MFAPipe

Version 0.1.0.0

Mark Borkum (ed.)
mark.borkum@pnnl.gov

Working Draft
June 4, 2017

Contents

List of Figures

List of Tables

List of Acronyms

13C-MFA ^{13}C Metabolic Flux Analysis

CSV Comma-Separated Values

EMU Elementary Metabolite Unit

FBA Flux Balance Analysis

MFA Metabolic Flux Analysis

MIP Mixed Integer Programming

SEM Standard Error of Measurement

Part I

Preface

Chapter 1

How to Read This Book

Ostensibly, this book is read straight through from start to finish.

1.1 Typographical Elements

Here, we describe the typographical elements that are used throughout this book.

1.1.1 Command-Line Invocations

Command-line invocations are given in monospace typeface with black foreground and gray background.

For example: open your terminal, type in `rm -rf` and press the return key. (This is just an example. Don't actually run this command! Running this command will remove the reference to every object in your filesystem.)

1.1.2 Source Code Snippets

Source code snippets are given in monospace typeface with black foreground and white background.

Line numbers are printed on the left-hand side of the source code snippet.

For example, the Haskell source code

```
1 map :: (a -> b) -> [a] -> [b]
2 map _ [] = []
3 map f (x : xs) = f x : map f xs
```

describes a higher-order function that applies a function to each element of a list. Line 1 gives the type signature of the function. Lines 2 and 3 give the body of the function. The two branches of the function correspond to the two constructors of the `Data.List` data type (empty and cons-list cell).

1.1.3 Notes

Note. Brief statements of facts, topics, or thoughts, look like this. Notes can be safely ignored.

1.1.4 Warnings

Warning. Brief statements of caveats, pitfalls, or common errors, look like this. Ignoring a warning is strongly discouraged.

Part II

MFAPipe

Chapter 2

Installing MFAPipe

2.1 Prerequisites

2.1.1 The Haskell Tool Stack

The best way to get started is to download and install The Haskell Tool Stack, available at:

`http://haskellstack.org/`

2.1.2 GLPK

Next, download and install the GNU Linear Programming Kit (GLPK), available at:

`https://www.gnu.org/software/glpk/`

Warning. Please ensure that the following packages are installed: `glpk` and `libglpk-dev`.

2.2 Source Code

The source code for MFAPipe is hosted online in a publicly accessibly GitHub repository, available at:

`https://github.com/EMSL-NMR-EPR/Haskell-MFAPipe-Executable`

2.2.1 Download Instructions

To clone the GitHub repository: open your terminal, type in `git clone https://github.com/EMSL-NMR-EPR/Haskell-MFAPipe-Executable.git`, and press

the return key.

The command creates a directory named "Haskell-MFAPipe-Executable", initializes a ".git" directory inside it, pulls down the contents of the GitHub repository, and checks out a working copy of the latest version.

Warning. The above assumes that the Git version control system is correctly installed on your machine and that the `git` executable is on the search path.

2.2.2 Install from Source

To navigate to the directory that contains the MFAPipe source code: open your terminal, type in `cd Haskell-MFAPipe-Executable`, and press the return key.

To build MFAPipe from the source code: open your terminal, type in `stack setup && stack build`, and press the return key. After MFAPipe has finished building: type in `stack install`, and press the return key.

MFAPipe is now installed on your machine.

Warning. The above assumes that The Haskell Tool Stack is correctly installed on your machine and that the `stack` executable is on the search path.

Chapter 3

Starting Out

3.1 Preliminaries

Alright, let's get started! If you're the sort of person who doesn't read introductions to things and you skipped it, you might want to read the previous chapter, because it explains what you need in order to follow this tutorial.

The first thing that we're going to do is test whether or not the MFAPipe executable is correctly installed. Open your terminal, type in `mfaPipe` and press the return key. You will be greeted with something like this:

```
No mode given and no default mode
```

Don't worry! Everything is fine. The MFAPipe executable is correctly installed.

The MFAPipe executable has many modes. The above greeting is the MFAPipe executable's way of saying "please specify a mode."

Note. For this tutorial, we'll be using the "mfa-levmar" mode. More on that story later.

Let's do another test. Open your terminal, type in `mfaPipe --version` and press the return key. You should see the following message:

```
MFAPipe v0.1.0.0, (C) 2016 Pacific Northwest National Laboratory
```

The message confirms that the MFAPipe executable is correctly installed, and that the version of the MFAPipe executable is = 0.1.0.0.

3.2 Introduction

In this tutorial, we'll work through an example of using MFAPipe to perform single labeling, steady state ^{13}C Metabolic Flux Analysis (13C-MFA) [citation needed] using the Levenberg-Marquardt

algorithm [citation needed].

The model for this tutorial (the chemical reaction network, experimental data, constraints, etc.) is derived from the *"simple example network"* from the literature [1]. The model was selected because: (i) it is prototypical, i.e., representative of the main patterns in model design; and, (ii) it has been investigated by multiple researchers [2, 3], i.e., the values of the metabolic fluxes for the chemical reactions of the network are known.

Note. Reading the cited articles is not a prerequisite for this tutorial.

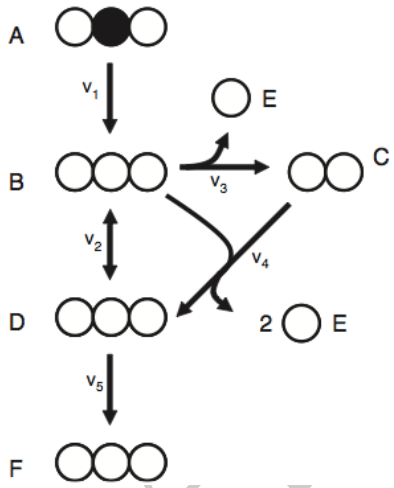

Figure 3.1: Depiction of intracellular compartment of chemical reaction network.

The intracellular compartment of the chemical reaction network, depicted in Figure 3.1, consists of the following:

- Five chemical reactions, denoted v_1, v_2, ..., v_5;
- One network substrate, labeled "A";
- Three network intermediates, labeled "B", "C" and "D"; and,
- Two network products, labeled "E" and "F".

The model has three subtleties: (i) the chemical reaction that is denoted by v_2 is reversible; (ii) the network product "E" is generated by multiple chemical reactions; and, (iii) in the context of the chemical reaction that is denoted by v_4, the stoichiometric coefficient of the network product "E" is $\neq 1$.

Our goal is two-fold: (i) to fit the values of the metabolic fluxes for the chemical reactions of the network to measurements of the isotopic labeling states of the network products, given the isotopic labeling states of the network substrates; and, (ii) to use the fitted values of the metabolic fluxes in order to calculate the isotopic labeling states of the network intermediates.

3.3 Experimental Data

We are given the following information about the intracellular compartment of the chemical reaction network:

1. The uptake rate of the network substrate "A" is = 100 (arbitrary units).
2. The isotopic labeling state of the network substrate "A" is 100% [2-^{13}C], i.e., the mass distribution of the second atom of the network substrate "A" is:
   - $\mathrm{A}_{\{2\},\,\mathrm{M+0}} = 0$ mol%
   - $\mathrm{A}_{\{2\},\,\mathrm{M+1}} = 100$ mol%
3. At isotopic steady state, the mass distribution of the network product "F" is:
   - $\mathrm{F}_{\{1,\,2,\,3\},\,\mathrm{M+0}} = 0.01$ mol% $\pm\, 0.003$ SEM
   - $\mathrm{F}_{\{1,\,2,\,3\},\,\mathrm{M+1}} = 80.08$ mol% $\pm\, 0.0025$ SEM
   - $\mathrm{F}_{\{1,\,2,\,3\},\,\mathrm{M+2}} = 19.83$ mol% $\pm\, 0.002$ SEM
   - $\mathrm{F}_{\{1,\,2,\,3\},\,\mathrm{M+3}} = 0.09$ mol% $\pm\, 0.0015$ SEM

3.3.1 Notation for Mass Fractions

The notation for mass fractions is

$$\mathrm{Label}_{\{1,\,2,\,\ldots,\,n\},\,\mathrm{M}+m} = p \pm x$$

where:

- $1, 2, \ldots, n \in \mathbb{Z}$ are the indices of the n atoms of the metabolite;
- $m \in \mathbb{Z}$ is the number of additional neutrons, viz., the number of mass shifts;
- $p \in [0,\, 1] \subset \mathbb{R}$ is the molar percentage; and,
- $x \in \mathbb{R}$ is the Standard Error of Measurement (SEM).

3.4 Ready, Set, Go!

In MFAPipe, models are completely specified using the FluxJS data format.

By the term "completely specified" we mean that every aspect of the model is a part of the FluxJS document, i.e., if a unit of information pertains to the model, then it is in the FluxJS document. Otherwise, it is not in the FluxJS document.

MFAPipe is a fully compliant FluxJS implementation. The high-level workflow for an MFAPipe-based investigation, depicted in Figure 3.2, is simple: FluxJS documents go in and zip archives come out.

Figure 3.2: Depiction of workflow for MFAPipe-based investigation.

The results of an MFAPipe-based investigation can be exactly reproduced from a combination of: (i) the FluxJS document for the model; (ii) the version of the MFAPipe executable; and, (iii) the command that was used in order to invoke the MFAPipe executable.

```
{
  "model_metadata": {
    "http://purl.org/dc/terms/title": "Simple example network",
    "http://purl.org/dc/terms/source": "http://www.sciencedirect.com/science/article/pii/S1096717606000061",
  },
  "model": {
    "isotopic_labeling_states_per_atom": 2,
    "reaction_network": {
      "v1": { "type": "mfa", "reaction": "A (abc) -> B (abc)" },
      "v2": { "type": "mfa", "reaction": "B (abc) <-> D (abc)" },
      "v3": { "type": "mfa", "reaction": "B (abc) -> C (bc) + E (a)" },
      "v4": { "type": "mfa", "reaction": "B (abc) + C (de) -> D (bcd) + E (a) + E (e)" },
      "v5": { "type": "mfa", "reaction": "D (abc) -> F (abc)" },
      "b1": { "type": "mfa", "reaction": "A.ext (abc) -> A (abc)" },
      "b2": { "type": "mfa", "reaction": "E (a) -> E.ext (a)" },
      "b3": { "type": "mfa", "reaction": "F (abc) -> F.ext (abc)" }
    },
    "experiments": {
      "exp1": {
        "isotopic_distributions": {
          "A.ext": [{ "%": 100, "isotopic_distribution": [[1,0],[0,1],[1,0]] }]
        },
        "measurements": {
          "F.ext#123,M+0": [{ "=": 0.0001, "weight": 333.3333333333333 }],
          "F.ext#123,M+1": [{ "=": 0.8008, "weight": 400 }],
          "F.ext#123,M+2": [{ "=": 0.1983, "weight": 500 }],
          "F.ext#123,M+3": [{ "=": 0.0009, "weight": 666.6666666666666 }],
          "F.ext#123,M+0 + F.ext#123,M+1 + F.ext#123,M+2 + F.ext#123,M+3": [{ "=": 1 }]
        }
      }
    },
    "constraints": {
      "bounds": {
        "b1.f": { "=": 100 }
      }
    }
  }
}
```

Figure 3.3: FluxJS document for model.

The FluxJS document for the model is given in Figure 3.3. When confronted with the nuts and bolts details of the programming, it is helpful to take a deep breath, and to proceed slowly and methodically. Let's walk through the FluxJS document for the model together, in bite-sized pieces.

Top-level (Lines 1–38). The top-level structure of the FluxJS document divides the information content into two sections: metadata and data.

Model metadata (Lines 2–5). Metadata for the model is asserted using the "model_metadata" data structure. In this tutorial, the "title" and "source" are asserted[1]; however, we could also have asserted, for instance, a bibliographic citation, the contact information of the model's authors, the chemical identifiers of the metabolites, and/or a description of the experimental protocol by which the measurements were obtained.

Model (Lines 6–37). The model is specified using the "model" data structure.

Number of isotopic labeling states per atom (Line 7). The first attribute of the "model" data structure, "isotopic_labeling_states_per_atom", specifies the number of isotopic labeling states per isotopic atom of metabolites in the model.

For 13C-MFA, where the isotopic tracer element is carbon, the value of this attribute is = 2, corresponding to the ^{12}C and ^{13}C isotopes.

Chemical reaction network (Lines 8–17). The second attribute of the "model" data structure, "reaction_network", is a data structure that, unsurprisingly, specifies the chemical reaction network for the model.

Chemical Reaction (Exchange)	Stoichiometry	Atom Transitions
v_1	A $\rightarrow$ B	abc $\rightarrow$ abc
v_2	B $\leftrightarrow$ C	abc $\leftrightarrow$ abc
v_3	B $\rightarrow$ C + E	abc $\rightarrow$ bc + a
v_4	B + C $\rightarrow$ D + E + E	abc + de $\rightarrow$ bcd + a + e
v_5	D $\rightarrow$ F	abc $\rightarrow$ abc

Table 3.1: Stoichiometry and atom transitions for intracellular compartment of chemical reaction network.

The intracellular compartment of the chemical reaction network (Table 3.1) consists of five exchange reactions, denoted v_1, v_2, ..., and v_5 (in FluxJS, metabolic flux variables for exchange reactions are prefixed with "v"), which consume and generate six intracellular metabolites, labeled "A", "B", ..., and "F".

The intracellular compartment of the network has one substrate, labeled "A", and two products, labeled "E" and "F"; hence, the extracellular compartment of the chemical reaction network (Table 3.2) consists of three transport reactions, denoted b_1, b_2 and b_3 (in FluxJS, metabolic flux variables for transport reactions are prefixed with "b"), which consume and generate three extracellular metabolites, labeled "A.ext", "E.ext" and "F.ext".

[1]The labels for the "title" and "source" metadata elements are drawn from Dublin Core [citation needed]. Using a metadata standard is good practice, as it promotes interoperability.

Chemical Reaction (Transport)	Stoichiometry	Atom Transitions
b_1	A.ext → A	abc → abc
b_2	E → E.ext	a → a
b_3	F → F.ext	abc → abc

Table 3.2: Stoichiometry and atom transitions for extracellular compartment of chemical reaction network.

Chemical reactions are encoded using a special-purpose syntax, and annotated with a "type" attribute. In this tutorial, all chemical reactions are of the "mfa" type.

	v1.f	v2.b	v2.f	v3.f	v4.f	v5.f	b1.f	b2.f	b3.f
A	-1	0	0	0	0	0	1	0	0
B	1	1	-1	-1	-1	0	0	0	0
C	0	0	0	1	-1	0	0	0	0
D	0	-1	1	0	1	-1	0	0	0
E	0	0	0	1	2	0	0	-1	0
F	0	0	0	0	0	1	0	0	-1
A.ext	0	0	0	0	0	0	-1	0	0
E.ext	0	0	0	0	0	0	0	1	0
F.ext	0	0	0	0	0	0	0	0	1

Table 3.3: Stoichiometric model for chemical reaction network.

After lexical analysis and parsing of the syntax, followed by processing of the resulting data structures, MFAPipe infers the stoichiometric model for the chemical reaction network (Table 3.3). The stoichiometric model is divided into quadrants, corresponding to the four combinations of intra- and extracellular metabolites and exchange and transport chemical reactions.

Experimental data (Lines 18–31). The third attribute of the "model" data structure, "experiments", is a data structure that asserts the isotopic labeling data that is required by the fitting algorithm: (i) the isotopic labeling states of the network substrates; and, (ii) the measurements of the isotopic labeling states of the network intermediates and products.

Experiments have unique names. In this tutorial, the name of the experiment is "exp1".

Isotopic labeling states of network substrates are specified by mass distributions for individual atoms (referred to as "isotopic distributions"). This enables the calculation of isotopomer and mass fractions for arbitrary network substrates, including mixtures, where each mixture part is associated with a percentage and isotopic distribution.

Measurements are specified by algebraic expressions and corresponding lists of weighted values, where the weight of a given value is, typically, the reciprocal of the SEM, and the weight of an "unweighted" value, say, a constant, is = 1.

In this tutorial, the mass distribution of the network substrate "F" is asserted as four separate measurements (lines 24–27). Moreover, it is asserted that, by definition, the sum of the mass distribution for the network substrate "F" is = 1 (line 28).

Constraints (Lines 32–36). The fourth attribute of the "model" data structure, "constraints", is a data structure that, again, unsurprisingly, asserts the linear and box constraints for the metabolic flux variables that are to be fitted by the Levenberg-Marquardt algorithm.

In this tutorial, the metabolic flux of the forwards direction of the exchange reaction b_1, denoted "b1.f", is fixed to a value of 100 (arbitrary units).

And we're done! See, it wasn't that bad after all, was it?

3.5 Invoking MFAPipe

In this tutorial, we are using the "mfa-levmar" mode, instructing the MFAPipe executable to perform single labeling, steady state 13C-MFA using the Levenberg-Marquardt algorithm.

The parameters for the Levenberg-Marquardt algorithm, such as the maximum number of iterations and the termination criteria, are optionally specified as command line arguments for the MFAPipe executable. If you do not specify a given parameter, then the MFAPipe executable will use the default value. (A detailed description of the Levenberg-Marquardt algorithm, its parameters and their default values is given in Chapter 6 of this manual.)

Open your terminal, type in `mfaPipe mfa-levmar --input=model.json --output=result.zip`, and press the return key. The input, a FluxJS document, is read from the file "model.json", and the output, a zip archive, is written to the file "result.zip".

The zip archive contains many files. Let's have a look inside.

3.6 Results

The MFAPipe executable generates zip archives, where every file in the archive uses the Comma-Separated Values (CSV) data format. CSV documents can be opened in any text editor, manipulated by almost all programming languages, and imported into almost all spreadsheet packages, e.g., Microsoft Excel.

Metabolic Flux Variable	Reported Value	Fitted Value
v1.f	100	100
v2.b	50	49.38184893971324
v2.f	110	109.45513197592477
v3.f	20	19.963358481894247
v4.f	20	19.96335848189424
v5.f	80	80.03664151810577
b1.f	100	100
b2.f	60	59.89007544568261
b3.f	80	80.03664151810577

Table 3.4: Fitted values of metabolic flux variables.

The output of the Levenberg-Marquardt algorithm, the fitted values of the metabolic flux variables is written to the file "./result/flux.csv". The output is summarized in Table 3.4. There is strong agreement between the reported and fitted values of the metabolic flux variables.

The results of the goodness-of-fit analysis of the weighted residuals, calculated using the fitted values of the metabolic flux variables, is written to the file "./result/statistics.csv". The exact p-value for the fit is = 0.112, indicating that, even though there is strong agreement, i.e., a "good" fit, the measurements are incorrectly weighted.

Warning. This tutorial demonstrates one of the major pitfalls of fluxomics investigations: syntactic validity of models does not imply veracity of results. (Garbage in, garbage out!)

3.7 Conclusion

Congratulations! You have reached the end of this tutorial. Together, we have used MFAPipe to perform single labeling, steady state 13C-MFA using the Levenberg-Marquardt algorithm.

This tutorial, however, has only scatched at the surface of MFAPipe's capabilities.

Keep reading. Let's explore MFAPipe's capabilities together.

Chapter 4

Flux Balance Analysis

4.1 Introduction

Flux Balance Analysis (FBA) is an analytic technique for simulating metabolism in metabolic networks by calculating steady state metabolic fluxes [4]. FBA simulations are formulated as optimization problems that use linear programming [5] in order to minimize or maximize the value of an objective function with respect to the values of the metabolic flux variables for the chemical reactions in the network. Typically, the objective function for an FBA simulation corresponds to the maximization of biomass formation [6].

4.2 Method

4.2.1 Stoichiometric Models

The central mathematical object of an FBA simulation is the stoichiometric model of the metabolic network: a matrix, whose elements are the stoichiometric coefficients of the metabolites with respect to the metabolic flux variables. Stoichiometric coefficients for reagents are negative; whereas, stoichiometric coefficients for products are positive.

	Exchange	Transport
Intracellular	...	...
Extracellular	...	...

Table 4.1: General form of stoichiometric model for arbitrary chemical reaction network.

The general form of a stoichiometric model (Table 4.2.1) is a block matrix, whose quadrants correspond to the four combinations of intra- and extracellular metabolites and exchange and transport reactions.

4.2.2 Stoichiometry Matrices

The stoichiometry matrix, denoted $\mathbf{S}$, is a block matrix of the two quadrants of the stoichiometric model that correspond to intracellular metabolites.

4.2.3 The Steady State Hypothesis

The results of FBA simulations must satisfy the steady state hypothesis $\mathbf{S} \cdot \mathbf{v} = \mathbf{0}$, where $\mathbf{v}$ is the column vector of the values of the metabolic flux variables.

4.2.4 Orthonormal Bases of Stoichiometry Matrices

Computing the orthonormal basis of the range space of the stoichiometry matrix (also referred to as the "kernel" or "nullspace"), denoted $\mathrm{Ker}\,(\mathbf{S})$, yields a new matrix, whose columns correspond to the free variables of the system, i.e., the degrees of freedom, such that $\mathbf{v} = \mathrm{Ker}\,(\mathbf{S}) \cdot \mathbf{u}$, where $\mathbf{u}$ is the column vector of the values of the free variables.

The steady state hypothesis can, therefore, be reformulated as $\mathbf{S} \cdot (\mathrm{Ker}\,(\mathbf{S}) \cdot \mathbf{u}) = \mathbf{0}$, where matrix multiplication is associated from right-to-left, since, for an arbitrary matrix $\mathbf{M}$, the matrix product $\mathbf{M} \cdot \mathrm{Ker}\,(\mathbf{M})$ is, by definition, $= \mathbf{0}$.

The advantage of this approach is that, as the number of free variables is, typically, lower than the number of metabolic flux variables, the optimization process is, typically, more computationally efficient. However, the disadvantage of this approach is that, because the free variables are not the metabolic flux variables, box constraints, such as lower and upper bounds, cannot be directly[1] applied.

4.2.5 Mathematical Formulation

Linear Programming

The mathematical formulation of FBA, as implemented by MFAPipe, is as follows:

- Let $\mathbf{v}$ be a column vector of the values of the metabolic flux variables, where the element $\mathbf{v}_i \in \mathbb{R}$ is the value of the metabolic flux variable referenced by the index $i \in \mathbb{Z}$.
- Let $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}$ be the objective function, represented as a column vector; a linear function of the vector of the values of the metabolic flux variables, where the element f_i is the coefficient of the metabolic flux variable referenced by the index $i \in \mathbb{Z}$.
- Let $\mathbf{S}$ be the stoichiometry matrix, where the element $\mathbf{S}_{i,\,j} \in \mathbb{R}$ is the stoichiometric coefficient of the metabolite and metabolic flux variable, referenced by, respectively, the indices i, $j \in \mathbb{Z}$.

[1] Some implementations of linear programming algorithms perform this optimization automatically, introducing slack variables behind the scenes in order to indirectly apply box constraints.

- Let $\mathbf{lb}$ and $\mathbf{ub}$ be, respectively, column vectors of lower and upper bounds for the values of the metabolic flux variables, where the elements $\mathbf{lb}_i$, $\mathbf{ub}_i \in \mathbb{R}$ are, respectively, the lower and upper bounds for the metabolic flux variable that is referenced by the index $i \in \mathbb{Z}$.
- Let $\mathbf{A}$ and $\mathbf{b}$ be, respectively, a matrix of linear constraints and a column vector of values, i.e., a linear system $\mathbf{A} \cdot \mathbf{v} = \mathbf{b}$, where the elements $\mathbf{A}_{i,\,j}$ and $\mathbf{b}_i$ are, respectively, the coefficient of the row and metabolic flux variable and the value of the row, referenced by, respectively, the indices i, $j \in \mathbb{Z}$.

Hence, the linear programming problem is

$$\begin{aligned} \max_{\mathbf{v}_i,\ \forall i \in \mathbb{Z}} \quad & \mathbf{f}^{\mathrm{T}} \cdot \mathbf{v} \\ \text{subject to} \quad & \left[\frac{\mathbf{S}}{\mathbf{A}}\right] \cdot \mathbf{v} = \left[\frac{\mathbf{0}}{\mathbf{b}}\right] \\ & \mathbf{lb}_i \leq \mathbf{v}_i \leq \mathbf{ub}_i,\ \forall i \in \mathbb{Z} \end{aligned} \tag{4.1}$$

which is, in prose, the maximization of the value of the objective function with respect to the values of the metabolic flux variables.

4.3 Conclusion

In this chapter, we have described in detail the FBA capabilities that are provided by MFAPipe.

Chapter 5

Metabolic Flux Analysis

5.1 Introduction

Metabolic Flux Analysis (MFA) is an analytic technique for simulating metabolism in metabolic networks by calculating steady state (stationary) and non-stationary metabolic fluxes [7]. MFA simulations are characterized by their consideration of the transfer of moieties containing isotopic tracer elements from one metabolite to another. MFA simulations are formulated as least square minimization problems.

5.2 Method

5.2.1 Representation of Isotopic Labeling States

Neutrons, Neutrons, Neutrons!

Representations of states of moieties of isotopic atoms are isomorphic to logical conjunctions of assertions of both the proton and neutron numbers of each atom, e.g., "the first atom has 1 proton and 0 neutrons, the second atom has 6 protons and 7 neutrons, etc."

In this way, the characterstics of the isotopic atoms are completely specified, and hence, the representation is an unambiguous (if verbose) description of the isotopic labeling state of the given moiety.

However, if there exists a bijection $f : \mathbb{Z} \to \mathbb{Z}$ from proton to *least* neutron number, e.g., $f(1) \stackrel{\text{def}}{=} 0$, $f(6) \stackrel{\text{def}}{=} 6$, etc., then representations of states of moieties of isotopic atoms are isomorphic to logical conjunctions of assertions of both the proton and *additional* neutron numbers, viz., mass shifts, e.g., "the first atom has 1 proton and 0 additional neutrons, the second atom has 6 protons and 1 additional neutron, etc."

In this way, the characteristics of the isotopic atoms are still completely specified; but, the total number of states that is required to represent each atom is reduced from positive infinity (the cardinality of the continuum) to the successor of the greatest additional neutron number.

Let m be a non-negative integer that denotes the greatest additional neutron number (either directly or, assuming the existence of a bijection from proton to least neutron number, indirectly), then the set of determinate isotopic labeling states of a given isotopic atom is the set of integral members of the interval $[0,\, m]$

$$\mathrm{St}_m \stackrel{\mathrm{def}}{=} \{0,\, 1,\, \ldots,\, m\}\,, \tag{5.1}$$

where St_m denotes the set of determinate isotopic labeling states.

The proportion of moieties in a given sample with a given mass, or, assuming the existance of a bijection from proton to least neutron number, a given number of mass shifts, is referred to as a "mass fraction."

The set of mass fractions, represented as a column vector, referred to as a "mass fraction vector," is the probability function for a discrete probability distribution, referred to as a "mass distribution."

Isotopomers

Isotopomers (a contraction of the term "isotopic isomers" or, more precisely, "isomers of isotopic atoms") [8] are representations of the isotopic labeling states of moieties, where the isotopic labeling states of every atom is determinate.

Note. The original formulation of isotopomers was motivated by carbon chemistry, with two determinate isotopic labeling states per atom, corresponding to the ^{12}C and ^{13}C isotopes. In this section, we give a generalized formulation.

Let n and m be non-negative integers that denote, respectively, the number of isotopic atoms in a given moiety and the greatest neutron number, then the set of isotopomers is the set of length-n vectors of determinate isotopic labeling states

$$\mathrm{Isotopomer}_{n,\,m} \stackrel{\mathrm{def}}{=} \{(a_1,\, a_2,\, \ldots,\, a_n) \mid \forall\, i : (i \in \{1,\, 2,\, \ldots,\, n\}) \wedge (a_i \in \mathrm{St}_m)\}\,, \tag{5.2}$$

where $\mathrm{Isotopomer}_{n,\,m}$ denotes the set of isotopomers.

The proportion of moieties in a given sample with a given isotopic labeling state (represented by a given isotopomer) is referred to as an "isotopomer fraction." The set of isotopomer fractions, represented as a column vector of $(m+1)^n$ entries, is referred to as an "isotopomer fraction vector."

Each entry of the isotopomer fraction vector gives the functional value of the corresponding minterm, i.e., each isotopomer fraction vector is a completely specified, Boolean function of n variables with $m+1$ values per variable.

Cumomers

Cumomers (a contraction of the term "cumulative isotopomers") [9] are representations of the isotopic labeling states of moieties, where the isotopic labeling states of some atoms are determinate, while others are indeterminate.

Note. The original formulation of cumomers was motivated by carbon chemistry, with two determinate isotopic labeling states per atom, corresponding to the ^{12}C and ^{13}C isotopes. Moreover, in the original formulation, the isotopic labeling state that corresponds to zero additional neutrons is deliberately elided by the authors. In this section, we give a generalized formulation.

Let n and m be non-negative integers that denote, respectively, the number of isotopic atoms in a given moiety and the greatest neutron number, then the set of cumomers (a contraction of the term "cumulative isotopomers") is the set of length-n vectors of either determinate or indeterminate isotopic labeling states, i.e.,

$$\text{Cumomer}_{n,\,m} \stackrel{\text{def}}{=} \{(a_1,\, a_2,\, \ldots,\, a_n) \mid \forall\, i : (i \in \{1,\, 2,\, \ldots,\, n\}) \wedge (a_i \in \text{St}_m \cup \{\top\})\} \tag{5.3}$$

where $\text{Cumomer}_{n,\,m}$ and $\top$ denote, respectively, the set of cumomers and the indeterminate isotopic labeling state.

Note. The indeterminate isotopic labeling state is the logical disjunction of every determinate isotopic labeling state. The correspondence between logical disjunction (in Boolean algebra) and addition (in the Galois field $\mathbb{F}_2$) is the origin of the word "cumulative" in the term "cumulative isotopomers."

The proportion of moieties in a given sample with a given isotopic labeling state (represented by a given cumomer) is referred to as a "cumomer fraction." The set of cumomer fractions, represented as a column vector of $(m+2)^n$, is referred to as a "cumomer fraction vector."

Each entry of the cumomer fraction vector gives the functional value of the corresponding minterm, i.e., each cumomer fraction vector is a completely specified, Boolean function of n variables with $m+2$ values per variable.

5.2.2 Elementary Metabolite Units

The Elementary Metabolite Unit (EMU) method [2] was originally proposed as a technique for reducing the total number of variables in systems of linear equations for MFA simulations; however, as we will now show, it can do so much more.

Note. The original formulation of the EMU method did not give a constructive definition for the set of EMUs of a given moiety. In this section, we give a generalized formulation.

Let n and m be non-negative integers that denote, respectively, the number of isotopic atoms in a given moiety and the greatest neutron number, then the set of cumomers of a given moiety can be exhaustively partitioned into 2^n mutually disjoint subsets, where the isotopic labeling states of the

atoms in each subset are determinate

$$\begin{aligned}\mathrm{EMU}_{n,\,m} &\stackrel{\text{def}}{=} \{\mathrm{EMU}_{n,\,m,\,N} \mid \forall\ N : (N \in \mathbb{P}\,(\{1,\,2,\,\ldots,\,n\}))\} \\ \mathrm{EMU}_{n,\,m,\,N} &\stackrel{\text{def}}{=} \left\{ a \mid (a \in \mathrm{Cumomer}_{n,\,m}) \wedge \left(\forall\ i : (i \in \{1,\,2,\,\ldots,\,n\}) \wedge \left(a_i \in \left\{ \begin{array}{lll} \mathrm{St}_m & \text{if} & i \in N \\ \{\top\} & \text{if} & i \notin N \end{array} \right) \right) \right\},\end{aligned} \tag{5.4}$$

where $\mathrm{EMU}_{n,\,m}$ denotes the set of EMUs for the given moiety, $\mathrm{EMU}_{n,\,m,\,N}$ denotes the set of cumomers for a given subset of atoms N, and $\mathbb{P}$ denotes the power-set function.

The cardinality of N is referred to as the size of the EMU or "EMU size".

Importantly, we can talk both *quo* and *qua* EMU; referring to either the EMU *per se*, i.e., the subset of atoms, or the EMU *per quad*, i.e., the corresponding subset of cumomers.

Suppose that we display a function $f : \{\{\mathbb{Z}\}\} \to S$ from the set of mutually disjoint subsets of atoms to the set of arbitrary objects S. What properties must the function f and its codomain S possess in order to be "valid" with respect to the EMU method?

In a recent investigation [citation needed], it was found that: (i) the function f must be a homomorphism when subsets of atoms are pairwise disjoint; (ii) the set S must be linear, with a commutative monoid structure; and, (iii) the dimension of elements of S must be well-defined function of n and m.

Set	Monoidal Combinator	Dimension Function
EMU sizes	Addition	$\phi\,(n,\,m) = 1$
Isotopomer fraction vectors	Cartesian product under multiplication	$\phi\,(n,\,m) = (m+1)^n$
Cumomer fraction vectors	Cartesian product under multiplication	$\phi\,(n,\,m) = (m+2)^n$
Mass fraction vectors	One-dimensional convolution	$\phi\,(n,\,m) = m+1$

Table 5.1: Codomains of EMU homomorphism.

The three conditions are not only satisfied by the sets of isotopomer, cumomer and mass fraction vectors, but also by the set of EMU sizes (Table 5.1).

Decomposition

The algorithm for the conversion of chemical reactions to sets of EMU-based reactions (referred to as "EMU reactions") is called "EMU decomposition."

The EMU decomposition algorithm is as follows:

1. Convert the chemical reaction to a set of chemical reactions in *normal form*;
2. For each normalized chemical reaction, construct the power-set of the set of atom transitions for each product, given the reagents;
3. For each set of atom transitions, construct the corresponding EMU reaction; and,
4. Partition the set of EMU reactions by size (the number of transitory atoms).

In practical terms, if we assume that there is a one-to-many correspondence between metabolites and atom configurations, and that atom configurations are tagged with probabilities, then *normalization* is the construction of the set of one-to-one correspondences, where each correspondence (not atom configuration) is tagged with a probability. The set of normalized chemical reactions is, therefore, the set of possible combinations of one-to-one correspondences, with the probability of the reaction (as a whole) being the product of the probabilities for the correspondences on each side of the arrow.

Constructing the power-set of the set of atom transitions has two key benefits: (i) it is exhaustive; and, (ii) it reduces the worst-case computational complexity of the algorithm to $\mathcal{O}\left(n^2\right)$.

Labeled, Directed Graph Representation

Sets of size-n EMU reactions (referred to as "EMU reaction networks") can be represented as adjacency matrices, viz., labeled, directed graphs, which, after permutation of the rows and columns, are of the form

$$\Omega_n = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \Omega_{n;\,2,1} & \Omega_{n;\,2,2} & \mathbf{0} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} & \mathbf{0} \end{bmatrix} \tag{5.5}$$

and vectors of vertices of the form

$$\omega_n = \begin{bmatrix} \omega_{n;\,1} \\ \omega_{n;\,2} \\ \omega_{n;\,3} \end{bmatrix}, \tag{5.6}$$

where the sub-vectors $\omega_{n;\,1}$, $\omega_{n;\,2}$ and $\omega_{n;\,3}$ are, respectively, the size-n substrates, intermediates and products. Vertex and edge labels are, respectively, EMUs and linear functions of metabolic flux variables.

Optimization

Given a set of EMUs, whose isotopic labeling states are to be determined, MFAPipe implements two directed graph optizations: vertex reachability analysis; and, edge smoothing.

Vertex Reachability Analysis. For a directed graph $G = (V,\, E)$, with a vertex set V and edge set E, the reachability relation of G is the transitive closure of E. [citation needed]

Note. In MFAPipe, the reachability relation is computed in linear time using a depth-first search. The set of EMUs is determined by bottom-up traversal of the algebraic expressions for measurements of isotopic labeling states, i.e., the graph is optimized for every EMU that is referenced as part of an expression.

Edge Smoothing. An operation on a vertex v with respect to the pair of edges $(e_1,\, e_2)$ incident on v that removes and replaces both edges with a new edge that connects the other endpoints of the pair. [citation needed]

Note. In MFAPipe, edge smoothing is performed on the depth-first, topological ordering of the graph for pairs of edges that correspond to the same type of chemical reaction: exchange and exchange; or, transport and transport.

For EMU reaction networks, edge smoothing acts upon pairs of edges $(p_1 \times e_1,\, p_2 \times e_2)$ that are associated with discrete probabilities. The label for the first edge is selected and the probabilities for both edges are multiplied, resulting in a new edge $(p_1 \times p_2) \times e_1$.

The Flux Balance Equation

Assuming a variational formulation of fluid dynamics for a bounded domain with a piecewise-smooth boundary and a general mass[1] conservation law [10], flux balance equations for size-n EMUs are of the form

$$f\left(\omega_n\right) = \operatorname{diag}\left(\operatorname{rowsum}\left(\begin{bmatrix} \Omega_{n;\,2,1} & \Omega_{n;\,2,2} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} \end{bmatrix}\right)\right) \cdot \begin{bmatrix} \omega_{n;\,2} \\ \omega_{n;\,3} \end{bmatrix} - \begin{bmatrix} \Omega_{n;\,2,1} & \Omega_{n;\,2,2} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} \end{bmatrix} \cdot \begin{bmatrix} \omega_{n;\,1} \\ \omega_{n;\,2} \end{bmatrix}, \tag{5.7}$$

where f is an arbitrary function, diag denotes a function that constructs a square matrix with the specified leading diagonal, and rowsum denotes a function that yields a column vector of the sums of the elements of each row of the specified matrix.

On the right-hand side of Equation 5.7, the left operand is the mass lumping matrix for intermediate and product size-n EMUs using the row sum technique; and, the right operand is the discrete transport operator for substrate and intermediate size-n EMUs.

The set of intermediate size-n EMUs is referenced by both operands. Since all matrix and vector elements are linear, the equation can be rearranged using elementary algebra to obtain

$$f\left(\omega_n\right) = \left(\operatorname{diag}\left(\operatorname{rowsum}\left(\begin{bmatrix} \Omega_{n;\,2,1} & \Omega_{n;\,2,2} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} \end{bmatrix}\right)\right) - \begin{bmatrix} \Omega_{n;\,2,2} & \mathbf{0} \\ \Omega_{n;\,3,2} & \mathbf{0} \end{bmatrix}\right) \cdot \begin{bmatrix} \omega_{n;\,2} \\ \omega_{n;\,3} \end{bmatrix} - \begin{bmatrix} \Omega_{n;\,2,1} \\ \Omega_{n;\,3,1} \end{bmatrix} \cdot \begin{bmatrix} \omega_{n;\,1} \end{bmatrix}. \tag{5.8}$$

On the right-hand side of Equation 5.8, the left operand is the consistent mass matrix for intermediate and product EMUs; and, the second matrix is the discrete transport operator for substrate EMUs.

The Steady State Flux Balance Equation

Assuming the steady state hypothesis $f\left(\omega_n\right) = 0$ yields

$$\left(\operatorname{diag}\left(\operatorname{rowsum}\left(\begin{bmatrix} \Omega_{n;\,2,1} & \Omega_{n;\,2,2} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} \end{bmatrix}\right)\right) - \begin{bmatrix} \Omega_{n;\,2,2} & \mathbf{0} \\ \Omega_{n;\,3,2} & \mathbf{0} \end{bmatrix}\right) \cdot \begin{bmatrix} \omega_{n;\,2} \\ \omega_{n;\,3} \end{bmatrix} = \begin{bmatrix} \Omega_{n;\,2,1} \\ \Omega_{n;\,3,1} \end{bmatrix} \cdot \begin{bmatrix} \omega_{n;\,1} \end{bmatrix}, \tag{5.9a}$$

which is consistent with the original definition [2, Equation 6]

$$\mathbf{A}_n \cdot \mathbf{X}_n = \mathbf{B}_n \cdot \mathbf{Y}_n, \tag{5.9b}$$

[1] Here, the term "mass" refers to the quantity of material for a given discretized term in the context of a flux equation.

where

$$\begin{aligned}
\mathbf{A}_n &= \operatorname{diag}\left(\operatorname{rowsum}\left(\begin{bmatrix} \Omega_{n;\,2,1} & \Omega_{n;\,2,2} \\ \Omega_{n;\,3,1} & \Omega_{n;\,3,2} \end{bmatrix}\right)\right) - \begin{bmatrix} \Omega_{n;\,2,2} & \mathbf{0} \\ \Omega_{n;\,3,2} & \mathbf{0} \end{bmatrix} \\
\mathbf{B}_n &= \begin{bmatrix} \Omega_{n;\,2,1} \\ \Omega_{n;\,3,1} \end{bmatrix} \\
\mathbf{X}_n &= \begin{bmatrix} \omega_{n;\,2} \\ \omega_{n;\,3} \end{bmatrix} \\
\mathbf{Y}_n &= \begin{bmatrix} \omega_{n;\,1} \end{bmatrix}.
\end{aligned} \tag{5.9c}$$

Implicit differentiation, which is allowable, given the linearity of the matrix and vector elements, yields

$$\frac{\mathrm{d}\mathbf{A}_n}{\mathrm{d}\mathbf{v}} \cdot \mathbf{X}_n + \mathbf{A}_n \cdot \frac{\mathrm{d}\mathbf{X}_n}{\mathrm{d}\mathbf{v}} = \frac{\mathrm{d}\mathbf{B}_n}{\mathrm{d}\mathbf{v}} \cdot \mathbf{Y}_n + \mathbf{B}_n \cdot \frac{\mathrm{d}\mathbf{Y}_n}{\mathrm{d}\mathbf{v}}, \tag{5.10}$$

where $\mathbf{v}$ is a column vector of the values of the metabolic flux variables. Differentiation is performed on every row in parallel, since, by definition, the matrices and vectors in the flux balance equation are separable.

5.2.3 The Jacobian Matrix

The Jacobian matrix, denoted $\mathbf{J}$, is the matrix of all first-order partial derivatives of a given vector-valued function. In MFA using the EMU method, the vector-valued function is the function that gives the measurements of the isotopic labeling states of the EMUs, i.e., the flux balance equation.

Note. In MFAPipe, measurements are algebraic expressions of isotopomer and mass fractions of EMUs, consisting of: constants; elementary functions; exponential and logarithmic functions; trigonometric functions; hyperbolic functions; and, any valid combination thereof. Partial derivatives are computed by forward-mode automatic differentiation [11]. The Jacobian matrix is, therefore, analytically defined, instead of being approximated by finite difference methods.

5.2.4 Mathematical Formulation

Least Squares

The mathematical formulation of parallel labeling, steady state MFA, as implemented by MFAPipe, is as follows:

- Let $\mathbf{v}$ be a column vector of the values of the metabolic flux variables, where the element $\mathbf{v}_i \in \mathbb{R}$ is the value of the metabolic flux variable referenced by the index $i \in \mathbb{Z}$.
- Let $f_n : \mathbb{R}^2 \to \mathbb{R}$ be the nth vector-valued function of the values of the metabolic flux variables, viz., the steady state flux balance equation.
- Let $\mathbf{S}$ be the stoichiometry matrix, where the element $\mathbf{S}_{i,\,j} \in \mathbb{R}$ is the stoichiometric coefficient of the metabolite and metabolic flux variable, referenced by, respectively, the indices i, $j \in \mathbb{Z}$.

- Let **lb** and **ub** be, respectively, column vectors of lower and upper bounds for the values of the metabolic flux variables, where the elements $\mathbf{lb}_i$, $\mathbf{ub}_i \in \mathbb{R}$ are, respectively, the lower and upper bounds for the metabolic flux variable that is referenced by the index $i \in \mathbb{Z}$.
- Let **A** and **b** be, respectively, a matrix of linear constraints and a column vector of values, i.e., a linear system $\mathbf{A} \cdot \mathbf{v} = \mathbf{b}$, where the elements $\mathbf{A}_{i,\,j}$ and $\mathbf{b}_i$ are, respectively, the coefficient of the row and metabolic flux variable and the value of the row, referenced by, respectively, the indices i, $j \in \mathbb{Z}$.

Hence, the least squares problem is

$$\begin{aligned} \min_{\mathbf{v}_i,\ \forall i \in \mathbb{Z}} \quad & \textstyle\sum_n \sum_i \left(\mathrm{E}\left[\mathbf{x}_{n;\,i}\right] - \mathbf{x}_{n;\,i}\right)^2 \text{ where } \mathbf{x}_n = f_n\left(\mathbf{v}\right) \\ \text{subject to} \quad & \left[\frac{\mathbf{S}}{\mathbf{A}}\right] \cdot \mathbf{v} = \left[\frac{\mathbf{0}}{\mathbf{b}}\right] \\ & \mathbf{lb}_i \leq \mathbf{v}_i \leq \mathbf{ub}_i,\ \forall i \in \mathbb{Z} \end{aligned} \tag{5.11}$$

which is, in prose, the minimization of the sum of squared residuals of the measurements, where E is the expectation function.

The term "parallel labeling" refers to the specification of an arbitrary number of vector-valued functions of the values of the metabolic flux variables.

The term "steady state" refers to the vector-valued functions being instances of the steady state flux balance equation, and the usage of the stoichiometry matrix as the linear constraints of the optimization.

Weighted Least Squares

Weighted least squares is a modification [12, Section 39.2] of the unweighted case, where the residuals and Jacobian are modified according to

$$\frac{\mathrm{E}\left[\mathbf{x}_{n;\,i}\right] - \mathbf{x}_{n;\,i}}{\sigma_{n;\,i}}, \tag{5.12}$$

where $\sigma_{n;\,i}$ is the weight of the given measurement.

Typically, weight is defined as the reciprocal of standard error of measurement. In this case, the weight of a constant is = 1.

Note. The examplar constant is a measurement that represents the sum of a mass distribution of a given moiety, which, by definition, is = 1.

5.2.5 Goodness of Fit

The goodness of fit of the residuals is evaluated using the Kolmogorov-Smirnov test [13], where p-values are calculated using the Marsaglia-Tsang-Wang exact alogorithm [14].

Derived Mathematical Objects

For both the weighted and unweighted cases, the output of least squares is: (i) the Jacobian matrix for each set of measurements at the point of termination, denoted $\mathbf{J}_n$; (ii) the column vector of the fitted values of the metabolic flux variables, denoted $\mathbf{v}$; and, (iii) the covariance matrix for the metabolic flux variables, denoted

$$\operatorname{Covar}(\mathbf{v}) = \begin{bmatrix} \operatorname{Covar}(\mathbf{v}_1,\, \mathbf{v}_1) & \operatorname{Covar}(\mathbf{v}_1,\, \mathbf{v}_2) & \dots \\ \operatorname{Covar}(\mathbf{v}_2,\, \mathbf{v}_1) & \operatorname{Covar}(\mathbf{v}_2,\, \mathbf{v}_2) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{5.13}$$

By definition, each element of the principal diagonal of the covariance matrix is the variance of the corresponding metabolic flux variable

$$\operatorname{Var}(\mathbf{v}) = \begin{bmatrix} \operatorname{Var}(\mathbf{v}_1) \\ \operatorname{Var}(\mathbf{v}_2) \\ \vdots \end{bmatrix} = \operatorname{diag}(\operatorname{Covar}(\mathbf{v})) = \begin{bmatrix} \operatorname{Covar}(\mathbf{v}_1,\, \mathbf{v}_1) \\ \operatorname{Covar}(\mathbf{v}_2,\, \mathbf{v}_2) \\ \vdots \end{bmatrix}. \tag{5.14}$$

Covariance matrices for each set of measurements are constructed [1, Equation 16] by taking the matrix product of the Jacobian matrix and the covariance matrix for the metabolic flux variables

$$\operatorname{Covar}(\mathbf{x}_n) = \mathbf{J}_n \cdot \left(\operatorname{Covar}(\mathbf{v}) \cdot (\mathbf{J}_n)^{\mathrm{T}} \right), \tag{5.15}$$

where matrix multiplication is associated from right-to-left.

The variances for each set of measurements are constructed in a similar fashion to those of the metabolic flux variables

$$\operatorname{Var}(\mathbf{x}_n) = \begin{bmatrix} \operatorname{Var}(\mathbf{x}_{n;\,1}) \\ \operatorname{Var}(\mathbf{x}_{n;\,2}) \\ \vdots \end{bmatrix} = \operatorname{diag}(\operatorname{Covar}(\mathbf{x}_n)) = \begin{bmatrix} \operatorname{Covar}(\mathbf{x}_{n;\,1},\, \mathbf{x}_{n;\,1}) \\ \operatorname{Covar}(\mathbf{x}_{n;\,2},\, \mathbf{x}_{n;\,2}) \\ \vdots \end{bmatrix}. \tag{5.16}$$

The contribution matrix [1, Equation 31] for each set of measurements gives the fractional contribution of the jth measurement to the local variance of the ith metabolic flux variable

$$\operatorname{Contrib}_{n;\,i,\,j} = \frac{(\mathbf{M}_{n;\,i,\,j})^2}{\operatorname{Var}(\mathbf{v}_i) \times \operatorname{Var}(\mathbf{x}_{n;\,j})} \text{ where } \mathbf{M}_n = \operatorname{Covar}(\mathbf{v}) \cdot (\mathbf{J}_n)^{\mathrm{T}}. \tag{5.17}$$

By definition, $\operatorname{Contrib}_{n;\,i,\,j} = 0$ when $\mathbf{J}_{n;\,i,\,j} = 0$, and $\operatorname{Contrib}_{n;\,i,\,j} = \mathrm{NaN}$ when either $\operatorname{Var}(\mathbf{v}_i) = 0$ or $\operatorname{Var}(\mathbf{x}_{n;\,j}) = 0$.

5.3 Conclusion

In this chapter, we have described in detail the MFA capabilities that are provided by MFAPipe.

Chapter 6

Invoking MFAPipe

6.1 fba-simplex (Flux Balance Analysis using Simplex)

The Simplex algorithm [15] is an iterative algorithm that solves a system of linear equations in each step, and terminates when either the optimum is reached, or when the solution becomes unfeasible.

The Simplex algorithm assumes that the values of the metabolic flux variables inhabit a continuous space, e.g., the real numbers.

To use the Simplex algorithm, specify the "fba-simplex" mode when invoking the MFAPipe executable, i.e., open your terminal, type `mfaPipe fba-simplex`, and press the return key.

Syntax	Description	Required?	Default Value
`--input=FILE`	Input file (FluxJS document)	Yes	n/a
`--output=FILE`	Output file (zip archive)	Yes	n/a
`--tm-lim=INT`	Maximum number of iterations	No	10000
`--presolve`	Presolve the system?	No	Yes

Table 6.1: Command line arguments for "fba-simplex" mode of MFAPipe executable.

Command line arguments for the "fba-simplex" mode of the MFAPipe executable are documented in Table 6.1.

The directory tree for the resulting zip archive is as follows:

```
/
├── reaction_network
│   ├── reaction.csv ..... Chemical reaction network
│   └── stoichiometry.csv ..... Stoichiometric model
└── result
    ├── flux.csv ..... Optimized metabolic fluxes
    └── objective_function.csv ..... Objective function value
```

6.2 fba-mip (Flux Balance Analysis using Mixed Integer Programming)

The Mixed Integer Programming (MIP) algorithm [16] is an iterative algorithm that solves a system of linear equations in each step, and terminates when either the optimum is reached, or when the solution becomes unfeasible.

The MIP algorithm assumes that the values of the metabolic flux variables are either integers or half-integers.

To use the MIP algorithm, specify the "fba-mip" mode when invoking the MFAPipe executable, i.e., open your terminal, type `mfaPipe fba-mip`, and press the return key.

Syntax	Description	Required?	Default Value
`--input=FILE`	Input file (FluxJS document).	Yes	n/a
`--output=FILE`	Output file (zip archive).	Yes	n/a
`--tm-lim=NUMBER`	Maximum number of iterations.	No	10000
`--presolve`	Presolve the system?	No	No

Table 6.2: Command line arguments for "fba-mip" mode of MFAPipe executable.

Command line arguments for the "fba-mip" mode of the MFAPipe executable are documented in Table 6.2.

The directory tree for the resulting zip archive is as follows:

```
/
├── reaction_network
│   ├── reaction.csv ..... Chemical reaction network
│   └── stoichiometry.csv ..... Stoichiometric model
└── result
    ├── flux.csv ..... Optimized metabolic fluxes
    └── objective_function.csv ..... Objective function value
```

6.3 mfa-levmar (Metabolic Flux Analysis using Levenberg-Marquardt)

The Levenberg-Marquardt algorithm [17] (weighted nonlinear least squares) is an iterative algorithm that solves a system of linear equations at each step, and terminates when either the optimum is reached, or when the solution becomes infeasible.

Note. The implementation of the Levenberg-Marquardt algorithm is provided by the "levmar" library by Manolis Lourakis [18].

To use the Levenberg-Marquardt algorithm, specify the "mfa-levmar" mode when invoking the MFAPipe executable, i.e., open your terminal, type `mfaPipe mfa-levmar`, and press the return key.

Syntax	Description	Required?	Default Value
`--input=FILE`	Input file (FluxJS document)	Yes	n/a
`--output=FILE`	Output file (zip archive)	Yes	n/a
`--max-iterations=INT`	Maximum iterations	No	100
`--scale-init-mu=NUM`	Scale factor for initial μ	No	1×10^{-3}
`--stop-norm-inf-jac-te=NUM`	Stopping thresholds for $\left\|\mathrm{J}^{\mathrm{T}} \cdot e\right\|_{\mathrm{inf}}$	No	1×10^{-17}
`--stop-norm2-dp=NUM`	Stopping thresholds for $\|\mathrm{Dp}\|_2$	No	1×10^{-17}
`--stop-norm2-e=NUM`	Stopping thresholds for $\|\mathrm{e}\|_2$	No	1×10^{-17}
`--delta=NUM`	Step for difference approximation of Jacobian	No	1×10^{-6}

Table 6.3: Command line arguments for "mfa-levmar" mode of MFAPipe executable.

Command line arguments for the "mfa-levmar" mode of the MFAPipe executable are documented in Table 6.3.

The directory tree for the resulting zip archive is as follows:

```
/
├── emu_reaction_network
│   └── <<n>>
│       ├── reaction.csv ..... Reaction network for size-n EMUs
│       └── stoichiometry.csv ..... Stoichiometric model for size-n EMUs
├── reaction_network
│   ├── reaction.csv ..... Chemical reaction network
│   └── stoichiometry.csv ..... Stoichiometric model
└── result
    ├── <<experiment>>
    │   ├── <<n>>
    │   │   ├── isotopomer_fraction.csv ..... Isotopomer fractions for size-n EMUs
    │   │   └── mass_fraction.csv ..... Mass fractions for size-n EMUs
    │   ├── contribution.csv ..... Contribution matrix
    │   ├── jacobian.csv ..... Jacobian matrix
    │   └── residual.csv ..... Weighted residuals
    ├── flux.csv ..... Fitted metabolic fluxes
    ├── flux_covariance.csv ..... Covariance matrix for fitted metabolic fluxes
    ├── info.csv ..... Information about fitting algorithm
    └── statistics.csv ..... Descriptive statistics
```

Note. The archive has subdirectories for each experiment and for each EMU size, denoted, respectively, "<<experiment>>" and "<<n>>".

Chapter 7

Examples

7.1 fba-mip (Flux Balance Analysis using Mixed Integer Programming)

In this tutorial, we'll work through an example of using MFAPipe to perform Flux Balance Analysis (FBA) using the Mixed Integer Programming (MIP) algorithm.

7.1.1 Model

The model for this tutorial is derived from the *"example network"* from the literature [19]. The model was selected because it is simple.

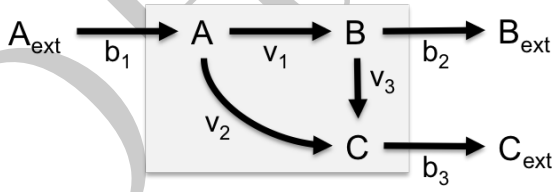

Figure 7.1: Depiction of intra- and extracellular compartments of chemical reaction network.

The intra- and extracellular compartments of the chemical reaction network, depicted in Figure 7.1, consist of the following:

- Three exchange chemical reactions, denoted v_1, v_2 and v_3;
- Three transport chemical reactions, denoted b_1, b_2 and b_3;
- Three intracellular metabolites, labeled "A", "B" and "C"; and,
- Three extracellular metabolites, labeled "A.ext", "B.ext" and "C.ext".

Chemical Reaction (Exchange)	Stoichiometry
v_1	A $\rightarrow$ B
v_2	A $\rightarrow$ C
v_3	B $\rightarrow$ C

Table 7.1: Stoichiometry for intracellular compartment of chemical reaction network.

Chemical Reaction (Transport)	Stoichiometry
b_1	A.ext $\rightarrow$ A
b_2	B $\rightarrow$ B.ext
b_3	C $\rightarrow$ C.ext

Table 7.2: Stoichiometry for extracellular compartment of chemical reaction network.

The intracellular compartment of the chemical reaction network has three exchange reactions (Table 7.1), and the extracellular compartment of the chemical reaction network has three transport reactions (Table 7.2). MFAPipe automatically infers the stoichiometric model for the chemical reaction network (Table 7.3).

7.1.2 Experimental Data

We are given the following information about the extracellular compartment:

1. The network substrate "A.ext" uptake rate is ≤ 10, i.e., that $b_1 \leq 10$.
2. The network products "B.ext" and "C.ext" are generated at a ratio of $3:1$, i.e., that $3 \times b_2 \leq b_3$.

The objective function for the simulation corresponds to biomass formation, i.e., maximization of the rates of production of the network products $z = b_2 + b_3$.

7.1.3 FluxJS Document

The FluxJS document for the model is given in Figure 7.2.

	v1.f	v2.f	v3.f	b1.f	b2.f	b3.f
A	-1	-1	0	1	0	0
B	1	0	-1	0	0	-1
C	0	1	1	0	-1	0
A.ext	0	0	0	-1	0	0
B.ext	0	0	0	0	0	1
C.ext	0	0	0	0	1	0

Table 7.3: Stoichiometric model for chemical reaction network.

```
{
  "model_metadata": {
    "http://purl.org/dc/terms/title": "Example network to illustrate flux balance analysis",
    "http://purl.org/dc/terms/source": "http://www.sciencedirect.com/science/article/pii/S0031942207002749"
  },
  "model": {
    "reaction_network": {
      "v1": { "type": "fba", "reaction": "A -> B" },
      "v2": { "type": "fba", "reaction": "A -> C" },
      "v3": { "type": "fba", "reaction": "B -> C" },
      "b1": { "type": "fba", "reaction": "A.ext -> A" },
      "b2": { "type": "fba", "reaction": "C -> C.ext" },
      "b3": { "type": "fba", "reaction": "B -> B.ext" }
    },
    "constraints": {
      "objective_function": {
        "max": "b2.f + b3.f"
      },
      "linear_constraints": {
        "b1.f": { "<=": 10 },
        "-3 b2.f + b3.f": { ">=": 0 }
      }
    }
  }
}
```

Figure 7.2: FluxJS document for model.

7.1.4 Invoking MFAPipe

Open your terminal, type in `mfaPipe fba-simplex --input=model.json --output=result.zip`, and press the return key. The input, a FluxJS document, is read from the file "model.json", and the output, a zip archive, is written to the file "result.zip".

7.1.5 Results

Metabolic Flux Variable	Reported Value	Maximized Value
v1.f	7.5	7.5
v2.f	2.5	2.5
v3.f	0	0
b1.f	10	10
b2.f	2.5	2.5
b3.f	7.5	7.5

Table 7.4: Fitted values of metabolic flux variables.

The results of the MIP algorithm, the fitted values of the metabolic flux variables (Table 7.4), is written to the file "./result/flux.csv".

Bibliography

[1] Maciek R Antoniewicz, Joanne K Kelleher, and Gregory Stephanopoulos. Determination of confidence intervals of metabolic fluxes estimated from stable isotope measurements. *Metabolic engineering*, 8(4):324–337, 2006.

[2] Maciek R Antoniewicz, Joanne K Kelleher, and Gregory Stephanopoulos. Elementary metabolite units (EMU): a novel framework for modeling isotopic distributions. *Metabolic engineering*, 9(1):68–86, 2007.

[3] Costas D Maranas and Ali R Zomorrodi. *Optimization Methods in Metabolic Networks*. John Wiley & Sons, 2016.

[4] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.

[5] Saul I Gass. *Linear programming*. Wiley Online Library, 1958.

[6] Adam M Feist and Bernhard Ø Palsson. The biomass objective function. *Current opinion in microbiology*, 13(3):344–349, 2010.

[7] Maciek R Antoniewicz. Methods and advances in metabolic flux analysis: a mini-review. *Journal of industrial microbiology & biotechnology*, 42(3):317–325, 2015.

[8] Craig R Malloy, A Dean Sherry, and F M Jeffrey. Evaluation of carbon flux and substrate selection through alternate pathways involving the citric acid cycle of the heart by ^{13}C NMR spectroscopy. *Journal of Biological Chemistry*, 263(15):6964–6971, 1988.

[9] Wolfgang Wiechert, Michael Möllney, Nichole Isermann, Michael Wurzel, and Albert A de Graaf. Bidirectional reaction steps in metabolic networks: III. Explicit solution and analysis of isotopomer labeling systems. *Biotechnology and Bioengineering*, 66(2):69–85, 1999.

[10] Dmitri Kuzmin and Jari Hämäläinen. Finite element methods for computational fluid dynamics: A practical guide. *SIAM Review*, 57(4):642, December 2015.

[11] Louis B Rall. The arithmetic of differentiation. *Mathematics Magazine*, 59(5):275–282, 1986.

[12] Brian Gough. *GNU Scientific Library Reference Manual - Third Edition*. Network Theory Ltd., 3rd edition, 2009.

[13] Frank J Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[14] Jingbo Wang, Wai Wan Tsang, and George Marsaglia. Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, 8(18), 2003.

[15] George B Dantzig, Alex Orden, Philip Wolfe, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

[16] Ralph Gomory. An algorithm for the mixed integer problem. Technical report, DTIC Document, 1960.

[17] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[18] Manolis I. A. Lourakis. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. *Foundation of Research and Technology*, 4:1–6, 2005.

[19] Rigoberto Rios-Estepa and Bernd Markus Lange. Experimental and mathematical approaches to modeling plant metabolic networks. *Phytochemistry*, 68(16):2351–2374, 2007.