Cellular Stoichiometric Models and Linear Programming

Flux Balance Analysis (FBA) is a computational approach that examines the flow of metabolites through metabolic networks under steady-state conditions. By representing cellular metabolism as a network of biochemical reactions and imposing mass balance constraints, FBA computes flux distributions that optimize an objective function—often the production of biomass or growth.

This method enables the prediction of how modifications, such as altering nutrient substrates or knocking out specific genes, affect cellular growth. To implement FBA, we require a linear programming (LP) solver, such as Gurobi, that can be integrated with MATLAB. This integration facilitates the setup and execution of the LP problem using a genome-scale metabolic model, thereby enabling the complete FBA process.

Learning objectives

(a) Formulate and solve LP for a large cellular stoichiometric model using Gurobi
(b) Interpret LP results with respect to growth rate and exchange fluxes
(c) Investigate possible existence of alternative optima
(d) Work with data structures, functions and tables in MATLAB

Please follow these instructions for installing the Gurobi Optimizer and setting up its MATLAB interface. If you have any questions regarding the installation of Gurobi and this project, please feel free to open a new discussion: https://github.com/jcwang587/math-modeling/discussions
(a) How do I install Gurobi Optimizer? (b) How do I install Gurobi for Matlab?

Part I – Setting Up and Solving the FBA Model

Load the E. coli metabolic model, which you can download here. Then, set up the FBA problem using the predefined flux bounds specified in the model, including the sparse stoichiometric matrix (`A`), the objective coefficients (`c`), the right-hand side (`b`), as well as the linear constraints and boundaries (`lb` and `ub`). Finally, call the Gurobi solver to solve the LP problem and save the result in a structure.

Part II – Identifying and Tabulating Non-Zero Fluxes

Find all the non-zero uptake and secretion fluxes, identified by their code names (including `EX`), from the previous linear programming result. Then, create two tables (using the `table` function), one for uptake fluxes and one for secretion fluxes, where the first column contains the flux name, and the second column lists their flux values.

Part III – Investigating Alternative Optima by Blocking Non-Zero Secretion Fluxes

Block each non-zero secretion flux and investigate the existence of alternative optima. Based on the previous results of the secretion index, block these fluxes by setting their upper boundaries to zero and then resolve the problem (using `outputflag=0`, so that no output information is printed on the screen). Obtain the growth rate under these conditions and check for the existence of alternative optima.

It is recommended to implement this using a `for` loop. You will need to reset the upper boundary to 1000 for each iteration. Finally, create a table that lists the blocked secretion fluxes and their corresponding growth rates.