Linear Systems Due: 1400 Hours, 27 March 1998

October 11, 2000

1. Consider a domain in which the input and output are two-dimensional (M rows and N column) arrays of positive or negative integers, like this:

```
1-1 1-1 1 0
1 1 1 1 1 0
2-3 4 5 6 6
0 0 1 1 1 1
```

Recall that the convolution is closely related to the correlation, which in the current case is easier to think about. Both are "shift, multiply, add" operations. Rows and columns are numbered from 0,0 to $M-1, N-1$.

We shall consider that we have an input array I that is correlated with the "system" or "operator" array T to yield an output array O. Generally we can think of the input as being big and the operator as being small. I think of the operator being slid over the input, and for each offset position its elements multiply those elements of the input array that they are on top of, and the resultant products are summed up to give the output for that particular offset.

If we imagine that I, T, O are embedded in

infinite arrays of zeros, with infinite "padding" of zeroes above, below, and to their left and right, we can write the correlation like this:

$$O(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} T(i,j) I(i+x, j+y).$$

A. Of course, most of the output with these infinite arrays is boring! What is the size of the output matrix O *for which* I *and* T *actually ovelap and generate some non-trivial output?*.

B. Write the C or C++ code that implements this equation *for finite two-dimensional arrays.* There will be six array bounds, named IRows, ICols, TRows, TCols, ORows, OCols, giving the sizes of the arrays. Generate, as in A, all the 'interesting' output, in which I, T overlap.

C. Does the formula in A and your code in B implement a linear system that obeys "homogeneity" $f(ax) = af(x)$ and "superposition" $f(x+y) = f(x) + f(y)$ laws?

D. Now suppose that you want to keep T much smaller than I, and you want O to be the same size as I, perhaps so you can apply several Ts in a row without having the output get unboundedly large. What choices do you have in implementing such a restriction? Is the system still linear? If your users expected a linear system, how would you describe the difference between the ideal one and your implementation?

E. Consider the input matrix I:

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 1 1 1 0 0
0 0 1 1 1 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```

```
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
```

What is the result of sending these two Is to the following operators $T1...T5$ (these are 1×2, 2×1, or 2×2 arrays). Say for each $T1...T5$ below whether it is acting like a low-pass filter, high-pass filter, differentiator, integrator, sharpener, smoother. Say which and why: several may be true. Thus ten answers are required.

```
T1.       T2.     T3.      T4.       T5.

          -1      -1  1     1 1     1 1
-1 1       1       1 -1             1 1
```

2.

I am making available a very quick and dirty simulation of a mass, spring, friction system. It is on the webpage for you to copy but the guts are simply this:

```
force = input + drive_force(t) - spring*pos - frict*fabs(vel);
accel = force/mass;
dv = accel*dt;
vel += dv;
pos += vel*dt + (0.5)* accel*dt*dt;
```

This is the most straightforward and stupid way to "integrate a differential equation" (to derive position and velocity from force, in this case). Industry standard is some form of Runga-Kutta method, which you get automatically in Mathematica, or from *Numerical Recipes*. We'll see what goes wrong soon, but at least the above "looks" reasonable, no? Another thing wrong with this simulation, which is related to the fixed `dt` above, is that the scaling of the units is completely random.

I wrote this in C and run gnuplot on the output. It occurs to me you could equally well use matlab, since you know that already.

My program takes input that sets up all the system parameters: the mass, friction coefficient, spring coefficient; the initial position and velocity of the mass; the gains of the controller (for me, P,I, and D) and the integrator decay constant; a delta time dt and a total time t for the simulation to run, etc.

A. Forgetting about the controller, and with zero exterior applied force, set up a mass-spring system with no friction. Set `dt = 0.1`, and `time` long enough for you to see what's going on. Pull the mass back

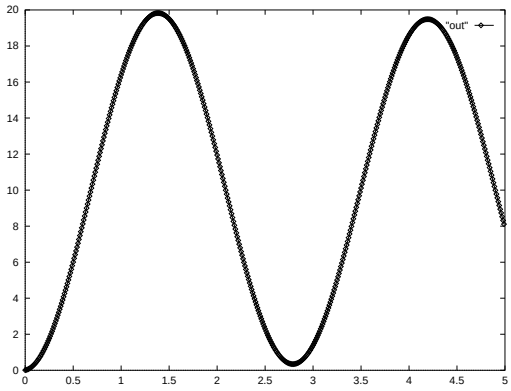

Figure 1: Hmmmmmmmm.....

on the spring a bit (set initial position non-zero), and let it go. What do you expect to happen? Plot the result. You should be surprized, or think I have a bug in the simulation code. Now change `dt` to .01 and repeat. Write up your experiences with pictures and tell me what you think is going on (hint in above paragraph).

B. OK so knowing that you need to keep `dt` relatively small and you're going to get a more or less inaccurate simulation over time, we can still make some progress. Here is a setting I used for my PID controller:

```
1.0 mass
0.0 friction
0.  spring
0.0 init. pos.
0.0 init vel.
5.0 Pgain
0.0 Igain
0.0 D gain
0.01  dt
5.0   time
0.9   integrator decay
10.0  desired position
```

What I seem to want to do here is apply proportional gain to the force on the mass so as to drive it from 0 to 10, with no springs or friction in the picture. Should work, eh? The behavior is shown in Fig. 1.

OK, changing several variables at once I add a nontrivial spring, a little damping, and "lots of" (note this gain depends on my choice of `dt`...ugly) derivative gain:

```
1.0 mass
0.2 friction
1.  spring
0.0 init. pos.
0.0 init vel.
```

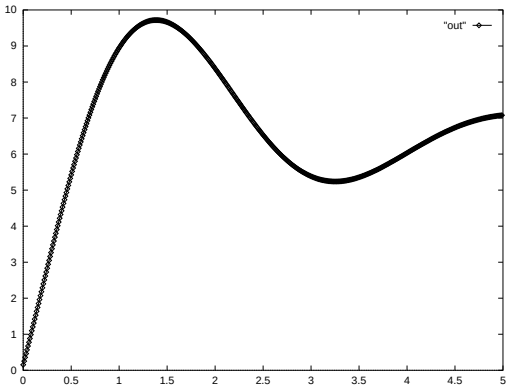

Figure 2: Explain?

```
2.0 Pgain
0.0 Igain
100.0 D gain
0.01  dt
5.0   time
0.9   integrator decay
10.0  desired position
```

The new behavior is shown in Fig. 2.

Explain the relevant phenomena demonstrated by these two "experiments".

C. In the above experiments, we assumed that we were sensing position and controlling force. These assumptions make intuitive sense, but you can also imagine sensing and controlling position, for example. What if you directly controlled position in the above experiments? Try it and see.

D. Pick some set of experiments whose outcome would clarify some question for you, perform the experiments, and explain the results to me. Make use of the graphics! For instance, add the time-varying disturbance force to the system and make a controller that tries to maintain a fixed position. Or try to match a desired constant velocity. Or try to match a time-varying position, or a combination of time-varying position and velocity.