CVX101 Homework 7 solutions

1. *Gradient and Newton methods.* Consider the unconstrained problem

$$\text{minimize} \quad f(x) = -\textstyle\sum_{i=1}^m \log(1 - a_i^T x) - \sum_{i=1}^n \log(1 - x_i^2),$$

with variable $x \in \mathbf{R}^n$, and $\mathbf{dom}\, f = \{x \mid a_i^T x < 1,\ i = 1, \ldots, m,\ |x_i| < 1,\ i = 1, \ldots, n\}$. This is the problem of computing the analytic center of the set of linear inequalities

$$a_i^T x \leq 1, \quad i = 1, \ldots, m, \qquad |x_i| \leq 1, \quad i = 1, \ldots, n.$$

Note that we can choose $x^{(0)} = 0$ as our initial point. You can generate instances of this problem by choosing a_i from some distribution on $\mathbf{R}^n$.

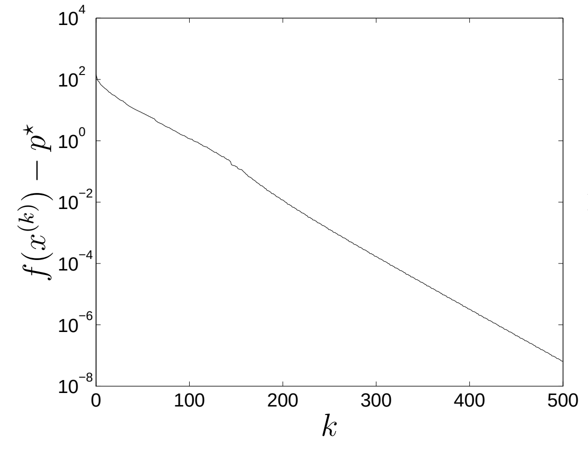
(a) Use the gradient method to solve the problem, using reasonable choices for the backtracking parameters, and a stopping criterion of the form $\|\nabla f(x)\|_2 \leq \eta$. Plot the objective function and step length versus iteration number. (Once you have determined $p^\star$ to high accuracy, you can also plot $f - p^\star$ versus iteration.) Experiment with the backtracking parameters α and β to see their effect on the total number of iterations required. Carry these experiments out for several instances of the problem, of different sizes.

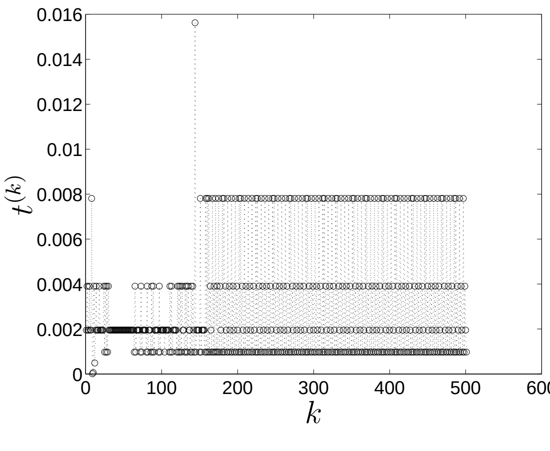
(b) Repeat using Newton's method, with stopping criterion based on the Newton decrement λ^2. Look for quadratic convergence. You do not have to use an efficient method to compute the Newton step, as in exercise 9.27; you can use a general purpose dense solver, although it is better to use one that is based on a Cholesky factorization.

Hint. Use the chain rule to find expressions for $\nabla f(x)$ and $\nabla^2 f(x)$.

Solution.

(a) *Gradient method.* The figures show the function values and step lengths versus iteration number for an example with $m = 200$, $n = 100$. We used $\alpha = 0.01$, $\beta = 0.5$, and exit condition $\|\nabla f(x^{(k)})\|_2 \leq 10^{-3}$.



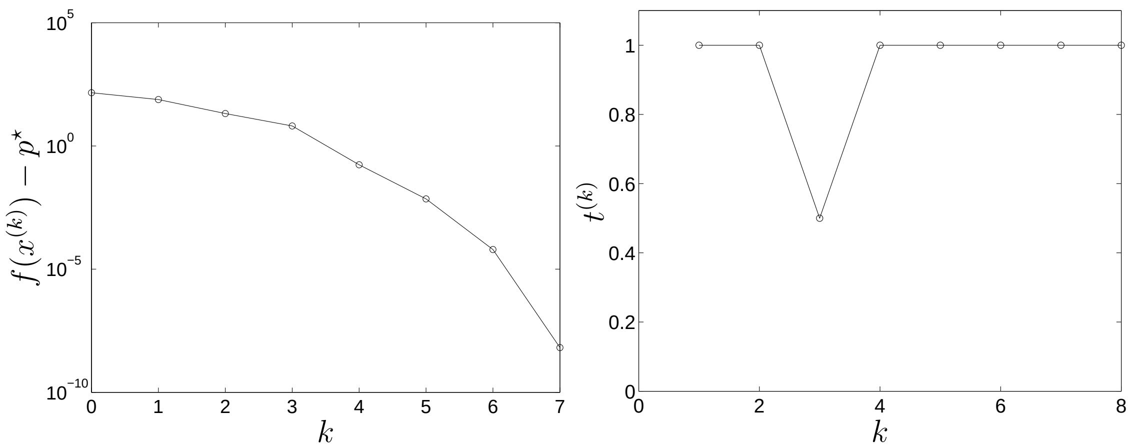


The following is a Matlab implementation.

```
ALPHA = 0.01;
BETA = 0.5;
MAXITERS = 1000;
GRADTOL = 1e-3;

x = zeros(n,1);
for iter = 1:MAXITERS
   val = -sum(log(1-A*x)) - sum(log(1+x)) - sum(log(1-x));
   grad =  A'*(1./(1-A*x)) - 1./(1+x) + 1./(1-x);
   if norm(grad) < GRADTOL, break; end;
   v = -grad;
   fprime = grad'*v;
   t = 1;  while ((max(A*(x+t*v)) >= 1) | (max(abs(x+t*v)) >= 1)),
      t = BETA*t;
   end;
   while ( -sum(log(1-A*(x+t*v))) - sum(log(1-(x+t*v).^2)) >  ...
           val + ALPHA*t*fprime )
      t = BETA*t;
   end;
   x = x+t*v;
end;
```

(b) *Newton method.* The figures show the function values and step lengths versus iteration number for the same example. We used $\alpha = 0.01$, $\beta = 0.5$, and exit condition $\lambda(x^{(k)})^2 \leq 10^{-8}$.



The following is a Matlab implementation.

```
ALPHA = 0.01;
BETA = 0.5;
MAXITERS = 1000;
```

```
NTTOL  = 1e-8;

x = zeros(n,1);
for iter = 1:MAXITERS
   val = -sum(log(1-A*x)) - sum(log(1+x)) - sum(log(1-x));
   d = 1./(1-A*x);
   grad =  A'*d - 1./(1+x) + 1./(1-x);
   hess =  A'*diag(d.^2)*A + diag(1./(1+x).^2 + 1./(1-x).^2);
   v = -hess\grad;
   fprime = grad'*v;
   if abs(fprime) < NTTOL, break; end;
   t = 1;  while ((max(A*(x+t*v)) >= 1) | (max(abs(x+t*v)) >= 1)),
      t = BETA*t;
   end;
   while ( -sum(log(1-A*(x+t*v))) - sum(log(1-(x+t*v).^2)) >  ...
          val + ALPHA*t*fprime )
      t = BETA*t;
   end;
   x = x+t*v;
end;
```

2. *Efficient solution of basic portfolio optimization problem.* This problem concerns the simplest possible portfolio optimization problem:

$$\begin{array}{ll} \text{maximize} & \mu^T w - (\lambda/2) w^T \Sigma w \\ \text{subject to} & \mathbf{1}^T w = 1, \end{array}$$

with variable $w \in \mathbf{R}^n$ (the normalized portfolio, with negative entries meaning short positions), and data μ (mean return), $\Sigma \in \mathbf{S}^n_{++}$ (return covariance), and $\lambda > 0$ (the risk aversion parameter). The return covariance has the factor form $\Sigma = FQF^T + D$, where $F \in \mathbf{R}^{n \times k}$ (with rank K) is the *factor loading matrix*, $Q \in \mathbf{S}^k_{++}$ is the factor covariance matrix, and D is a diagonal matrix with positive entries, called the *idiosyncratic risk* (since it describes the risk of each asset that is independent of the factors). This form for Σ is referred to as a 'k-factor risk model'. Some typical dimensions are $n = 2500$ (assets) and $k = 30$ (factors).

   (a) What is the flop count for computing the optimal portfolio, if the low-rank plus diagonal structure of Σ is *not* exploited? You can assume that $\lambda = 1$ (which can be arranged by absorbing it into Σ).

   (b) Explain how to compute the optimal portfolio more efficiently, and give the flop count for your method. You can assume that $k \ll n$. You do not have to give the best method; any method that has linear complexity in n is fine. You can assume that $\lambda = 1$.

Hints. You may want to introduce a new variable $y = F^T w$ (which is called the vector of factor exposures). You may want to work with the matrix

$$G = \begin{bmatrix} \mathbf{1} & F \\ 0 & -I \end{bmatrix} \in \mathbf{R}^{(n+k)\times(1+k)},$$

treating it as dense, ignoring the (little) exploitable structure in it.

(c) Carry out your method from part (b) on some randomly generated data with dimensions $n = 2500$, $k = 30$. For comparison (and as a check on your method), compute the optimal portfolio using the method of part (a) as well. Give the (approximate) CPU time for each method, using `tic` and `toc`. *Hints.* After you generate D and Q randomly, you might want to add a positive multiple of the identity to each, to avoid any issues related to poor conditioning. Also, to be able to invert a block diagonal matrix efficiently, you'll need to recast it as sparse.

(d) *Risk return trade-off curve.* Now suppose we want to compute the optimal portfolio for M values of the risk aversion parameter λ. Explain how to do this efficiently, and give the complexity in terms of M, n, and k. Compare to the complexity of using the method of part (b) M times. *Hint.* Show that the optimal portfolio is an affine function of $1/\lambda$.

Solution.

(a) We compute the optimal portfolio by solving the (linear) KKT system

$$\begin{bmatrix} \Sigma & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} w \\ \nu \end{bmatrix} = \begin{bmatrix} \mu \\ 1 \end{bmatrix},$$

where $\nu \in \mathbf{R}$ is the Lagrange multiplier associated with the the budget constraint $\mathbf{1}^T w = 1$. The flop count is $(1/3)(n+1)^3$, which after dropping non-dominant terms is the same as $(1/3)n^3$.

(b) There are several ways to describe an efficient method, and they are all related. Here is one. We introduce a new variable $y = F^T w$ (which represents the factor exposures for the portfolio w), and solve the problem

$$\begin{array}{ll} \text{maximize} & \mu^T w - (1/2) w^T D w - (1/2) y^T Q y \\ \text{subject to} & \mathbf{1}^T w = 1, \quad F^T w = y, \end{array}$$

with variables w, y. The KKT system is

$$\begin{bmatrix} D & 0 & \mathbf{1} & F \\ 0 & Q & 0 & -I \\ \mathbf{1}^T & 0 & 0 & 0 \\ F^T & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ y \\ \nu \\ \kappa \end{bmatrix} = \begin{bmatrix} \mu \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

We will use standard block elimination, as described in §C.4 of the textbook. To match the notation of that section, we define

$$A_{11} = \begin{bmatrix} D & 0 \\ 0 & Q \end{bmatrix}, \quad A_{12} = \begin{bmatrix} \mathbf{1} & F \\ 0 & -I \end{bmatrix}, \quad A_{21} = A_{12}^T, \quad A_{22} = 0,$$

(with $A_{22} \in \mathbf{R}^{(k+1)\times(k+1)}$),

$$b_1 = \begin{bmatrix} \mu \\ 0 \end{bmatrix}, \qquad b_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(where the 0s in these expressions have dimension k), and

$$x_1 = \begin{bmatrix} w \\ y \end{bmatrix}, \qquad x_2 = \begin{bmatrix} \nu \\ \kappa \end{bmatrix}.$$

Now we solve the linear equations above by elimination. The Schur complement is

$$\begin{aligned} S &= A_{22} - A_{21}A_{11}^{-1}A_{12} \\ &= -\begin{bmatrix} \mathbf{1}^T & 0 \\ F^T & -I \end{bmatrix} \begin{bmatrix} D^{-1} & 0 \\ 0 & Q^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & F \\ 0 & -I \end{bmatrix} \\ &= -\begin{bmatrix} \mathbf{1}^T D^{-1}\mathbf{1} & \mathbf{1}^T D^{-1} F \\ F^T D^{-1}\mathbf{1} & F^T D^{-1} F + Q^{-1} \end{bmatrix}. \end{aligned}$$

The dominant part of forming S is computing $F^T D^{-1} F$, which costs nk^2 flops. (Computing Q^{-1} is order k^3, which is non-dominant.) We also compute the reduced righthand side,

$$\begin{aligned} \tilde{b} &= b_2 - A_{21}A_{11}^{-1}b_1 \\ &= \begin{bmatrix} 1 - \mathbf{1}^T D^{-1}\mu \\ -F^T D^{-1}\mu \end{bmatrix}, \end{aligned}$$

with negligible cost (even ignoring the fact that $F^T D^{-1}$ was already computed in forming S). Now we solve the reduced system $Sx_2 = \tilde{b}$, which costs $(1/3)(k+1)^3$, and so is non-dominant. We break up x_2 into its first component, which is ν, and its last k components, κ. Finally, we find x_1 from

$$\begin{aligned} x_1 &= A_{11}^{-1}(b_1 - A_{12}x_2) \\ &= \begin{bmatrix} D^{-1}(\mu - \nu\mathbf{1} - F\kappa) \\ Q^{-1}\kappa \end{bmatrix}, \end{aligned}$$

which has negligible cost. In fact, we only need the top part of x_1:

$$w = D^{-1}(\mu - \nu\mathbf{1} - F\kappa).$$

The overall flop count is nk^2 flops, quite a bit more efficient than the other method, which requires $(1/3)n^3$ flops.

(c) The code below carries out both methods. The times for the two methods are around 2 seconds (slow method) and 60 milliseconds (fast method). With a real language (*e.g.*, C or C++), the difference would be even more dramatic.

(d) Now we put λ back in to the KKT system:

$$\begin{bmatrix} \lambda\Sigma & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} w \\ \nu \end{bmatrix} = \begin{bmatrix} \mu \\ 1 \end{bmatrix},$$

Divide the first block row by λ, and change variables to $\tilde{\nu} = \nu/\lambda$ to get

$$\begin{bmatrix} \Sigma & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{bmatrix} w \\ \tilde{\nu} \end{bmatrix} = \begin{bmatrix} \mu/\lambda \\ 1 \end{bmatrix}.$$

This shows that the optimal portfolio is an affine function of $1/\lambda$. Therefore, w has the form $w = w_0 + (1/\lambda)w_1$, where w_0 is the optimal portfolio for $\mu = 0$ (or $\lambda = \infty$), and $w_0 + w_1$ is the optimal portfolio for $\lambda = 1$. We just solve for the optimal portfolio for these *two* values of λ (or any other two, for that matter!), and then we can construct the solution for any other value with $2n$ flops.

Thus we perform two solves, which costs $(2/3)nk^2$; then we need $2nM$ flops to find all the optimal portfolios. The complexity couldn't be any smaller, since to just write down the M optimal portfolios, we touch nM real numbers.

3. *Sizing a gravity feed water supply network.* A water supply network connects water supplies (such as reservoirs) to consumers via a network of pipes. Water flow in the network is due to gravity (as opposed to pumps, which could also be added to the formulation). The network is composed of a set of n nodes and m directed edges between pairs of nodes. The first k nodes are supply or reservoir nodes, and the remaining $n-k$ are consumer nodes. The edges correspond to the pipes in the water supply network.

We let $f_j \geq 0$ denote the water flow in pipe (edge) j, and h_i denote the (known) altitude or height of node i (say, above sea level). At nodes $i = 1, \ldots, k$, we let $s_i \geq 0$ denote the flow into the network from the supply. For $i = 1, \ldots, n-k$, we let $c_i \geq 0$ denote the water flow taken out of the network (by consumers) at node $k+i$. Conservation of flow can be expressed as

$$Af = \begin{bmatrix} -s \\ c \end{bmatrix},$$

where $A \in \mathbf{R}^{n\times m}$ is the incidence matrix for the supply network, given by

$$A_{ij} = \begin{cases} -1 & \text{if edge } j \text{ leaves node } i \\ +1 & \text{if edge } j \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

We assume that each edge is oriented from a node of higher altitude to a node of lower altitude; if edge j goes from node i to node l, we have $h_i > h_l$. The pipe flows are determined by

$$f_j = \frac{\alpha \theta_j R_j^2 (h_i - h_l)}{L_j},$$

where edge j goes from node i to node l, $\alpha > 0$ is a known constant, $L_j > 0$ is the (known) length of pipe j, $R_j > 0$ is the radius of pipe j, and $\theta_j \in [0, 1]$ corresponds to the valve opening in pipe j.

Finally, we have a few more constraints. The supply feed rates are limited: we have $s_i \leq S_i^{\max}$. The pipe radii are limited: we have $R_j^{\min} \leq R_j \leq R_j^{\max}$. (These limits are all known.)

(a) *Supportable consumption vectors.* Suppose that the pipe radii are fixed and known. We say that $c \in \mathbf{R}_+^{n-k}$ is supportable if there is a choice of f, s, and θ for which all constraints and conditions above are satisfied. Show that the set of supportable consumption vectors is a polyhedron, and explain how to determine whether or not a given consumption vector is supportable.

(b) *Optimal pipe sizing.* You must select the pipe radii R_j to minimize the cost, which we take to be (proportional to) the total volume of the pipes, $L_1 R_1^2 + \cdots + L_m R_m^2$, subject to being able to support a set of consumption vectors, denoted $c^{(1)}, \ldots, c^{(N)}$, which we refer to as consumption scenarios. (This means that any consumption vector in the convex hull of $\{c^{(1)}, \ldots, c^{(N)}\}$ will be supportable.) Show how to formulate this as a convex optimization problem. *Note.* You are asked to choose *one* set of pipe radii, and N sets of valve parameters, flow vectors, and source vectors; one for each consumption scenario.

(c) Solve the instance of the optimal pipe sizing problem with data defined in the file `grav_feed_network_data.m`, and report the optimal value and the optimal pipe radii. The columns of the matrix C in the data file are the consumption vectors $c^{(1)}, \ldots, c^{(N)}$.

Hint. $-A^T h$ gives a vector containing the height differences across the edges.

Solution.

(a) Let P be the set of vectors $(f, \theta, s, c) \in \mathbf{R}^m \times \mathbf{R}^m \times \mathbf{R}^k \times \mathbf{R}^{n-k}$ that satisfy

$$Af = \begin{bmatrix} -s \\ c \end{bmatrix}, \quad 0 \preceq \theta \preceq 1, \quad 0 \preceq f, \quad 0 \preceq c, \quad 0 \preceq s \preceq S^{\max}, \quad f = D\theta,$$

where $D = D_1 D_2$, with $D_1 = -\alpha\, \mathbf{diag}(A^T h)\, \mathbf{diag}(1/L_1, \ldots, 1/L_m)$ and $D_2 = \mathbf{diag}(R_1^2, \ldots, R_m^2)$. The set P is clearly a polyhedron. Therefore, the set of supportable consumption vectors c, which is just a projection of P, is also a polyhedron.

To determine whether or not a consumption vector $c \in \mathbf{R}_+^{n-k}$ is supportable, we can solve the following feasibility problem:

$$\begin{array}{ll} \text{find} & (f, \theta, s) \\ \text{subject to} & 0 \preceq f \\ & 0 \preceq \theta \preceq 1 \\ & Af = (-s, c) \\ & 0 \preceq s \preceq S^{\max} \\ & f = D\theta. \end{array}$$

(b) The problem that we would like to solve is

$$\begin{array}{ll} \text{minimize} & L_1 R_1^2 + \cdots + L_m R_m^2 \\ \text{subject to} & R^{\min} \preceq R \preceq R^{\max} \\ & 0 \preceq f^{(p)}, \quad p = 1, \ldots, N \\ & 0 \preceq \theta^{(p)} \preceq 1, \quad p = 1, \ldots, N \\ & Af^{(p)} = (-s^{(p)}, c^{(p)}) \quad p = 1, \ldots, N \\ & 0 \preceq s^{(p)} \preceq S^{\max}, \quad p = 1, \ldots, N \\ & f^{(p)} = D_1 D_2 \theta^{(p)}, \quad p = 1, \ldots, N. \end{array}$$

This problem, however, is not convex in the variables $R, f^{(p)}, s^{(p)}, \theta^{(p)}$ since the last inequality constraint is not a convex constraint.

The first trick is to not use the valve parameters directly and replace θ with $\mathbf{1}$, *i.e.*

$$f^{(p)} \leq D_1 D_2 \mathbf{1}.$$

After finding the flows and pipe radii, we can simply set

$$\theta^{(p)} = (D_1 D_2)^{-1} f^{(p)} \in [0, 1]^m.$$

This is a valid transformation, since D_1 and D_2 are diagonal and invertible, so θ can be recovered uniquely.

However, the constraint $f^{(p)} \leq D_1 D_2 \mathbf{1}$ is still not convex because D_2 is quadratic in R. The second trick is to do a change of variables $z = D_2 \mathbf{1} = (R_1^2, \ldots, R_m^2)$. This is a valid change of variables since $R \succeq 0$.

Using these transformations, we obtain the equivalent problem

$$\begin{array}{ll} \text{minimize} & L_1 z_1 + \cdots + L_m z_m \\ \text{subject to} & (R_j^{\min})^2 \leq z_j \leq (R_j^{\max})^2, \quad j = 1, \ldots, m \\ & 0 \preceq f^{(p)}, \quad p = 1, \ldots, N \\ & Af^{(p)} = (-s^{(p)}, c^{(p)}) \quad p = 1, \ldots, N \\ & 0 \preceq s^{(p)} \preceq S^{\max}, \quad p = 1, \ldots, N \\ & f^{(p)} \preceq D_1 z, \quad p = 1, \ldots, N, \end{array}$$

which is now a convex problem.

(c) The following Matlab script solves the problem.

The optimal value was found to be 313.024 and the pipe radii are given in the table below.

pipe	radius
1	0.5
2	0.5
3	2.21
4	0.5
5	0.5
6	0.5
7	1.93
8	1.23
9	0.5
10	2.5
11	2.5
12	0.5
13	0.5
14	1.20
15	0.5
16	2.5
17	1.05
18	1.89
19	1.73
20	0.5

4. *Flux balance analysis in systems biology.* Flux balance analysis is based on a very simple model of the reactions going on in a cell, keeping track only of the gross rate of consumption and production of various chemical species within the cell. Based on the known stoichiometry of the reactions, and known upper bounds on some of the reaction rates, we can compute bounds on the other reaction rates, or cell growth, for example.

   We focus on m metabolites in a cell, labeled $M_1, \ldots, M_m$. There are n reactions going on, labeled $R_1, \ldots, R_n$, with nonnegative reaction rates $v_1, \ldots, v_n$. Each reaction has a (known) stoichiometry, which tells us the rate of consumption and production of the metabolites per unit of reaction rate. The stoichiometry data is given by the *stoichiometry matrix* $S \in \mathbf{R}^{m \times n}$, defined as follows: S_{ij} is the rate of production of M_i due to unit reaction rate $v_j = 1$. Here we consider consumption of a metabolite as negative production; so $S_{ij} = -2$, for example, means that reaction R_j causes metabolite M_i to be consumed at a rate $2v_j$.

   As an example, suppose reaction R_1 has the form $M_1 \to M_2 + 2M_3$. The consumption rate of M_1, due to this reaction, is v_1; the production rate of M_2 is v_1; and the production rate of M_3 is $2v_1$. (The reaction R_1 has no effect on metabolites $M_4, \ldots, M_m$.)

This corresponds to a first column of S of the form $(-1, 1, 2, 0, \ldots, 0)$.

Reactions are also used to model flow of metabolites into and out of the cell. For example, suppose that reaction R_2 corresponds to the flow of metabolite M_1 into the cell, with v_2 giving the flow rate. This corresponds to a second column of S of the form $(1, 0, \ldots, 0)$.

The last reaction, R_n, corresponds to biomass creation, or cell growth, so the reaction rate v_n is the cell growth rate. The last column of S gives the amounts of metabolites used or created per unit of cell growth rate.

Since our reactions include metabolites entering or leaving the cell, as well as those converted to biomass within the cell, we have conservation of the metabolites, which can be expressed as $Sv = 0$. In addition, we are given upper limits on *some* of the reaction rates, which we express as $v \preceq v^{\max}$, where we set $v_j^{\max} = \infty$ if no upper limit on reaction rate j is known. The goal is to find the maximum possible cell growth rate (*i.e.*, largest possible value of v_n) consistent with the constraints

$$Sv = 0, \qquad v \succeq 0, \qquad v \preceq v^{\max}.$$

The questions below pertain to the data found in `fba_data.m`.

(a) Find the maximum possible cell growth rate $G^\star$, as well as optimal Lagrange multipliers for the reaction rate limits. How sensitive is the maximum growth rate to the various reaction rate limits?

(b) *Essential genes and synthetic lethals.* For simplicity, we'll assume that each reaction is controlled by an associated gene, *i.e.*, gene G_i controls reaction R_i. Knocking out a set of genes associated with some reactions has the effect of setting the reaction rates (or equivalently, the associated $v^{\max}$ entries) to zero, which of course reduces the maximum possible growth rate. If the maximum growth rate becomes small enough or zero, it is reasonable to guess that knocking out the set of genes will kill the cell. An *essential gene* is one that when knocked out reduces the maximum growth rate below a given threshold $G^{\min}$. (Note that G_n is always an essential gene.) A *synthetic lethal* is a pair of non-essential genes that when knocked out reduces the maximum growth rate below the threshold. Find all essential genes and synthetic lethals for the given problem instance, using the threshold $G^{\min} = 0.2G^\star$.

Solution. There's not much to do here other than write the code for the problem. For determining the essential genes and synthetic lethals, we simply loop over pairs of reactions to zero out.

```
% flux balance analysis in systems biology

fba_data;
```

```
% part (a): find maximum growth rate
cvx_begin
variable v(n);
dual variable lambda;
maximize (v(n));
S*v == 0;
v >= 0;
lambda: v <= vmax;
cvx_end
Gstar=cvx_optval;
vopt=v;

[v vmax lambda]

% part (b): find essential genes and synthetic lethals
cvx_quiet('true');
G = zeros(n,n);
for i=1:n
for j=i:n
cvx_begin
variable v(n);
S*v == 0;
v >= 0;
v <= vmax;
v(i) == 0;
v(j) == 0;
maximize (v(n))
cvx_end
G(i,j)=cvx_optval;
G(j,i)=cvx_optval;
end
end

G<0.2*Gstar
```

We find the maximum growth rate to be $G^\star = 13.55$. Only three reactions are limited: R_1, R_3, and R_5. All other Lagrange multipliers are zero, of course; these have optimal values 0.5, 0.5, and 1.5, respectively. So it appears that the limit on reaction R_5 is the one that would have the largest effect on the optimal growth rate, for a small change in the limit.

We see that the essential genes are G_1 and G_9. The synthetic lethals are

$$\{G_2, G_3\}, \quad \{G_2, G_7\}, \quad \{G_4, G_7\}, \quad \{G_5, G_7\}.$$

5. *Online advertising displays.* When a user goes to a website, one of a set of n ads, labeled $1, \ldots, n$, is displayed. This is called an *impression.* We divide some time interval (say, one day) into T periods, labeled $t = 1, \ldots, T$. Let $N_{it} \geq 0$ denote the number of impressions in period t for which we display ad i. In period t there will be a total of $I_t > 0$ impressions, so we must have $\sum_{i=1}^n N_{it} = I_t$, for $t = 1, \ldots, T$. (The numbers I_t might be known from past history.) You can treat all these numbers as real. (This is justified since they are typically very large.)

   The revenue for displaying ad i in period t is $R_{it} \geq 0$ per impression. (This might come from click-through payments, for example.) The total revenue is $\sum_{t=1}^T \sum_{i=1}^n R_{it} N_{it}$. To maximize revenue, we would simply display the ad with the highest revenue per impression, and no other, in each display period.

   We also have in place a set of m contracts that require us to display certain numbers of ads, or mixes of ads (say, associated with the products of one company), over certain periods, with a penalty for any shortfalls. Contract j is characterized by a set of ads $\mathcal{A}_j \subseteq \{1, \ldots, n\}$ (while it does not affect the math, these are often disjoint), a set of periods $\mathcal{T}_j \subseteq \{1, \ldots, T\}$, a target number of impressions $q_j \geq 0$, and a shortfall penalty rate $p_j > 0$. The *shortfall* s_j for contract j is

$$s_j = \left(q_j - \sum_{t \in \mathcal{T}_j} \sum_{i \in A_j} N_{it} \right)_+ ,$$

   where $(u)_+$ means $\max\{u, 0\}$. (This is the number of impressions by which we fall short of the target value q_j.) Our contracts require a total penalty payment equal to $\sum_{j=1}^m p_j s_j$. Our net profit is the total revenue minus the total penalty payment.

   (a) Explain how to find the display numbers N_{it} that maximize net profit. The data in this problem are $R \in \mathbf{R}^{n \times T}$, $I \in \mathbf{R}^T$ (here I is the vector of impressions, not the identity matrix), and the contract data $\mathcal{A}_j$, $\mathcal{T}_j$, q_j, and p_j, $j = 1, \ldots, m$.

   (b) Carry out your method on the problem with data given in `ad_disp_data.m`. The data $\mathcal{A}_j$ and $\mathcal{T}_j$, for $j = 1, \ldots, m$ are given by matrices $A^{\text{contr}} \in \mathbf{R}^{n \times m}$ and $T^{\text{contr}} \in \mathbf{R}^{T \times m}$, with

$$A_{ij}^{\text{contr}} = \begin{cases} 1 & i \in \mathcal{A}_j \\ 0 & \text{otherwise,} \end{cases} \qquad T_{tj}^{\text{contr}} = \begin{cases} 1 & t \in \mathcal{T}_j \\ 0 & \text{otherwise.} \end{cases}$$

   Report the optimal net profit, and the associated revenue and total penalty payment. Give the same three numbers for the strategy of simply displaying in each period only the ad with the largest revenue per impression.

Solution.

(a) The constraints $N_{it} \geq 0$, $\sum_{i=1}^{n} N_{it} = I_t$ are linear. The contract shortfalls s_j are convex functions of N, so the net profit

$$\sum_{t=1}^{T}\sum_{i=1}^{n} R_{it}N_{it} - \sum_{j=1}^{m} p_j s_j$$

is concave. So we have a convex optimization problem.

(b) We can express the vector of shortfalls as

$$s = \left(q - \mathbf{diag}\left(A^{\mathrm{contr}\,T} N T^{\mathrm{contr}}\right)\right)_+$$

The following code solves the problem:

```
clear all;
% ad display problem
ad_disp_data;

% optimal display
cvx_begin
    variable N(n,T)
    s = pos(q-diag(Acontr'*N*Tcontr));
    maximize(R(:)'*N(:)-p'*s)
    subject to
        N >= 0;
        sum(N)' == I;
cvx_end
opt_s=s;
opt_net_profit = cvx_optval
opt_penalty = p'*opt_s
opt_revenue = opt_net_profit+opt_penalty

% display, ignoring contracts
cvx_begin
    variable N(n,T)
    maximize(R(:)'*N(:))
    subject to
        N >= 0;
        sum(N)' == I;
cvx_end
greedy_s = pos(q-diag(Acontr'*N*Tcontr));
greedy_net_profit = cvx_optval-p'*greedy_s
greedy_penalty = p'*greedy_s
greedy_revenue = cvx_optval
```

The performance of the optimal and greedy strategies is given in the table below. The optimal strategy gives up a bit of (gross) revenue, in order to pay substantially less penalty, and ends up way ahead of the greedy strategy.

strategy	revenue	penalty	net profit
optimal	268.23	37.66	230.57
greedy	305.10	232.26	72.84

6. *Ranking by aggregating preferences.* We have n objects, labeled $1, \dots, n$. Our goal is to assign a real valued rank r_i to the objects. A *preference* is an ordered pair (i, j), meaning that object i is preferred over object j. The ranking $r \in \mathbf{R}^n$ and preference (i, j) are *consistent* if $r_i \geq r_j + 1$. (This sets the scale of the ranking: a gap of one in ranking is the threshold for preferring one item over another.) We define the *preference violation* of preference (i, j) with ranking $r \in \mathbf{R}^n$ as

$$v = (r_j + 1 - r_i)_+ = \max\{r_j + 1 - r_i, 0\}.$$

We have a set of m preferences among the objects, $(i^{(1)}, j^{(1)}), \dots, (i^{(m)}, j^{(m)})$. (These may come from several different evaluators of the objects, but this won't matter here.)

We will select our ranking r as a minimizer of the total preference violation penalty, defined as

$$J = \sum_{k=1}^{m} \phi(v^{(k)}),$$

where $v^{(k)}$ is the preference violation of $(i^{(k)}, j^{(k)})$ with r, and ϕ is a nondecreasing convex penalty function that satisfies $\phi(u) = 0$ for $u \leq 0$.

(a) Make a (simple, please) suggestion for ϕ for each of the following two situations:

i. We don't mind some small violations, but we really want to avoid large violations.

ii. We want as many preferences as possible to be consistent with the ranking, but will accept some (hopefully, few) larger preference violations.

(b) Find the rankings obtained using the penalty functions proposed in part (a), on the data set found in `rank_aggr_data.m`. Plot a histogram of preference violations for each case and *briefly* comment on the differences between them. Give the number of positive preference violations for each case. (Use `sum(v>0.001)` to determine this number.)

Remark. The objects could be candidates for a position, papers at a conference, movies, websites, courses at a university, and so on. The preferences could arise in several ways. Each of a set of evaluators provides some preferences, for example by rank ordering a subset of the objects. The problem can be thought of as aggregating the preferences given by the evaluators, to come up with a composite ranking.

Solution.

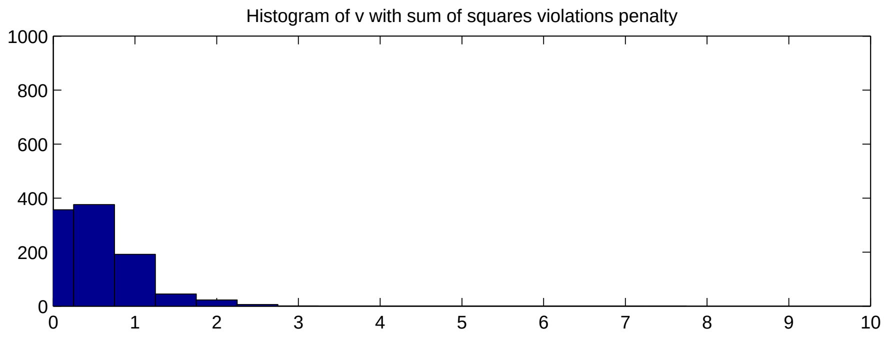
(a) Here are some simple suggestions. For the first case (i), we take a quadratic penalty (for positive violation): $\phi(u) = u_+^2$. For the second case (ii), we take a linear penalty (for positive violations): $\phi(u) = u_+$. If the violations were two-sided, these would correspond to an ℓ_2 norm and an ℓ_1 norm, respectively.

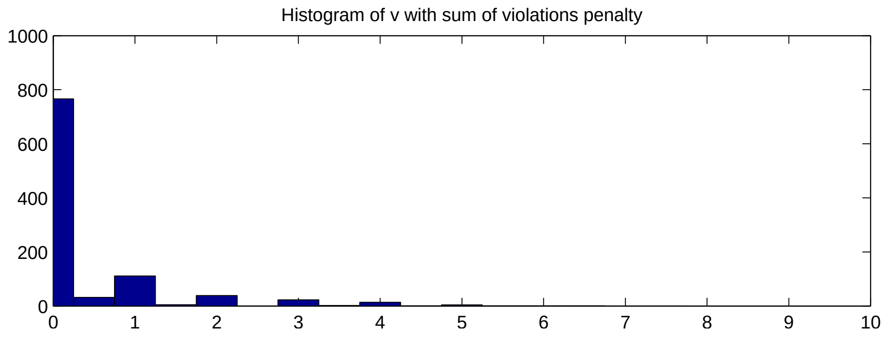
(b) The problem that we need to solve is

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^m \phi((v^{(k)})_+) \\ \text{subject to} & v^{(k)} = r_{j^{(k)}} + 1 - r_{i^{(k)}}. \end{array}$$

The following Matlab code solves the problem for the particular problem instance using the penalty functions proposed in part (a):

The resulting residual histograms are shown below.





For case (i), the violations are often small, but rarely zero; of the 1000 preferences given, 781 are violated. But the largest violation is below 3. For case (ii), only 235 preferences are violated, but we have a few larger violations, with values up to around 7.