CS170A — Winter 2003

Final Examination

Due: noon Thursday March 20, 2003

D. Stott Parker (`stott@cs.ucla.edu`)

In order to obtain full credit, you are asked to submit hardcopy of all plots/graphs with your solutions.

You are to do this assignment individually, by yourself. The hardcopy you submit for grading must include a signed statement certifying that it is your work and only your work.
You may use any materials on the Internet except the work of other students in CS170A, past or present, as long as you state clearly what you have used.

All files below should be in the directory `http://www.seas.ucla.edu/cs170a/final/` (or, equivalently, `~cs170ata/www/final/` on SEASnet).

1 Newton's Method

The seventh Tchebycheff polynomial is defined by

$$T_7(z) \;=\; 64\,z^7 - 112\,z^5 + 56\,z^3 - 7\,z \qquad \text{for } z \in [-1,1].$$

This polynomial has 7 real roots in the interval $[-1,1]$.

1. Use Newton's method to find and print each of the 7 roots.

   Hint: It may help to automate the use of Newton's method. Remember that, given a function f, Newton's method attempts to find a value z such that $f(z) = 0$ by executing the iteration

   $$z_{k+1} \;=\; g(z_k) \qquad \text{where } g(z) \;=\; z \;-\; f(z)/f'(z).$$

   for some intelligently-chosen initial value z_0, which can often be chosen by looking at a plot of f.

   For example, the definition

   ```
   Newtonize  :=  f -> unapply( simplify( z - f(z)/diff(f(z),z) ), z);
   ```

   finds the function g for any given function f. Then, to compute $\sqrt[3]{a}$ using Newton's method for any nonnegative real value a, the following program will work:

   ```
   g := Newtonize( z -> z^3 - a );

   a := 2.0;    # or whatever we want to find the cube root of
   zold := 0.0;
   z  := 1.0;   # an inspired initial value
   while (abs((z-zold)/z) >= 10^(1-Digits))
      do
         zold := z;
         z := g(z)
      od;
   ```

2. Generate a plot showing to which of the 7 roots of $T_7(z)$ your Newton's method implementation converges when it is given each complex starting value $z_0 = (x + i\,y)$ in the 101×101 grid with

   $x \in [-1.00, -0.98, -0.96, \ldots, 0.96, 0.98, 1.00]$,
   $y \in [-1.00, -0.98, -0.96, \ldots, 0.96, 0.98, 1.00]$.

   Hint: In class we studied `~cs170ata/www/final/newton.m` (`http://www.seas.ucla.edu/cs170a/final/newton.m`) which generated a similar plot for $f(z) = z^3 - 1$.

2 Pink Noise

The program ~cs170ata/www/final/Noise.m (http://www.seas.ucla.edu/cs170a/final/Noise.m) studied in class generates white, brown, and pink noise. This problem studies two nice uses of pink noise.

1. **one-dimensional noise**

   The goal: to generate some pink noise, and play it as music.

   Pink noise is defined as noise having a power spectrum that decreases like $1/f$. That is, the f-th Fourier coefficient of the noise has complex absolute value approximating $a \cdot 1/f + b$, where a and b are constant. One theory argues that pink noise is aesthetically pleasing.

   Actually, 'off-pink' noise — having a power spectrum that decreases like $1/f^{\alpha}$, for some constant $\alpha \in [0, 2]$ — is also supposedly aesthetically pleasing. Brownian noise has $\alpha = 2$; White noise has $\alpha = 0$.

   Pink noise can be generated using something like the code below, which first generates a $1/f^{\alpha}$ spectrum, the performs an inverse FFT to get back a real (i.e., non-imaginary) signal.
   The program http://www.seas.ucla.edu/cs170a/final/Noise.m (also available on SEASnet as ~cs170ata/www/final/Noise.m) implemented this for $\alpha = 1$:

   ```
   alpha = 1;  % pure pink noise

   n = 1024;  % for example

   f = (1:n/2)';
   randphase = exp(-2i * pi .* rand(n/2,1));
   noise = randn(n/2,1) .* randphase;
   pinkspectrum = zeros(n,1);
   pinkspectrum(1:n/2) = (1./(f^.alpha)) .* noise;
   for i=2:n/2; pinkspectrum(n+2-i) = conj(pinkspectrum(i)); end

   close all
   loglog(f,abs(pinkspectrum(1:n/2)), f, 1./f, 'r')
   title('the power spectrum of pink noise drops off as 1/f')

   pink = real(ifft(pinkspectrum));

   plot(pink)
   title('pink noise')
   ```

   The function http://www.seas.ucla.edu/cs170a/final/NoiseMusic.m (also available on SEASnet as ~cs170ata/www/final/NoiseMusic.m) takes a vector of 'noise' as input, and plays it as music, by emitting each value in the input as a musical note. It is crude, so feel free to improve it. For example, you can make it vary the time duration of the notes; currently it gives all notes the same duration.

   Generate pink music sequences for several values of α in $[0, 2]$, and determine which ones are the most 'aesthetically pleasing' when their NoiseMusic soundtrack is played. For your favorite, plot the soundtrack (as a function) and annotate the plot, explaining what parts you like, and why.

2. **two-dimensional noise**

   The goal: generate some 2-dimensional pink noise, and plot it as a 'pink mountain'.

   Where the code above generates a vector of pink noise of length n, this time generate a *matrix* of Pink noise.

   You can do this by generalizing the code above to produce a $n \times n$ matrix PinkSpectrum with a central loop over i and j defining the matrix in something like the following way:

   ```
   ...
   for i=2:(n/2)
      for j=2:(n/2)
         f = sqrt(i^2 + j^2);
         PinkSpectrum(i,j) = (1/f)^alpha * PinkSpectrum(i,j) * randphase(i,j);
         ...
   ```

   After this PinkNoise should simply be the 2-dimensional inverse FFT (ifft2()) of PinkSpectrum. The 2-dimensional FFT of a matrix implements an FFT on each row and on each column of the matrix.

   After you have done this, generate a .gif image showing the resulting PinkNoise matrix as a 3-d surface.

   Hint: The following Maple code creates a .gif image showing the contents of an $n \times n$ matrix PinkNoise:

```
# store any plot output in file "myimage.gif", in 512x512 gif format:
plotsetup(gif,plotoutput="myimage.gif",plotoptions="height=512,width=512");

with(plots):
# plot the matrix in color, with row 1 at the top, and row n at the bottom
plot3d( PinkNoise[(n-ni+1),j],  j=1..n, ni=1..n, grid=[n,n],
             style=patch, color=PinkNoise[(n-ni+1),j] );
```

3 Period 3 implies Chaos

A basic result of chaos theory is that any iteration of the form

$$x_{k+1} \;=\; h(x_k) \qquad (x_k \text{ real})$$

is inherently unstable if h has **period 3** (that is, there is some real value x such that $h(h(h(x))) = x$, but $h(x) \neq x$). Instability here means that a small change in the initial value of the iteration can cause a large change in the output of the iteration. As a result, the iteration above can become something like a random number generator.

For example, suppose $h(x) = x^2 - 2$. This function has period 3, since for a particular value of x near -1.879, $h(h(h(x))) = x$. Specifically, with Maple:

```
> h := x -> x^2 - 2;
                                    2
                         h := x -> x  - 2

> x := -1.8793852416;
                        x := -1.8793852416

> h(x);
                            1.532088886

> h(h(x));
                            0.347296355

> h(h(h(x)));
                           -1.879385242

> h(h(h(x))) - x;
                                 0
```

Notice it is not necessary that $h(h(h(x))) = x$ for *all* values of x, but only that $h(h(h(x))) = x$ for *some* value of x.

Determine which of the following functions h (if any) have period 3:

- $h(x) \;=\; 2\,x\,(1-x)$
- $h(x) \;=\; 3\,x\,(1-x)$
- $h(x) \;=\; 4\,x\,(1-x)$

4 Optimization

In the third homework assignment, you were asked to find a least squares fit for the dataset giving 9 attributes of 391 cars called `autos.m` (or `autos.maple`). The start of `autos.m` looks like this:

```
% Columns of the autos matrix:
% 1 -- MPG
% 2 -- Cylinders
% 3 -- Displacement
% 4 -- Horsepower
```

```
% 5 -- Weight
% 6 -- Acceleration
% 7 -- Year
% 8 -- Origin
% 9 -- Make

% MPG Cyl Disp   Hpwr   Wt    Accel Yr Org Make
autos = [ ...
13.0, 8, 360.0, 175.0, 3821., 11.0, 73, 1, 1; ...
14.0, 8, 304.0, 150.0, 3672., 11.5, 73, 1, 1; ...
14.0, 8, 304.0, 150.0, 4257., 15.5, 74, 1, 1; ...
15.0, 6, 258.0, 110.0, 3730., 19.0, 75, 1, 1; ...
15.0, 8, 304.0, 150.0, 3892., 12.5, 72, 1, 1; ...
```

All values in the files are numeric (the last two columns use integer codes).

1. Use nonlinear least squares to find the coefficients α, β, γ, δ, a, b, c that yield a model

$$1/\text{MPG} \;=\; \alpha \cdot \text{horsepower}^{\,a} \;+\; \beta \cdot \text{displacement}^{\,b} \;+\; \gamma \cdot \text{weight}^{\,c} \;+\; \delta$$

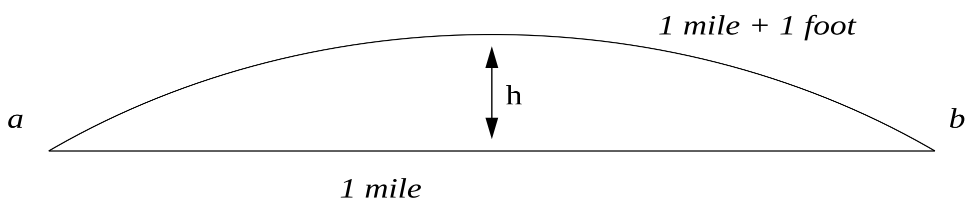
   with minimal error.

2. Is finding these coefficients a convex optimization problem? (as defined in `http://www.seas.ucla.edu/cs170a/lecture8/Optimization.pdf`)

3. How does the error of your model compare with that of the least-squares model, which assumes $a = b = c = 1$?

5 Train Tracks

Suppose a new steel train track is installed, running from point a to point b, which are 1 mile apart. To save money, the track was pinned down with spikes only at these two points. At night a CS170A student comes and cuts the track halfway between a and b, inserts an extra 1-foot section of track, and welds the pieces together. As a result of this insertion, the track bows up by some amount h in a circular arc. The situation looks like this:



The question is: what is h, the height of the track halfway between a and b?

6 Extra Credit

The class notes on Optimization (`http://www.seas.ucla.edu/cs170a/lecture8/Optimization.pdf`) includes a sample application of optimization in fitting a ‘mixture’ of two gaussian densities to a histogram of observed geyser duration times for *Old Faithful*.

If you look at the figure showing the fit of the gaussians to the histogram is not very good — the histogram is more ‘skewed’ than the gaussians.

In the class web site, the file `eruptiondurations.m` (Matlab version) gives the eruption duration times (in seconds) for Old Faithful. Find a better fit to the data by considering a different distribution.

For example, consider using the *lognormal* distribution instead of the normal (gaussian) distribution.