project_4

April 21, 2023

0.0.1 Introduction

In this project, you will be continuing your work with the *Escherichia coli* core model and investigate how the growth phenotype of the model changes when varying one or multiple exchange fluxes (i.e., production envelopes and phenotype phase planes). Additionally, you will be conducting a gene essentiality analysis on the model where the deletion of a gene can be simulated by simply constraining the lower and upper flux bounds of its associated reactions to 0. Gene deletion analysis is a common application of genome-scale metabolic models and allow us to assess the effects of a gene deletion on the metabolic phenotype of an organism before initiating expensive and time-consuming experiments in the lab. I recommend taking a look at the cobrapy documentation before starting with the project, particularly the section on deletion analysis found here.

[ ]:
```
import json
import random
from IPython.display import Image
from math import comb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from cobra.io import read_sbml_model
from cobra.flux_analysis import (production_envelope, single_gene_deletion,␣
 ↪double_gene_deletion)
from phpp import PhenotypePhasePlane, PhPP
```

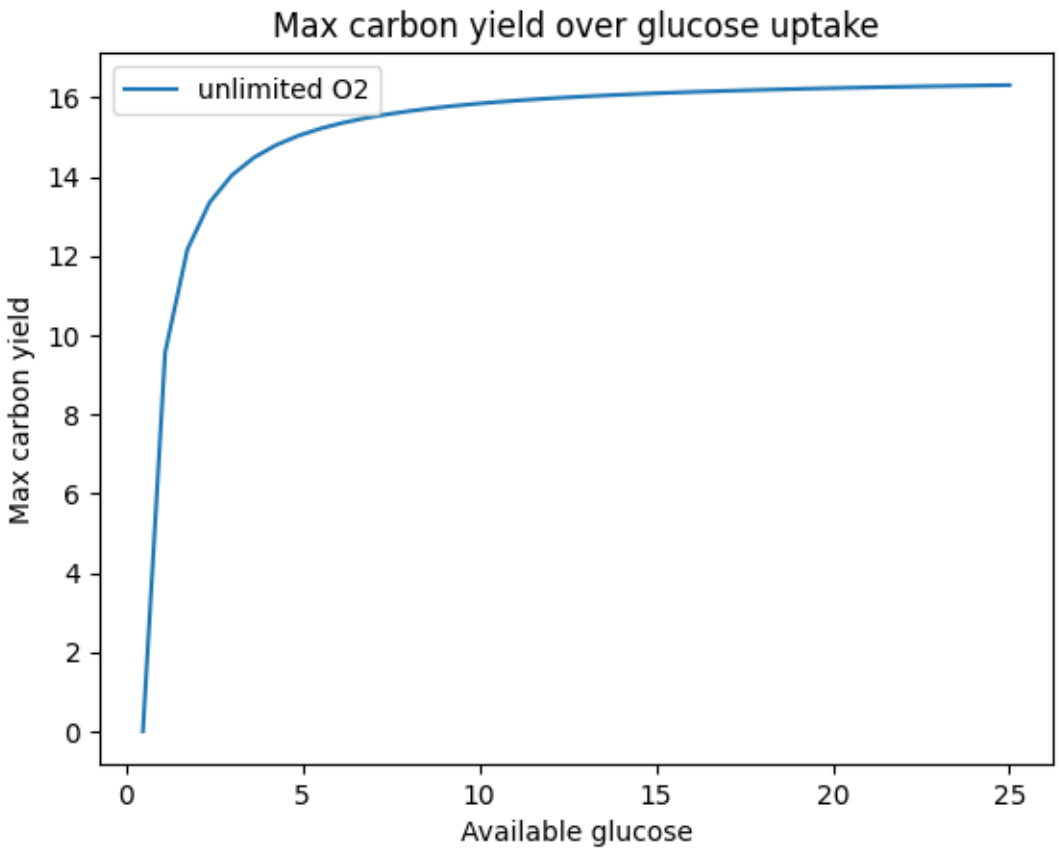
[ ]:
```
ecoli_model = read_sbml_model("ecoli_core_model.xml")

def foo(model, o2_lower_bound, o2_upper_bound,
        carbon_source, carbon_source_lower_bound,
        carbon_source_upper_bound, no_glucose=False):
    local_model = model.copy()
    local_model.reactions.get_by_id("EX_o2_e").lower_bound = o2_lower_bound
    local_model.reactions.get_by_id("EX_o2_e").upper_bound = o2_upper_bound
    local_model \
        .reactions.get_by_id(carbon_source).lower_bound =␣
 ↪carbon_source_lower_bound
    local_model \
```

```
        .reactions.get_by_id(carbon_source).upper_bound =␣
↪carbon_source_upper_bound
    # I think that glucose is available by default. Change that:
    if (no_glucose):
        local_model.reactions.get_by_id("EX_glc__D_e").lower_bound = 0

    local_model.optimize()

    prod_env = production_envelope(local_model, [carbon_source],
                                   carbon_sources=carbon_source, points=40)
    prod_env[str(carbon_source)+".transformed"] =  -1*prod_env[carbon_source]

    name_dict = {"EX_glc__D_e" : "glucose",
                 "EX_succ_e" : "succinate"}

    chart = plt.plot(prod_env[(str(carbon_source)+".transformed")],␣
↪prod_env["carbon_yield_maximum"], )
    return chart
```

0.0.2 1.1 Phenotype phase plane analysis

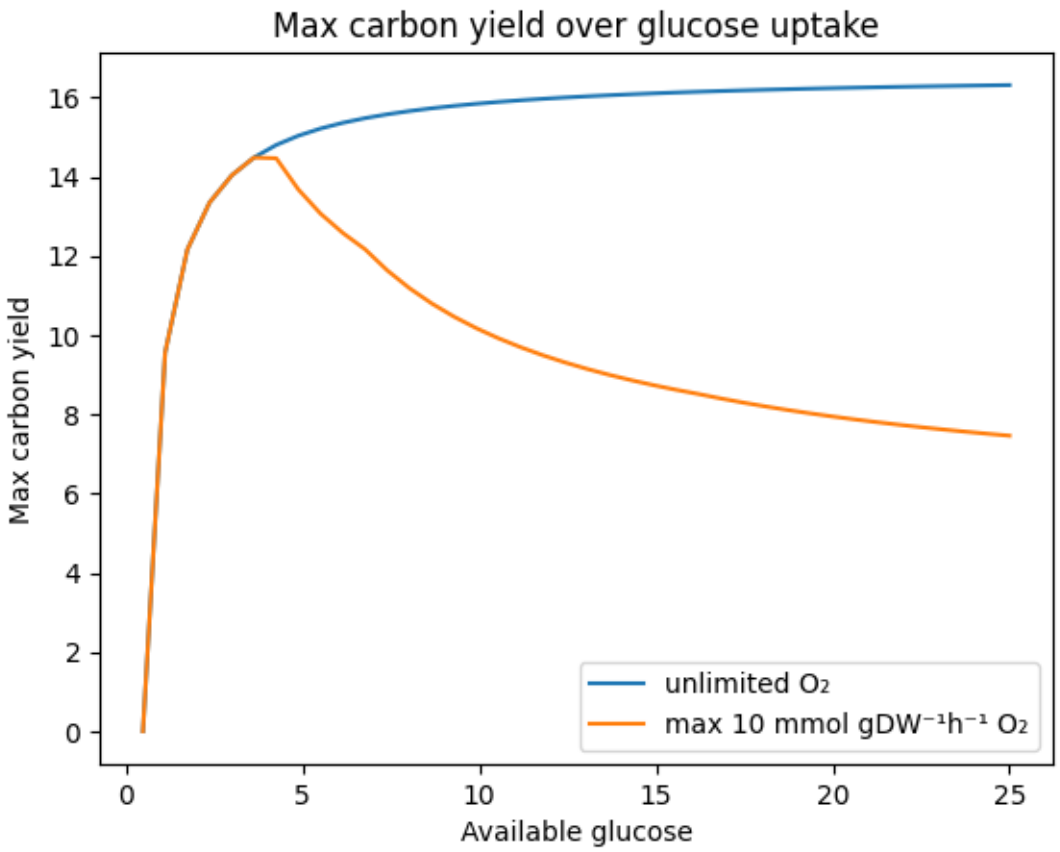
(i) Using the cobrapy function `production_envelope` (see documentation), calculate the growth rate as a function of glucose uptake flux when oxygen is unlimited for the *E. coli* core model. Plot growth vs. glucose uptake flux from 0 to -25 mmol gDW-1 h-1 using 40 plot points.

[ ]:
```
fig, ax = plt.subplots()
high_O2 = foo(ecoli_model, -1000, 0, "EX_glc__D_e", -25, 0)
ax.set_title(f"Max carbon yield over glucose uptake")
ax.set_xlabel(f"Available glucose")
ax.set_ylabel("Max carbon yield")
plt.legend(
    ["unlimited O2",
    "max 10 mmol gDW\u207B\u00B9h\u207B\u00B9 O2"])
plt.show()
```



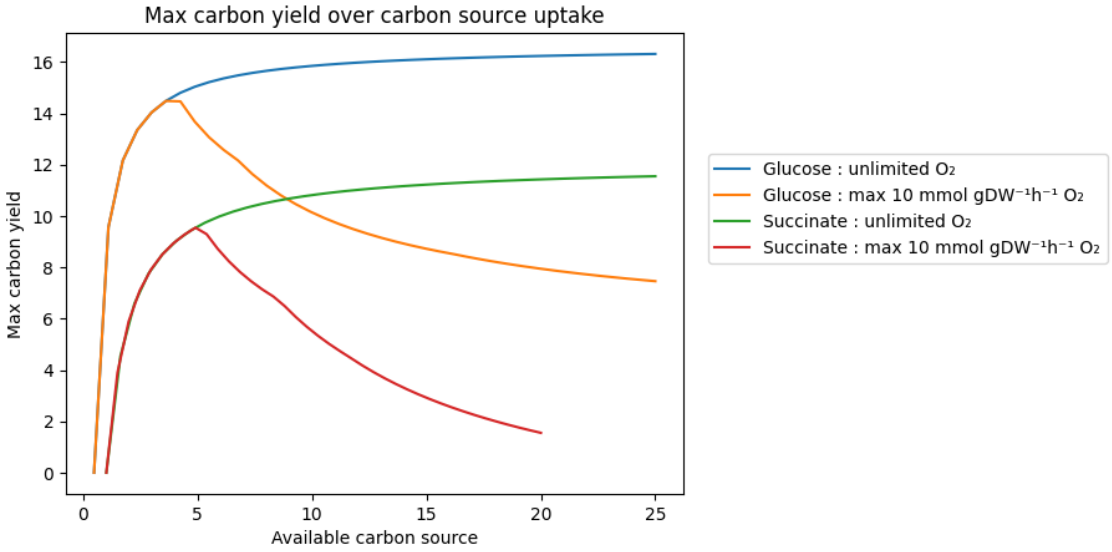
(ii) Same as in (i), but now limit the oxygen uptake flux to -10 mmol gDW-1 h-1 . Compare and explain the differences with the results from (i).

[ ]:
```
fig, ax = plt.subplots()
high_O2_glu = foo(ecoli_model, -1000, 0, "EX_glc__D_e", -25, 0)
Low_O2_glu = foo(ecoli_model, -10, 0, "EX_glc__D_e", -25, 0)
ax.set_title(f"Max carbon yield over glucose uptake")
ax.set_xlabel(f"Available glucose")
ax.set_ylabel("Max carbon yield")
plt.legend(["unlimited O\u2082", "max 10 mmol gDW\u207B\u00B9h\u207B\u00B9␣
 ↪O\u2082"])
plt.show()
```



(iii) Same as in (i) and (ii), but with succinate as the only carbon source. Compare and explain the differences with the results from (i) and (ii). Use the calculated growth rates from Table 1 in Project 3 to argue for your results.

[ ]:
```
fig, ax = plt.subplots()
high_O2_glu = foo(ecoli_model, -1000, 0, "EX_glc__D_e", -25, 0)
Low_O2_glu = foo(ecoli_model, -10, 0, "EX_glc__D_e", -25, 0)
high_O2_suc = foo(ecoli_model, -1000, 0, "EX_succ_e", -25, 0, True)
Low_O2_suc = foo(ecoli_model, -10, 0, "EX_succ_e", -25, 0, True)
ax.set_title(f"Max carbon yield over carbon source uptake")
ax.set_xlabel(f"Available carbon source")
ax.set_ylabel("Max carbon yield")
plt.legend(
    ["Glucose : unlimited O\u2082",
     "Glucose : max 10 mmol gDW\u207B\u00B9h\u207B\u00B9 O\u2082",
     "Succinate : unlimited O\u2082",
     "Succinate : max 10 mmol gDW\u207B\u00B9h\u207B\u00B9 O\u2082"],
     loc=(1.04,0.5))
plt.show()
```



Commonly, we classify the linear segments of the production envelopes by their *shadow prices*, which are defined as

$

$$\pi_i = -\frac{\partial Z}{\partial b_i}. \tag{1}$$

$

The shadow prices π_i defines the sensitivity of the objective function Z to changes in the availability of a given metabolite i. b_i defines the violation of the mass balance constraint of metabolite i and is equivalent to an uptake reaction. In other words, it quantitatively describes how the objective function value would change if we could import more of an exchangeable metabolite. A negative π_i indicates that increasing the availability of a metabolite i will increase the optimal objective value, while a positive π_i indicates that the optimal objective value will decrease.

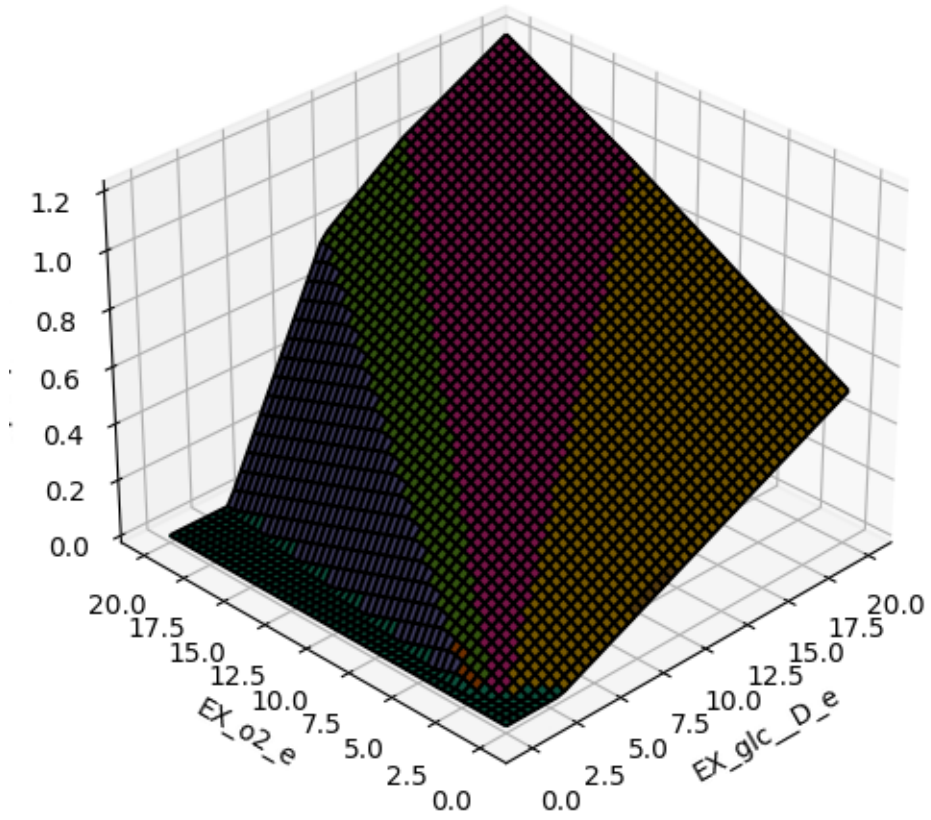
While production envelopes plotted above are interesting in their own right, it is often more informative to investigate the growth phenotype when adjusting two exchange fluxes simultaneously. This is called phenotype phase plane analysis (PhPP). Rather than forming linear segments, PhPP results in the formation of two-dimensional planes which are defined by their metabolic phenotype and shadow prices of the two variable exchange metabolites.

(iv) Start by plotting a PhPP and associated shadow prices for glucose and oxygen using the *E. coli* core model and the `PhPP` function and associated plotting functions found on Blackboard (`phpp.py`). Use a `flux_range` evenly distributed between 0 and -20 with 50 data points. Note that you will be needing the Python package `palettable`.

[ ]:
```
flux_range = np.linspace(0, -20, 50)
plane = PhPP(
    model=ecoli_model,
    rxn_x="EX_glc__D_e",
```

```
    rxn_y="EX_o2_e",
    rxn_x_range=flux_range,
    rxn_y_range=flux_range)
```

[ ]:
```
# Plot the phenotypic phase planes:
plane.plot_PhPP()
```



We can observe that the shadow prices π_i are constant in each of the phase planes. Consequently, we can determine the properties of the different phases by calculating

$

$$\alpha = -\frac{\pi_x}{\pi_y}, \tag{2}$$

$

the ratio between the shadow prices associated with the two exchange fluxes x and y (here, glucose and oxygen). We can then define four types of phases in PhPP: 1. If $\alpha < 0$, the objective function is limited by both exchange fluxes (*dual substrate limitation*). 2. If $\alpha > 0$, the region is called a *futile phase*: one of the exchange substrates will have a positive shadow price. Characterized by wasteful

metabolic operations (substrate consumed does not improve Z). 3. If either $\pi_x = 0$ or $\pi_y = 0$, the region is described by *single substrate limitation*. 4. *Infeasible phases* due to a combination of stoichiometric limitations and/or substrate availability (no growth).

(v) How many unique phases does this PhPP have? Classify these according to the four types defined above.

[ ]:
```
def alpha(sp_x, sp_y):
    if sp_x == 0 or sp_y == 0:
        return 0
    else:
        return -(sp_x/sp_y)

def handle_alpha(alpha):
    if alpha < 0:
        return "Dual substrate limitation"
    elif alpha > 0:
        return "Futile phase"
    elif alpha == 0:
        return "Single substrate limitation"
    else:
        return "Infeasible phases"  # This will never run \
        # according to the criteria in the task description!

# Identify the segments:
plane.segment()

# Print output:
print(f"Number of unique phases {len(plane.phases)}. The phases are:")
counter = 0
for phase in plane.phases:
    counter += 1
    print(f"# Phase {counter}:")
    print("x={x: >7.4f}, y={y: >7.4f}, \
\tshadow price 1: {sp1: >7.4f}, \
shadow price 2: {sp2: >7.4f}"  \
            .format(
                x=phase[0][0],
                y=phase[0][1],
                sp1=phase[1],
                sp2=phase[2]))
    print(f"The phase is described as: \
        {handle_alpha(alpha=alpha(phase[1], phase[2]))}\n")
```

```
Number of unique phases 7. The phases are:
# Phase 1:
x= 0.0000, y= 0.0000,   shadow price 1:  0.0000, shadow price 2:  0.0000
The phase is described as:         Single substrate limitation
```

```
# Phase 2:
x=-0.8163, y=-2.4490,   shadow price 1: -0.0498, shadow price 2: -0.0223
The phase is described as:         Dual substrate limitation

# Phase 3:
x=-0.8163, y=-3.6735,   shadow price 1: -0.1373, shadow price 2:  0.0229
The phase is described as:         Futile phase

# Phase 4:
x=-1.2245, y=-1.6327,   shadow price 1: -0.0325, shadow price 2: -0.0325
The phase is described as:         Dual substrate limitation

# Phase 5:
x=-1.2245, y=-2.4490,   shadow price 1: -0.0459, shadow price 2: -0.0230
The phase is described as:         Dual substrate limitation

# Phase 6:
x=-1.6327, y=-1.2245,   shadow price 1: -0.0305, shadow price 2: -0.0360
The phase is described as:         Dual substrate limitation

# Phase 7:
x=-7.3469, y= 0.0000,   shadow price 1: -0.0305, shadow price 2: -0.0581
The phase is described as:         Dual substrate limitation
```

From the code output I guess that Phase 1 can actually be described as a Dual substrate limitation, given that both substrate levels are 0.

(vi) Considering the reasonable assumption of growth optimality, where do you think *E. coli* strains will be located on this PhPP in vivo?

I would say that *E. coli* could be found on many location of this PhPP depending on its environment. *E. coli* live in the gut where there is hypoxic/anoxic conditions. The nutrient availability may also vary a lot depending on when the host ate. Free-living *E. coli* could in comparison have more stable nutrient availability in e.g. the soil.

0.0.3 1.2 Gene deletion analysis

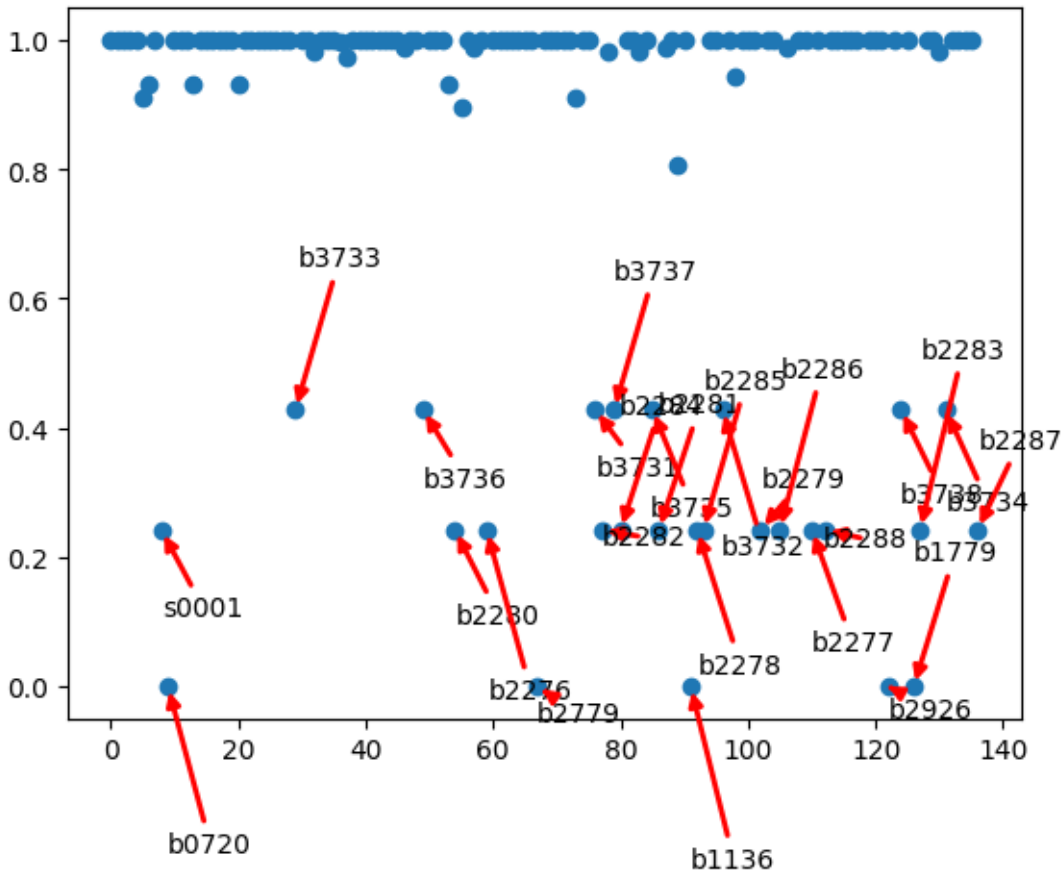
(i) Conduct a single gene deletion study for all the genes in the *E. coli* core model. Use the default defined nutrient environment (i.e. aerobic glucose).

(a) By making a scatter plot of the relative growth rates (mutant growth / wild type growth) for all gene deletions, define what you consider to be an essential gene.

[ ]:
```
deletion_results = single_gene_deletion(ecoli_model)
```

[ ]:
```
wt_growth = ecoli_model.optimize().objective_value
deletion_results["relative_growth_rate"] = deletion_results.growth/wt_growth

fig, ax = plt.subplots()
```

```
ax.scatter(list(range(deletion_results.shape[0])), deletion_results.
  ↪relative_growth_rate)
# Add
arrowprops=dict(arrowstyle='-|>', color="red", linewidth=2)
for i, txt in enumerate(deletion_results.ids):
    txt = str(txt)[2:-2]
    if deletion_results.relative_growth_rate[i] < 0.5:
        x = list(range(deletion_results.shape[0]))[i]
        y = deletion_results.relative_growth_rate[i]
        ax.annotate(
            txt,
            xy=(x,y),
            xytext=(
                x+random.uniform(-0.3, 0.3),
                y+random.uniform(-0.3, 0.3)),
            arrowprops=arrowprops
            )
plt.show()
```



My first thought would be that important genes are genes that when knocked out cause a significant drop in growth. As one can see on the graph there are many of these. However, a truly essential gene would be genes that when knocked out results in a growth rate almost equal to 0. In this project this threshold has been set to 10^{-10}.

(b) Identify the name and function of all the essential genes you found. While checking which reactions these genes are associateed with in the model is useful, it can also be an idea to check biochemical databases for informations such as EcoCyc.

[ ]:
```
# Define an essentiality threshold
essentiality_threshold = 10**(-10)

for row in range(deletion_results.shape[0]):
    if deletion_results.relative_growth_rate[row] < essentiality_threshold:
        gene = str(deletion_results.ids[row])[2:-2]
        display(getattr(ecoli_model.genes, gene))
        print("Relative growth rate of: {gr:.4f}"\
            .format(
                gr=deletion_results.relative_growth_rate[row]
                ))
        print()

# NB! Used later! Store a df with only essential genes:
df_essential_KO = deletion_results[deletion_results.relative_growth_rate <␣
 ↪essentiality_threshold]
```

```
<Gene b0720 at 0x1e9e52ad120>

Relative growth rate of: -0.0000


<Gene b2779 at 0x1e9e52adf90>

Relative growth rate of: -0.0000


<Gene b1136 at 0x1e9e52ad570>

Relative growth rate of: -0.0000


<Gene b2926 at 0x1e9e52ae020>

Relative growth rate of: 0.0000


<Gene b1779 at 0x1e9e52ad8d0>

Relative growth rate of: 0.0000
```

b0720 - CS: Enzyme: citrate synthase Gene: gltA Function: Synthesize citrate (metabolite in e.g. TCA cycle) Pathways: glyoxylate cycle TCA cycle I (prokaryotic) Mixed acid fermentation

b2779 - ENO Enzyme: enolase
Gene: eno Function: Catalyze the conversion (both direction) of metabolite in glycolysis and/or gluconeogenesis Pathways: gluconeogenesis I glycolysis II (from fructose 6-phosphate) glycolysis I (from glucose 6-phosphate)

b1136 - ICDHyr Enzyme: isocitrate dehydrogenase Gene: icd Function: Catalyze reaction in TCA cycle Pathways: TCA cycle I (prokaryotic) Mixed acid fermentation

b2926 - PGK Enzyme: phosphoglycerate kinase
Gene: pgk Function: Catalyze reaction in glycolysis and gluconeogenesis Pathways: gluconeogenesis I glycolysis II (from fructose 6-phosphate) glycolysis I (from glucose 6-phosphate)

b1779 - GAPD Enzyme: glyceraldehyde-3-phosphate dehydrogenase Gene: gapA Function: Catalyze reaction in glycolysis and gluconeogenesis Pathways: gluconeogenesis I glycolysis II (from fructose 6-phosphate) glycolysis I (from glucose 6-phosphate)

(c) Give plausible explanations for why they are essential.

So, all the essential genes seems to encode enzymes that are either part of the glycolysis/gluconeogenesis, or the TCA cycle. It would make sense that these are essential in the default ecoli core model, as the sole energy source is glucose. I came across something about ecoli being able to survive on some metabolite if one of the enzyme was inhibited, probably by using other (parts) of pathways. However, that is of course only relevant if the other energy sources are available.

(ii) Change the single carbon source from glucose to acetate and redo the analysis in (a) (no need to go into detail on the function of each gene, comment on the general trend). Also, identify the essential genes that are common to both nutrient environments, and those that are specific to acetate. Explain why this is so.

[ ]:
```
ecoli_model_acetate = ecoli_model.copy()
ecoli_model_acetate.reactions.get_by_id("EX_ac_e").lower_bound =␣
 ↪ecoli_model_acetate.reactions.get_by_id("EX_glc__D_e").lower_bound
ecoli_model_acetate.reactions.get_by_id("EX_glc__D_e").lower_bound = 0
ecoli_model_acetate.optimize()

deletion_results_acetate = single_gene_deletion(ecoli_model_acetate)

wt_growth_acetate = ecoli_model_acetate.optimize().objective_value
deletion_results_acetate["relative_growth_rate"] = deletion_results_acetate.
 ↪growth/wt_growth_acetate

for row in range(deletion_results_acetate.shape[0]):
    if deletion_results_acetate.relative_growth_rate[row] <␣
 ↪essentiality_threshold:
        gene = str(deletion_results_acetate.ids[row])[2:-2]
        display(getattr(ecoli_model.genes, gene))
        print("Relative growth rate of: {gr:.4f}"\
            .format(
                gr=deletion_results_acetate.relative_growth_rate[row]
```

```
                ))
        print()
```

```
<Gene b4015 at 0x1e9e52ae680>
Relative growth rate of: 0.0000

<Gene b4025 at 0x1e9e52ae6b0>
Relative growth rate of: 0.0000

<Gene b2779 at 0x1e9e52adf90>
Relative growth rate of: -0.0000

<Gene b2926 at 0x1e9e52ae020>
Relative growth rate of: 0.0000

<Gene b3919 at 0x1e9e52ae530>
Relative growth rate of: -0.0000

<Gene b1136 at 0x1e9e52ad570>
Relative growth rate of: -0.0000

<Gene b1779 at 0x1e9e52ad8d0>
Relative growth rate of: 0.0000
```

As when glucose was the energy source, the essential genes with acetate as energy source are all linked to pathways of energy-metabolism. The genes that are in common are: ENO, ICDHyr, PGK, and GAPD. This makes sense given that acetate can be converted to acetyl-CoA, which is the metabolite added to "start" the TCA cycle.

(iii) Conduct a complete double-gene deletion study of the model in the original nutrient environment.

[ ]:
```
ddeletion_results = double_gene_deletion(ecoli_model)
ddeletion_results["relative_growth_rate"] = ddeletion_results.growth/wt_growth
```

(a) What is the fraction of essential double-gene knockouts?

[ ]:
```
total_pair_KO = ddeletion_results.shape[0]
essential_pair_KO = ddeletion_results[ddeletion_results.relative_growth_rate <␣
↪essentiality_threshold].shape[0]
fraction = essential_pair_KO/total_pair_KO
```

```
print("The fraction of essential double-gene knockouts is {frac:.5f}".
 ↪format(frac=fraction))
print(f"There are {essential_pair_KO} of essential double-gene knockouts in␣
 ↪total.")
```

```
The fraction of essential double-gene knockouts is 0.06273
There are 593 of essential double-gene knockouts in total.
```

(b) Which genes are involved in the largest number of essential double-knockout combinations? Any idea why?

[ ]:
```
df_essential_pair_KO = ddeletion_results[
    ddeletion_results.relative_growth_rate < essentiality_threshold
    ]
individual_genes = set()
for pair in df_essential_pair_KO.ids:
    individual_genes = individual_genes.union(pair)
individual_genes = list(individual_genes)

dic = {key:0 for key in individual_genes}

occurrences = 0
for key in dic.keys():
    occurrences = 0

    for pair in df_essential_pair_KO.ids:
        if key in pair:
            occurrences += 1

    dic[key] = occurrences

# Sort dictionary by occurrences
dic = dict(sorted(dic.items(), key=lambda item: item[1], reverse=True))

# Top-10:
for elm in range(10):
    gene = list(dic.keys())[elm]
    print(f"\nThe gene {list(dic.keys())[elm]} are involved in {dic[gene]} of␣
 ↪essential double-knockout combinations.")
    display(getattr(ecoli_model.genes, gene))
```

```
The gene b0720 are involved in 135 of essential double-knockout combinations.
```

```
<Gene b0720 at 0x1e9e52ad120>
```

```
The gene b1136 are involved in 135 of essential double-knockout combinations.
```

```
<Gene b1136 at 0x1e9e52ad570>

The gene b2926 are involved in 106 of essential double-knockout combinations.
<Gene b2926 at 0x1e9e52ae020>

The gene b2779 are involved in 106 of essential double-knockout combinations.
<Gene b2779 at 0x1e9e52adf90>

The gene b1779 are involved in 106 of essential double-knockout combinations.
<Gene b1779 at 0x1e9e52ad8d0>

The gene b3956 are involved in 12 of essential double-knockout combinations.
<Gene b3956 at 0x1e9e52ae5f0>

The gene b1761 are involved in 7 of essential double-knockout combinations.
<Gene b1761 at 0x1e9e52ad870>

The gene b0722 are involved in 6 of essential double-knockout combinations.
<Gene b0722 at 0x1e9e52ad180>

The gene b3213 are involved in 6 of essential double-knockout combinations.
<Gene b3213 at 0x1e9e52ae1a0>

The gene b2914 are involved in 6 of essential double-knockout combinations.
<Gene b2914 at 0x1e9e52adfc0>
```

So, the first five essential double-knockout combinations that showcase in more than two occurrences are all genes that we have already identified for a single gene knockout.

(iv) Same analysis as in (iii), but this time with acetate as the single carbon source.

[ ]:
```
ddeletion_results_acetate = double_gene_deletion(ecoli_model_acetate)
```

[ ]:
```
ddeletion_results_acetate["relative_growth_rate"] = ddeletion_results_acetate.
 ↪growth/wt_growth_acetate

total_pair_KO = ddeletion_results_acetate.shape[0]
essential_pair_KO = ddeletion_results_acetate[ddeletion_results_acetate.
 ↪relative_growth_rate < essentiality_threshold].shape[0]
fraction = essential_pair_KO/total_pair_KO
```

```
print("The fraction of essential double-gene knockouts is {frac:.5f}".
 ↪format(frac=fraction))
print(f"There are {essential_pair_KO} of essential double-gene knockouts in␣
 ↪total.")

df_essential_pair_KO_acetate =␣
 ↪ddeletion_results_acetate[ddeletion_results_acetate.relative_growth_rate <␣
 ↪essentiality_threshold]
individual_genes = set()
for pair in df_essential_pair_KO_acetate.ids:
    individual_genes = individual_genes.union(pair)
individual_genes = list(individual_genes)

dic = {key:0 for key in individual_genes}

occurrences = 0
for key in dic.keys():
    occurrences = 0

    for pair in df_essential_pair_KO_acetate.ids:
        if key in pair:
            occurrences += 1

    dic[key] = occurrences

dic = dict(sorted(dic.items(), key=lambda item: item[1], reverse=True))

# Top-10:
for elm in range(10):
    gene = list(dic.keys())[elm]
    print(f"\nThe gene {list(dic.keys())[elm]} are involved in {dic[gene]} of␣
 ↪essential double-knockout combinations.")
    display(getattr(ecoli_model.genes, gene))
```

```
The fraction of essential double-gene knockouts is 0.07966
There are 753 of essential double-gene knockouts in total.

The gene b4025 are involved in 110 of essential double-knockout combinations.
```

<Gene b4025 at 0x1e9e52ae6b0>

```
The gene b3919 are involved in 110 of essential double-knockout combinations.
```

<Gene b3919 at 0x1e9e52ae530>

```
The gene b2926 are involved in 110 of essential double-knockout combinations.
```

```
<Gene b2926 at 0x1e9e52ae020>

The gene b2779 are involved in 110 of essential double-knockout combinations.
<Gene b2779 at 0x1e9e52adf90>

The gene b1779 are involved in 110 of essential double-knockout combinations.
<Gene b1779 at 0x1e9e52ad8d0>

The gene b1136 are involved in 109 of essential double-knockout combinations.
<Gene b1136 at 0x1e9e52ad570>

The gene b4015 are involved in 104 of essential double-knockout combinations.
<Gene b4015 at 0x1e9e52ae680>

The gene b1761 are involved in 9 of essential double-knockout combinations.
<Gene b1761 at 0x1e9e52ad870>

The gene b3403 are involved in 9 of essential double-knockout combinations.
<Gene b3403 at 0x1e9e52ae230>

The gene b2987 are involved in 8 of essential double-knockout combinations.
<Gene b2987 at 0x1e9e52ae0e0>
```

(v) Use the results from (i) and (iii) to identify only the synthetic lethal gene combinations. How many are there?

[ ]:
```python
single_KO_genes = [str(gene)[2:-2] for gene in list(df_essential_KO.ids)]

# First find the KO pairs that are not one of the 5
# essential genes found in the single gene KO:
synthetic_lethal = []
for i in df_essential_pair_KO.ids:
    if len(i) != 2:  # Skip elements with len 1!
        continue
    if (list(i)[0] not in single_KO_genes) and (list(i)[1] not in␣
 ↪single_KO_genes):
        synthetic_lethal.append(list(i))

print(f"All of these 5 essential genes (found by single gene KO) \
can be combined to form synthetic lethal gene combinations. \
The genes are: {single_KO_genes}")
```

```
print("The pair of synthetic lethal gene combinations \
that do not include the 5 essential genes found \
in the single gene KO are:")
for pair in synthetic_lethal:
    print(pair)

# The number of times one can combine the
# 5 essential genes (found in the single gene knockout)
# is the binomial coefficient (5 choose 2), which is 10.
print(f"\nThe total number of synthetical lethal gene combinations is␣
 ↪{len(synthetic_lethal)+comb(5, 2)}")
```

```
All of these 5 essential genes (found by single gene KO) can be combined to form
synthetic lethal gene combinations. The genes are: ['b0720', 'b2779', 'b1136',
'b2926', 'b1779']
The pair of synthetic lethal gene combinations that do not include the 5
essential genes found in the single gene KO are:
['s0001', 'b3956']
['b0721', 'b3956']
['b0722', 'b3956']
['b1276', 'b0118']
['b4090', 'b2914']
['b2935', 'b2465']
['b0723', 'b3956']
['b3213', 'b1761']
['b3212', 'b1761']
['b3236', 'b3956']
['b0724', 'b3956']
['b3870', 'b1297']
['b2987', 'b3493']
['b4015', 'b3956']
['s0001', 'b0451']

The total number of synthetical lethal gene combinations is 25
```

(vi) Select a partial essential gene from (a) (i.e. a gene which when deleted reduces the relative growth rate of the model, but not below the chosen essentiality threshold). Visualize its optimal flux distribution in Escher and compare it to the wild type flux distribution. Discuss and explain the observed differences in flux phenotypes.

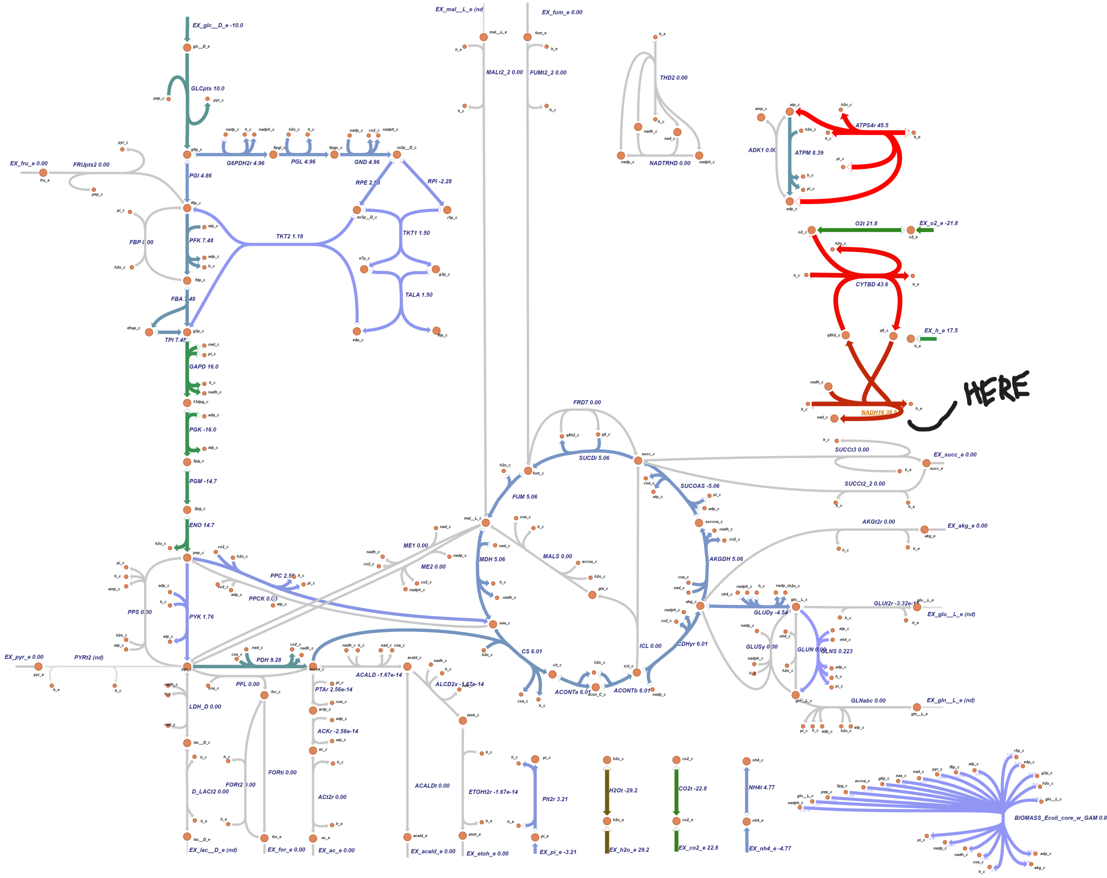
[ ]:
```
# Df of partial essential genes:
partial_essential = deletion_results[(deletion_results.relative_growth_rate < 0.
 ↪6) & (deletion_results.relative_growth_rate > 0.2)]
#display(partial_essential)

# An arbitrary choice resulted in the gene: b2282
```

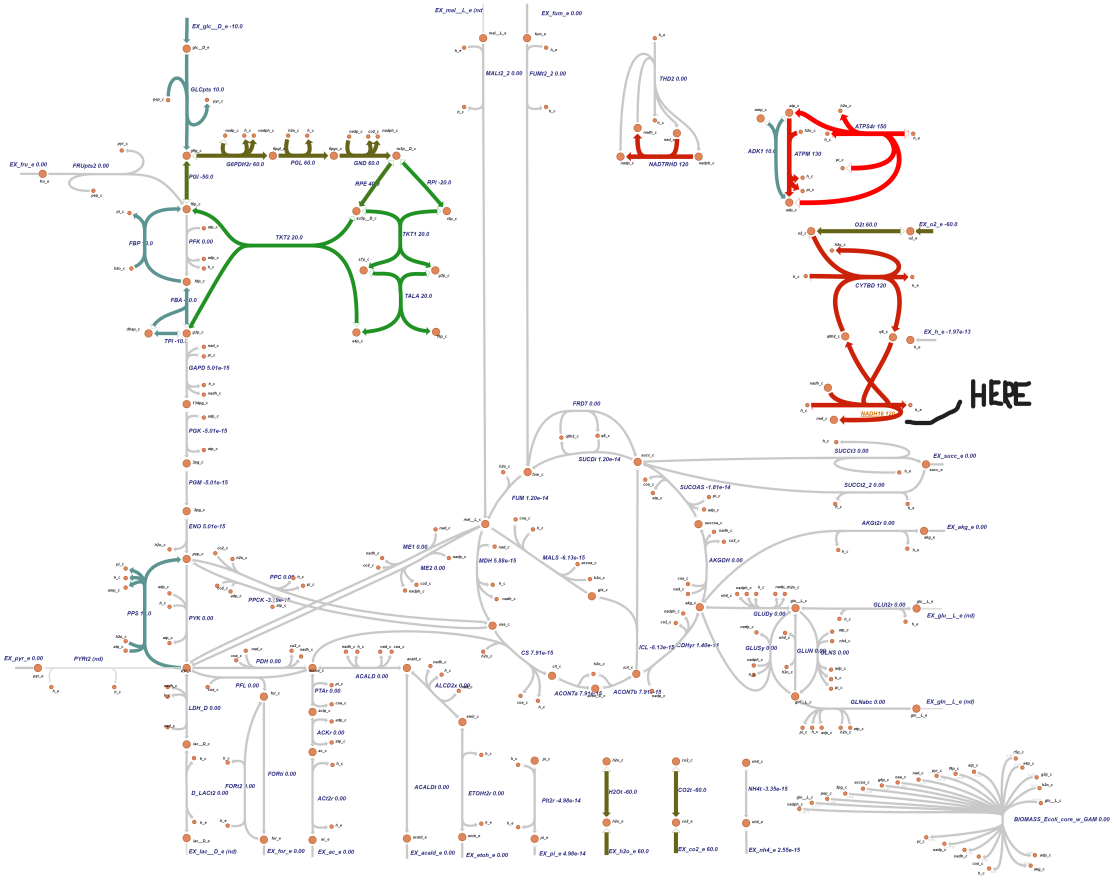
```
# From: https://www.geeksforgeeks.org/how-to-convert-python-dictionary-to-json/
# Wild type flux:
dic = {reac.id : flx.flux for reac,flx in zip(ecoli_model.reactions,␣
 ↪ecoli_model.reactions)}

with open("b2282_wt_flux.json", "w") as outfile:
    json.dump(dic, outfile)


max_b2282_flux_model = ecoli_model.copy()
max_b2282_flux_model.objective = list(ecoli_model.genes.b2282.reactions)[0].id
max_b2282_flux_model.optimize()

# Wild type flux:
dic = {reac.id : flx.flux for reac,flx in zip(max_b2282_flux_model.reactions,␣
 ↪max_b2282_flux_model.reactions)}

with open("b2282_optimized_flux.json", "w") as outfile:
    json.dump(dic, outfile)
```

[ ]:
```
reac_id = list(ecoli_model.genes.b2282.reactions)[0].id

print(f"The wild type flux of {reac_id} is {getattr(ecoli_model.optimize().
 ↪fluxes, reac_id)}.")
print(f"The optimized flux of {reac_id} is {getattr(max_b2282_flux_model.
 ↪optimize().fluxes, reac_id)}.")
```

```
The wild type flux of NADH16 is 38.534609650515435.
The optimized flux of NADH16 is 120.0.
```

[ ]:
```
display(Image("wt_flux.png"))
display(Image("optimized_flux.png"))
```


EX_glc__D_e -10.0
GLCpts 10.0
PGI 4.96
G6PDH2r 4.96
PGL 4.96
GND 4.96
RPE 2.51
RPI -2.28
TKT2 1.18
TKT1 1.50
TALA 1.50
PFK 7.48
FBP 0.00
EX_fru_e 0.00
FRUpts2 0.00
GAPD 16.0
PGK -16.0
PGM -14.7
ENO 14.7
PPC 2.50
PPCK 0.00
PPS 0.00
PYK 1.76
EX_pyr_e 0.00
PYRt2 (nd)
PDH 9.28
PFL 0.00
LDH_D 0.00
D_LACt2 0.00
FORti 0.00
PTAr 2.56e-14
ACKr -2.56e-14
ACt2r 0.00
ACALD -1.67e-14
ALCD2x -1.67e-14
ACALDt 0.00
ETOHt2r -1.67e-14
PIt2r 3.21
H2Ot -29.2
CO2t -22.8
NH4t 4.77
EX_lac__D_e (nd)
EX_for_e 0.00
EX_ac_e 0.00
EX_acald_e 0.00
EX_etoh_e 0.00
EX_pi_e -3.21
EX_h2o_e 29.2
EX_co2_e 22.8
EX_nh4_e -4.77
EX_mal__L_e (nd)
EX_fum_e 0.00
MALt2_2 0.00
FUMt2_2 0.00
THD2 0.00
NADTRHD 0.00
ADK1 0.00
ATPM 8.39
ATPS4r 45.5
O2t 21.8
EX_o2_e -21.8
CYTBD 43.6
EX_h_e 17.5
NADH16 38.5
HERE
FRD7 0.00
SUCDi 5.06
FUM 5.06
MDH 5.06
ME1 0.00
ME2 0.00
MALS 0.00
CS 6.01
ACONTa 6.01
ACONTb 6.01
ICL 0.00
ICDHyr 6.01
AKGDH 5.06
SUCOAS -5.06
SUCCt3 0.00
SUCCt2_2 0.00
EX_succ_e 0.00
AKGt2r 0.00
EX_akg_e 0.00
GLUDy -4.54
GLUt2r -3.32e-15
EX_glu__L_e (nd)
GLUSy 0.00
GLUN 0.00
GLNS 0.223
GLNabc 0.00
EX_gln__L_e (nd)
BIOMASS_Ecoli_core_w_GAM 0.874



If I understood my model correctly I am optimizing for the generation of NAD+. However, then I do not understand why the TCA cycle is "deactivated"