

Exam 1

Due: 2:30 PM April 10, 2008

- **You must work alone on this exam. If you have any questions, ask the TAs or the professor, but do not discuss the exam at all, with anyone else.** Your work should be based on the material in the course and the course materials (including everything on the cs004 website and discussion board); do not look at other sources, including the Web.
- **Do not post to the discussion board while the exam is out.** Visit a TA on hours or mail questions to `cs004tas@cs.brown.edu`. Read the discussion board, however, as clarifications will be posted there.
- **To receive a grade above 0 on the exam, abide by the following statement. Then print out a copy of this page, sign the statement and turn it in to the cs004 hand-in bin (the one on the second floor by the stairs). We really won't give you a grade on the exam if this is not done.**

All work on this exam is my own. I have not discussed the problems or their solutions with anyone except the course staff. I did not consult any materials other than those officially provided by the course staff.

Signed: X_____

Prob.	Poss. Points	Points	Grader
1	20		
2	15		
3	25		
4	20		
5	20		
Total	100		

Tips

- For all the problems that require code, **do not change the names of any procedures from those given in the problem**. You are welcome to define and use any helper procedures you want. You are also welcome to use anything we've defined or used in any previous lecture or assignment.
- Simplify your code and make sure it's clean and appropriately commented. As in homeworks, we are always looking for the best answer on exam problems.
- Don't wait until the last minute. Get started early, this way you can take advantage of TA hours throughout the week.

Submitting your exam

To submit your code electronically, make sure that in your `home` directory you have a `course` directory and that in that there is a `cs004` directory. You should save your code in `.m` files and your written responses in `.txt` files, then save them in a directory in your `cs004` directory called `midterm`. To summarize, you will place all your work into the directory

```
/u/<login>/course/cs004/midterm
```

Don't forget to change <login> to be your login name. After you have your work saved in the proper directory you can run the hand-in script. Open a terminal, then type

```
cs004_handin midterm
```

If you have any questions about submitting your work, email your question to `cs004tas@cs.brown.edu` or ask a TA during hours. No late submissions will be accepted.

Problem 1

“Great fleas have little fleas upon their backs to bite ’em, And little fleas have lesser fleas, and so ad infinitum. ” - Augustus De Morgan

Just as with fleas, little fish get eaten by bigger fish. And when they do, they don't get the opportunity to reproduce, and there's less food for the big fish next year. The following equations describe this situation:

$$\begin{aligned} s_{new} &= s_{old} + s_{old}(A - C \cdot b_{old}) \\ b_{new} &= b_{old} - b_{old}(G - H \cdot s_{old}) \end{aligned}$$

where s is the number of small fish, and b is the number of big fish. The constants A, C, G , and H determine how the populations are related. We can rewrite the first equation as:

$$s_{new} - s_{old} = A \cdot s_{old} - C \cdot s_{old} \cdot b_{old}$$

so that we see that the change in the population of the small fish

- depends on a constant A , which determines the rate of growth in the absence of bigger fish that eat the small ones;
- depends on a constant C times the product of the number of small and big fish, since we assume that big fish eat small ones at a rate proportional to how often they come in contact (which is proportional to the number of each kind of fish); and
- doesn't take into account the possibility of the small fish dying of old age or other factors.

We can rewrite the second equation as:

$$b_{new} - b_{old} = -G \cdot b_{old} + H \cdot s_{old} \cdot b_{old}$$

so that we see that the change in the population of the big fish

- depends on a constant G , which determines the proportion of big fish that die from old age; and

→ depends on a constant H times the product of the number of small and big fish, since we assume that the population of big fish grows at a rate proportional to how often they come in contact with small fish (which is proportional to the number of each kind of fish).

- a. (10 points) Write a function

```
function [slist, blist] = simulateFish(s, b, A, C, G, H, n)
```

which simulates the changing population of fish over the course of n years, starting with initial populations s and b ; the outputs should contain the population of the fish in each of the n years, so `slist(1) = s`, and `blist(1) = b`.

Note: You should solve this problem with a `for`-loop; there's no easy way to do it with matrix-at-a-time operations.

- b. (5 points) Write a second function

```
function plotFish(slist, blist)
```

that, given two equal-length vectors of fish populations, plots them both on the same graph, using different color lines, suitable labels and key, has a title, etc. — something you'd be proud to put in a scientific paper.

- c. (5 points) Write a script file called `testFish.m` that tests your two functions with the following values:

$$s = 11, b = 20, A = .12, C = .006, G = .1, H = .01, n = 100$$

You should see the populations of small and large fish varying somewhat over time. To keep all the numbers manageable, we've expressed the number of small and big fish in millions, so $s = 11$ means "there are 11 million small fish", and similarly for b . That means that it makes perfect sense to write something like $s = 4.5538$, which means "the number of small fish is 4,553,800." In the comments of this file, describe qualitatively how the populations vary over a period of 100 years; describe the relationship between peaks in the small fish and large fish populations.

Problem 2

(15 points) You are the only dating expert for the Ja'arsblattts on the planet of Xzfarjima. The Ja'arsblattt population has only one gender (like amoebae). You will match each individual based on its responses to 63 'yes or no' questions. To do this, you will be given a matrix of every candidate's answers to these 63 questions. When asked to find a match for a new Ja'arsblatttian, you will search the matrix for the candidate with the most answers in common. Due to the aging of the Ja'arsblatttian population, more of them are becoming eligible singles every day. Your dating service is rapidly growing more popular, so you need a Matlab function `findMate(X)` that will help you do your job. `X` is the vector of preferences (the list of yes/no answers, encoded as ones and zeros) of the Ja'arsblattt to match. There is a csv file which holds the responses to the 63 questions for 125 Ja'arsblatttian individuals, available at `/course/cs004/pub/midterm/dating.csv`. (It's an array of 125×63 ones and zeros).

To complete this problem, do the following:

- Import `/course/cs004/pub/midterm/dating.csv` using `csvread`. This is the matrix of the candidates' answers.
- Write a function `best_mate = findMate(X)` that takes in a vector `X` of 63 yes/no answers for a new Ja'arsblattt and returns the row index of the best matching candidate in the dating array. If there is more than one best possible match, you need only return one of them.

The best solution to this problem does not use loops or if-statements. A solution that uses loops or if-statements will earn partial credit, but not full credit.

Problem 3

Suppose that some company manufactures widgets. The cost of manufacturing x widgets is given by a function $C(x)$. The market price per widget, assuming that there are x of them made, is $p(x)$. If all x of the widgets are sold, the total revenue to the company is $x \cdot p(x)$. However, there may also be a tax to pay on the items. The amount of tax per item is t . The total profit for the company is then:

$$U(x, t) = x \cdot p(x) - C(x) - t \cdot x$$

For this problem, please put the code for parts a-d in a file called `widgets.m`. You may create additional M-files for helper functions if you so choose.

It's common to assume that the price $p(x)$ decreases with x . We'll assume that:

$$p(x) = 1 + \frac{1}{\sqrt{x}}$$

- a. (5 points) Plot this price function for $x = 1$ to 100. Label everything nicely.
- b. (5 points) It's also common to assume that the cost of production (per item) goes down as the number of items produced increases. We'll assume that:

$$C(x) = x \cdot \left(1 + \frac{1}{x}\right)$$

(Note that the function $C(x)$ is not decreasing - it is equal to the cost to produce x items.) Plot $C(x)$ for $x = 1$ to 100. Again, label nicely.

- c. (10 points) Now make a surface plot (using `surf`) of the function $U(x, t)$ as x goes from 1 to 100, and as t , the tax rate, varies from 0 to $\frac{1}{10}$ in increments of $\frac{1}{100}$. Label nicely.
- d. (5 points) We generally assume that the company will adjust production to make the greatest possible profit, given a fixed rate of taxation. Examine the graph and describe in a comment how the optimal level of production varies as the tax rate increases. You can limit your discussion to what's going on in the graph, rather than an analysis of the economics behind it. Write in clear, complete sentences. Put your answer in comments in your `widgets.m` file.

Problem 4

A typical baseball player comes to bat about 600 times during the baseball season. Typical batting averages are about .275 (i.e. the player gets a hit about 27.5% of the time he comes to bat). One frequently hears of players “in a slump,” (i.e. failing to get a hit many times in a row). Let’s examine the question of whether slumps are in fact to be expected. Or, in other words, if you had a biased coin - one that came up heads only 27.5% of the time - would you expect to have long runs of “tails” in a sequence of 600 coin-tosses? The following code:

```
function y = coin(p, rows, cols)
    u = rand(rows, cols);
    y = (u < p);
end
```

generates a `rows x cols` array of coin-flips, with ‘1’ representing heads, and ‘0’ representing tails. A fraction `p` of them (on average) will be heads. Thus using `p = 0.5` results in “fair coin tosses”, while using `p = 0.275` gets biased coin tosses, in which only 27.5% of the coins will be expected to come up heads.

- a. (10 pts) Write a function

```
function y = slump(x)
% count longest sequence of zeros in the row vector x
```

that finds the length of the longest sequence of zeros in the row-vector `x`; if there are no zeros, the function should return 0. Though this can be done without a loop, it is a bit tricky. We will give full credit for a solution that uses a for-loop to pass *once* over the indices of `x`.

- b. (10 pts) Write a function

```
function y = testslump(p, n)
% compute the expected length of a slump in a player with
% batting average p, by simulating n seasons of 600 at-bats each.
```

that generates `n` sequences of 600 hits/misses with probability `p` of getting a hit, computes the length of the longest slump in each one,

and then average those lengths. Set y equal to that result, which (for large n) is the “expected length of a slump for a player with batting average p ”.

Note: You should generate all n sets of 600 hits/misses at once, but you’ll almost certainly have to use a loop to compute the length of the worst slump in each season.

Problem 5

(15 points) As the only crew member on Captain Redbeard's pirate ship with programming skills, you have been appointed chief ballistics scientist. Whenever an enemy ship is spotted on the horizon, a crew member in the crow's nest determines the distance to the target. She shouts two numbers, `target_near` and `target_far`, that indicate the distance from your ship to the enemy's near side and the enemy's far side, respectively. A crew member manning the cannons reports `v0`, the predicted initial velocity of the cannonball (based on the amount of gunpowder in the cannon). It is your job to calculate the angle above the horizon to set your cannons in order to hit the enemy, as well as how long it will take for the projectile to reach their ship (so you have enough time to prepare for battle, if necessary).

Write a function

```
function plotTrajectory(target_near, target_far, v0)
```

that, for a range of angles between 5 and 85 degrees in 10 degree increments, displays the predicted trajectories of a cannonball fired with initial velocity `v0`. All trajectories should be displayed on the same plot. Those trajectories that represent a 'hit' should be colored red, and those that represent a 'miss' should be colored blue. Any trajectory that has a range r such that `target_near` $\leq r \leq$ `target_far` is considered a hit. Use at least 100 time steps to plot each trajectory, and add a title and axis labels to the graph. Your function should also print the angles and the corresponding travel times of the 'hit' projectiles to the command window.

You may find the following excerpt from the *Pirate's Guide to Ballistics Calculations* helpful:

When yer seekin' teh hit an enemy ship, matey, heed ye these instructions:

$$\begin{aligned}x(t) &= v_0 \cos(\theta)t \\ y(t) &= v_0 \sin(\theta)t - \frac{1}{2}gt^2 \\ g &= 9.81m/s\end{aligned}$$

If it be travel time yeh want, use ye:

$$t = \frac{2v_0 \sin(\theta)}{g}.$$

