Square Drumhead

Completed and Analyzed in class, March 25, 2025

This is the fourteenth notebook for you to complete. It is our first notebook that is has a two-dimensional network of masses. We'll make those two dimensional network be the *x* and *y* directions. The two-dimensional network of masses will oscillate vertically (in the *z* direction).

Initial Conditions

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

```
In[25]:= 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[29]:= nx = 3; (* 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 = 4; (* 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 has to simulate a grid of 391 masses. *)
```

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

```
In[79]:= modex = 1;
modey = 2;
maxz = 0.5;
initialzs =
  Table[maxz Sin[Pi modex (j - 1) / nx] Sin[Pi modey (k - 1) / ny], {j, nx + 1}, {k, ny + 1}];
initialvs = Table[0, {k, ny + 1}, {j, nx + 1}];
initialConditions = {tInitial, initialzs, initialvs};
```

```
In[91]:= initialzs // Grid
```

Out[91]=

0.	0.	0.	0.	0.
0.	0.433013	0.	-0.433013	0.
0.	0.433013	0.	-0.433013	0.
0.	0.	0.	0.	0.

Formulas for the Accelerations — Recap from Theory

The acceleration formula

$$a_{j,k} = -\omega_0{}^2 \left(z_{j,k+1} + z_{j,k-1} + z_{j+1,k} + z_{j-1,k} - 4\, z_{j,k}\right)$$

is valid except for the ends, 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 has already been taken in the initialConditions above.

Implementing the Accelerations

In[67]:=
```
omega0 = 4 Pi;

(* The following is on the right track, but it doesn't work for the ends *)

a[j_, k_, allzs_] := -omega0^2 If[j == 1 || j == nx + 1 || k == 1 || k == ny + 1,
   0, (* no acceleration at the edges *)
   allzs[[j, k + 1]] + allzs[[j, k - 1]] + allzs[[j + 1, k]] + allzs[[j, k - 1]] - 4 allzs[[j, k]]
  ]
```

Second-Order Runge-Kutta — Implementation

Now we implement Runge-Kutta 2:

In[44]:=
```
initialzs
```

Out[44]=
```
{{0., 0., 0., 0., 0.}, {0., 0.433013, 0., -0.433013, 0.},
 {0., 0.433013, 0., -0.433013, 0.}, {0., 0., 0., 0., 0.}}
```

In[54]:=
```
Range[{nx + 1, ny + 1}]
```

Out[54]=
```
{{1, 2, 3, 4}, {1, 2, 3, 4, 5}}
```

In[56]:=
```
Array[{#1, #2} &, {nx + 1, ny + 1}]
```

Out[56]=
```
{{{1, 1}, {1, 2}, {1, 3}, {1, 4}, {1, 5}}, {{2, 1}, {2, 2}, {2, 3}, {2, 4}, {2, 5}},
 {{3, 1}, {3, 2}, {3, 3}, {3, 4}, {3, 5}}, {{4, 1}, {4, 2}, {4, 3}, {4, 4}, {4, 5}}}
```

In[71]:=
```
initialzs
```
Out[71]=
```
{{0., 0., 0., 0.}, {0., 0.433013, 0.433013, 0.},
 {0., 0., 0., 0.}, {0., -0.433013, -0.433013, 0.}, {0., 0., 0., 0.}}
```
In[86]:=
```
as = Table[a[j, k, initialzs], {j, 1, nx + 1}, {k, 1, ny + 1}];
```

In[87]:=
```
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]];
```

Thread: Objects of unequal length in {{0., 0., 0., 0., 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0., 0., 0., 0.}} + {{0., 0., 0., 0.}, {0., 0., 0., 0.}, {0., ≪2≫, 0.}, {0., 0., 0., 0.}, {0., 0., 0., 0.}} cannot be combined.

Part: Part 3 of {{0., 0., 0., 0., 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0., 0., 0., 0.}} + {{0., 0., 0., 0.}, {0., 0., 0., 0.}, {0., ≪2≫, 0.}, {0., 0., 0., 0.}, {0., 0., 0., 0.}} does not exist.

Part: Part 3 of {{0., 0., 0., 0., 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0., 0., 0., 0.}} + {{0., 0., 0., 0.}, {0., 0., 0., 0.}, {0., ≪2≫, 0.}, {0., 0., 0., 0.}, {0., 0., 0., 0.}} does not exist.

Part: Part 3 of {{0., 0., 0., 0., 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0.433013, 0., −0.433013, 0.}, {0., 0., 0., 0., 0.}} + {{0., 0., 0., 0.}, {0., 0., 0., 0.}, {0., ≪2≫, 0.}, {0., 0., 0., 0.}, {0., 0., 0., 0.}} does not exist.

General: Further output of Part::partw will be suppressed during this calculation.

Thread: Objects of unequal length in {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}} + {{0., 0., 0., 0., 0.}, {0., ≪3≫, 0.}, {0., −0.315827 (2 Plus[≪2≫][[3, 1]] − 4 Plus[≪2≫][[3, 2]] + ({≪4≫} + {≪5≫})[[3, 3]] + ({≪4≫} + {≪5≫})[[4, 2]]), −≪20≫ ≪1≫, −≪20≫ (≪4≫ + ≪1≫), 0.}, {0., 0., 0., 0., 0.}} cannot be combined.

Thread: Objects of unequal length in {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}} + {{0., 0., 0., 0., 0.}, {0., ≪3≫, 0.}, {0., −0.315827 (≪1≫), −≪20≫ (≪1≫), −0.315827 (≪1≫), 0.}, {0., 0., 0., 0., 0.}} + {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}} cannot be combined.

General: Further output of Thread::tdlen will be suppressed during this calculation.

Out[88]=

```
$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. Space the masses equally across the x and y axes of the cuboid.

```
halfHeight = 1;
halfDepth = 4;
halfWidth = 3;
xspacing = 2 halfWidth / (nx + 1);
yspacing = 2 halfDepth / (ny + 1);
region = {FaceForm[{Blue, Opacity[0.04]}], Cuboid[
    {-halfWidth, -halfDepth, -halfHeight}, {halfWidth, halfDepth, halfHeight}]};
drumheadGraphic[zs_] := Graphics3D[Flatten[{
     {region},
     Table[
      {Point[{0, 0, 0}]},
      {{j, nx + 1}, {k, ny + 1}}
     ]},
    1]];
drumheadGraphic[initialxs]
```

Animating the 3D Graphics

It’s also nice to have an animation, arranged so that the default duration of the animation is the actual duration of the animation:

```
Animate[drumheadGraphic[zs〚step〛],
 {step, 0, steps, 1}, DefaultDuration → tFinal - tInitial]
```