

# KINETICS ON NETWORKS OF ENZYMATIC REACTIONS

FRANCISCO BLANCO-SILVA

Department of Mathematics, Purdue University

ABSTRACT. We will present an explanation of the notion of **network** in Biochemistry, related to the solution of kinetics in enzymatic reactions. We have developed a series of python utilities designed to solve dynamical systems of “biochemical-aware” differential equations related to such networks. Some tests are performed over known metabolic pathways. The main idea is based in the paper [MK], which gives a very clear introduction to the problem.

## INTRODUCTION

An excellent definition of biochemical network can be read in [MK]:

The notion of a network is fundamentally a mathematical one: a set of specific things connected by a set of quantitative relations. In biology, the term is sometimes used to refer to genes connected by their regulatory relationships; or metabolites, connected by enzymatic reactions; or receptors, connected by messenger molecules; or proteins, connected by noncovalent binding reactions. All of these examples are special cases of a biochemical network.

The most important properties of networks (on any of its forms) are **structure** and **function**. Let us offer a rough notion of its intuitive meaning now, together with a few examples:

**Structure of a Biochemical Network.** Roughly speaking, the **structure** of a network is its topology. This can be studied through the point of view of Graph Theory, for instance, and obviously the results will improve our computational treatment of the problems related to the given networks in a more efficient way (sometimes cutting the complexity by huge factors!). An excellent exposition can be found in [SP], in which general biochemical networks are translated into the realm of Linear Algebra.

Another example; in [McS], an approach to structure is carried away with “Circuit Simulation”: Transitions are explained in terms of signal timing, and time delays. As it can be easily inferred, a “Logic Theory” point of view is implicit over this paper. This one is highly influenced by [K], from more than two decades before.

**Function of a Biochemical Network.** A very abstract and broad term; it depends very closely on the kind of network we are dealing with. Basically they are either dynamical or chemical properties. These can be efficiently handled with classical dynamical systems, together with basic chemistry and of course every little dirty trick from linear algebra. Quite often we will end up with manipulation of huge matrices (if we are lucky sparse ones), and in many situations the use of parallel computing to solve our problems.

---

This problem was introduced to the author by Prof. Toni Kazic in a seminar series on Scientific Computing and Computational Biology during the IMA Summer Workshop held in the University of Kentucky in July’02. The testing of the method with metabolic pathways, as well as all the information related to most of the biochemical aspects of them were provided by M.D. Pilar Huecas.

We have found several papers illustrating the computational manipulation of networks and its applications; among them, the already cited [MK], and the complex (but beautifully explained and illustrated) [F-R].

### ENZYMATIC REACTIONS

In order to exemplify this idea, we have chosen to study the dynamics of a biochemical network of enzymatic reactions: We are given, from a database, a set of chemical reactions involving related enzymes. Each of these take certain amounts of one or several compounds (substrate), and after the reaction, output one or several reactants (notice that the reaction can proceed in either way, depending mainly on the concentration of the substrate and reactant). Notice also that both input and output of those reactions might be used by different enzymes in another related reactions.

In this way, if we could “freeze” the reactions at any given time, and measure the concentration of each component being used in this network, we would be able to answer questions of the kind: “Which component decays faster?”, “when will the reactions stop?” (provided the enzymes don’t “die”), “what are the initial concentrations that produce certain phenomena?”, etc.

All these can be answered from a “Dynamical Systems” point of view. Each of the chemical reactions offer a special differential equation involving not only the concentration of the components, but also the concentration of the enzyme and a few more parameters (to be explained below). This differential equations are constrained by the fact that none of the concentrations (=variables) can drop below zero. These are just a small example of the so-called **differential-algebraic** systems of equations. A survey on this topic can be read in [BHP].

It is clear now the object of this part of the project: Given a network of chemical reactions produced by enzymes, included initial concentrations of components and enzymes, modelize the dynamical behaviour using a differential-algebraic system of equations. The outputs will be, for instance, graphs of the change in the concentration of substrate or reactant A with respect to time.

Of great interest is also the procedure of finding the network itself (hard problem!). For instance, given a certain protein A, find all known enzymatic reactions related to the manipulation of this protein. Arrange them in a network, and study its dynamical properties. It is clear that the first part involves taking information from a database, and a second part involves manipulating that information so we get a network.

Let us proceed with a description of the mathematics below the problem, and the approach followed to get a solution:

#### Enzyme Kinetics.

This section is compiled from [G-G]; actually, most of it is a modified quotation of section 11.2 there.

Chemical kinetics is the study of the rates of chemical reactions. Consider a simple reaction  $A \rightarrow P$ ; although we treat this reaction as a simple one-step conversion of  $A$  to  $P$ , it more likely occurs through a sequence of elementary reactions, each of which is a simple molecular process, as in  $A \rightarrow I \rightarrow J \rightarrow P$ , where  $I$  and  $J$  represent intermediates in the reaction. Precise description of all of the elementary reactions in a process is necessary to define the overall reaction mechanism for  $A \rightarrow P$ . Let us assume that  $A \rightarrow P$  is an elementary spontaneous and essentially irreversible reaction. The **velocity** of the reaction is the amount of  $P$  formed or the amount of  $A$  consumed per time; that is,

$$v = \frac{d[P]}{dt} = -\frac{d[A]}{dt}$$

The mathematical relationship between reaction rate and concentration of reactants is the **rate law**. For this simple case, the rate law is

$$v = -\frac{d[A]}{dt} = k[A],$$

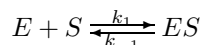
for some constant value  $k > 0$ .

Consider now a more complex reaction where two molecules must react to yield products:  $A+B \rightarrow P+Q$ . Assuming this is an elementary reaction, the velocity can be determined from the rate of disappearance of either  $A$  or  $B$ , or the rate of appearance of  $P$  or  $Q$ :

$$v = -\frac{d[A]}{dt} = -\frac{d[B]}{dt} = \frac{d[P]}{dt} = \frac{d[Q]}{dt}$$

The rate law is  $v = k[A][B]$ . In the general case,  $xA + yB \rightarrow \text{products}$ , the velocity obeys the general rate law  $v = k[A]^x[B]^y$ .

Unfortunately, for the case we are studying, the reactions are not elementary; in 1913, L. Michaelis and M. L. Menten proposed a general theory of enzyme action based on the assumption that the enzyme  $E$  and its substrate  $S$  associate reversibly to form an enzyme-substrate  $ES$ :



This association/dissociation is assumed to be in rapid equilibrium, and  $K_S$  is the **enzyme-substrate dissociation constant**. At equilibrium,

$$k_{-1}[ES] = k_1[E][S]$$

and

$$K_S = \frac{[E][S]}{[ES]} = \frac{k_{-1}}{k_1}$$

Product  $P$  is formed in a second step when  $ES$  breaks down to yield  $E + P$ .



$E$  is then free to interact with another molecule of  $S$ .

The interpretations of Michaelis and Menten were refined and extended in 1925 by Briggs and Haldane, by assuming the concentration of the enzyme-substrate complex  $ES$  quickly reaches a constant value in such a dynamic system. That is,  $ES$  is formed as rapidly from  $E + S$  as it disappears by its two possible fates: dissociation to regenerate  $E + S$ , and reaction to form  $E + P$ . This assumption is termed the **steady-state assumption** and is expressed as

$$\frac{d[ES]}{dt} = 0$$

A more complex reasoning involving the maximum velocity attained for such reactions, and the Michaelis constant  $K_m = (k_{-1} + k_2)/k_1$ , leads us to more involved but at the same time more accurate differential equations modeling the behaviour of our enzymatic reactions. For instance, if  $[E_T] = [E] + [ES]$  denotes the total amount of enzyme (which is fixed), then we have

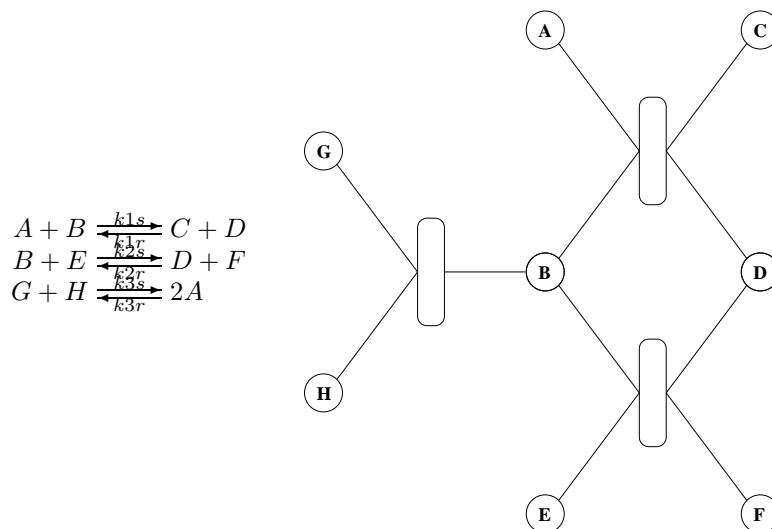
$$v = \frac{k_2[E_T][S]}{K_m + [S]}$$

The product  $k_2[E_T]$  has a special meaning: when  $[S]$  is high enough to saturate all of the enzyme, the velocity of the reaction is maximal, and at that moment the amount of  $[ES]$  is equal to the total enzyme concentration  $[E_T]$ . We have thus a maximum velocity of  $V_{\max} = k_2[E_T]$ , and we can also write

$$v = \frac{V_{\max}[S]}{k_m + [S]}$$

### Network Kynetics.

Now consider the following simple example of a network of several related enzymatic reactions. Assume for simplicity that each reaction is elementary, spontaneous but by no means irreversible (enzymatic reactions seldom are):



We have, for each compound, the following rates (notice we have denoted with  $X$  instead of  $[X]$  the concentration of the compounds  $X$  for ease of notation):

$$\begin{aligned}
 \frac{dA}{dt} &= k_{1r} \cdot CD - k_{1s} \cdot AB \\
 \frac{dB}{dt} &= k_{1r} \cdot CD - k_{1s} \cdot AB + k_{2r} \cdot DF - k_{2s} \cdot BE - k_{3r} \cdot B^2 + k_{3s} \cdot GH \\
 \frac{dC}{dt} &= -k_{1r} \cdot CD + k_{1s} \cdot AB \\
 \frac{dD}{dt} &= -k_{1r} \cdot CD + k_{1s} \cdot AB - k_{2r} \cdot DF + k_{2s} \cdot BE \\
 \frac{dE}{dt} &= k_{2r} \cdot BE - k_{2s} \cdot DF \\
 \frac{dF}{dt} &= -k_{2r} \cdot BE + k_{2s} \cdot DF \\
 \frac{dG}{dt} &= k_{2r} \cdot GH - k_{2s} \cdot B^2 \\
 \frac{dH}{dt} &= k_{2r} \cdot GH - k_{2s} \cdot B^2
 \end{aligned}$$

Let us explain how we got each equation; notice for example the first one: it expresses how the compound  $A$  is affected by the network of enzymatic reactions. As it can be inferred from the picture,  $A$  only gets affected by the first reaction, therefore the rate of the concentration of  $A$  with respect to time is directly proportional to  $[A][B]$  (with positive rate  $k_{1s}$ , since those decrease and affect negatively to the growth) and directly proportional to  $[C][D]$  (with negative  $k_{1r}$ , since they increase).

Notice also that compounds  $B$  and  $D$  are shared by several enzymes (3 and 2 respectively), so this affects in a similar way at the rate of change of their concentration with respect to time:

$$\frac{dB}{dt} = \underbrace{k_{1r} \cdot CD - k_{1s} \cdot AB}_{\text{first enzyme}} + \underbrace{k_{2r} \cdot DF - k_{2s} \cdot BE}_{\text{second enzyme}} + \underbrace{k_{3s} \cdot GH - k_{3r} \cdot B^2}_{\text{third enzyme}}$$

### The ENZYME database.

ENZYME is a repository of information relative to the nomenclature of enzymes. It is primarily based on the recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (IUBMB) and it describes each type of characterized enzyme for which an EC (Enzyme Commission) number has been provided. It contains the following data:

- (1) EC number.

- (2) Recommended name.
- (3) Alternative names (if any).
- (4) Catalytic activity.
- (5) Cofactors (if any).
- (6) Pointers to the SWISS-PROT entrie(s) that correspond to the enzyme (if any).
- (7) Pointers to disease(s) associated with a deficiency of the enzyme (if any).

The main source for the data in the ENZYME database comes from recommendations of the Nomenclature Committee of the IUBMB [1]. A minor part of the data has been extracted from the literature. We have downloaded a copy of the database from the site indicated in the previous citation to our folders. Each entry in the database is composed of lines. Different types of lines, each with its own format, are used to record the various types of data which make up the entry. The general structure of a line is the following:

<u>Char.Posn.</u>	<u>Contents</u>
1 to 2	Two-character line code. Indicates the type of information contained in the line.
3 to 5	Blank
6 up to 78	Data

The currently used line types, along with their respective line codes, are listed below:

ID	Identification	Begins each entry; 1 per entry
DE	Description (official name)	$\geq 1$ per entry
AN	Alternate name(s)	$\geq 0$ per entry
CA	Catalytic activity	$\geq 0$ per entry
CF	Cofactor(s)	$\geq 0$ per entry
CC	Comments	$\geq 0$ per entry
DI	Disease(s) associated with the enzyme	$\geq 0$ per entry
PR	Cross-references to PROSITE	$\geq 0$ per entry
DR	Cross-references to SWISS-PROT	$\geq 0$ per entry
//	Termination line	Ends each entry; 1 per entry

Some entries do not contain all of the line types, and some line types occur many times in a single entry. Each entry must begin with an identification line (ID) and end with a terminator line (//).

Of course, for the practical part of this project, not all the information from the database is needed; basically we get along with the ID and CA entries; therefore, we have created a python dictionary that assigns to each EC entry its catalytic activity (the reaction); it is easier and faster using this dictionary than the database file ENZYME.DAT. We have also created utilities to search this database, output and handle different information:

`Enzyme[string]`. This is the dictionary performs a call to the dictionary. It takes as input the ec number of the requested reaction (as a string), and offers the corresponding enzymatic reaction:

```
>>> Enzyme['1.15.1.1']
'2 peroxide radical + 2 H(+) = O(2) + H(2)O(2)'
```

`LookFor(string)`. This module performs a search on the dictionary `Enzyme`, and collects in a list each of the ec numbers of those reactions in which the compounds containing the *string* are present either as substrate or product. There is no need for wildcards; in the example below we input the

string 'galactose'. The function will look for \*galactose and galactose\*.

```
>>> set=LookFor('galactose')
>>> set
['1.1.1.120', '2.7.7.12', '2.7.7.10', '2.7.1.52', '2.4.1.134', '1.1.1.
48', '3.2.1.22', '3.2.1.23', '1.1.3.9', '2.4.1.37', '2.4.1.38', '2.7.8
.6', '2.4.1.23', '2.4.1.22', '2.4.1.90', '2.7.8.18', '5.1.3.18', '5.3.
1.26', '2.4.1.50', '2.4.1.47', '2.4.1.46', '2.4.1.45', '2.4.1.44', '2.
.94.1.40', '2.4.1.74', '3.2.1.108', '2.4.16', '2.4.1.124', '2.4.1.123',
, '2.4.1.122', '3.1.6.4', '2.6.1.59', '2.4.1.60', '2.4.1.62', '2.4.1.13
3', '2.4.1.87', '2.4.1.86', '2.4.1.137', '2.5.1.5', '3.2.1.46', '3.2.1
.47', '1.1.1.186', '2.4.1.156', '2.4.1.151', '3.2.1.85', '3.2.1.83', '2.
4.1.167', '5.1.3.2', '2.7.1.6', '2.4.1.179', '2.7.7.32', '2.4.1.205']
```

EquationParser(*string*). The dictionary Enzyme offers the information in ENZYME format, while the output of this function breaks the equation into substrate and reactant, and also indicates if several molecules are present on each side; this nice feature make much easier the manipulation “chemical reaction” → “differential equation”:

```
>>> EquationParser('1.15.1.1')
('1.15.1.1', ['peroxide radical', 'peroxide radical', 'H(+)', 'H(+)'],
['O(2)', 'H(2)O(2)'])
```

CollectCompounds(*list*). Given a network of parsed reactions —as a list of strings, this module collects all the compounds on those reactions, displaying them in lexicographical order: numbers before in natural quasi-order (i.e.  $1 < 11 < 2 < 211 < 3 < \dots$ ), then letters in alphabetical.

```
>>> network=[]
>>> for ec in set: network.append(EquationParser(ec))
...
>>> compounds=CollectCompounds(network)
>>> for cmp in compounds: print cmp
...
1,2-diacylglycerol
1,6-beta-D-galactosylgalactogen
1-(beta-D-galactosyl)-2-(2-hydroxyacyl)sphingosine
1-0-alpha-D-galactosyl-D-myo-inositol
2-(2-hydroxyacyl)sphingosine
...
```

## DYNAMICAL SYSTEMS OF NETWORKS OF ENZYMATIC REACTIONS

We have all the ingredients to pose the problem: Given a network of enzymatic reactions,

- (1) Find a (biochemically-correct) system of differential equations expressing the rate of change of each of the compounds with respect to time.
- (2) Solve the previous system.
- (3) Visualization: plot graphs with, for example, the concentration of each compound with respect to time.

Let us explain in detail each of the subproblems, explaining the approach we followed to get a solution and showing a few examples of how our code works in those cases.



```

import Numeric
def function(x,k):
    y=Numeric.zeros(106,Numeric.Float)
    y[0]=+(-k[43,0]*x[61]*x[0]+k[43,1]*x[57]*x[7])*Numeric.sign(x[61]*x[0])
    y[1]=-(-k[46,0]*x[61]*x[85]+k[46,1]*x[57]*x[1])*Numeric.sign(x[61]*x[85])
    y[2]=-(-k[19,0]*x[61]*x[4]+k[19,1]*x[57]*x[2])*Numeric.sign(x[61]*x[4])
    ...
    y[103]=+(-k[12,0]*x[61]*x[103]+k[12,1]*x[57]*x[101])*Numeric.sign(x[61]*x[103])
    y[104]=+(-k[3,0]*x[61]*x[104]+k[3,1]*x[57]*x[11])*Numeric.sign(x[61]*x[104])
    y[105]=+(-k[11,0]*x[61]*x[105]+k[11,1]*x[63]*x[71])*Numeric.sign(x[61]*x[105])
return y

```

### Solution of the Dinamical System. Tests.

In order to solve this system numerically, we have chosen a simple Runge-Kutta of fourth order, for both its simplicity of coding and robustness. A careful reading over our module `SolveNetwork` reveals the coding philosophy:

- (1) It accepts as input a **network** of enzymatic reactions (as a set of parsed equations), an array of initial conditions (`InitialConditions`), the dimension of which must match the dimension of `cmp=CollectCompounds(network)`; an array of enzymatic reactions constants (`K`) with dimensions `len(network)×2`—the first column offers the substrate rate, while the second offers the reactant rate; initial and final times (`t0` and `tf`); and number of steps to be carried away by the Runge-Kutta procedure (`steps`).
- (2) The numerical solution is stored in a new array (`y`) of dimesions `len(cmp)×steps`; this array is the returned value of the function.
- (3) But `SolveNetwork` also writes to disk the graph information of each concentration, in `GNUplot` format. For each compound it creates the file `cmp.k`, where `k` ranges from 0 to `len(cmp)−1`.

We have tested our code against small tricky networks prepared to have this effect, and it has performed consecuently, stopping those reactions in which substrate dissapeared, affecting the ones related to the previous ones, but keeping untouched the non-affected ones.

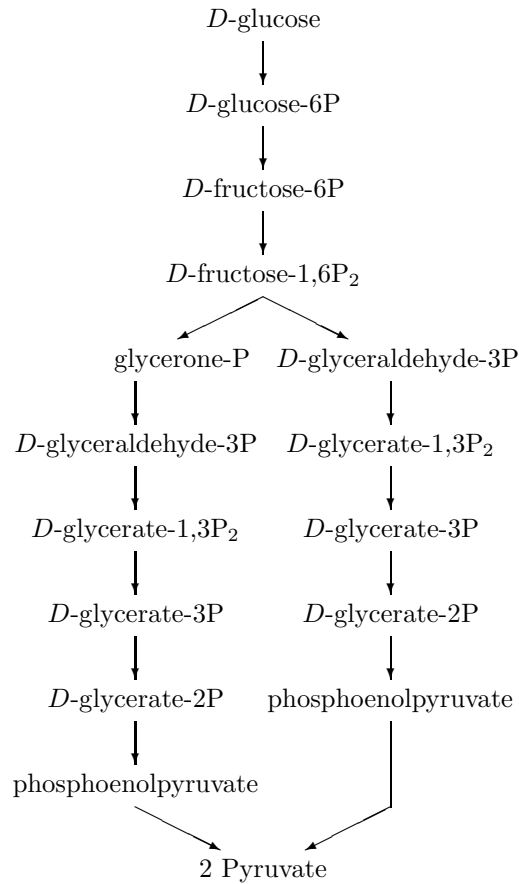
A further test was performed; we listed all compounds present only in substrates, and realized that their concentrations where properly decreasing at fast pace. We listed also compounds only in reactants, and noticed that they kept increasing, except when a zero event was reached in a substrate related to them. These checks are a good exponent of the well behaviour of our code. A deeper study with a test example will be explained below.

### Visualization and Test Case: Embden-Meyerhof glycolitic pathway.

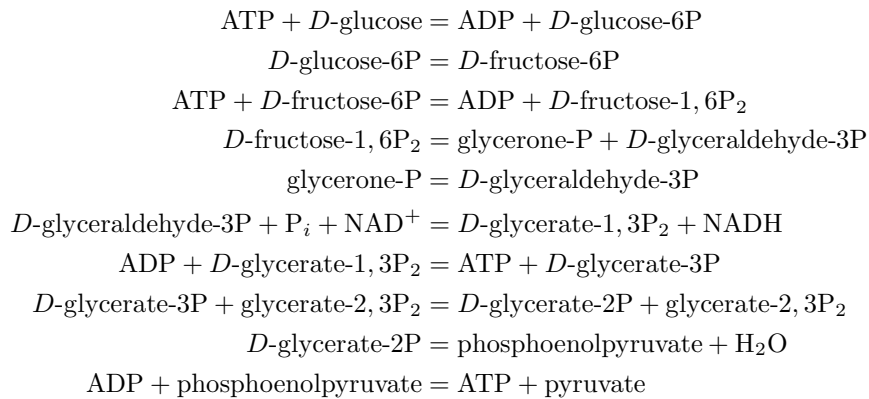
As our visualization tool, we decided to use `GNUplot`; writing a file that can be interpreted by this program is an easy task. Besides, Python has a module that helps interact with this program: `gnuplot.py`. We have written a utility that allows us to plot the concentration of each compound with respect to time. The module `PlotConcentration` takes as input the solution array offered by `SolveNetwork`, the list of compounds of the network, and a key word:

- (1) If `key` is the string `'all'`, then all the concentrations are displayed in order.
- (2) One can choose any of the compounds to be singled out, by calling it in `key`. For instance, the command `PlotConcentration(y,c,'ATP')` will plot the concentration of the ATP in the given network.
- (3) Sometimes writing the complete name of the compound is a tedious matter; one can choose `key` to be the index of the required compound. For instance, `PlotConcentration(y,c,7)` will plot the concentration of `c[7]`.

We have tested our code with the Embden-Meyerhof glycolitic pathway, in which glucose is converted into energy. This pathway consists on two phases: in the first one, glucose is broken down to two molecules of glyceraldehyde-3-phosphate. In the second phase, those two molecules are converted to two molecules of pyruvate.



The enzymatic reactions are listed below:



It is always annoying to use different names for the very same compound; in this case, each of the three sources in which we obtained the information about this pathway used their own notation, what makes researching tedious quite often. As a means of trying to unify the three of them, we have followed the recommendations of ExpaSY, although for coding purposes it is clear we cannot choose, since we have to satisfy the ENZYME database requirements. Notice the difference; this is the output of the previous reactions from the ENZYME database:

2.7.1.2: 'ATP + D-glucose = ADP + D-glucose 6-phosphate'  
 5.3.1.9: 'D-glucose 6-phosphate = D-fructose 6-phosphate'  
 2.7.1.11: 'ATP + D-fructose 6-phosphate = ADP  
 + D-fructose 1,6-bisphosphate'  
 4.1.2.13: 'D-fructose 1,6-bisphosphate = glyceraldehyde 3-phosphate  
 + D-glyceraldehyde 3-phosphate'  
 5.3.1.1: 'D-glyceraldehyde 3-phosphate = glyceraldehyde 3-phosphate'  
 1.2.1.12: 'D-glyceraldehyde 3-phosphate + phosphate + NAD(+)  
 = 3-phospho-D-glyceroyl phosphate + NADH'  
 2.7.2.3: 'ATP + 3-phospho-D-glycerate = ADP  
 + 3-phospho-D-glyceroyl phosphate'  
 5.4.2.1: '2-phospho-D-glycerate + 2,3-diphosphoglycerate  
 = 3-phospho-D-glycerate + 2,3-diphosphoglycerate'  
 4.2.1.11: '2-phospho-D-glycerate = phosphoenolpyruvate + H(2)O'  
 2.7.1.40: 'ATP + pyruvate = ADP + phosphoenolpyruvate'

Notice also that in the ENZYME reactions **ec:5.3.1.1**, **ec:2.7.2.3**, **ec:5.4.2.1**, and **ec:2.7.1.40**, the substrates and products are not in the same order as in the pathway; an adjustment is therefore necessary prior to the manipulation of the enzymatic rate constants  $k_s$  and  $k_r$ .

Microorganisms, plants, and animals (including humans) carry out the 10 reactions of glycolysis in more or less similar fashion, although the rates of the individual reactions and the means by which they are regulated differ from species to species. The most significant difference among species, however, is the way in which the product pyruvate is utilized, but that doesn't concern this case.

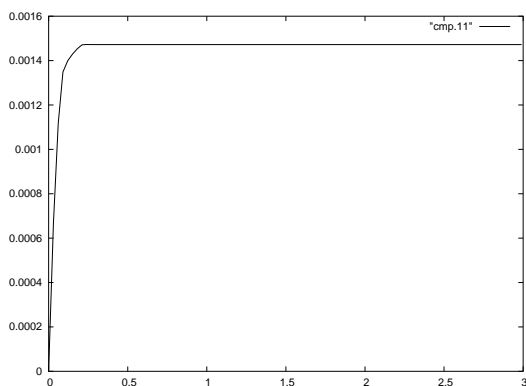
We obtained both the values of the constants  $K_m$ , and a set of standard steady-state concentrations of all the components from [G-G]. They refer there to those values in humans at conditions of pH 7 and 37°C (the normal body temperature). For example, in erythrocytes, the concentration of ATP is typically 1850  $\mu M$ , the concentration of ADP is 145  $\mu M$ , and the value of  $K_m$  for the hexokinase (ec: 2.7.1.2) is approximately 0.1  $mM$ .

The values  $K_m$  give us freedom to choose just one of the required  $k_s$  and  $k_r$  (see above in section **Enzyme Kinetics**). A random generator was then used to get the values of the constants  $k_r$ , and the corresponding values  $k_s$  are then inferred; we have to admit that we are completely clueless so as which range to choose for those  $k_r$ , and how this choice affects the output of our solver.

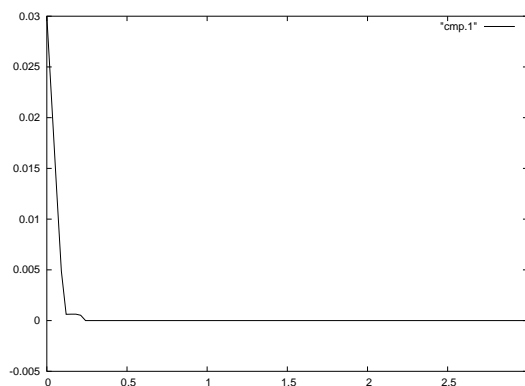
compound	init. conc. ( $nM$ )
2,3-diphosphoglycerate	4.000
2-phospho-D-glycerate	0.030
3-phospho-D-glycerate	0.120
3-phospho-D-glyceroylphosphate	5.000
ADP	0.140
ATP	1.850
D-fructose1,6-bisphosphate	0.031
D-fructose6-phosphate	0.014
D-glucose	5.000
D-glucose6-phosphate	0.083
D-glyceraldehyde3-phosphate	0.019
H(2)O	0.000
NAD(+)	5.000
NADH	0.000
glyceronephosphate	0.140
phosphate	1.000
phosphoenolpyruvate	0.023
pyruvate	0.000

$ec \#$	$k_r(nM)$	$k_s(nM)$
2.7.1.2	0.05896235	0.94289084
5.3.1.9	0.46270551	0.18425005
2.7.1.11	0.99509258	0.77118606
4.1.2.13	0.73096511	0.71323266
5.3.1.1	0.30008683	0.53397941
1.2.1.12	-0.36994987	-0.1655352
2.7.2.3	-0.53042004	-0.5017144
5.4.2.1	-0.27204919	-0.59356926
4.2.1.11	0.85654551	0.16472936
2.7.1.40	-0.65080809	-0.28294028

Below are the graphs of the concentrations of those compounds in a simulated experiment of 3 seconds (and 100 steps per second) starting with the previous initial conditions.

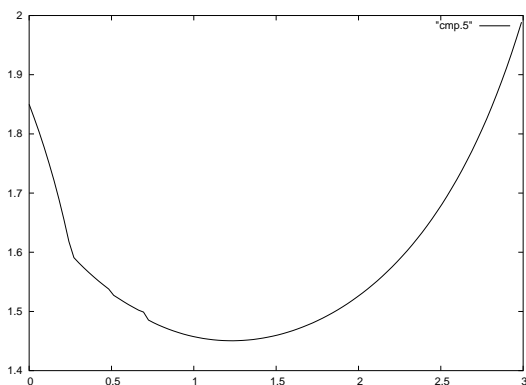


**Fig.1** Concentration of water for  $t$  between 0 and 3 seconds

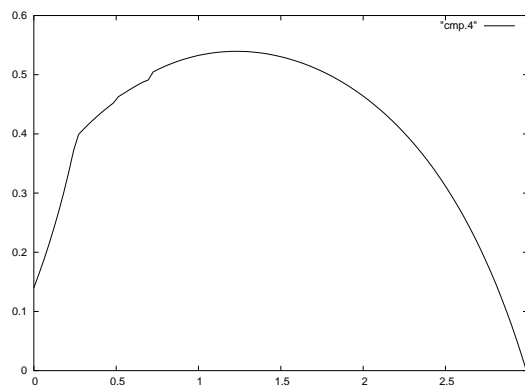


**Fig.2** Concentration of 2-phospho-D-glycerate for  $t$  between 0 and 3 seconds

The enzymatic reaction enolase (ec: 4.2.1.11) catalyzes the formation of phosphoenolpyruvate from 2-phosphoglycerate. The reaction in essence involves a dehydration (namely, the removal of one water molecule) to form the enol structure. Notice that initially there was no water in the network; it is created at a very fast rate by this reaction, and once the substrates vanish, the concentration of  $H_2O$  remains constant through the rest of the process.



**Fig.3** Concentration of ATP for  $t$  between 0 and 3 seconds

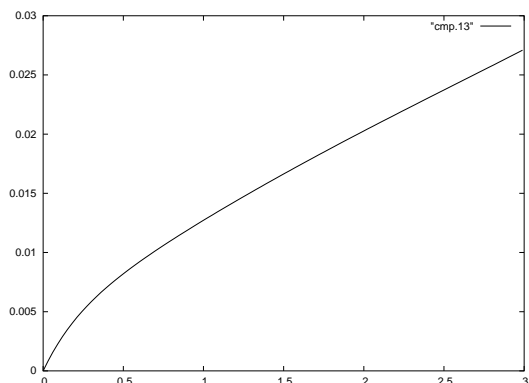


**Fig.4** Concentration of ADP for  $t$  between 0 and 3 seconds

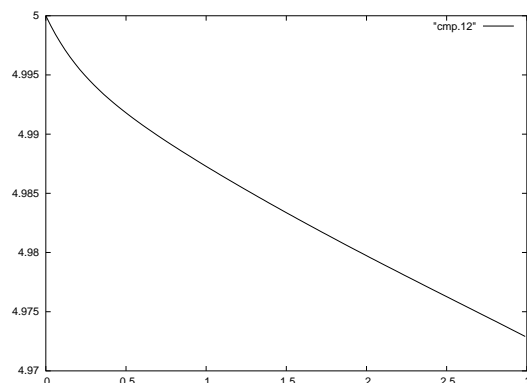
The glycolysis pathway requires two priming ATP molecules to start the sequence of reactions. This ATP is synthesized using the metabolic free energy contained in the glucose molecule. After

the first stage of glycolysis, the energy released is used in the second phase to synthesize more ATP. Altogether, four new molecules are produced; therefore, a net yield of 2 ATPs per glucose is realized. Notice how that effect is clear from the graphs of the concentrations of both ATP and ADP with respect to time.

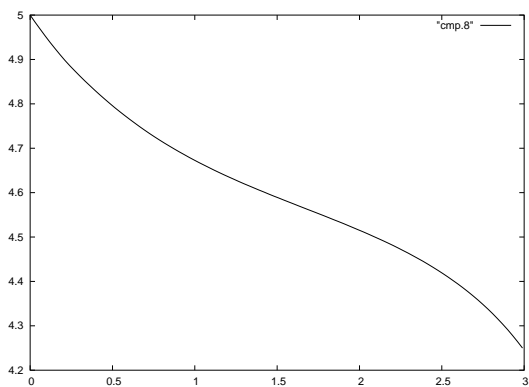
In addition to ATP, the products of glycolysis are NADH and pyruvate. Notice in the graphs how the concentrations of  $\text{NAD}^+$  and NADH are related in this way, by the enzymatic reaction phosphoglycerate kinase (ec: 1.2.1.12).



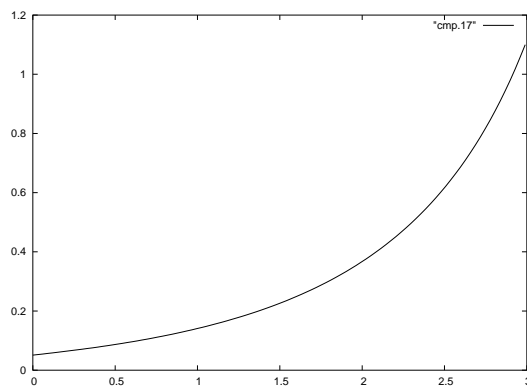
**Fig.5** Concentration of NADH  
for  $t$  between 0 and 3 seconds



**Fig.6** Concentration of  $\text{NAD}^+$   
for  $t$  between 0 and 3 seconds



**Fig.7** Concentration of D-glucose  
for  $t$  between 0 and 3 seconds



**Fig.8** Concentration of pyruvate  
for  $t$  between 0 and 3 seconds

#### FURTHER TESTS. CONCLUSION

We designed a second experiment, this time with very little biological meaning, but complex enough to test the speed and robustness of our code.

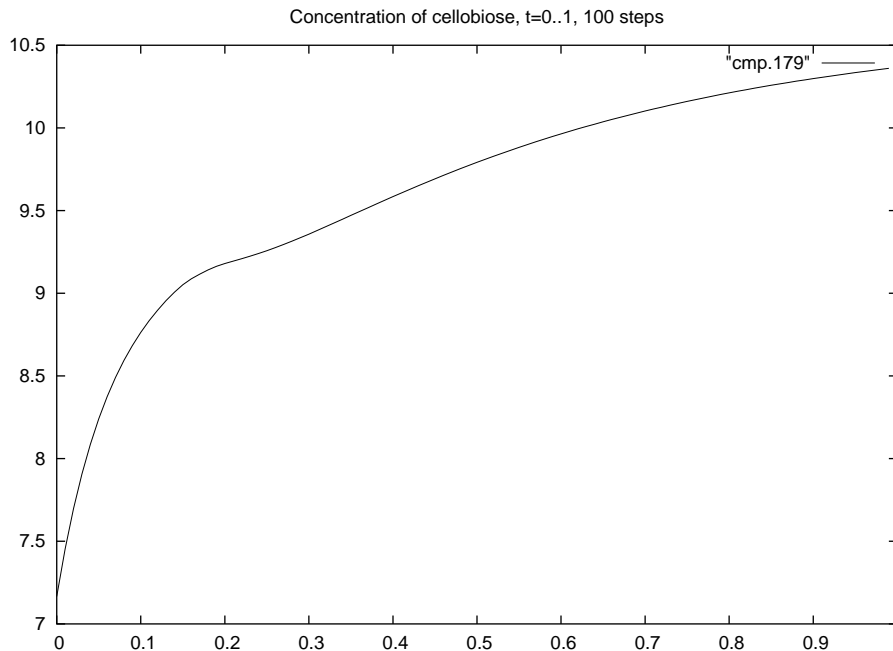
#### “Too-sweet-for-my-test” pathway.

We perform a search on **Enzyme** of all those reactions related to compounds with names containing the string 'glucose'. We keep all those reactions in the list **set**. This fake pathway consists on 157 reactions, and a total of 281 compounds. We give completely random concentrations for those compounds, and also for each of the enzymatic rates  $k_r$  and  $k_s$ , and solve the corresponding system. The following was run in our laptop, with a

```

>>> set=LookFor('glucose')
>>> network=[]
>>> for reaction in set: network.append(EquationParser(reaction))
...
>>> compounds=CollectCompounds(network)
>>> IC=Numeric.zeros(len(compounds),Numeric.Float)
>>> for k in len(compounds): IC[k]=10*random.random()
>>> K=Numeric.zeros((len(set),2),Numeric.Float)
>>> for k in range(len(set)):
...   for j in range(2): K[k,j]=random.random()
>>> a=time.time()
>>> solution=SolveNetwork(network,IC,K,0,1,100)
>>> time.time()-a
32.942000031471252
>>> Plot(solution,compounds,179)

```



### Conclusion.

The method proposed in this report solves coherently the given network: the concentrations of all the compounds, as computed here, have the expected behaviour. Notice that the numerical solver was based on one of the simpler choices of differential equations modelizing the kinetics of enzymatic equations. In case we desire to use more complicated models, it is enough to change two lines of code (in `SolveNetwork`). Another serious improvement can be made in the numerical method of solution; we have used a simple Runge-Kutta with four coefficients, but for some stiffer systems of differential equations this method may prove unstable. The tools developed in [BHP] are a good exposition of how to improve in this sense.

### REFERENCES

- [BHP] Peter N. Brown, Alan C. Hindmarsh and Linda R. Petzold, *Consistent Initial Condition Calculation for Differential-Algebraic Systems (preprint)* (1996).
- [F-R] Micheline Fromont-Racine, Andrew E. Mayes, Adeline Brunet-Simon, Jean-Christophe Rain, Alan Colley, Ian Dix, Laurence Decourty, Nicolas Joly, Florence Ricard, Jean D. Beggs and Pierre Legrain, *Genome-wide protein interaction screen reveal functional networks involving SM-like proteins*, *Yeast* (2000).

- [G-G] Reginald H. Garrett and Charles M. Grisham, *Biochemistry*, Saunders College Publishing, Harcourt Brace College Publishers, 1995.
- [I] Enzyme Nomenclature, *NC-IUBMB*, Academic Press, New York, 1992, (see <http://www.chem.qmul.ac.uk/iubmb/enzyme/>).
- [K] Stuart Kauffman, *The Large Scale Structure and Dynamics of Gene Control Circuits*, J. Theor. Biol. (1974).
- [McS] Harley H. McAdams, Lucy Shapiro, *Circuit Simulation of Genetic Networks*, SCIENCE (Aug 4<sup>th</sup>, 1995).
- [MK] Avanthi Mummaneni and Toni Kazic, *The Architectural Dynamics of Cellular Biochemistry and Molecular Biology*, Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics. (2002).
- [SP] Christophe H. Schilling and Bernhard O. Palsson, *The underlying pathway structure of biochemical reaction networks*, Proc. Natl. Acad. Sci. USA (1998).

150 NORTH UNIVERSITY ST. WEST LAFAYETTE IN 47907-2067

*E-mail address:* [fbs@math.purdue.edu](mailto:fbs@math.purdue.edu)