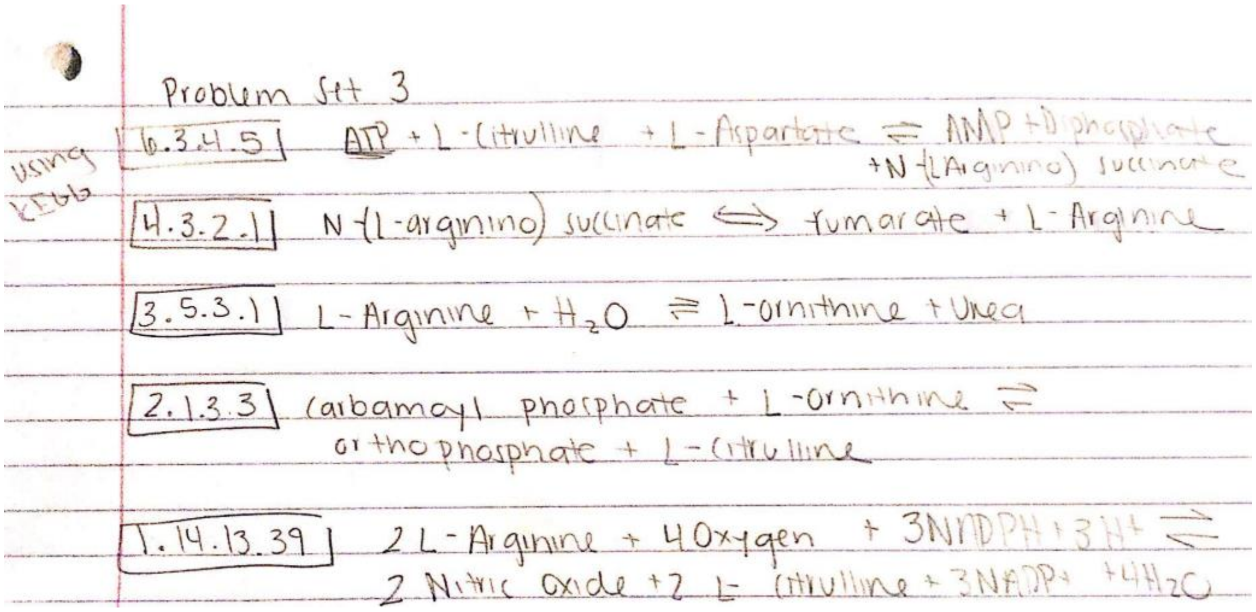
Nicole Newberger

CHEME 5440 Problem Set 3

Part A:

The stoichiometric matrix with headers can be found in the file “5440 PS3.xlsx” Reactions (v’s) are listed in the first 6 columns (reversible reaction, v5, split into 2) and the remaining columns are the exchange reactions.

Reactions taken from KEGG to build stoichiometric matrix:



Problem Set 3

using KEGG

$\boxed{6.3.4.5}$ ATP + L-Citrulline + L-Aspartate $\rightleftharpoons$ AMP + Diphosphate + N-(L-Arginino) succinate

$\boxed{4.3.2.1}$ N-(L-arginino) succinate $\Leftrightarrow$ fumarate + L-Arginine

$\boxed{3.5.3.1}$ L-Arginine + H_2O $\rightleftharpoons$ L-ornithine + Urea

$\boxed{2.1.3.3}$ Carbamoyl phosphate + L-ornithine $\rightleftharpoons$ orthophosphate + L-Citrulline

$\boxed{1.14.13.39}$ 2 L-Arginine + 4 Oxygen + 3NADPH + $3H^+$ $\rightleftharpoons$ 2 Nitric oxide + 2 L-Citrulline + $3NADP^+$ + $4H_2O$

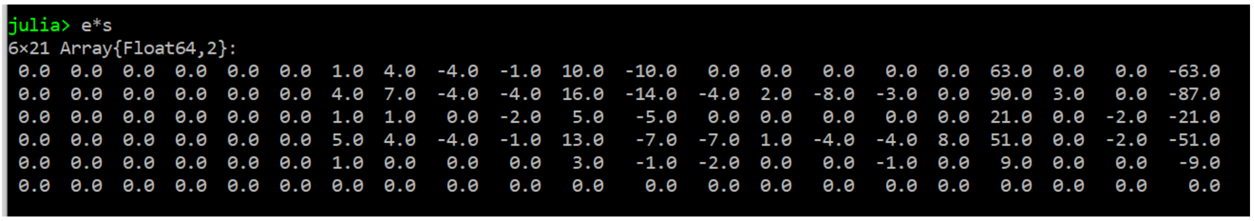
Part B:

In order to determine if the stoichiometric matrix created in part A is elementally balanced, the elemental/atom matrix was constructed. This matrix, with the appropriate headers, can be found in the second tab of the “5440 PS3.xlsx” file.

Julia was used to multiply elemental matrix*stoichiometric matrix, to ensure a matrix of zeros resulted, indicating elemental balance. I used the DelimitedFiles package in Julia to read in my csv files and convert them into matrices. I used the following code:

```
julia > using DelimitedFiles

julia > s = readdlm(“Stoichiometric Matrix.csv” ‘,’)

julia > e = readdlm(“Elemental Matrix.csv” ‘,’)

julia > e*s
```

This code returned the following matrix:



```
julia> e*s
6×21 Array{Float64,2}:
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  4.0  -4.0  -1.0  10.0  -10.0   0.0  0.0   0.0   0.0  0.0  63.0  0.0   0.0  -63.0
 0.0  0.0  0.0  0.0  0.0  0.0  4.0  7.0  -4.0  -4.0  16.0  -14.0  -4.0  2.0  -8.0  -3.0  0.0  90.0  3.0   0.0  -87.0
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  1.0   0.0  -2.0   5.0   -5.0   0.0  0.0   0.0   0.0  0.0  21.0  0.0  -2.0  -21.0
 0.0  0.0  0.0  0.0  0.0  0.0  5.0  4.0  -4.0  -1.0  13.0   -7.0  -7.0  1.0  -4.0  -4.0  8.0  51.0  0.0  -2.0  -51.0
 0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0   0.0   0.0   3.0   -1.0  -2.0  0.0   0.0  -1.0  0.0   9.0  0.0   0.0   -9.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0   0.0   0.0   0.0    0.0   0.0  0.0   0.0   0.0  0.0   0.0  0.0   0.0    0.0
```

This shows that the first 6 columns, the ones that represent our reactions, are all zeroes. We do not have to worry about the exchange reactions since we are not considering external species. Therefore, the stoichiometric matrix is elementally balanced.

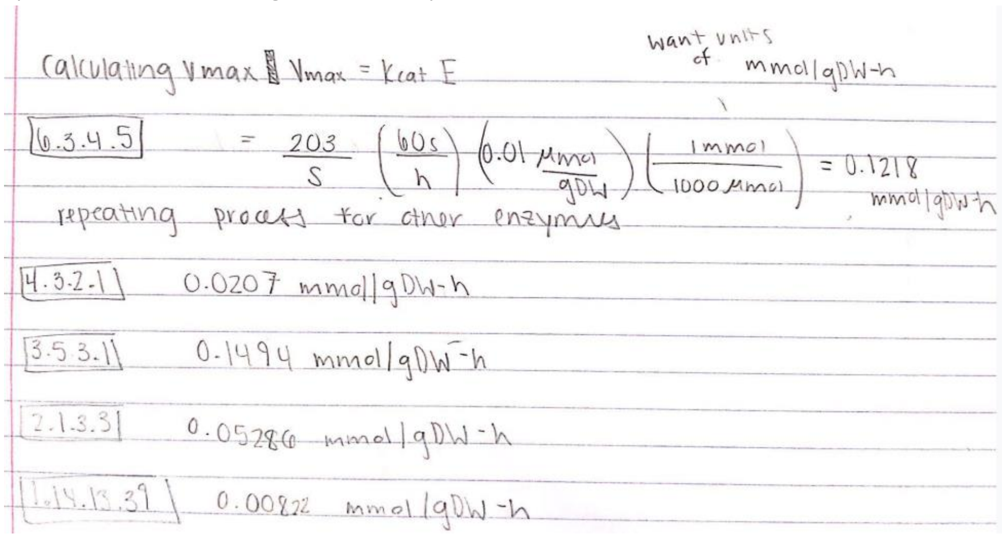
Part C:

The flux.jl script was used to calculate the maximum rate of urea production.

The following inputs were used:

- \`S\` - stoichiometric matrix determined in part A (18x21)

- \`[Lv,Uv]\` - (21 x 2) array: For v's a lower bound of 0 was used and an upper bound of v_{max} was used. v_{max} was calculated by multiplying the k_{cat} of each enzyme by the enzyme concentration E, specified as 0.01umol gDW^{-1} in this problem.

  Calculating vmax $V_{max} = k_{cat} E$ — want units of mmol/gDW-h

  $\boxed{6.3.4.5}$ $= \frac{203}{s}\left(\frac{60s}{h}\right)\left(0.01\frac{\mu mol}{gDW}\right)\left(\frac{1 mmol}{1000 \mu mol}\right) = 0.1218$ mmol/gDW-h

  repeating process for other enzymes

  $\boxed{4.3.2.1}$ 0.0207 mmol/gDW-h

  $\boxed{3.5.3.1}$ 0.1494 mmol/gDW-h

  $\boxed{2.1.3.3}$ 0.05286 mmol/gDW-h

  $\boxed{1.14.13.39}$ 0.00822 mmol/gDW-h

  A better upper bound can be found using Michaelis-Menton kinetics. These saturation constants can be found by using the K_m and metabolite concentrations from the Park et. Al. data set. However, the problem will first be run with the looser constraint of v_{max}.

  For the b's, the bounds -10 to 10 were used, as specified in the problem. The resulting matrix can be seen in "Flux Bounds.csv" (units: mmol/gDW-hr)

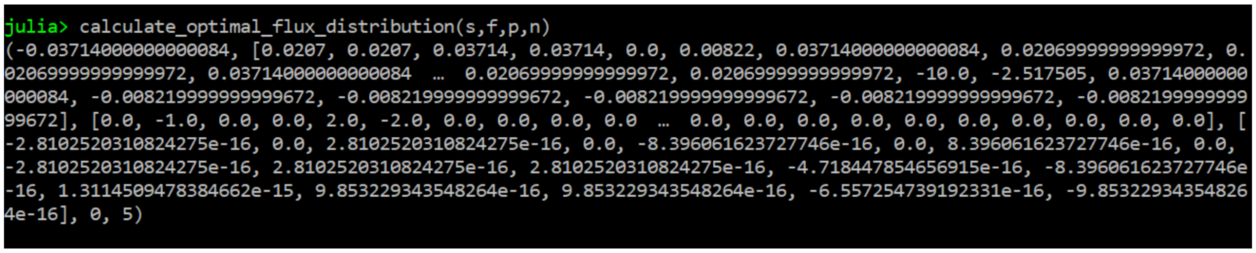
- \`[Lx,Ux]\` - (18 x 2) array of zeros – steady state ["Species Bounds.csv"]

- `c` - (21 x 1) vector holding indexes for objective vector – since we want to maximize b_4 (urea production) a vector with all values equal to 0 except value corresponding to b_4, set equal to -1 [“Objective Vector.csv”]

In order to run the Flux.jl file with the above inputs the following code was used: (**for some reason, when I tried using DelimitedFiles to import “Objective Vector.csv” it would show up as a 2 dimensional array (Array{Float64, 2}) which could not be taken as an input. I therefore had to create this array manually as 1 dimensional within Julia (see line 5 of code below)

```
julia> using DelimitedFiles

julia> s = readdlm(“Stoichiometric Matrix.csv”, ‘,’)

julia> f = readdlm(“Flux Bounds.csv”, ‘,’)

julia> p = readdlm(“Species Bounds.csv”, ‘,’)

julia> n = Float64[0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

julia> include(“Flux.jl”)

julia> calculate_optimal_flux_distribution(s,f,p,n)
```

This code returned the following output:



```
julia> calculate_optimal_flux_distribution(s,f,p,n)
(-0.03714000000000084, [0.0207, 0.0207, 0.03714, 0.03714, 0.0, 0.00822, 0.03714000000000084, 0.02069999999999972, 0.
02069999999999972, 0.03714000000000084  …  0.02069999999999972, 0.02069999999999972, -10.0, -2.517505, 0.03714000000
000084, -0.008219999999999672, -0.008219999999999672, -0.008219999999999672, -0.008219999999999672, -0.0082199999999
99672], [0.0, -1.0, 0.0, 0.0, 2.0, -2.0, 0.0, 0.0, 0.0, 0.0  …  0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [
-2.8102520310824275e-16, 0.0, 2.8102520310824275e-16, 0.0, -8.396061623727746e-16, 0.0, 8.396061623727746e-16, 0.0,
-2.8102520310824275e-16, 2.8102520310824275e-16, 2.8102520310824275e-16, -4.718447854656915e-16, -8.396061623727746e
-16, 1.3114509478384662e-15, 9.853229343548264e-16, 9.853229343548264e-16, -6.557254739192331e-16, -9.85322934354826
4e-16], 0, 5)
```

These outputs are as follows:

Outputs:

`objective_value` - value of the objective function at the optimum

`calculated_flux_array` - R x 1 flux array at the optimum

`dual_value_array` - R x 1 dual values

`uptake_array` - M x 1 array of S*v

`exit_flag` = 0 if optimal

`status_flag` = 5 if optimal

Therefore, the objective value when using v_{max} as the upper flux bound is -0.03714 mmol/gDW-hr.

This solution can be improved by determining saturation terms and using those as upper flux bounds instead.

Saturation terms can be determined using the following formula:

$$v = v_{max}(\frac{[S]}{K_M + [S]}$$

From the Park et. Al. paper, I could find the following metabolite concentrations

- Aspartate: 1.49E-2 M
- Arginine: 5.69E-4 M
- Ornithine: 1.01E-5 M

Using Brenda, K_M values for some enzymes were determined:

- Argininosuccinate synthase – (aspartate substrate) 0.12-0.18 mM
- Arginase – (arginine substrate) 0.02-146 mM
- Ornithine carbamoyltransferase – (ornithine subtrate) 0.001-350 mM
- Nitric-oxide synthase – (arginine substrate) 0.0019 - 68.5 mM

The following saturation terms could therefore be calculated (Since some of the K_M ranges are extreme, calculate for upper and lower values)

<u>6.3.4.5</u>

Lower K_M

v = 0.1208 mmol/gDW h

Upper K_M

v = 0.1203 mmol/gDW h

<u>3.5.3.1</u>

Lower K_M

v = 0.1443 mmol/gDW h

Upper K_M

v = 5.800E-4 mmol/gDW h

<u>2.1.3.3</u>

Lower K_M

v = 0.0481 mmol/gDW h

Upper K_M

v = 1.525E-6 mmol/gDW h

1.14.13.39

Lower K_M

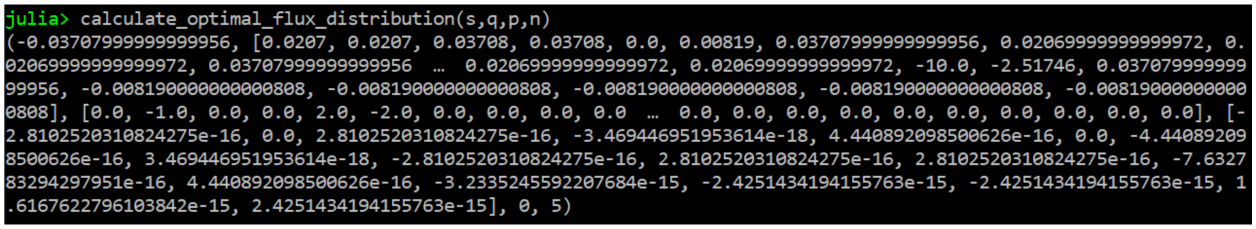
v = 0.00819 mmol/gDW h

Upper K_M

v = 6.771E-4 mmol/gDW h

The new flux bounds using the different Km values are saved as “Flux Bounds Lower Km.csv” and “Flux Bounds Upper Km.csv”

I ran the file with the “Flux Bounds Lower Km.csv” as the flux bounds array and a similar result was obtained:



```
julia> calculate_optimal_flux_distribution(s,q,p,n)
(-0.03707999999999956, [0.0207, 0.0207, 0.03708, 0.03708, 0.0, 0.00819, 0.03707999999999956, 0.02069999999999972, 0.
02069999999999972, 0.03707999999999956  …  0.02069999999999972, 0.02069999999999972, -10.0, -2.51746, 0.037079999999
99956, -0.008190000000000808, -0.008190000000000808, -0.008190000000000808, -0.008190000000000808, -0.00819000000000
0808], [0.0, -1.0, 0.0, 0.0, 2.0, -2.0, 0.0, 0.0, 0.0, 0.0  …  0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], [-
2.8102520310824275e-16, 0.0, 2.8102520310824275e-16, -3.469446951953614e-18, 4.440892098500626e-16, 0.0, -4.44089209
8500626e-16, 3.469446951953614e-18, -2.8102520310824275e-16, 2.8102520310824275e-16, 2.8102520310824275e-16, -7.6327
83294297951e-16, 4.440892098500626e-16, -3.2335245592207684e-15, -2.4251434194155763e-15, -2.4251434194155763e-15, 1
.6167622796103842e-15, 2.4251434194155763e-15], 0, 5)
```

The same code could also be run using the “Flux Bounds Upper Km.csv” file.