# KETCHUP’s kinetic formulation

KETCHUP automatically constructs an optimization formulation according to the Pandas DataFrame information. For benchmark purposes, KETCHUP formulates the optimization problem with reaction rates defined as decomposed elementary reaction steps. The following is an overview of elemental reaction steps.

Similar to Methods section 2.2.1 we use a generalized scheme for one substrate and one product is illustrated below:

where or is the forward or reverse rate at each elementary step *l*, respectively, E is the free enzyme complex, S is the substrate, P is the product, ES and EP is the enzyme bound to the substrate or product, respectively. However, we also include regulatory steps for competitive inhibitor , non-competitive inhibitor , and activator , as indicated in the following equations.

Competitive inhibition is modeled as binding to the free enzyme complex,

.

Noncompetitive inhibition is modeled as binding to the enzyme-substrate complex,

.

Activators are required to bind to the free enzyme before the enzyme binds to any substrate,

,

where E\* is the inactive enzyme form.

Before providing the formulation, we first introduce the definition of the following sets, variables, and parameters:

**Sets**

I = {metabolites}

J = {reactions}

Hj = {elementary reaction steps of reaction j}

m = {conditions where m = 1 is typically set to reference state}

**Variables**

= forward kinetic parameter for elementary step h ∈ H­j in reaction j

= reverse kinetic parameter for elementary step h ∈ Hj in reaction j

= flux of reaction j

= forward flux for elementary step h ∈ Hj in reaction j

= reverse flux for elementary step h ∈ Hj in reaction j

= fraction of enzyme abundance involved at elementary step h ∈ Hj in reaction j

= concentration of metabolite involved at elementary step h ∈ Hj in reaction j

**Parameters**

= Stoichiometric coefficient of reaction j ∈ J and metabolite I ∈ I

= measured experimental flux of reaction j ∈ J

= standard deviation of measured experimental flux of reaction j ∈ J

Ej = {relative enzyme total values for reaction j ∈ J }

The objective function of the optimization problem aims to minimize the squared distance between measured and experimental reaction flux weighted by squared standard deviation value of each corresponding reaction flux. This function results in reactions with tighter confidence interval having a larger contribution to the objective function. The formulation is described as:

Minimize: ,

Subject to:

Equation (1) in the formulation enforces steady-state conservation of mass across all metabolites. Equations (2) and (3) represent the rate law for all forward and reverse elementary reactions that follows mass-action kinetics, respectively. These rate laws involve the relative metabolite concentrations with the enzyme fractional abundances on binding or release steps and only the enzyme fraction term for conversion steps. The forward and reverse fluxes locked into the presented net flux reaction rates which are part of the objective function by Equation (4). Equation (5) ensures all enzyme fractional abundances sum to the total enzyme relative enzyme concentration for reaction j. Because this formulation uses steady-state data we assume that not only the flux rates are at metabolic steady-state but also the chance in fractional abundance for each enzyme complex are also in steady-state. Equation (6a), (6b), and (6c) enforces non-negativity of relative metabolite concentrations, kinetic parameters, and enzyme fractional abundances, respectively. These constraints also enforce a user-specified upper limit to the possible values these variables can hold.

By default, all mutant relative metabolite concentrations and enzyme sums are normalized to the wild-type’s concentrations. Wild-type strain’s (i.e., reference) metabolite enzyme concentrations are set equal to one and the mutant strains have their relative metabolite concentration initialized to one and relative enzyme sums fixed to one for the unknown enzyme concentrations. However, if enzyme concentrations are available for the mutants then the values can be adjusted accordingly. If absolute enzyme concentrations for the reference are available, those values can be used instead of the value of one.

# IPOPT iterative strategy

We summarize below the relevant portions of IPOPT algorithm for determination of local optimal solutions in KETCHUP. For detailed information on IPOPT please see its original publication (Wächter & Biegler, 2006).

KETCHUP estimates *in vivo* kinetic parameters by using Pyomo (Bynum et al., 2021; Hart et al., 2011) to construct a nonlinear optimization formulation that is solved by the Interior Point OPTimizer (IPOPT) (Wächter & Biegler, 2006), a nonlinear programming (NLP) solver that utilizes an interior point line search filter. The IPOPT algorithm approximates a set of solutions by iteratively solving a set of barrier problems. The barrier problems are first transformed from the original optimization formulation problem P (given in standard form):

,where *f(x)* is the objective function (*i.e.*,SSR), *cp(x)* are equality constraints (*i.e.*, rate laws, enzyme total sums, stoichiometry, net flux), and *x* are the variables (*i.e.*, kinetic parameters, metabolite concentrations, enzyme fractional abundances).

The formulation for the barrier problem B transformed from the original formulation is:

The transformed formulation implicitly removes the nonnegativity constraints of the original formulation by augmenting the objective function with a barrier term , where *µ* is the barrier parameter that is iteratively updated until it converges to zero. This barrier parameter influences how far the solution *x* is from the boundaries of the optimization problem and reaching an optimal solution when *µ* is zero (Vanderbei, 2001). By iterating in the interior of the feasible region, this algorithm allows for faster convergence for large-scale problems compared to other NLP algorithms such as Sequential Quadratic Programming (Błaszczyk et al., 2007). See Supplemental Text 2 for more details of IPOPT.

Problem B is solved iteratively at decreasing values of the barrier parameter until it satisfies the termination criteria discussed below. At each updated barrier parameter value , a set of primal-dual equations are formed and iteratively solved for . The iterations *h* (outside loop) and *y* (inner loop) update µh and , respectively.

The set of primal-dual equations for problem B are set as:

where λ ∈ ℝn, corresponds to the Lagrangian multiplier for the equality constraint in problem B, *X* and *Z* is diagonal matrix for the vector and , respectively, and e is a vector of all ones for the appropriate dimension. Variable () is an auxiliary variable introduced to circumvent the differentiation of from becoming too large and ill-conditioning the Hessian matrix (Potra & Wright, 2000). Note that variable *z* ∈ ℝn can be also interpreted as the Lagrangian multiplier for inequality constraints of problem P and that the primal-dual equations form the Karush-Kuhn-Tucker (KKT) conditions (Kuhn & Tucker, 1951) of problem P when .

To find solutions satisfying KKT optimality conditions at , IPOPT employs an iterative Newton-Raphson method based on solving the following system of linear equations:

,

where I is the identity matrix of appropriate dimensions and ( are search direction vectors for .W­y is the Hessian of the Lagrangian for problem P,

.

IPOPT avoids solving the nonsymmetric system of linear equations directly by first solving a smaller symmetric system,

where (derived by eliminating the last row of the original system of linear equations) and subsequently calculating the vector from,

.

Setting up the smaller symmetric system of linear equations allows for use of efficient linear solvers (see below). The computed search directions are then used to calculate the next iteration of solutions as:

,

for step sizes (the z variable takes on a different step size than other variables). However, step sizes must be calculated to determine the next, these step sizes are calculated as:

},

for a “fraction-to-the-boundary” parameter τy ∈ (0,1) that is decided by:

,

where is a predefined constant (default 0.99). Note that the step length of *z* is automatically calculated but the other variables require backtracking line-search procedure that uses a decreasing sequence of trial steps (with q = 0,1,2…). A line-filter method is used to determine the acceptance of a trial point. Filter methods consider if a trial point is accepted if it leads to sufficient progress towards minimizing either the objective function or equality constraint violation , where denotes a fixed vector norm (*e.g.*, for a vector x̅, ) . IPOPT employs additional rules to help improve the search for trial points:

1. If the current iterate is feasible but did not make sufficient progress towards either of the two requirements, then as long as the objective function value has a sufficient decrease, then the trial point is accepted.
2. A “filter” set Fy(Θ, φ) is initialized at the beginning of the algorithm and at each iteration, the set is augmented with previously combinations of constraint violation and objective function values to prevent future iterates from returning to the neighborhood of xk.
3. In the case of no feasible trial step size where , then IPOPT calls upon *feasible restoration phase* that attempts to find a new iterate xy+1 that by minimizing constraint violation until a new acceptable iteration is found or the problem is found to be locally infeasible.

Compared to traditional line-search algorithms, IPOPT employs a filter method that allows for larger step sizes and the introduction of the *feasibility restoration phase* allows for detection of local infeasibilities that may be resulted from ill-posed problems.

Using set of primal-dual barrier problems, the optimality error is defined as:

,

where denotes a fixed vector norm (*e.g.*, for a vector x̅, ).

The algorithm terminates if the problem is infeasible (discussed above) or if the approximate solution of the original problem P (when µ=0) at current iteration y satisfies:

,

where ε­tol is a user provided tolerance. This optimality error is also used to update the barrier parameter µh when it satisfies the tolerance:

,

for a constant κε > 0. The barrier parameter is then updated as:

with constants ∈ (0,1) and ∈ (1,2). This update rule prevents from becoming much smaller than the user-specified tolerance εtol and helps prevent numerical difficulties when approach the optimal solution of the overall problem (near the bounds).

# Benchmark of linear solvers

The performance and robustness of the IPOPT algorithm depends on the linear solver used to compute the search directions at each iteration and the problem being solved. Because linear systems resulting from kinetic parameter estimations problems are known to be ill-conditioned while many converged solutions exhibit degeneracy (Hu et al., 2023), it is important to benchmark different linear solvers IPOPT uses. Although MUMPS is the default linear solver for IPOPT, we found that recompiling IPOPT so that it uses other linear solvers (*i.e.,* HSL’s MA57, MA86, and MA97) not only significantly reduces solve times but obtains solutions with better fits than MUMPS. The main difference between these solvers lies in their efficiency to factorize different type of matrices (*e.g.,* combination of (un)symmetric, positive/negative, (in)definite matrices), the size of the matrices containing the formulated problem (*e.g.,* small, medium, or large-scaled problems), and capability of utilizing multiple CPU cores to solve the problem (*i.e.,* serial vs parallel). We benchmarked these linear solvers with the large-scale *E. coli* model k-ecoli307 and detailed the results in Table S1. All problems were solved with the same settings for IPOPT discussed in Section 2.1.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Linear solvers | | | | |
|  | MUMPS | MA57 | MA86 | MA97 | MA97 |
| Number of CPU cores | 1 | 4 | 4 | 1 | 4 |
| Average solve time (min) | 146.76 | 52.85 | 118.30 | 49.43 | 31.09 |
| Average SSR | 708.43 | 1470.61 | 923.70 | 1231.53 | 225.95 |
| Fastest solve time (min) | 52.31 | 15.84 | 47.77 | 7.60 | 7.35 |
| Lowest SSR | 2.027 | 2.16 | 1.546 | 2.889 | 2.47 |
| % solution converged (out of 500) | 11.4 | 13.2 | 15.0 | 14.6 | 13.8 |

**Table S1:** Benchmark of different linear solvers used to solve the kinetic parameter estimation formulation for the large-scale model k-ecoli307. All linear solvers ran under the same IPOPT settings discussed in the Methods section.

These results show that the HSL’s linear solvers were consistently able to converge more solutions than the MUMPS solver and amongst the HSL solvers, MA97 yielded the best average solve time and lowest average SSR. We note that although MA86 has a higher percentage of converged solutions and converged to the lowest SSR, it on average requires at least 3 times longer to solve than MA97 and does not consistently converge to lower SSR solutions. More importantly, MA86 is unable to deterministically regenerate results from a provided initial condition when utilizing multiple cores because parallelization (may) randomly reorder parallel operations to improve solve performance (*HSL. A Collection of Fortran Codes for Large Scale Scientific Computation.*, n.d.). Overall, running any one of the HSL solvers with one core would significantly increase each solution’s solve time.

# King-Altman method for converting elementary kinetics to Michaelis-Menten constants

Reaction-decomposed elementary step kinetics is the most fundamental kinetic description at the molecular level capable of distinctly capturing binding, release, and allostery steps of the reaction. For enzymatic steps, the reactants bind to the enzyme one step at a time until they are converted to products and the converted now enzyme bound products are subsequently releases each product one step at a time. The order of the binding and release of metabolites with the enzyme complex depends on the user provided enzyme mechanism. The benefit of elementary step kinetics for allosteric regulation is apparent as inhibitors can be designated to bind to the free enzyme complex (*i.e.,* competitive inhibition) or enzyme-substrate complex (*i.e.,* uncompetitive inhibition). It is possible to mathematically model non-competitive inhibition (both competitive and uncompetitive inhibition) as well as the contribution of each type of inhibition if known (we assume each type of inhibition equally contributes).

For an example one substrate S, one product P reaction catalyzed by enzyme E:

For each reaction rate *v* in the forward and reverse directions *f* and *r*, respectively, the mass action kinetics equation can be described as such:

|  |  |
| --- | --- |
| Forward elementary steps | Reverse elementary steps |
|  |  |
|  |  |
|  |  |

The enzyme level ETot is expressed as the total enzyme sum of all the enzyme complexes:

(KA 1)

To convert the elementary kinetic parameters to their equivalent Michaelis-Menten constants, we assume the elementary reactions are in quasi steady-state and the following enzyme complexes with this quasi steady-state assumption form the following equations:

(KA2) (KA3)

Substituting KA1 into KA2 and KA3, we find that the net flux v of this reversible reaction to be:

where

|  |  |
| --- | --- |
| Michaelis-Menten constant | Elementary Constant |
|  |  |
|  |  |
|  |  |
|  |  |

For reactions with multiple substrates and products, the same procedure can be used to derive Michaelis-Menten constants. A schematic method has been developed to derive the higher-order elementary reactions by (King & Altman, 1956); this method allows for the automated conversion of elementary kinetic parameters to their Michaelis-Menten equivalents without having the explicitly solve algebraic equations.

A graph of a graph of a graph

Description automatically generated with medium confidence

**Supplementary Figure SF1**: Recapitulation plot of kinetic model fitted with a.) batch and b.) chemostat data simultaneously. Despite perfect recapitulation for batch fluxomics data, chemostat fluxomics data departs significantly for TCA cycle reactions and PTAr (a reaction utilizing acetyl-CoA).