Cubical Grid of Masses

Completed and Analyzed in class, April 1, 2025

This is the sixteenth notebook for you to complete. It is our first notebook that has a three-dimensional network of masses. Since we live in three space dimensions, it is of course important to actually study it. However, most of the intuition about three-dimensional systems can actually be developed using one and two-dimensional systems.

Also, this is our last notebook in which we will do the hard coding work of modeling a continuous system by creating a system with a large but finite number of masses. After this, we are going to let Mathematica do that hard work!

How can Mathematica do the hard work for us?! Well, from a mathematics perspective, the types of problems we are studying are called differential equations or partial differential equations. They have been studied numerically for decades (actually a full century), and by hand for a century before that. Libraries of software routines have been developed to do what we have been doing, and Mathematica has those software routines. So we can just choose a physics problem, learn what differential equations are applicable to it, specify the differential equations to Mathematica, and it will (under the hood) do the work that we have been doing of chopping up the continuous system into little pieces and little time steps.

I HAVEN'T GOTTEN GOING ON THIS NOTEBOOK YET. IT IS JUST A COPY OF A PREVIOUS NOTEBOOK WITH A NEW TITLE.

THIS DOCUMENT EXISTS AS A PLACEHOLDER. DON'T BOTHER LOOKING AT IT FURTHER.

Initial Conditions

Set up the duration, **steps,** and **deltaT**:

In[ ]:=
```
tInitial = 0.0;
tFinal = 10.0;
steps = 5000;
deltaT = (tFinal - tInitial) / steps;
```

Set up the size of the grid (enough masses so that the grid looks like a drumhead, but not so many that we tax our computers):

```
In[ ]:= nx = 18; (* There is actually going to be 19, but both x edges will be fixed. *)
(* So the net number that are actually moving will be 17 in the x-direction. *)
ny = 24; (* There is actually going to be 25, but both y edges will be fixed. *)
(* So the net number that are actually moving will be 23 in the y-direction. *)
(* 17 * 23 means that the computer is simulating a grid of 391 masses. *)
(* It is doing this for 5000 time steps so in all your computer is having to *)
(* compute and render about 2000000 particle positions. *)
```

We are going to make initial conditions that are a product of sine functions. What sine function specifically is specified by the modes.

```
In[ ]:= modex = 2;
modey = 3;
maxz = 1.0;
initialzs =
  Table[maxz Sin[Pi modex (j - 1) / nx] Sin[Pi modey (k - 1) / ny], {j, nx + 1}, {k, ny + 1}];
initialvs = Table[0, {j, nx + 1}, {k, ny + 1}];
initialConditions = {tInitial, initialzs, initialvs};
```

Formulas for the Accelerations — Theory

The acceleration formula

$$a_{j,k} = v_0{}^2(z_{j,k+1} + z_{j,k-1} + z_{j+1,k} + z_{j-1,k} - 4\,z_{j,k})$$

is valid except for the edges, and we have to handle those separately.

Fixed Edges

A drumhead is normally fixed at the edges, and we are going to deal with the edges by just freezing the edge masses to have $z = 0$. So $z_{1,k} = 0$, $z_{n_x+1,k} = 0$, $z_{j,1} = 0$, and $z_{j,n_y+1} = 0$.

Conceptually, you can think of the index j as running from 0 to n_x and the index k as running from from 0 to n_y, but that goes against the grain of the way Mathematica indexes its arrays, so we are going to have to be super-careful about off-by-one errors. The index j will run from 1 to $n_x + 1$ and the index k will run from 1 to $n_y + 1$.

You can see that the necessary care was already taken in the initialConditions above.

Implementing the Acceleration

In[ ]:=
```
v0 = 4 Pi;

a[j_, k_, allzs_] := v0^2 If[j == 1 || chattanooga,
   0, (* no acceleration at the edges *)
   allzs[[j, k + 1]] + choo
  ]
```

Second-Order Runge-Kutta — Implementation

In[ ]:=
```
rungeKutta2[cc_] := (
  curTime = cc[[1]];
  curzs = cc[[2]];
  curvs = cc[[3]];
  newTime = curTime + deltaT;
  zsStar = curzs + curvs deltaT / 2;
  as = Table[a[j, k, zsStar], {j, 1, nx + 1}, {k, 1, ny + 1}];
  newvs = curvs + as deltaT;
  newzs = curzs + (curvs + newvs) deltaT / 2;
  {newTime, newzs, newvs}
 )

rk2Results = NestList[rungeKutta2, initialConditions, steps];

rk2ResultsTransposed = Transpose[rk2Results];
zs = rk2ResultsTransposed[[2]];
```

Out[ ]=

```
$Aborted
```

••• Part: Part 2 of rk2Results^T does not exist.

3D Graphics

We need a graphics implementation with $(n_x + 1)(n_y + 1)$ masses. We'll space the masses equally across the x and y axes of the cuboid and draw grid lines connecting them.

```
halfHeight = 1;
halfDepth = 4;
halfWidth = 3;
xspacing = 2 halfWidth / nx;
yspacing = 2 halfDepth / ny;
cuboid = {FaceForm[{Blue, Opacity[0.04]}], Cuboid[
    {-halfWidth, -halfDepth, -halfHeight}, {halfWidth, halfDepth, halfHeight}]};
drumheadGraphic[zs_] := Graphics3D[Flatten[{
     {cuboid},
     Table[
      Point[{-halfWidth + (j - 1) xspacing, -halfDepth + (k - 1) yspacing, zs[[j, k]]}],
      {j, nx + 1}, {k, ny + 1}
     ],
     Table[
      Line[{{-halfWidth + (j - 1) xspacing, -halfDepth + (k - 1) yspacing, zs[[j, k]]},
        {-halfWidth + (j - 1) xspacing, -halfDepth + k yspacing, zs[[j, k + 1]]}}],
      {j, nx + 1}, {k, ny}
     ],
     (* All the points and all the grid lines that go in the y-direction *)
     (* are already done. Add the grid lines that go in the x-direction. *)
     choo
    }, 1]];
drumheadGraphic[initialzs]
```

Animating the 3D Graphics

The default duration of the animation is the duration of our simulation:

In[ ]:=
```
Animate[drumheadGraphic[zs[[step]]],
 {step, 0, steps, 1}, DefaultDuration → tFinal - tInitial]
```

Completely Different Initial Conditions — The Mallet Strike

Your CPU is probably working hard just displaying the animation above over and over. Pause it or comment it out, because if it is running, it will drain its ability to also work on the problem below.

Look back at the initial conditions where initialzs and initialvs were defined.

Those initial conditions were very special and designed to illustrate modes of a drumhead. Now you are going to do something completely different. First, *the initialzs are going to be zero.* Then we are going to whack the drum with a mallet centered at malletx, mallety:

Where it says “glenn miller” “and his orchestra” implement something that is -maxv at the center of the mallet and decays away to nothing or nearly nothing at the edge of the mallet.

If you are interested in a function that is 1 at its center and decays away smoothly, try something like:

```
In[ ]:= malletvaluesx = N[Table[Exp[-(j - malletx)^2 / malletradius^2], {j, 1, nx + 1}]];
```

```
In[ ]:= malletvaluesy = N[Table[Exp[-(k - mallety)^2 / malletradius^2], {k, 1, ny + 1}]];
```

You will need to take the product of these functions to get something that decays away in both directions.

```
maxv = 20.0;
malletx = nx / 3;
mallety = ny / 3;
malletradius = 2;
newinitialzs = glenn miller;
newinitialvs = - maxv and his orchestra;
newInitialConditions = {tInitial, newinitialzs, newinitialvs};
```

Visualize the velocities generated by the mallet strike:

```
In[ ]:= drumheadGraphic[newinitialvs]
```

```
In[ ]:= newrk2Results = NestList[rungeKutta2, newInitialConditions, steps];

newrk2ResultsTransposed = Transpose[newrk2Results];
newzs = newrk2ResultsTransposed[[2]];
```

```
In[ ]:= Animate[drumheadGraphic[newzs[[step]]],
  {step, 0, steps, 1}, DefaultDuration → tFinal - tInitial]
```