Chaos Session III – Coupled oscillators

Goals for the session:

1. Look at one or two real systems that exhibit chaotic behavior
2. Program a coupled oscillator simulation, covering the use of arrays in EJS
3. Look at the double pendulum simulation to investigate chaos
4. Spend some time working on your own simulation project

1. Real-life systems, with Peter (60 minutes)

Hele-Shaw instabilities in fluids, and/or electrochemical deposition.

2. Creating a coupled oscillator simulation (45 minutes)

Start by loading into EJS the spring.xml file that is in EJS in the Manuals folder inside of the examples folder. Instead of one mass on one spring, we will modify it to have a few balls connected by springs. We will use arrays to simplify our life, and learn a little about loops. Also, we'll use two phantom balls, one at each end of the chain, that will not move.

variable	value		description	dimension
m	1.0		mass	
k	1.0		spring constant	
n	5	int	maximum number of balls	
x[i]	0.0		x-position of the balls	[n+2]
y	0.0		y-position	
vx[i]	0.0		x-velocity	[n+2]
ax[i]	0.0		x-acceleration	[n+2]
t	0.0		time	
dt	0.02		time increment	
length[i]	1.0		spring length	[n+1]
q	20.0		damping parameter	

A sample "for" loop

```
for (int i = 0; i <=n+1; i++)
{
  x[i] = i * (6.0/(n+1));
}
```

i starts at 0, gets incremented by 1 each time, and the loop repeats as long as i is less than or equal to n + 1.

Our initialization page looks like:

```
ax[0] = 0.0;
ax[n+1] = 0.0;

for (int i = 0; i <=n+1; i++)
{
  x[i] = i * (6.0/(n+1));
}
```

The first two lines will ensure that the phantom balls at the ends of the chain will not move.

The for loop places the balls into their initial positions. The balls are spaced equally from x = 0 to x = 6. If n = 2 (that is, if there are two balls that will move), how many times does the program run through the loop?

Now let's move to the Evolution page. Here, even though we have multiple objects, we only have to write two equations – we write them in array form, though, so the program applies them to each ball.

d vx[i] / dt = ax[i] – (1/q) * vx[i]

d x[i] / dt = vx[i]

Our constraints page looks like:

```
for (int i = 1; i <= n; i++ )
{
  ax[i] = -(k/m)*(x[i] - x[i-1]) + (k/m)*(x[i+1] - x[i]);
}

for (int i = 0; i <= n; i++)
{
  length[i] = x[i+1] - x[i];
}
```

The first loop finds the acceleration of each ball, and the second figures out the length of each spring.

In the View, we will use:

In the Drawing Panel, a SpringSet and a ParticleSet

Properties of the SpringSet (representing all the springs):

Elements: n+1

X: x
Y: y
Size X: length
Size Y: 0.0
Draggable: false

Properties of the ParticleSet

Elements: n+2
X: x
Y: y
Size X: 0.2
Size Y: 0.2
Draggable: true

Fix the various graphs, and the program should run.

So far, we have no way to change the number of balls. One way to so this is with a set of radio buttons. Add five radio buttons to the ButtonsPanel

The properties of the first radio button should look like:

Variable: true
Selected: false
Text: N = 1

Action On: (Use the first link and type in:
n = 1;
_initialize();

The others should look the same, except use N = 2, n = 2; N = 3, n = 3, etc. Only the last one should have Selected = true.

Save as coupled_oscillators_v1 (posted on-line at the wiki site)

We can easily turn our simulation into a simulation of driven harmonic oscillators by adding two more variables:

A	0.5	amplitude of the driving force
omega_drive	1.0	angular frequency of the driving force

And then adding this line to the constraints page:

```
x[0] = A*Math.sin(omega_drive*t);
```

That should do it.

Save as coupled_oscillators_v2 (posted on the wiki site)

The simulation allows us to investigate normal modes: the number of natural oscillation modes is generally equal to the number of oscillating objects.

3. The double pendulum and chaos (5 minutes)

There is also a double pendulum simulation (see a real version on the first floor of SCI) posted on the wiki site. Aside from a couple of horrendous evolution equation, the pendulum simulation basically takes the single pendulum simulation and adds another pendulum.

4. Spend time working on your own simulation project (40 minutes)

We can brainstorm together regarding any issues you have.