculated from reaction fluxes via the transition matrix *V2M* from fluxes to metabolites ($V2M_{i,j}$ = {$S_{i,j}$ if $S_{i,j} > 0$, 0 elsewhere} more details are given in Fig. 1C), similarly reaction fluxes can be calculated from metabolite production fluxes using the matrices *M2V* and $W^{sst}$. For a given metabolite *i* and a reaction *j*, the weight $w_{ij}$ (in matrix $W^{sst}$) represents the fraction of metabolite *i* production flux going to reaction *j,* and is therefore flux distribution dependent. Here we assume that the branching ratios remain similar between flux distributions, then attempt to learn "typical" ratios from flux data.

To learn the weights $w_{ij}$, one first needs to translate the model into a neural-network-like architecture (Fig. 1D). Precisely, in Fig. 1D we start from an initial set of given fluxes and then propagate knowledge about the fluxes through the entire network, each layer corresponding to one step in flux propagation. Mathematically, each layer is composed of two simple operations that update the *M* and *V* vectors, respectively representing metabolites production fluxes and reaction fluxes. Those operations are repeated until convergence, with $v_1$ being constant because it is known from the start, and the equilibrium is reached in cycles (for instance $\{v_2, v_4, v_5\}$). The network shown in Fig. 1D is the unrolled representation of the Recurrent Neural Network (named RNN-W) depicted in Fig. 1E. As it will be discussed later, the matrix *W* can be learned through training, assuming the learned weights are those of the matrix $W^{sst}$ (Fig. 1C) the steady state fluxes of RNN-W iterated 30 times (or more) equal to those obtained solving MFA with CBM's linear programs (Fig. 1F).

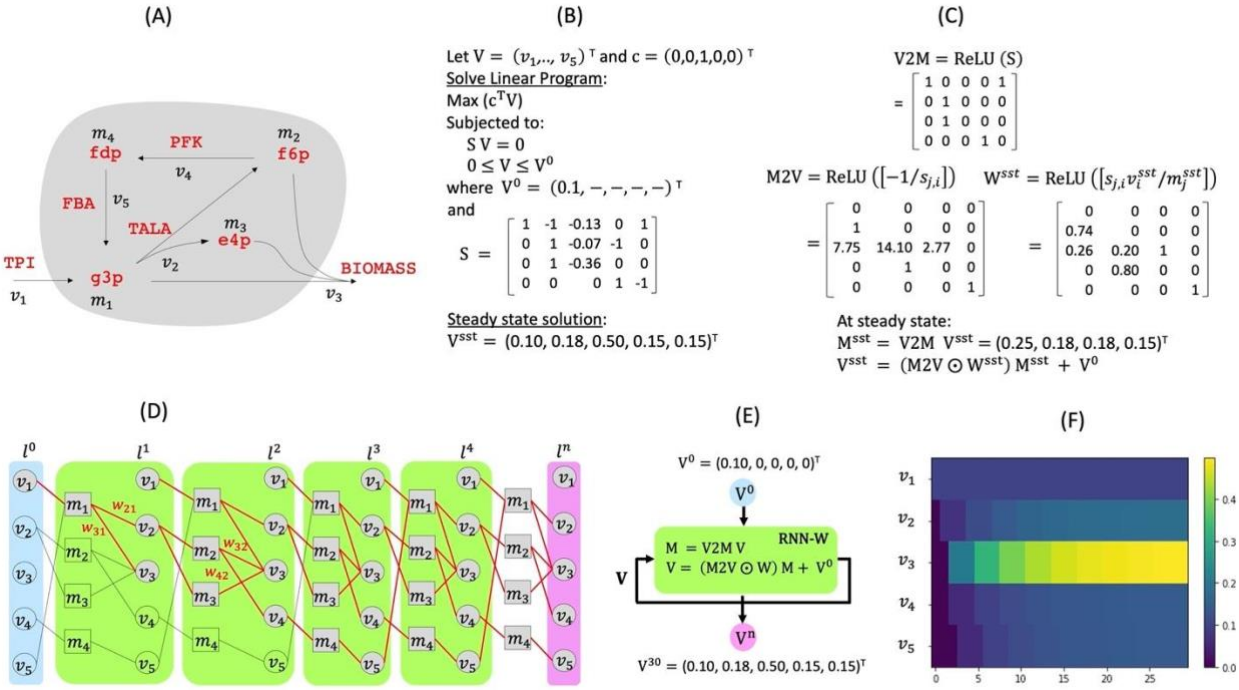


*Fig. 1. Computing steady-state fluxes with stoichiometric and neural models. **(A)** Simple stoichiometric model. The model corresponds to the upper Glycolysis pathways of E. coli (iML1515 model extracted from the BiGG database* [25]*). The model is unidirectional, all fluxes values are positive. **(B)** Steady-state solution fluxes maximizing biomass production. At steady-state, the reaction fluxes ($v_i$) must satisfy stationarity conditions that guarantee mass balance of all metabolites, this is depicted by the equation $SV = 0$, where $S$ is the stoichiometric matrix representing the connectivity of the model and $V$ the vector of fluxes to be calculated. $V^0$ is the medium represented by a vector of nutrient uptake fluxes (here $v_1$= TPI = 0.1, symbol "—" indicates that no value is provided). The steady-state solution $V^{sst}$ is calculated by solving a linear program maximizing the objective $c^TV = v_3$ = BIOMASS, here we used the COBRApy package* [26] *to compute $V^{sst}$. **(C)** Stoichiometric model matrices. $V2M$ is the matrix to compute metabolite production fluxes from reaction fluxes, $M2V$ is a matrix to compute reaction fluxes from the production fluxes of the reaction substrates. When a metabolite is the substrate of several reactions, each reaction gets a fraction of the metabolite production flux, this is depicted in matrix $W^{sst}$. $M^{sst}$ is the vector of metabolite production fluxes at steady-state, the operator ⊙ stands for element-wise matrix product, and ReLU(x)=max (0, x). **(D)** Unrolled neural network built from the model. In this theoretical example an initial flux vector is set, representing only the fluxes in the exchange reactions, is mapped to a flux vector covering the entire network. In the initial layer ($l^0$) only $v_1$ has a value ($v_1^0$). In layer 1, $v_1$ value is passed to $m_1$, the production flux for metabolite 1, $m_1^1 = v_1^0$.Subsequently a fraction ($w_{21}$) of $m_1$ goes to $v_2$ and the other fraction ($w_{31}$) to $v_3$, therefore $v_2^1 = w_{21}m_1^1$, $v_3^1 = w_{31}m_1^1$ with $w_{21}$+$w_{31} = 1$. Additionally, $v_1$ remains as in $l^0$: $v_1^1 = v_1^0$. In layer 2, $v_1$ continues to feed $m_1$, $v_2$ is passed on to $m_2$ and $m_3$ , therefore, $m_2^2 = v_1^1$ and $m_3^2 = v_1^1$, $m_2$ then goes to $v_3$ and $v_4$, and as in the previous layer we have $v_3^2 = w_{32}m_2^2$, $v_4^2 = w_{42}m_2^2$ with $w_{32}$+$w_{42} = 1$, other fluxes remain the same as in $l^1$, i.e., $v_2^2 = w_{21}m_1^2$, $v_3^2 = w_{31}m_1^2$ $v_1^2 = v_1^1$. In layer 3, $m_3$ receives input from $v_4$ which in turn activates $v_5$. In layer 4,$m_1$receives input from both $v_1$ and $v_5$: $m_1^4 = v_1^3 + v_5^3$. **(E)** Recurrent neural network (RNN) representation. At each iteration step (l) $V^l$ and $M^l$ are computed using matrices V2M and M2V of panel C, while the matrix W can be learned by training with experimental data or model simulations. For example, setting $v_1^0$ = 0.1 and searching weights for which $v_3^n$ = 0.5 one finds $w_{21}$ = 0.74, $w_{31}$ = 0.26, $w_{32}$ = 0.20, and $w_{42}$ = 0.80. Taking these weights, the $V^n$values obtained for n=30 match those of panel B. **(F)** Heatmap for flux values up to 30 RNN iterations.*

While the RNN-W method is simple to implement it has several drawbacks. First, the method is not performant enough for learning on large metabolic models and datasets, notably because of the large number of parameters (cf. Table 1, architecture RNN-W). More problematic is the fact that a fixed set of weights – corresponding to assumed fixed branching ratios in flux distributions - may not fit all instances of a training set, since these branching ratios are typically changing (for example, in a switch between overflow metabolism and respiration). Supplementary Fig. S1 shows an example in which two flux distributions, with different uptake fluxes, lead to different weights. Consequently, we cannot assume that our model will precisely represent very diverse flux distributions. To overcome this

shortcoming, we next present an alternative RNN method for solving CBM optimization problems. These other methods are much closer to the LP optimization behind CBM: their input can be exact or upper bounds on fluxes, and an objective reaction (or set of reactions) to optimize can be set – or not.

Our second method is inspired by the works of Ghasabi-Oskoei *et al.* [21] and Yang *et al.* [22]. The method is still a core method that aims to 'replace' CBM but is not the AMN itself. The method makes use of RNN to solve linear and quadratic problems, using exact constraint bounds (EB) in Ghasabi-Oskoei *et al.* and upper-bounds (UB) in Yang *et al.* In the same way as the RNN-W method, RNN-EB and RNN-UB iteratively compute fluxes to come closer to the steady-state solution. However, calculations are more sophisticated (Fig. 2A-B) and integrate the same objective function as in classical CBM (cf. Method - Solving LP/QP with RNN methods - for further details). These RNNs can achieve almost identical results as pFBA. In both cases we used the same initial conditions as with pFBA: known intake fluxes with EB and upper bound for unknown intake fluxes with UB. The results obtained by the two RNNs are compared with the flux value to optimize (growth rate in Fig. 2C) or with the global regression coefficient ($R^2$) on all fluxes (Fig. 2D). While the match with pFBA is excellent for the prediction of growth rates, it becomes a bit worse when comparing the predictions for intracellular fluxes. This is due to the fact that the flux solutions of pFBA can be non-unique: many pFBA calculated steady state flux distributions may lead to the same growth rate, so for any given growth rate there is no guarantee that the RNN calculated fluxes will match the pFBA fluxes.

While RNN-EB and -UB perform well, their main weakness is the number of iterations needed to reach satisfactory performances, at least 400,000 (Fig. 2D). Since our goal is to integrate such RNNs in a learning architecture, this drawback has to be tackled. As illustrated in Fig. 3A, our solution is to improve our initial guesses for fluxes, by training a prior network (a classical dense ANN architecture) to compute initial values for all fluxes ($V^0$) from medium intake fluxes ($V^{in}$). This prior network is trained along with the RNN with the goal of finding $V^0$ values that are as close as possible to the optimum flux values (pFBA steady state fluxes). This is achieved by setting the iteration number of the RNN to low values.

In the remainder of the paper, we name Artificial Metabolic Network (AMN), the hybrid model shown in Fig. 3A composed of a prior dense network and an RNN. We note that the RNN acts as a mechanistic layer as it contains the stoichiometry and bounds of a metabolic network. In Fig 3B and 3C we present two examples of AMNs making use of RNN-EB and RNN-W respectively, further details can be found in the section Method - AMN architectures and parameters. Performances of our AMNs are compared with other more classical architectures in Table 1. Overall, we find that the architecture RNN-EB provides the best performances, especially for cross-validation independent test sets.

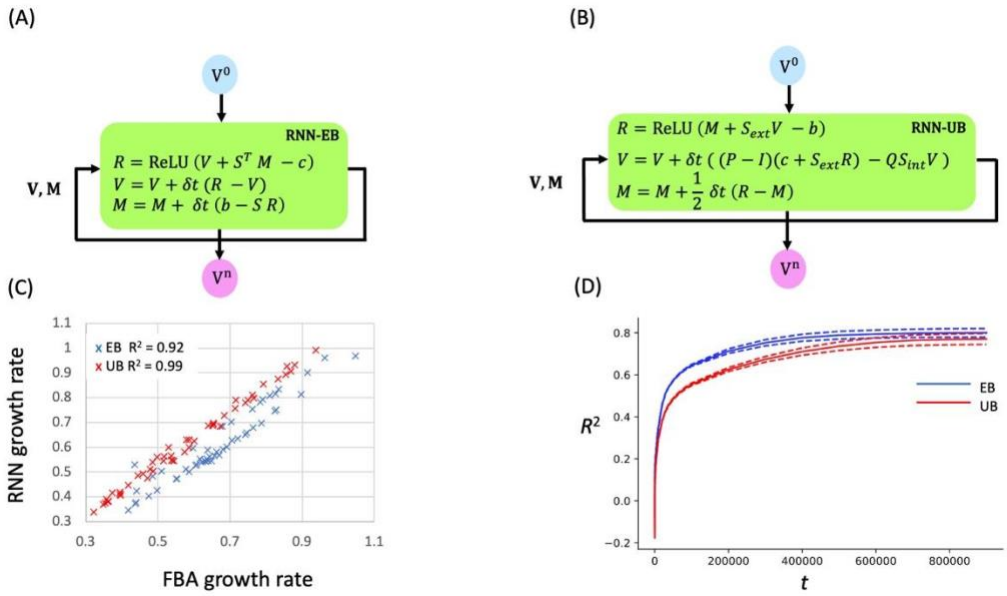


*Fig. 2. RNN architectures and performances. (A) RNN-EB. The constrained linear optimization problem (eqs. 1 and 2 in Method section) was solved with a gradient based method that iteratively update flux values and that is proven to converge to the same solution as (1). The solution is called "exact bounds" (EB) because the intake fluxes are supposed to be known* [21]*. The notations are those of Fig. 1 for V, $V^0$, S and c, while b = V2M $V^0$. M is a vector representing the variables of the dual problem (also named metabolite shadow prices* [27]*), R is a vector used to simplify the notations, and δt is a constant (representing the increment at the $t^{th}$ iteration). (B) RNN-UB. The general principle of this method is the same as the previous one. However, the equation system (eq. 3 in Method section) that is used allows for inequality boundaries, so it is called "upper bounds" (UB). This method makes used of an architecture developed by Yang et al.* [22] *solving linear and quadratic problems given upper bounds. $S_{ext}$ and $S_{int}$ are the stoichiometric matrices of the network taking respectively external and internal metabolites and reaction fluxes. These matrices are computed from S by removing rows and columns (cf. Fig. S2). $Q = S_{int}^T(S_{int}S_{int}^T)^{-1}$, and $P = QS_{int}$ (cf. Method – Solving LP/QP with RNNs). (C) AMNs predicted growth rate vs. FBA calculated growth rate on the same inputs and the same model (E. coli core) for both RNN-EB and RNN-UB. (D) Regression coefficient $R^2$ improvement for all reaction fluxes of E. coli core model with the number of iterations (t).*

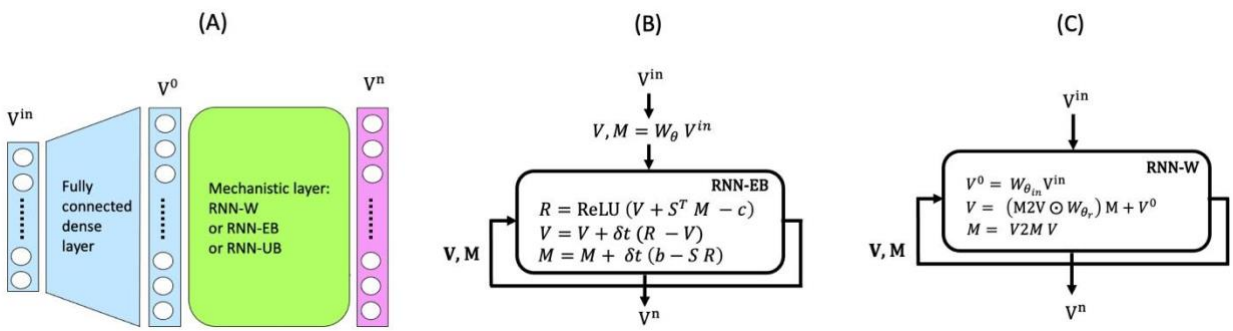


*Fig. 3. AMN architectures enabling training. (A) Basic architecture principle. $V^{in}$ is the vector representing upper or exact bounds on exchange reactions, i.e., the medium uptake rates. Each white circle represents one reaction. $V^0$ contains the initial fluxes predicted by a dense, fully connected layer. $V^n$ is the final flux vector, according to the mechanistic constraints applied on $V^0$ and resulting from S: mass and charge conservation imposed by SV=0. The architectures RNN-W, RNN-EB and RNN-UB of Figs. 1 and 2 can be used for the mechanistic layer. (B) Architecture example for RNN-EB. Here the weight matrix $W_\theta$ is obtained via training to compute initial values for V and M, then the RNN-EB cell is run iteratively to return $V^n$. (C) Architecture example for RNN-W. Here the fully connected dense layer is plugged inside the recurrent cell containing the mechanistic layer. At each iteration $V^0$ is computed via $W_{\theta_{in}}$ obtained through training. V and M are updated via $W_{\theta_r}$ (also obtained via training) and the values of V and M at the previous iteration. M is calculated using the M2V matrix defined in Fig. 1.*

***Table 1. AMN performance**. All SBML models from different strains were downloaded from the BiGG database [25]. AMN architectures and parameters section. (1) Training set size and ratio of variable compounds in medium turned on, method used when running COBRApy* [26] *(with pFBA) (2) All: all fluxes are calculated, Biomass: only the growth rate is calculated. (3) Number of trainable parameters and epochs, in all cases dropout = 0.25, batch size = 5, the optimizer is Adam and the loss function is the mean squared error between predicted and provided fluxes to which the value IISVII² is added (which should be near zero) (4) Measured on 2.4 GHz 8-Core Intel Core i9. (5) Regression coef. and IISVII² values for X-fold = 5-fold on training and validation sets (6) Regression coef. and IISVII² values on an independent test set of size 1000.*

| SBML model | Size<br>Ratio<br>Method(1) | Measure(2)<br>Size | NN type<br>Architecture<br>Timestep | Nbr<br>Param(3)<br>Nbr epoch | Time<br>per epoch<br>(s) (4) | $R^2$<br>X-fold(5) | $Q^2$<br>IISVII²<br>X-fold(5) | $Q^2$<br>IISVII²<br>test(6) |
|---|---|---|---|---|---|---|---|---|
| Ecoli_core | 10000<br>0.5<br>pFBA | All<br>154 | ANN<br>Dense<br>n/a | 3.61k<br><br>1000 | 1.0 | 0.92 ± 0.00 | 0.92 ± 0.00<br><br>0.22 ± 0.06 | 0.94<br><br>5.54 |
| Ecoli_core | 10000<br>0.5<br>pFBA | Biomass<br>1 | ANN<br>Dense<br>n/a | 3.61k<br><br>500 | 1.0 | 0.96 ± 0.02 | 0.95 ± 0.02<br><br>n/a | 0.96<br><br>n/a |
| Ecoli_core | 10000<br>0.5<br>pFBA | All<br>154 | ANN<br>SV<br>n/a | 3.61k<br><br>1000 | 2.0 | 0.92 ± 0.01 | 0.92 ± 0.01<br><br>0.16 ± 0.03 | 0.95<br><br>2.52 |
| Ecoli_core | 10000<br>0.5<br>pFBA | Biomass<br>1 | ANN<br>SV<br>n/a | 3.61k<br><br>500 | 2.0 | 0.98 ± 0.01 | 0.98 ± 0.01<br><br>0.04 ± 0.01 | 0.99<br><br>0.85 |
| Ecoli_core | 10000<br>0.5<br>pFBA | All<br>154 | AMN<br>RNN-EB<br>1 | 4.74k<br><br>1000 | 6.3 | 0.82 ± 0.12 | 0.83 ± 0.11<br><br>1.64 ± 0.48 | 0.94<br><br>0.54 |
| Ecoli_core | 10000<br>0.5<br>pFBA | Biomass<br>1 | AMN<br>RNN-EB<br>5 | 4.74k<br><br>500 | 6.3 | 1.00 ± 0.00 | 1.00 ± 0.00<br><br>0.03 ± 0.02 | 0.99<br><br>0.45 |
| Ecoli_core | 10000<br>0.5<br>pFBA | Biomass<br>1 | AMN<br>RNN-W<br>5 | 14.32k<br><br>500 | 11.0 | 0.98 ± 0.01 | 0.98 ± 0.01<br><br>0.02 ± 0.01 | 0.70<br><br>0.48 |
| iML1515 | 10000<br>0.5<br>pFBA | Biomass<br>1 | AMN<br>RNN-EB<br>1 | 289.43k<br><br>500 | 60.0 | 1.00 ± 0.00 | 1.00 ± 0.00<br><br>0.06 ± 0.02 | 1.00<br><br>0.04 |

Once an AMN has been trained on a large dataset of pFBA simulations, we can exploit it in subsequent, further learning, this time on experimental data. Precisely, we used the iML1515 AMN of Table 1 for characterizing and predicting the growth of *E. coli* on many different media compositions. To that end, we grew *E. coli* DH5-alpha in 73 different media compositions, with M9-glycerol as a basis and a wide variety of possibly added nutrients (see Methods - Culture conditions). As already mentioned in the introduction section, the intake rates of *E. coli* nutrients (*i.e.*, the nutrient exchange reactions values),

as well as their relation to external concentrations, remain largely unknown: the quantitative rate for each compound may vary between growth media, unlike in the FBA calculations, where the same upper bound (or zero, if a compound is absent) is used in all cases. This is a common flaw of classical FBA (one that is only partially remedied in satFBA or FBA with molecular crowding [28]). To use the architectures of Fig. 3, we thus need to first convert nutrient concentrations into uptake rates. This is achieved by adding a dense layer that precedes the AMN (Fig 4A). In this architecture the AMN is no longer trainable, only the prior dense layer is, and the AMN acts as a reservoir as in reservoir computing [29].

The predictive power of our AMN-reservoir was assessed by a regression coefficient $R^2 = 0.96 \pm 0.02$ on training sets; and $R^2 = 0.78$ (Fig. 4B) on a test set build from aggregated validation sets during 10-fold cross-validation. We obtained similar performance with a state-of-the-art black-box model (XGBoost [30]) and a simple multivariate regression model, with the Ordinary Least Squares (OLS) method of the statsmodels [31] python package. However, those black-box models have no mechanistic insights to propose, and no possible modifications that have a biological sense, like a knockout of a gene that would delete a reaction in the metabolic network, which can be accounted for in our AMNs. To compare with an 'out-of-the-box' CBM approach, we first used an arbitrary value of 10 for the present compounds, and 0 for the absent compounds. Applying these values on the corresponding exchange reactions upper bounds, and optimizing the biomass reaction rate produced results uncorrelated with experimental measurements (Fig. 4C). To enhance these results, we optimized the upper bound values of exchange reactions corresponding to absent or present compounds in the medium (see Methods - Optimization of exchange reactions upper bounds in FBA). It resulted in a $R^2 = 0.47$ performance (Fig. 4D), showing a poor fit and weak predictive power (no cross-validation scheme here), indicating AMNs have a purpose in augmenting the capabilities of CBM without costly experimental work.

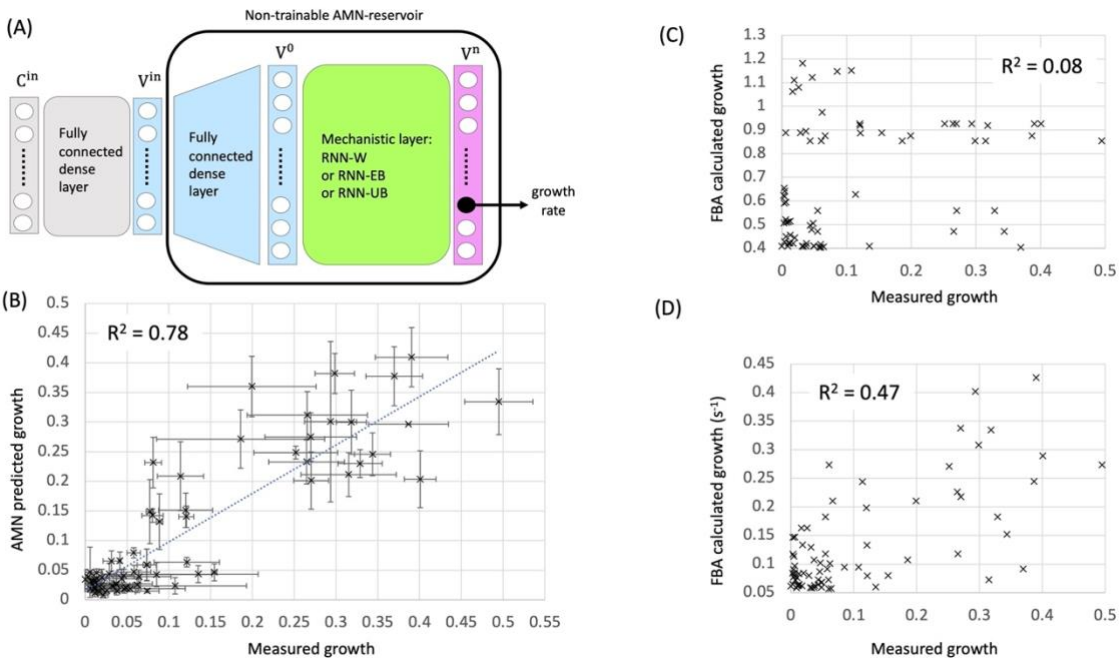


*Fig. 4. Benchmarking growth rate predictions with experimental measurements for E. coli (strain DH5-alpha, model iML1515). **(A)** AMN used in reservoir computing. The non-trainable reservoir (rounded box) is the AMN shown in Figure 3.B (with performances given the last row of Table 1) and is connected to a prior trainable network which purpose is to compute medium intake fluxes ($V^{in}$) from the concentration ($C^{in}$) of metabolites added to medium. Taking as input $V^{in}$, the non-trainable reservoir prints out all fluxes including the growth rate. **(B)** AMN (RNN-EB architecture) predicted growth rate vs. measured growth rate. All points plotted correspond to predicted values not present in the training set. This was compiled using all predicted values obtained for each validation sets in 10-fold stratified cross validation repeated 3 times with different initial random seeds. **(C)** COBRApy calculated growth rate vs. measured growth rate. COBRApy was run taking as input upper bound intake fluxes for added metabolites in medium. Intake flux values were set to arbitrary values (0 when the metabolite was absent in the medium and 10 when it was present). **(D)** COBRApy calculated growth rate vs. measured growth rate with optimized medium intake fluxes (see section Method - Optimization of exchange reaction - for details on optimization procedure).*

# Discussion and conclusion

In this study, we showed that metabolic networks can be used as a learning architecture for neural network approaches. Previous work on RNN for constrained optimization was re-used and adapted to enable machine learning methods (such as backpropagation of errors) within metabolic networks. For improved scalability and adaptability, we first trained an AMN-reservoir on large, simulated pFBA datasets. We then exploited the reservoir to improve the predictive power of MFA on growth rate datasets of *E. coli.* Figure 4 shows good predictive power of AMN, far outperforming those obtained by standard CBM solvers (like COBRApy pFBA solver in Fig. 2C and 2D).

Determining the uptake rates is a core experimental work needed for making constraint-based MFA realistic. Here, we developed an approach that gets rid of such needs for reaching plausible fluxes distributions. We did so by backpropagating the error on the growth rate, to find complex relationships between the medium concentrations and the medium uptake rates. To that end, we demonstrated the

high predictive power of AMNs, and their re-usability in classical CBM approaches. Indeed, the constraint-based methods developers and users can now make use of our AMN-reservoir method for solving the medium uptake rate issues. In this regard, supplementary Table S2 gives uptake rates for the metabolites used in our benchmarking work with *E. coli* (Fig. 4), which can directly be used with CBM software products like COBRApy.

As mentioned earlier, in classical CBM, plausible flux distributions can be found by constraining some fluxes to measured flux values. Here, we avoided costly experimental flux determinations and found plausible flux distributions just by learning the consensual metabolic behavior of an organism in response to its environment. We also learned the relationship between medium nutrient concentrations and the actual nutrient uptake by the bacteria, eventually revealing complex regulations between *E. coli'*s environment to its steady-state metabolic phenotype. Added to the possible unveiling of biological mechanisms, AMNs can also be exploited for industrial applications. Indeed, since one can design any objective function to optimize with AMNs, they can be used to search optimum media for the bioproduction of compounds of interest as well as microorganism-based decision-making devices for the multiplexed detection of metabolic biomarker or environmental pollutants.

# Methods

### Making metabolic networks suitable for neural computations

AMNs should have a property that is not guaranteed by usual metabolic models: all reactions must be unidirectional. In other words, AMN computations have been developed for models showing positive-only fluxes. We wrote a standardization script that loads an SBML model into COBRApy and screens for all two-sided reactions, then duplicating them into two separate reactions; and writes a new version of the model with bi-directional reactions split into separate forward and backward reactions. To avoid confusion, we add a suffix for these reactions, either “for” or “rev” respectively designating the original forward reaction and the reversed reaction. For more clarity, the exchange reactions were also duplicated, even if encoded as one-sided, and their suffix was set to “i” for inflow reactions, and “o” for outflow reactions.

**Generation of training sets with COBRApy**

Training sets of metabolic flux distributions were generated using models downloaded from the BiGG database [25]. The models were used to generate data using COBRApy [26] following a precise set of rules. First, we identified essential exchange reactions for the models we used (*E. coli* core and iML1515). Precisely, if one of these reactions has its upper bound set to zero, the biomass reaction optimization is impossible, even if all other exchange reactions are set to a high value, e.g., 1000. In other words, we identified the minimal uptake fluxes that enable growth, according to the models. During data generation, the upper bounds on these reactions were always set to 1000.

Two carbon sources always had their exchange reaction turned on, glycerol and arabinose, which were given an upper bound of 10. Then, we selected a set of 27 exchange reactions to be the variables of our training set: the 20 canonical proteinogenic amino acids (L-histidine, L-leucine, L-phenylalanine, L-lysine, L-tyrosine, L-valine, L-leucine, L-methionine, L-isoleucine, L-glutamine, L-cysteine, L-serine, L-alanine, L-tryptophan, L-asparagine, L-proline, L-threonine, Glycine, L-arginine, L-aspartate), 4 sugars (Melibiose, Sucrose, Trehalose, Lactose) and 3 acids (Acetate, Pyruvate, Citrate). Note that these exchange reactions corresponding compounds are used as the possibly added compounds in the experimental datasets.

These constitute the so-called set of variable exchange reactions (VER). On these reactions, we first applied a binomial drawing rule B($n$, $p$) with $n$=27 and $p \in [0, 1]$, a tunable parameter related to the ratio of 'activated' VER (to have an upper bound larger than zero). Consequently, the mean number of activated variable exchange reactions was close to $n$ x $p$. For each of those activated VER, one out of several levels were randomly drawn from a uniform law, to give the upper bound actual value. In the workflow, the number of levels and the maximum value are tunable parameters. To retrieve the FBA-solved fluxes, we used COBRApy optimization on the biomass reaction without any constraints on the network reactions. In the end, each point (x, y) in the dataset consisted of a vectors of values for all the VER (X), and all the fluxes obtained running FBA or pFBA with COBRApy (Y).

**Solving LP/QP with RNN methods**

When intake fluxes are known (EB method), the linear optimization problem solved with FBA for a metabolic network with $n$ fluxes and $m$ metabolites can be written as:

$$\begin{aligned} Max: c^T V \\ st: SV = b \\ V \geq 0 \end{aligned} \tag{1}$$

and its dual form:

$$\begin{aligned} Min: b^T M \\ st: S^T M \leq c^T \end{aligned} \tag{2}$$

In the case where intake fluxes are unknown (UB and general case for FBA) it can be written as:

$$\begin{aligned} Max: c^T V \\ st:\ S_{ext} V \leq b \ and\ S_{int} V = 0 \\ V \geq 0 \end{aligned} \tag{3}$$

with *V* the vector of fluxes, *M* its dual referred to as shadow prices of metabolites, *S* the stoichiometric matrix, *b* a vector of dimension *m* with $b_i$ corresponding to input flux of the *ith* metabolite and *c* the objective vector of dimension *n*. Keep in mind that b takes positive values only for metabolites transported by exchange reactions, consequently $b_i = 0$ for all internal metabolites. Using the notation of Fig. 1 we note that $b = M2V\ V^0$, where $V^0$ is the vector of intake fluxes. In this work *c* is a null vector with 1 for the 'biomass reaction' flux.

The usual method to solve this linear programming problem is the Simplex algorithm [32], which is fast and scalable. But, Simplex has no gradient, nor backpropagation compatibility. Thus, it couldn't be integrated in AMNs, which are designed to be hybrid models.

Another way to solve this problem is through recurrent neural networks (RNN). As this method relies on gradient descent and is compatible with backpropagation. The two methods that were used are illustrated in Fig. 2A and 2B. The general solving happens by iteratively updating *V* the vector of all reaction fluxes and *M* the dual vector of shadow prices:

$$\begin{aligned} V_{t+1} = V_t + \delta t dV \\ M_{t+1} = M_t + \delta t dM \end{aligned} \tag{4}$$

With $dV$ and $dM$ depend on the architecture (RNN-EB or RNN-UB).

For RNN-EB, the constraints are simplified to $SV = b$ and the solution used is given in Figure 2A. For the UB method, the stoichiometric matrix *S* is split into $S_{int}$ the part of *S* for internal fluxes and $S_{ext}$ for exchange reaction fluxes. In addition, terms to ensure the positivity of the fluxes were added to $S_{ext}$ (cf.

Fig S2). The resulting equations are given in Fig. 2B. As the matrix $S_{int}\,S_{int}^{T}$ should be invertible, we guarantee this property by giving its initial form the row echelon form. As a result, *M* in the UB method is of dimension $p<m$.

Both methods were proven to converge to the global optimum independently of the initialization. The time step was arbitrarily fixed to $3.0\ 10^{-4}$ for EB and $5.0^{-2}$ for UB and it was multiplied by 0.7 every $10^4$ iterations if the objective value was not decreasing.

For both Fig. 2C and Fig. 2D the results were obtained with 50 simulations and $10^7$ iterations. The reference values were obtained using COBRApy and *E. coli* core model. Each replicate was generated with a different medium computed with the training set generation method (cf. previous section).

In the first part of this work (*i.e.*, Fig. 2), RNNs are used with the maximization of growth rate as an objective function. Whereas, in the second part (*i.e.,* Figs. 3-4, Table 1) RNNs are embedded within larger architectures for training purposes, the objective function is no longer needed and $c=0$.

**AMN architectures and parameters**

We propose three AMNs architectures as described in Fig. 3A (*i.e.*, composed of a dense layer and a mechanistic layer): RNN-EB, RNN-UB and RNN-W. RNN-EB and RNN-UB have a mechanistic layer as described in Fig. 2A and 2B. The architecture of RNN-W is shown in Fig. 3C. In the architecture 'ANN dense' and 'ANN SV' we omit the mechanistic layer and simply use the output of the first dense layer as the final output of the model. For all architecture, we use the Mean Squared Error (MSE) on all fluxes as the objective function to minimize while learning. In the architecture 'ANN SV' and in all RNN architectures we added to the MSE loss function, the term $\|S V\|^2$ to enforce the metabolic network constraint, *i.e.,* the mass balance of internal metabolites.

To summarize the parameters used when training ANN and AMN can be decomposed into different categories:

1. Simulated data parameters. As described in the previous section (Generation of training sets with COBRApy), we can tune the size of the training set to be generated. We can also modify the mean number of activated VER per data point, the maximum value each VER can take, and the number of levels *(i.e.*, the resolution) between 0 and a maximum. We can also modify the actual VER list, but this modifies the architecture of the model (initial layer size), so we kept the same list in the present work.

2. Learning parameters. During learning on simulated data, an AMN has a small set of parameters to tune: the loss function, the number of epochs, the batch size, and the X-fold validation, where X is the number of folds.
3. Architecture parameters. One can also select the architecture of the model, amongst the 5 benchmarked architectures: 'RNN-W', 'RNN-EB', 'RNN-UB', 'ANN Dense' and 'ANN SV'.

**Optimization of exchange reactions upper bounds in FBA**

The goal of this optimization was to find the best VER fluxes to match experimentally determined growth rates, by using 'out-of-the-box' FBA, simply informing the presence or absence of the flux according to the experimental medium composition. As a result, the optimized fluxes values were assigned (as an upper bound) to the corresponding exchange reaction only when the experimental data point has the compound present. The Mean Squared Error (MSE) between all the FBA predictions and all the experimentally determined growth rates was the function to minimize. For that, the input vector to optimize was composed of the upper bounds for all VER, but also arabinose and glycerol (which are present in all media compositions). The Broyden, Fletcher, Goldfarb, and Shanno (BFGS) algorithm [33] was used under the SciPy library, as a local optimization method, starting from an initial vector filled of the value 5. Final optimized uptake values are shown in supplementary Table S1.

**Culture conditions and growth rate determination**

The basic medium for culturing E. coli *DH5-α* was a M9-glycerol medium prepared with those final concentrations: 100µM CaCl2, 2mM MgSO4, 1X M9 salts (3 g.L-1 KH2PO4, 8.5 g.L-1 Na2HPO4 2H2O, 0.5 g.L-1 NaCl, 1g.L-1 NH4Cl), 1X trace elements (15 mg.L-1 Na2EDTA 2H20, 4.5 mg.L-1 ZnSO4 7H2O, 0.3 mg.L-1 CoCl2 6H2O, 1 mg.L-1 MnCl2 4H2O, 1 mg.L-1 H3BO3, 0.4mg.L-1 Na2MoO4 2H20, 3 mg.L-1 FeSO4 7H2O, 0.3 mg.L-1 CuSO4 5H2O; solution adjusted at pH=4 and stored at 4°C), 1 mg.L-1 Thiamine-HCl and 0.5g.L-1 glycerol. The additional compounds that could be added (the VER, See Methods - Generation of training sets with COBRApy) were all set to a final concentration of 0.1 g.L-1. The pH was adjusted at 7.4 prior to a 0.22µm filter sterilization of the medium. Pre-cultures were recovered from glycerol -80°C stocks, grew in Luria-Bertani (LB) broth overnight, then used as inoculate in 200µL M9-glycerol (supplemented with variable compounds) in 96 U-bottom wells plates. The temperature was set to 37°C in a plate reader (BioTek HTX Synergy), with continuous orbital shaking, allowing aerobic growth for 24 hours. A monitoring every 10 minutes of the optical density at 600 nm was performed and the steepest part of the growth curve was retrieved for growth rate determination, by a simple logistic regression.

**Acknowledgements**

JLF would like to acknowledge funding provided by the ANR funding agency grant numbers ANR-18-CE44-0015 (SynBioDiag project) and ANR-21-CE45-0021-01 (AMN project). LF is supported by INRAE's MICA department and the by INRAE's metaprogram BIOLPREDICT. BM is supported by an Ecole Normale Supérieure Scholarship. JLF thanks Aymeric Gaudin (CentraleSupélec Engineering School) for early development in reservoir computing with AMN, and Ivan Radkevich (University of Paris Saclay Systems & Synthetic Biology master program student) for his work on custom RNN cells loading and saving.

**Author contributions and codes information**

LF and JLF wrote the core of the text of the manuscript. JLF designed the study and wrote all the AMN codes used to produce Figs. 1, 3, 4, S1 and Tables 1 and S2. BM wrote the RNN codes producing Figs. 2 and S2 and wrote the corresponding part in the method section. LF wrote the code transforming SBML models into unidirectional networks and the code to produce Table S1. LF also performed all experimental work reported in Fig. 4 and wrote the corresponding experimental method section. WL contributed to designing the project and was involved in the discussions. All authors read, edited, and approved the manuscript. All AMN codes make use of COBRApy, numpy, Pandas, Tensorflow, Sklearn and Keras libraries. All codes are available within Google Colab notebooks and will be released upon journal publications.

**References**

[1] D. M. Camacho, K. M. Collins, R. K. Powers, J. C. Costello, and J. J. Collins, "Next-Generation Machine Learning for Biological Networks," *Cell*, vol. 173, no. 7, Art. no. 7, Jun. 2018, doi: 10.1016/j.cell.2018.05.015.

[2] P. Carbonell, T. Radivojevic, and H. García Martín, "Opportunities at the Intersection of Synthetic Biology, Machine Learning, and Automation," *ACS Synth. Biol.*, vol. 8, no. 7, pp. 1474–1477, Jul. 2019, doi: 10.1021/acssynbio.8b00540.

[3] J.-L. Faulon and L. Faure, "In silico, in vitro, and in vivo machine learning in synthetic biology and metabolic engineering," *Current Opinion in Chemical Biology*, vol. 65, pp. 85–92, Dec. 2021, doi: 10.1016/j.cbpa.2021.06.002.

[4] S. G. Wu, K. Shimizu, J. K.-H. Tang, and Y. J. Tang, "Facilitate Collaborations among Synthetic Biology, Metabolic Engineering and Machine Learning," *ChemBioEng Reviews*, vol. 3, no. 2, Art. no. 2, 2016, doi: 10.1002/cben.201500024.

[5] J. L. Reed and B. Ø. Palsson, “Thirteen Years of Building Constraint-Based In Silico Models of Escherichia coli,” *Journal of Bacteriology*, vol. 185, no. 9, Art. no. 9, May 2003, doi: 10.1128/JB.185.9.2692-2699.2003.
[6] S. Niedenführ, W. Wiechert, and K. Nöh, “How to measure metabolic fluxes: a taxonomic guide for (13)C fluxomics,” *Curr Opin Biotechnol*, vol. 34, pp. 82–90, Aug. 2015, doi: 10.1016/j.copbio.2014.12.003.
[7] A. M. Willemsen *et al.*, “MetDFBA: incorporating time-resolved metabolomics measurements into dynamic flux balance analysis,” *Mol Biosyst*, vol. 11, no. 1, Art. no. 1, Jan. 2015, doi: 10.1039/c4mb00510d.
[8] N. Alghamdi *et al.*, “A graph neural network model to estimate cell-wise metabolic flux using single-cell RNA-seq data,” *Genome Res.*, p. gr.271205.120, Jul. 2021, doi: 10.1101/gr.271205.120.
[9] M. Kim, N. Rai, V. Zorraquino, and I. Tagkopoulos, “Multi-omics integration accurately predicts cellular state in unexplored conditions for Escherichia coli,” *Nat Commun*, vol. 7, p. 13090, Oct. 2016, doi: 10.1038/ncomms13090.
[10] J. E. Lewis and M. L. Kemp, “Integration of machine learning and genome-scale metabolic modeling identifies multi-omics biomarkers for radiation resistance,” *Nat Commun*, vol. 12, no. 1, p. 2700, May 2021, doi: 10.1038/s41467-021-22989-1.
[11] G. Zampieri, S. Vijayakumar, E. Yaneske, and C. Angione, “Machine and deep learning meet genome-scale metabolic modeling,” *PLoS Comput Biol*, vol. 15, no. 7, Art. no. 7, Jul. 2019, doi: 10.1371/journal.pcbi.1007084.
[12] A. Sahu, M.-A. Blätke, J. J. Szymański, and N. Töpfer, “Advances in flux balance analysis by integrating machine learning and mechanism-based models,” *Comput Struct Biotechnol J*, vol. 19, pp. 4626–4640, 2021, doi: 10.1016/j.csbj.2021.08.004.
[13] N. Baker *et al.*, “Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence,” USDOE Office of Science (SC), Washington, D.C. (United States), Feb. 2019. doi: 10.2172/1478744.
[14] A. Nilsson, J. M. Peters, B. Bryson, and D. A. Lauffenburger, “Artificial neural networks enable genome-scale simulations of intracellular signaling,” Sep. 2021. doi: 10.1101/2021.09.24.461703.
[15] R. Anantharaman *et al.*, “Stably Accelerating Stiff Quantitative Systems Pharmacology Models: Continuous-Time Echo State Networks as Implicit Machine Learning,” Oct. 2021. doi: 10.1101/2021.10.10.463808.
[16] J. H. Lagergren, J. T. Nardini, R. E. Baker, M. J. Simpson, and K. B. Flores, “Biologically-informed neural networks guide mechanistic modeling from sparse experimental data,” *PLOS Computational Biology*, vol. 16, no. 12, Art. no. 12, Dec. 2020, doi: 10.1371/journal.pcbi.1008462.
[17] C. Culley, S. Vijayakumar, G. Zampieri, and C. Angione, “A mechanism-aware and multiomic machine-learning pipeline characterizes yeast cell growth,” *PNAS*, vol. 117, no. 31, Art. no. 31, Aug. 2020, doi: 10.1073/pnas.2002959117.
[18] L. Jin, S. Li, B. Hu, and M. Liu, “A survey on projection neural networks and their applications,” *Applied Soft Computing*, vol. 76, pp. 533–544, Mar. 2019, doi: 10.1016/j.asoc.2019.01.002.
[19] J. J. Hopfield and D. W. Tank, “‘Neural’ computation of decisions in optimization problems,” *Biol. Cybern.*, vol. 52, no. 3, Art. no. 3, Jul. 1985, doi: 10.1007/BF00339943.
[20] J. Wang and V. Chankong, “Recurrent neural networks for linear programming: Analysis and design principles,” *Computers & Operations Research*, vol. 19, no. 3, pp. 297–311, Apr. 1992, doi: 10.1016/0305-0548(92)90051-6.
[21] H. Ghasabi-Oskoei and N. Mahdavi-Amiri, “An efficient simplified neural network for solving linear and quadratic programming problems,” *Applied Mathematics and Computation*, vol. 175, no. 1, pp. 452–464, Apr. 2006, doi: 10.1016/j.amc.2005.07.025.

[22] Y. Yang, J. Cao, X. Xu, M. Hu, and Y. Gao, “A new neural network for solving quadratic programming problems with equality and inequality constraints,” *Mathematics and Computers in Simulation*, vol. 101, pp. 103–112, Jul. 2014, doi: 10.1016/j.matcom.2014.02.006.
[23] S. Müller, G. Regensburger, and R. Steuer. Resource allocation in metabolic networks: kinetic optimization and approximations by FBA. Biochemical Society Transactions, 43:1195–1200, 2015.
[24] S. Schuster, T. Pfeiffer and D. Fell (2008), Is maximization of molar yield in metabolic networks favoured by evolution? J. Theor. Biol. 252, 497-504
[25] *http://bigg.ucsd.edu/*. http://bigg.ucsd.edu/
[26] A. Ebrahim, J. A. Lerman, B. O. Palsson, and D. R. Hyduke, “COBRApy: COnstraints-Based Reconstruction and Analysis for Python,” *BMC Syst Biol*, vol. 7, p. 74, Aug. 2013, doi: 10.1186/1752-0509-7-74.
[27] A. Varma and B. O. Palsson, “Metabolic capabilities of Escherichia coli: I. synthesis of biosynthetic precursors and cofactors,” *J Theor Biol*, vol. 165, no. 4, pp. 477–502, Dec. 1993, doi: 10.1006/jtbi.1993.1202.
[28] Beg, Q. K., Vazquez, A., Ernst, J., de Menezes, M. A., Bar-Joseph, Z., Barabási, A. L., & Oltvai, Z. N. (2007). Intracellular crowding defines the mode and sequence of substrate uptake by Escherichia coli and constrains its metabolic activity. *Proceedings of the National Academy of Sciences*, *104*(31), 12663-12668.
[29] G. Tanaka *et al.*, “Recent advances in physical reservoir computing: A review,” *Neural Networks*, vol. 115, pp. 100–123, Jul. 2019, doi: 10.1016/j.neunet.2019.03.005.
[30] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
[31] S. Seabold and J. Perktold, “Statsmodels: Econometric and Statistical Modeling with Python,” Austin, Texas, 2010, pp. 92–96. doi: 10.25080/Majora-92bf1922-011.
[32] H. Karloff, “The Simplex Algorithm,” in *Linear Programming*, H. Karloff, Ed. Boston, MA: Birkhäuser, 1991, pp. 23–47. doi: 10.1007/978-0-8176-4844-2_2.
[33] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical Programming*, vol. 45, no. 1, Art. no. 1, Aug. 1989, doi: 10.1007/BF01589116.

# Supplementary Materials

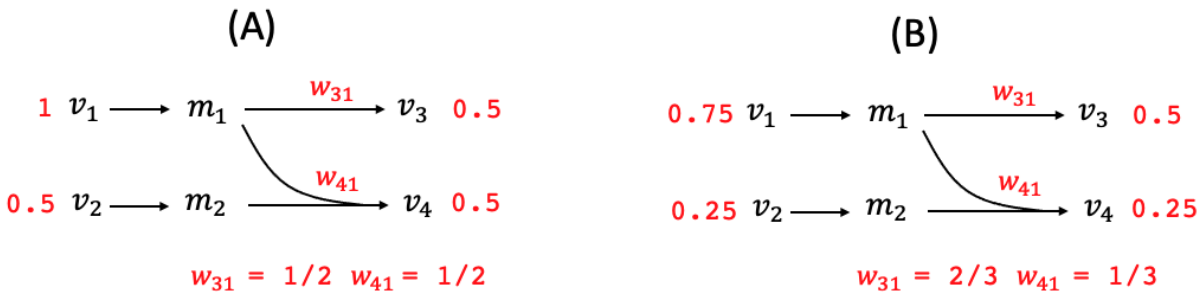


***Fig. S1. Different weights for different intake fluxes****. In the two cases all flux values ($v_i$) satisfy the steady state constraints (SV = 0, cf. Fig. 1.B). Following the equations provided in Fig. 1.C, the production fluxes for $m_1$ and $m_2$ are respectively 1 and 0.5 in (A) and 0.75 and 0.25 in (B). The reaction for flux $v_4$ is taking two substrates $m_1$ and $m_2$ and the value for $v_4$ is the minimum metabolite production flux (i.e., the rate limiting metabolite among $m_1$ and $m_2$). Consequently, the value for $v_4$ is 0.5 (A) and 0.25 (B). Therefore, the fraction ($w_{41}$) of $m_1$ contributing to $v_4$ is 1/2 (A) and 1/3 (B). The weights are different in panel (A) and (B) as they depend on the intake flux values.*

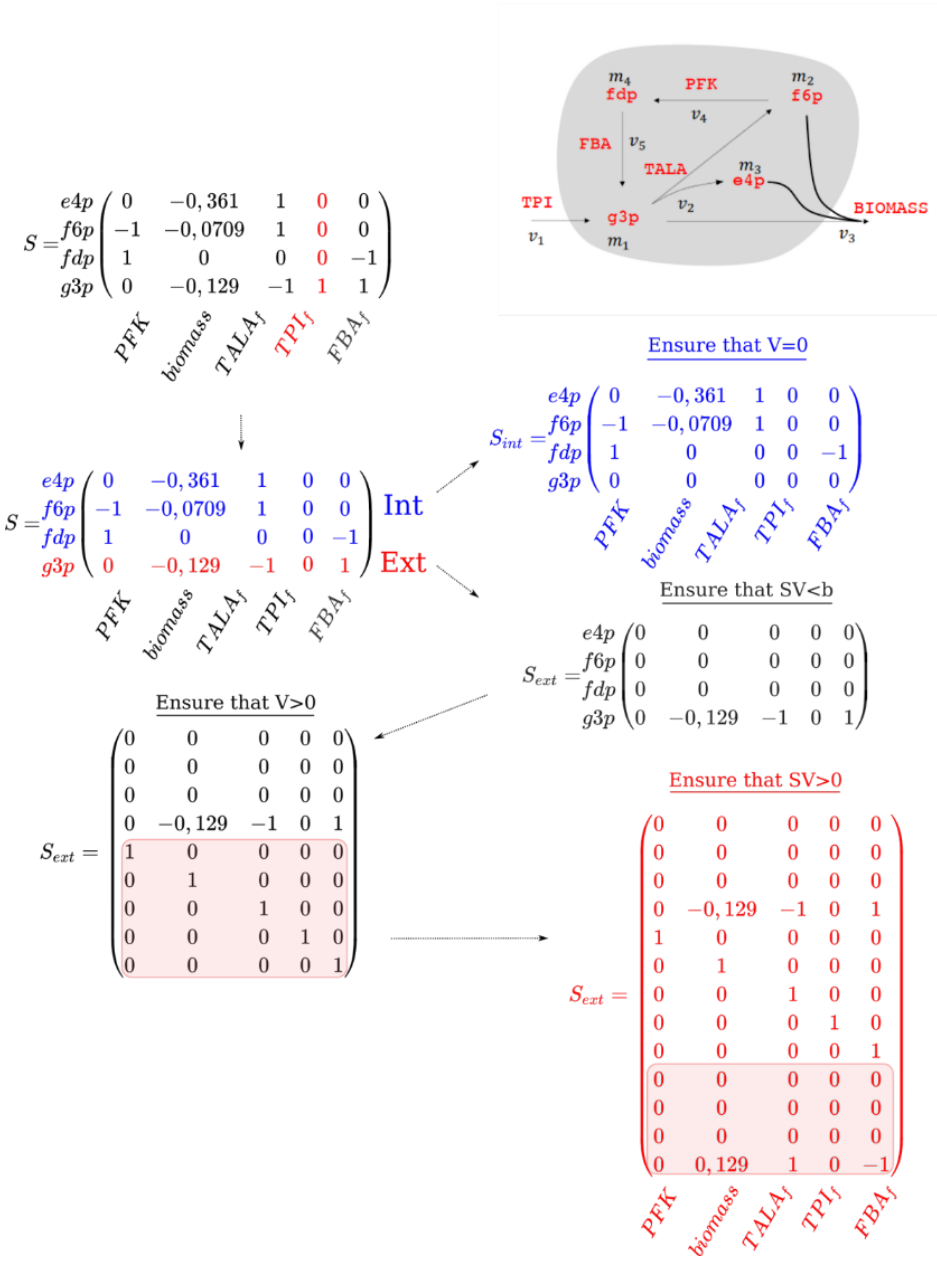


*Fig. S2. Matrices used with the UB method.* *We show here an example for the same model as in Fig. 1. The goal here is to produce two matrices from S, one where we remove the stoichiometric coefficients of the exchange reactions (here the only exchange reaction is TPI, the red column of the top-left matrix); the other where we only keep exchange reactions. Thus, the stoichiometric matrix S was split between reactions involving internal metabolites ($S_{int}$) and exchange reactions, importing or exporting external (medium) metabolites in or out of the cell ($S_{ext}$). In $S_{int}$ external reactions (TPI) and involved metabolites (g3p) are zeroed out. An identity matrix is added to $S_{ext}$ to ensure that V≥0 and $S_{ext}$ is filled with negative value to ensure that intake fluxes are positive.*

*Table S1. Optimized intake fluxes for 'out-of-the-box' FBA. 'VER ID' indicates the column of variable exchange reactions IDs in the unidirectional iML1515 model. 'Name' indicates the actual human-readable name of the compound. The third line is the optimized upper bound value for this compound, in mmol/gDW/h.*

| VER ID | Name | Optimized UB | VER ID | Name | Optimized UB |
|---|---|---|---|---|---|
| EX_his__L_e_i | L-histidine | 1.38644249 | EX_pro__L_e_i | L-proline | 0.19732778 |
| EX_leu__L_e_i | L-leucine | 1.04517755 | EX_thr__L_e_i | L-threonine | 0.18261263 |
| EX_phe__L_e_i | L-phénylalanine | 0.46916847 | EX_gly_e_i | Glycine | 1.18544455 |
| EX_lys__L_e_i | L-lysine | 0.91378495 | EX_arg__L_e_i | L-arginine | 0.34324712 |
| EX_tyr__L_e_i | L-tyrosine | 1.32356027 | EX_glu__L_e_i | L-glutamate | 0.28930693 |
| EX_val__L_e_i | L-valine | 0.82434214 | EX_melib_e_i | Melibiose | 1.20124412 |
| EX_asp__L_e_i | L-aspartate | 0.16357178 | EX_sucr_e_i | Sucrose | 0.48828341 |
| EX_met__L_e_i | L-methionine | 0.46414337 | EX_tre_e_i | Trehalose | 0.84498494 |
| EX_ile__L_e_i | L-isoleucine | 0.4522181 | EX_lcts_e_i | Lactose | 0.08696407 |
| EX_gln__L_e_i | L-glutamine | 0.11281941 | EX_ac_e_i | Acetate | 1.43337434 |
| EX_cys__L_e_i | L-cystéine | 0.32868508 | EX_pyr_e_i | Pyruvate | 1.74551093 |
| EX_ser__L_e_i | L-serine | 0.84441828 | EX_cit_e_i | Citrate | 1.88332571 |
| EX_ala__L_e_i | L-alanine | 0.96374981 | EX_glyc_e_i | Glycerol | 0.87195784 |
| EX_trp__L_e_i | L-tryptophan | 0.19073675 | EX_arab__L_e_i | Arabinose | 0.51011534 |
| EX_asn__L_e_i | L-asparagine | 0.3260942 | | | |

***Table S2. Intake fluxes predicted by the architecture of Fig. 4****. The Table provides in the two first columns the measured (TRUE) and COBRApy calculated growth rate (PRED) for 38 different media for which measured growth rate ≥ 0.05. The other columns provide intake fluxes for medium metabolites. The intake flux values have been calculated by the grey dense layer of Figure 4A. The intake flux values can differ quite a lot for one medium to another, as intake fluxes of metabolites are not independent from one another. Intake flux values are in mmol/gDW/h. Values are provided for minimum ($m_i$) and variable ($v_i$) medium metabolites. The compound name / ID are given below the Table. The Table below is also provided as a supplementary .csv file.*

| TRUE | PRED | m1 | m2 | m3 | m5 | m6 | m7 | m8 | m9 | m10 | m11 | m13 | m15 | m16 | m17 | m18 | m20 | m21 | m24 | v1 | v2 | v11 | v16 | v21 | v23 | v24 | v26 | v27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9E-02 | 9E-02 | 8E-02 | 9E-01 | 7E-04 | 6E-05 | 7E-04 | 3E-05 | 8E-04 | 5E-04 | 3E-05 | 6E-05 | 2E-06 | 6E-07 | 2E-02 | 9E-01 | 2E-02 | 5E-04 | 2E+00 | 2E+00 | | | | | | | 1E+00 | | |
| 1E-01 | 1E-01 | 1E-01 | 7E-01 | 9E-04 | 8E-05 | 9E-04 | 4E-05 | 1E-03 | 6E-04 | 4E-05 | 8E-05 | 3E-06 | 8E-07 | 3E-02 | 1E+00 | 2E-02 | 6E-04 | 3E+00 | 3E+00 | | | | | | | 6E-01 | | |
| 6E-02 | 6E-02 | 6E-02 | 1E+00 | 5E-04 | 4E-05 | 5E-04 | 2E-05 | 5E-04 | 3E-04 | 2E-05 | 4E-05 | 2E-06 | 4E-07 | 2E-02 | 7E-01 | 1E-02 | 3E-04 | 2E+00 | 2E+00 | | | | | | | 6E-01 | | |
| 2E-01 | 1E-01 | 1E-01 | | 1E-03 | 1E-04 | 1E-03 | 5E-05 | 1E-03 | 8E-04 | 5E-05 | 1E-04 | 4E-06 | 1E-06 | 4E-02 | 2E+00 | 3E-02 | 8E-04 | 4E+00 | 4E+00 | | | | | | | 4E-01 | | |
| 1E-01 | 1E-01 | 1E-01 | 1E+00 | 1E-03 | 9E-05 | 1E-03 | 4E-05 | 1E-03 | 6E-04 | 4E-05 | 9E-05 | 3E-06 | 9E-07 | 3E-02 | 1E+00 | 2E-02 | 6E-04 | 3E+00 | 3E+00 | | | | | | | 9E-01 | | |
| 6E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 8E-03 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | 3E-03 | | | | | | | | |
| 8E-02 | 8E-02 | 8E-02 | 1E+00 | 6E-04 | 5E-05 | 7E-04 | 3E-05 | 7E-04 | 4E-04 | 3E-05 | 6E-05 | 2E-06 | 6E-07 | 2E-02 | 9E-01 | 2E-02 | 4E-04 | 2E+00 | 2E+00 | | | | | 1E+00 | | | | |
| 8E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 8E-03 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | | | | | | | | | |
| 6E-02 | 6E-02 | 6E-02 | 1E+00 | 5E-04 | 4E-05 | 5E-04 | 2E-05 | 5E-04 | 3E-04 | 2E-05 | 4E-05 | 2E-06 | 4E-07 | 2E-02 | 7E-01 | 1E-02 | 3E-04 | 2E+00 | 2E+00 | | | | | | | 6E-01 | | |
| 9E-02 | 9E-02 | 9E-02 | 9E-01 | 7E-04 | 6E-05 | 7E-04 | 3E-05 | 8E-04 | 5E-04 | 3E-05 | 6E-05 | 2E-06 | 6E-07 | 2E-02 | 1E+00 | 2E-02 | 5E-04 | 2E+00 | 2E+00 | | | | | | 1E+00 | | | |
| 7E-02 | 4E-02 | 4E-02 | | 3E-04 | 3E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 3E-05 | 1E-06 | 3E-07 | 1E-02 | 4E-01 | 8E-03 | 2E-04 | 2E+00 | 2E+00 | | | | | | | | 1E+00 | |
| 8E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 8E-03 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | | | | | | | | | |
| 7E-02 | 6E-02 | 5E-02 | | 4E-04 | 4E-05 | 5E-04 | 2E-05 | 5E-04 | 3E-04 | 2E-05 | 4E-05 | 1E-06 | 4E-07 | 1E-02 | 6E-01 | 1E-02 | 3E-04 | 2E+00 | 2E+00 | | | | | | | | | 2E+00 |
| 7E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 1E-04 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | | | 8E-03 | | | | | | |
| 6E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 8E-03 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | 3E-03 | | | | | | | | |
| 1E-01 | 1E-01 | 1E-01 | | 1E-03 | 9E-05 | 1E-03 | 4E-05 | 1E-03 | 7E-04 | 4E-05 | 9E-05 | 3E-06 | 9E-07 | 3E-02 | 1E+00 | 2E-02 | 7E-04 | 4E+00 | 4E+00 | | 6E-02 | | | | | | | |
| 5E-02 | 3E-02 | 3E-02 | | 3E-04 | 2E-05 | 3E-04 | 1E-05 | 3E-04 | 2E-04 | 1E-05 | 2E-05 | 8E-07 | 2E-07 | 8E-03 | 3E-01 | 6E-03 | 2E-04 | 2E+00 | 2E+00 | | | | 1E-01 | | | | | |
| 5E-01 | 5E-01 | 5E-01 | 3E-01 | 4E-03 | 4E-04 | 4E-03 | 2E-04 | 4E-03 | 3E-03 | 2E-04 | 4E-04 | 1E-05 | 4E-06 | 1E-01 | 5E+00 | 1E-01 | 3E-03 | 1E+01 | 1E+01 | | | | | 2E+00 | | | | |
| 4E-01 | 4E-01 | 4E-01 | 2E-01 | 3E-03 | 3E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 3E-04 | 1E-05 | 3E-06 | 1E-01 | 4E+00 | 8E-02 | 2E-03 | 5E+00 | 5E+00 | | | | | | | 5E+00 | | |
| 4E-01 | 4E-01 | 4E-01 | 3E-01 | 3E-03 | 3E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 3E-04 | 1E-05 | 3E-06 | 1E-01 | 4E+00 | 8E-02 | 2E-03 | 5E+00 | 5E+00 | | | | | | 5E+00 | | | |
| 3E-01 | 3E-01 | 3E-01 | 3E-01 | 2E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 2E-04 | 8E-06 | 2E-06 | 8E-02 | 3E+00 | 6E-02 | 2E-03 | 4E+00 | 4E+00 | | | | | 4E+00 | | | | |
| 3E-01 | 3E-01 | 3E-01 | 3E-01 | 2E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 2E-04 | 8E-06 | 2E-06 | 8E-02 | 3E+00 | 6E-02 | 2E-03 | 4E+00 | 4E+00 | | | | | 4E+00 | | | | |
| 3E-01 | 3E-01 | 3E-01 | 5E-01 | 2E-03 | 2E-04 | 2E-03 | 1E-04 | 2E-03 | 1E-03 | 9E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 6E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | 4E+00 | | | | |
| 3E-01 | 3E-01 | 3E-01 | 3E-01 | 2E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 2E-04 | 8E-06 | 2E-06 | 8E-02 | 3E+00 | 6E-02 | 2E-03 | 4E+00 | 4E+00 | | | | | | | 4E+00 | | |
| 2E-01 | 2E-01 | 2E-01 | 1E+00 | 2E-03 | 1E-04 | 2E-03 | 7E-05 | 2E-03 | 1E-03 | 7E-05 | 1E-04 | 5E-06 | 1E-06 | 5E-02 | 2E+00 | 4E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | | 2E+00 | | | |
| 3E-01 | 3E-01 | 3E-01 | | 2E-03 | 2E-04 | 2E-03 | 9E-05 | 2E-03 | 1E-03 | 9E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 5E-02 | 1E-03 | 7E+00 | 7E+00 | | | | | | | | 7E-01 | |
| 4E-01 | 4E-01 | 3E-01 | | 3E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 3E-04 | 9E-06 | 3E-06 | 9E-02 | 4E+00 | 7E-02 | 2E-03 | 9E+00 | 9E+00 | | | | | | | | | |
| 3E-01 | 3E-01 | 3E-01 | | 3E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 2E-04 | 8E-06 | 2E-06 | 8E-02 | 4E+00 | 6E-02 | 2E-03 | 8E+00 | 8E+00 | | | | | | | | | 8E-01 |
| 3E-01 | 3E-01 | 3E-01 | 8E-01 | 2E-03 | 2E-04 | 2E-03 | 9E-05 | 2E-03 | 1E-03 | 8E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 5E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | | 3E+00 | | | |
| 1E-01 | 1E-01 | 1E-01 | 1E+00 | 1E-03 | 9E-05 | 1E-03 | 4E-05 | 1E-03 | 6E-04 | 4E-05 | 9E-05 | 3E-06 | 9E-07 | 3E-02 | 1E+00 | 2E-02 | 6E-04 | 3E+00 | 3E+00 | | | | | | | 9E-01 | | |
| 2E-01 | 2E-01 | 2E-01 | 2E+00 | 1E-03 | 1E-04 | 2E-03 | 6E-05 | 2E-03 | 1E-03 | 6E-05 | 1E-04 | 5E-06 | 1E-06 | 5E-02 | 2E+00 | 4E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | | | 1E+00 | | |
| 1E-01 | 1E-01 | 1E-01 | | 1E-03 | 9E-05 | 1E-03 | 4E-05 | 1E-03 | 6E-04 | 4E-05 | 9E-05 | 3E-06 | 9E-07 | 3E-02 | 1E+00 | 2E-02 | 6E-04 | 3E+00 | 3E+00 | | | | | | | | | 3E+00 |
| 3E-01 | 3E-01 | 3E-01 | 8E-01 | 2E-03 | 2E-04 | 2E-03 | 9E-05 | 2E-03 | 1E-03 | 8E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 5E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | | 3E+00 | | | |
| 4E-01 | 4E-01 | 4E-01 | 3E-01 | 3E-03 | 3E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 3E-04 | 1E-05 | 3E-06 | 1E-01 | 4E+00 | 8E-02 | 2E-03 | 5E+00 | 5E+00 | | | | | | 5E+00 | | | |
| 3E-01 | 3E-01 | 3E-01 | 6E-01 | 2E-03 | 2E-04 | 2E-03 | 9E-05 | 2E-03 | 1E-03 | 9E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 5E-02 | 1E-03 | 4E+00 | 4E+00 | | | | | | 3E+00 | | | |
| 3E-01 | 3E-01 | 3E-01 | | 3E-03 | 2E-04 | 3E-03 | 1E-04 | 3E-03 | 2E-03 | 1E-04 | 2E-04 | 9E-06 | 2E-06 | 9E-02 | 4E+00 | 7E-02 | 2E-03 | 8E+00 | 8E+00 | | | | | | | | 5E+00 | |
| 1E-01 | 1E-01 | 1E-01 | | 9E-04 | 8E-05 | 9E-04 | 4E-05 | 1E-03 | 6E-04 | 4E-05 | 8E-05 | 3E-06 | 8E-07 | 3E-02 | 1E+00 | 2E-02 | 6E-04 | 3E+00 | 3E+00 | | | | | | | | | 2E+00 |
| 3E-01 | 3E-01 | 3E-01 | | 2E-03 | 2E-04 | 2E-03 | 9E-05 | 2E-03 | 1E-03 | 9E-05 | 2E-04 | 7E-06 | 2E-06 | 7E-02 | 3E+00 | 5E-02 | 1E-03 | 6E+00 | 6E+00 | | | | | | | | | 3E+00 |

| ID | Name | ID | Name | ID | Name | ID | Name | ID | Name |
|---|---|---|---|---|---|---|---|---|---|
| m1 | EX_pi_e_i | m11 | EX_cu2_e_i | m21 | EX_o2_e_i | v7 | EX_asp__L_e_i | v17 | EX_thr__L_e_i |
| m2 | EX_co2_e_i | m12 | EX_sel_e_i | m22 | EX_tungs_e_i | v8 | EX_met__L_e_i | v18 | EX_gly_e_i |
| m3 | EX_fe3_e_i | m13 | EX_cobalt2_e_i | m23 | EX_slnt_e_i | v9 | EX_ile__L_e_i | v19 | EX_arg__L_e_i |
| m4 | EX_h_e_i | m14 | EX_h2o_e_i | m24 | EX_glyc_e_i | v10 | EX_gln__L_e_i | v20 | EX_glu__L_e_i |
| m5 | EX_mn2_e_i | m15 | EX_mobd_e_i | v1 | EX_his__L_e_i | v11 | EX_cys__L_e_i | v21 | EX_melib_e_i |
| m6 | EX_fe2_e_i | m16 | EX_so4_e_i | v2 | EX_leu__L_e_i | v12 | EX_ser__L_e_i | v22 | EX_sucr_e_i |
| m7 | EX_zn2_e_i | m17 | EX_nh4_e_i | v3 | EX_phe__L_e_i | v13 | EX_ala__L_e_i | v23 | EX_tre_e_i |
| m8 | EX_mg2_e_i | m18 | EX_k_e_i | v4 | EX_lys__L_e_i | v14 | EX_trp__L_e_i | v24 | EX_lcts_e_i |
| m9 | EX_ca2_e_i | m19 | EX_na10_e_i | v5 | EX_tyr__L_e_i | v15 | EX_asn__L_e_i | v25 | EX_ac_e_i |
| m10 | EX_ni2_e_i | m20 | EX_cl_e_i | v6 | EX_val__L_e_i | v16 | EX_pro__L_e_i | v26 | EX_pyr_e_i |
| | | | | | | | | v27 | EX_cit_e_i |