Bayesian inference for realistic thermokinetic models of cell metabolism

What we want to do and where we need help

Table of contents

This document explains how our group, Quantitative Modelling of Cell Metabolism, thinks about modelling networks of metabolic reactions. It also explains where we think our approach could be improved and what kind of help we need in order to carry out these improvements.

Our modelling approach

Our aim is to synthesise information from the following sources:

- Quantitative measurements of the cell's metabolites, enzymes and fluxes
- A thermokinetic model that describes how each reaction in the metabolic network works in terms of some unknown parameters.
- Quantitative information about the parameters of the thermokinetic model.

We believe that taking all of this information into account at the same time will yield insights that can be used to guide the design of new cell strains.

By treating this problem as a case of Bayesian statistical inference, we can conveniently partition these information sources into separate sub-models:

- A measurement model or 'likelihood' describing what is learned from any measurement.
- A generative model, encompassing our thermokinetic model, that describes how measureable quantities are generated from underlying unobserved parameters.
- A prior model describing what is known about the unobserved parameters.

The rest of this section sets out how we flesh out these sub-models, starting with the generative model as it is by far the most interesting.

Our generative model

Thermokinetic models in general

A thermokinetic model describes the rates (usually referred to as 'fluxes') of the chemical reactions in a metabolic network parametrically, in terms of the concentrations of the network's metabolites, some kinetic parameters describing the network's reactions, some thermodynamic parameters and some parameters describing the network's boundary conditions. Equation 1 therefore captures the general form of a thermokinetic model:[1]

$$v = f_v(x, \theta) \tag{1}$$

[1] In Equation 1, the term x represents a vector of metabolite concentrations, the term represent parameter vectors, and v is a vector of real numbers, one per reaction

Thermokinetic models of cell metabolism are often contrasted with constraint-based models. Constraint-based models do not describe metabolic fluxes parametrically; instead they simply identify constraints that exclude certain flux values.

Many thermokinetic models are possible, depending on how the modeller trades off detail against other factors such as computational feasibility, model identification and ease of curation.

One thing that almost all thermokinetic models have in common is the assumption that the stoichiometric coefficient—that is, the number of molecules created or destroyed—of every metabolite in every reaction is known and collected in a stoichiometric matrix S with a row for each metabolite and a column for each reaction.

Our thermokinetic model

The thermokinetic model that we chose decomposes into factors as shown in Equation 2. The full model is presented here for

completeness and to illustrate the general nature of the equations that we use, but the details can safely be skipped over.

$$f_v(x, \theta) = Enzyme \cdot k_{cat} \cdot Reversibility \cdot Saturation \cdot Allostery \cdot Phosphorylation \tag{2}$$

Each of the terms on the right hand side of Equation 2 is a function of x and θ representing a conceptually distinct aspect of how an enzyme-catalysed reaction works.

The term *Enzyme* is a vector of non-negative real numbers representing the concentration of the enzyme catalysing each reaction.

The term k_{cat} is a vector of non-negative real numbers representing the amount of flux carried per unit of saturated enzyme.

The term *Reversibility* is a vector of real numbers capturing the impact of thermodynamics on the reaction's flux, as shown in Equation 3.[2]

$$\begin{aligned} Reversibility &= 1 - \exp\left(\frac{\Delta_r G' + RT \cdot S^T \ln(x)}{RT}\right) \\ \Delta_r G' &= S^T \Delta_f G' \end{aligned} \tag{3}$$

In Equation 3 the term T represents the temperature in Kelvin, R is the gas constant, $\Delta_r G'$ is a vector representing the Gibbs free energy change of each reaction in standard conditions and $\Delta_f G'$ is a vector representing the standard condition Gibbs free energy change of each metabolite's formation reaction, or in other words each metabolite's 'formation energy'.

Note that the thermodynamic effect on each reaction is derived from metabolite formation energies. This formulation is helpful because, provided that all reactions' rates are calculated from the same formation energies, they are guaranteed to be thermodynamically consistent.

The term *Saturation* in equation Equation 2 is a vector of non-negative real numbers representing, for each reaction, the fraction of enzyme that is saturated, i.e. bound to one of the

[2] Equation 3 is simplified a bit; the formulation we use in practice takes into account special cases where a reactions that consumes or produces water or is affected by membrane potential. See here for the code that is actually used.

reaction's substrates. To describe saturation we use Equation 4, which is taken from Liebermeister, Uhlendorf, and Klipp (2010).

$$
\begin{aligned}
Saturation_r &= a \cdot \text{free enzyme ratio} \\
a &= \prod_{\text{s substrate}} \frac{x_s}{km_{rs}} \\
\text{free enzyme ratio} &= \begin{cases} \prod_{\text{s sustrate}} (\frac{x_s}{km_{rs}})^{S_s r} + \sum_{\text{c inhibitor}} \frac{x_c}{ki_{rc}} & r \text{ irreversible} \\ -1 + \prod_{\text{s sustrate}} (\frac{x_s}{km_{rs}})^{S_s r} + \sum_{\text{c inhibitor}} \frac{x_c}{ki_{rc}} + \prod_{\text{p product}} (\frac{x_p}{km_{rp}})^{S_p r} & r \text{ reversible} \end{cases}
\end{aligned}
\tag{4}
$$

The term *Allostery* in Equation 2 is a vector of non-negative numbers describing the effect of allosteric regulation on each reaction. Allosteric regulation happens when binding to a certain molecule changes an enzyme's shape in a way that changes its catalytic behaviour. We use Equation 5, originally from Popova and Sel'kov (1975), to describe this phenomenon.

$$
\begin{aligned}
Allostery_r &= \frac{1}{1 + tc_r \cdot (\text{free enzyme ratio}_r \cdot \frac{Qtense}{Qrelaxed})^{subunits}} \\
Qtense &= 1 + \sum_{\text{i inhibitor}} \frac{x_i}{dc_{ri}} \\
Qrelaxed &= 1 + \sum_{\text{a activator}} \frac{x_a}{dc_{ra}}
\end{aligned}
\tag{5}
$$

Finally, the term *Phosphorylation* in Equation 2 captures an important effect whereby enzyme activity is altered due to a coupled process of phosphorylation and dephosphorylation.

$$
\begin{aligned}
Phosphorylation_r &= (\frac{\beta}{\alpha + \beta})^{subunits} \\
\alpha &= \sum_{\text{p phosphoylator}} kcat_p \cdot concp_p \\
\beta &= \sum_{\text{p dephosphoylator}} kcat_d \cdot concd_d
\end{aligned}
\tag{6}
$$

Steady state assumption

As well as assuming that the fluxes in our metabolic network will behave as our thermokinetic model expects, we also assume that the system was measured in a steady state, so that every non-boundary metabolite's concentration was not changing. This assumption is represented mathematically in Equation 7.

$$S \cdot f_v(x, \theta) = 0 \tag{7}$$

The steady state assumption removes degrees of freedom equal to the rank of S from our model, so that v is constrained to lie in the right null space of S. Usually, given θ it is possible can solve Equation 7 to find a steady state metabolite concentration x_{steady}.

Our full generative model

Our full generative model starts with a parameter vector θ. It then calculates the steady state metabolite concentration vector x_steady by solving $@eq - steady$.[3] Metabolite concentrations are now available and can be compared with measurements. To find flux values to compare with measurements we simply calculate $v_{steady} = f_v(x_{steady}, \theta)$.

[3] Note that in practice we do not solve Equation 7 directly but instead use ODE simulation - see section Section 4 for details

Measurement model

For measurements of metabolite and enzyme concentrations we use independent lognormal regression models:

$$\begin{aligned} y_x &\sim LN(\ln(x_{steady}), \sigma_x) \\ y_{enzyme} &\sim LN(\ln(Enzyme), \sigma_{enzyme}) \end{aligned} \tag{8}$$

For measurements of steady state fluxes we use independent linear regression models:

$$y_v \sim N(f_v(x_{steady}, \theta), \sigma_v) \tag{9}$$

{#eq-conc-measurement-model}

We ensure that flux measurements correspond to the flux modes of the measured network, so that the same pathway is never measured twice. See chapter 10 of Palsson (2015), entitled "the left null space" for a discussion of this issue.

We assume that the measurement errors σ_x, σ_{enzyme} and σ_v are known for each measurement.

These measurements y_x and y_{enzyme} are typically derived from quantitative metabolomics and proteomics experiments. Our choice of measurement model for these measurements is imperfect in at least these ways:

- The measurement error is not known, and is in fact very difficult to estimate.
- The measurements are not independent, as they are typically far more reliable as to differences—both between metabolites in the same experiment and between the same metabolite in different experiments—than they are as to absolute concentrations.

These problems also apply to flux measurements, but there is another issue: flux meausrements are derived from k

Prior model

For kinetic parameters including km, $kcat$, ki and parameters governing regulation we use independent informative lognormal prior distributions based on information gleaned from online databases, literature searches and intuition.

For boundary conditions including unbalanced metabolite concentrations, boundary fluxes and enzyme concentrations we use informative independent lognormal or normal prior distributions depending on the natural sign constraints of the variable (for example fluxes are not constrained to be positive, so in this case we use independent normal prior distributions). We sometimes use measurements to determine informative prior distributions for boundary conditions.

For thermodynamic parameters—i.e. metabolite formation energies—we use an informative multivariate normal distribution that is derived from equilibrium constant measurements reported in the NIST TECRDB Goldberg, Tewari, and Bhat (2004). The distribution is calculated using the component contribution method Noor et al. (2013) as implemented in the software equilibrator Beber et al. (2021). In future we would like to use our own software to generate these priors.

Implementation

Full details of how we implement Bayesian thermokinetic modelling can be found in our software package Maud.

Input specification

The required input for our Bayesian thermokinetic model is as follows:

- A qualititative specification of the thermokinetic model, including reaction stoichiometries, enzyme and compartment mappings and regulations.
- Records of experiments including measurements (and their errors) and enzyme knockouts.
- Information about parameters including priors and initial values. In future it will also be possible to fix a parameter's value.
- Configuration including MCMC hyperparameters, solver tolerances and units.

We created a custom format with which these things can be specified in toml files. Maud can read these files, perform validation and represent them as Python objects.

Posterior sampling

We carry out Bayesian inference by posterior sampling using adaptive Hamiltonian Monte Carlo as provided by Stan.

To our knowledge this is the only viable approach. Realistic models have too many parameters for many Bayesian computation approaches, including rejection sampling, ABC, Metropolis-Hastings and Gibbs sampling, while a similar study St. John et al. (2018) indicates that variational inference is unlikely to provide satisfactory approximations in this case.

We believe that the non-linear, multi-parameter equations shown in Section 2 create particular problems for MCMC sampling because they induce complex correlations in the posterior distribution. Consequently, although adaptive Hamiltonian Monte Carlo makes Bayesian inference for realistic thermokinetic models possible, many leapfrog steps are typically required per sample, as the sampler must traverse the posterior distribution in small steps in order to avoid discretisation errors.

Solving the steady state problem

As mentioned in Section 3, in our generative model steady state metabolite concentrations are calculated from parameter values by solving the steady state equation Equation 7. Since adaptive Hamiltonian Monte Carlo requires gradients of the posterior distribution, it is also necessary to calculate sensitivities of the steady state problem solution with respect to all parameters.

Our approach to this problem is to choose a starting concentration vector x_0 and a simulation time t, then find x_t using numerical ODE integration. To verify whether x_t is a steady state we then evaluate $S \cdot f_v(x_t, \theta)$ and check if the result is sufficiently close to zero.

Stan provides an interface to the Sundials ODE solver CVODES, including gradient calculations.

For the systems we have investigated, this method works better than solving the steady state problem using an algebra solver.

Analysis of results

After sampling is complete, Maud uses the Bayesian analysis library arviz to transform the results into an InferenceData object and save it as a json file, along with a range of files containing debug information. These files can be used to validate the computation and to draw conclusions about the measured system.

Our problems

There are several features we would like to add to Maud in order to model new phenomena, including promiscuous enzymes, varying compartment volumes and different enzyme mechanisms. In addition, as mentioned above our measurement models are somewhat unrealistic and should ideally be replaced with models that more accurately reflect the information contained in a multi-omics analysis, particularly in the case of data from 13C labelling experiments. There are also new features that would make the modelling process easier, including fixing arbitrary parameters values, automatically loading input data and adding more validation. We are pretty confident that we can incrementally add all these features within the current framework.

However our biggest problems require more fundamental changes: we would like to fit larger networks faster, and we would like to guarantee that our models appropriately respect known conservation relationships.

Problem 1: speed

Due to speed constraints, the ballpark range for the largest systems we can practically model is around 15 reactions and state variables: see the example maud input for the methionine cycle for an illustration. This is large enough that Maud can describe some non-trivial metabolic networks, but relatively small improvements would dramatically expand the range of

applicability. To illustrate, it is possible to describe the important subsystem comprising glycolysis and pentose phosphate pathway in *E. coli* using about 20 reactions, and `e_coli_core`, the smallest model in the UCSD BIGG database, has 95 reactions.

The main thing that limits Maud's speed is the need to solve the steady state problem many times. To illustrate, a typical run with 2000 iterations, each with 500 gradient evaluations requires the steady state problem to be solved 1000000 times.

Problem 2: conservation relationships

Conservation relationships exist when a network conserves the total abundance of a combination of metabolites. For example, central carbon metabolism does not change the total amount of the metabolites ATP, AMP and ADP, but only changes the distribution of the total pool among these three forms. In general any concentration vector x must satsify a set of mass conservation equations

$$L \cdot vol(x) = a \tag{10}$$

{#eq-conservation}

where L is the left null space of the stoichiometric matrix (i.e. a matrix satisfying $L \cdot S = 0$), vol is a function that translates concentrations into abundances and a is a vector of pool abundances.[4]

The practical importance of conservation relationships for our problem is that, in our current implementation, the vector a is fixed by the starting metabolite concentration, which is effectively hard-coded[5]. If it is inappropriate—for example if due to insufficient total ATP, AMP and ADP a reaction cannot flow in the correct direction—there is no way for the model to correct for this. We therefore think there are cases where our implementation prevents a steady state from being found even though one exists.

[4] See chapter 10 ('The Left Null Space of S') of Palsson (2015) for further discussion.

[5] To be precise the starting concentration for each metabolite is set before sampling as either the measured value if available or a default value

Potential solutions

Hardware

We have not been able to improve Maud's performance dramatically by using high performance computers or by exploiting parallelism as the key bottlenecks—it is difficult to ensure steady state solving.

Compiler

Another possibility is to explore compiler optimisations as described here. We have not had much success, though there is definitely room for further optimisation in this area.

ODE solver tolerances

There is a dramatic change in speed depending on the chosen ODE solver tolerances. We have not been able to find a systematic way to choose these, so we let the user configure these per run.

Adjoint ODE solver

Stan provides an interface to the CVODE bdf solver with gradients calculated using adjoint sensitivity analysis (the default approach, which we use, instead calculates gradients by solving an augmented ODE system). We expected using this solver to lead to improved speed and tried to use it when it was released, but we were not able to achieve large improvements, and reliability was worse.

Algebra solver

It is possible in principle to solve the steady state problem using an algebra solver rather than via our forward simulation method. We have not been able to make this approach work reliably as the available solvers require a good starting guess.

Starting concentration vector

A clear problem with our approach to solving the steady state problem is that we use a bad starting metabolite concentration vector. There are several potential speed improvements from choosing the starting concentration with escalating potential for improvement (and difficulty):

1. Find a better starting concentration vector before beginning MCMC and use the better starting concentration vector the same way we use our current one.
2. Use the warmup MCMC iterations to improve the starting concentration vector, then use the improved vector for the sampling iterations.
3. Choose a new starting vector every MCMC iteration or every few iterations.
4. Choose a new starting concentration vector programmatically on each gradient evaluation based on current parameter values.
5. Choose a new starting concentration vector on each gradient evaluation based on the result of the previous gradient evaluation.

Option 1 could be achieved simply by substituting the code that chooses the starting concentration for a better alternative.

Options 2 and 3 require running Stan multiple times with custom starting conditions. This isn't a huge challenge.

Option 4 is easy in principle as it only requires writing some Stan code. However we don't know how to use the current parameter values to find a good starting concentration.

Option 5 is difficult as it requires re-writing Stan's underlying C++ code.

DAE system

We think the correct way to take into account conservation relationsips is to cast the steady state problem as a differential algebraic equation system.

Stan provides an interface to the Sundials IDAS DAE solver.

References

Beber, Moritz E., Mattia G. Gollub, Dana Mozaffari, Kevin M. Shebek, and Elad Noor. 2021. "eQuilibrator 3.0 – a Platform for the Estimation of Thermodynamic Constants." *arXiv:2103.00621 [q-Bio]*, February. http://arxiv.org/abs/2103.00621.

Goldberg, Robert N., Yadu B. Tewari, and Talapady N. Bhat. 2004. "Thermodynamics of Enzyme-Catalyzed Reactions—a Database for Quantitative Biochemistry." *Bioinformatics* 20 (16): 2874–77. https://doi.org/10.1093/bioinformatics/bth314.

Liebermeister, Wolfram, Jannis Uhlendorf, and Edda Klipp. 2010. "Modular Rate Laws for Enzymatic Reactions: Thermodynamics, Elasticities and Implementation." *Bioinformatics* 26 (12): 1528–34. https://doi.org/10.1093/bioinformatics/btq141.

Noor, Elad, Hulda S. Haraldsdóttir, Ron Milo, and Ronan M. T. Fleming. 2013. "Consistent Estimation of Gibbs Energy Using Component Contributions." Edited by Daniel A. Beard. *PLoS Computational Biology* 9 (7): e1003098. https://doi.org/10.1371/journal.pcbi.1003098.

Palsson, Bernhard Ø. 2015. *Systems Biology: Constraint-based Reconstruction and Analysis*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9781139854610.

Popova, S. V., and E. E. Sel'kov. 1975. "Generalization of the Model by Monod, Wyman and Changeux for the Case of a Reversible Monosubstrate Reaction." *FEBS Letters* 53 (3): 269–73. https://doi.org/10.1016/0014-5793(75)80034-2.

St. John, Peter, Jonathan Strutz, Linda J Broadbelt, Keith E J Tyo, and Yannick J Bomble. 2018. "Bayesian Inference of Metabolic Kinetics from Genome-Scale Multiomics Data." *bioRxiv*, October. https://doi.org/10.1101/450163.