Agent-Based Fabric Modeling Using Differential Equations

Joseph Rusinko and Hannah Swan
Winthrop University

Keywords: NetLogo, Software, Draping, Mass spring systems
Manuscript received on August 31, 2012; published on November 25, 2012.

Abstract: We use an agent-based modeling software, NetLogo, to simulate fabric drape by applying a modified mass spring system. This model provides an application of harmonic motion to textiles and fashion, fields not typically discussed in the undergraduate differential equations classroom. Euler's method is coded into the model to solve a system of ordinary differential equations describing the fabric's position over time. Our interactive NetLogo model allows students to visualize the behavior of the system and to experiment with parameters. We show an example of the success of our program. Students can also experiment with other numerical methods.

1 Introduction

The Tacoma Narrows Bridge is a popular example of harmonic motion in an introductory differential equations class. This visual representation of resonance helps students understand the importance of damping, an element in the second order differential equations, and the behavior of solutions to these equations [3]. Visual tools and examples help students understand concepts, as well as captivate interest [9]. Here we present an application of harmonic motion in textiles and fashion: fields not normally represented in math and science textbooks.

The fashion industry often uses computer aided design as a low cost and efficient tool for flat pattern drafting, creating the shapes that will later be cut from fabric. Innovative fashion designers like Alexander McQueen, however, begin their design process with physically draping on the form, manipulating a flat, essentially two dimensional fabric over a three dimensional mannequin to explore shape. They will record every step of this process with pictures and sketches to experiment with possible garments before drafting the patterns [8]. Although at times there is no alternative to hand draping, many hours and yards of fabric could be saved by a realistic computer simulation of fabric draping. This would allow designers from the entire spectrum of the fashion business to explore many different shapes very quickly and make the recording process more efficient.

Simulation of fabric draping has been considered outside of the fashion industry, especially in video games [4, 2] and film [7]. Real-time interaction between fabric and solid forms is necessary within video games, which often results in a loss of realism. In any simulation, there is a delicate balance between realism and speed. This is evident in a particular simulation that allows the user to pick up a silk scarf in a computer environment and move it around various obstacles with real-time responses in fabric shape [4].

We chose to model a particular drape, a circular sheet of fabric draped over a pin placed at the circle's center. This representation models all of the forces involved in fabric drapery. There is gravity pulling the fabric down; the tendency of the fabric to resist bending; and interaction with a fixed region such as a dress form, body or, in this case, a pin. The fabric is represented as a series of mass points connected by springs, creating a mesh that forms the surface of the fabric. There are, therefore, hundreds of mass points and springs, each with a number of differential equations modeling the mass point's motion, leading to large systems of differential equations. The solutions to these systems would be difficult to understand analytically. Therefore, we solve the system using Euler's method to plot new positions of the mass points at each time step giving a dynamic visual of the fabric falling to a resting position.

We use NetLogo, a free and easy-to-use modeling environment, to create the simulation of our chosen drape [10]. NetLogo allows us to set up agents, representing our mass points, to which we can assign rules. NetLogo programs contain a Code tab and an Interface tab. We can set up buttons in the Interface tab that run procedures in the code. There are also switches, sliders, and inputs which control various parameters, and monitors which display model behaviors. To write the rules for the agents to follow, we use the NetLogo programming language in the Code tab. We have encoded the mass spring system and numerical integration, which describe the motion of the fabric.

In this paper, in Section 2, we develop a dynamic model of yarn draped over a point. Then, in Section 3, we show how to encode Euler's method in NetLogo to solve the corresponding system of differential equations. This system is used to construct (Section 4) and analyze (Section 5) mesh models of a circular sheet of fabric draped over a point. We conclude by discussing possible applications of the model and uses for NetLogo in the classroom.

2 Mass Spring Systems

2.1 Classic Mass Spring Systems

The movement of a mass, m, connected to a spring is affected by the restoring force, $R(x)$, of the spring, a damping force acting on velocity, $D(x')$ and any external forces, $F(t)$. The one dimensional displacement of a single mass point from its starting position is described by the solution, $x(t)$, to the second order differential equation,

$$mx'' = R(x) + D(x') + F(t).$$

This model can be extended to systems with arbitrary numbers of masses and springs. For the system pictured in Figure 1, with two masses linked by three springs fixed on either end, we let $x_1(t)$ and $x_2(t)$ be the displacement from their resting position of m_1 and m_2. Before considering damping or external forces, each mass is only acted on by the restoring forces of each spring. The restoring force follows Hooke's law, which tells us that the force of a stretched spring is proportional to its displacement. Therefore the force is determined both by displacement and a spring constant, k, which measures the stiffness of each spring. For the system in Figure 1, our differential equations are,

$$\begin{aligned} m_1 x_1'' &= -k_1 x_1 + k_2(x_2 - x_1) \\ m_2 x_2'' &= -k_2(x_2 - x_1) + k_3 x_2, \end{aligned}$$

as $(x_2 - x_1)$ determines the length of the spring between the two masses.

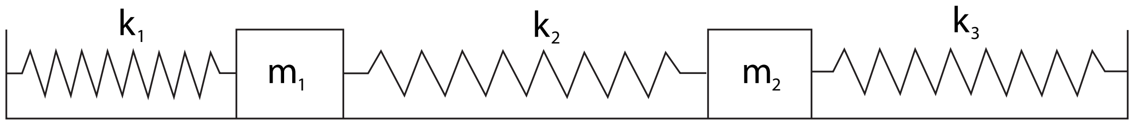

Figure 1: Mass Spring System

We now have a system of ordinary differential equations whose solutions, $x_1(t)$ and $x_2(t)$, measure the displacement of each mass over time [5]. The curves, $x_1(t)$ and $x_2(t)$, represent the separate displacements of each mass. Although this gives students a visual representation of the movement of the masses, simultaneously plotting the positions would further clarify the relationship between the connected masses.

2.2 Mass Spring System Used to Model Yarn

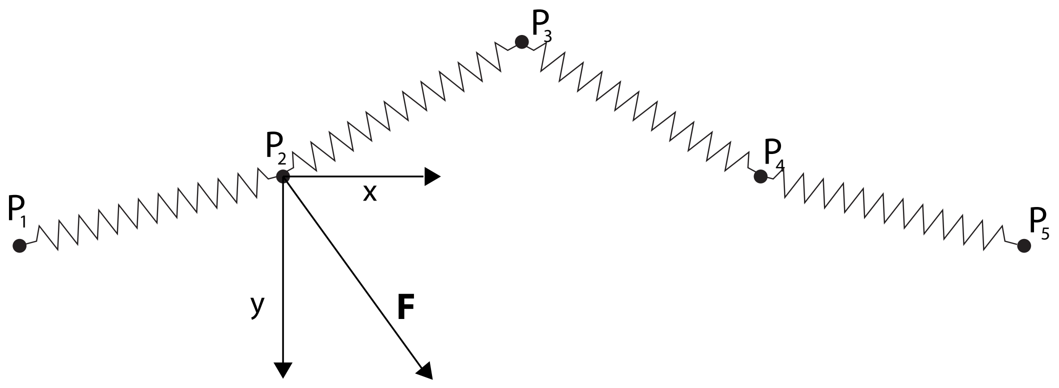

Figure 2: Yarn

Mass spring systems allow us to model the position of a piece of yarn draped from a pinned point. We use five points, P_1, P_2,..., P_5 labeled left to right, each with mass m as picture in Figure 2. Springs sharing a common spring constant k connect consecutive mass points. The model begins with the mass points equally spaced along a horizontal line, ensuring that each spring also has the same natural length. We represent the pin by fixing the position of the middle point, P_3. To model the motion of the yarn, we need to derive the equations for each mass point. As the equations for each point are similar, we derive them for the point,

P_2. The construction of the equations describing the other points follows a similar argument, the results of which are shown at the end of this section.

Like the classical mass spring system from Section 2.1, there are a number of forces acting on any point. For P_2, the springs connecting it to P_1 and P_3 apply restoring forces, $\mathbf{F_1}$ and $\mathbf{F_3}$, following Hooke's law. Letting d_{ij} be the distance between P_i and P_j, and l, the natural length of the springs, these forces pull in the direction of the prospective mass points with magnitudes:

$$\|\mathbf{F_1}\| = k(d_{12} - l) \text{ and } \|\mathbf{F_3}\| = k(d_{23} - l).$$

There is also an external force, gravity, or $m\mathbf{g}$, pulling P_2 down. The forces acting on P_2 sum to one vector, $\mathbf{F} = \mathbf{F_1} + \mathbf{F_3} - m\mathbf{g}$.

To properly see the behavior of the mass spring system we would like to plot the positions of the mass points at each time step. To do this, we must decompose each of the forces into x and y components and then use Euler's method to approximate a solution to the corresponding system of differential equations.

Here, we derive the y component, F_y, of the force acting on P_2. First we consider the contribution of $\mathbf{F_1}$ which acts along d_{12} as pictured in Figure 3.

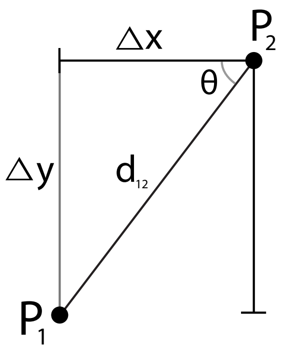

Figure 3: Decomposition of F_1

To compute the y component of this force, we multiply by the sine of the angle, θ, between our force and the horizontal direction vector. The substitution, $\sin\theta = \frac{\Delta y}{d_{12}}$, leads to the following equation for the force of F_1 in the y direction,

$$F_{1_y} = k(d_{12} - l)\frac{\Delta y}{d_{12}}.$$

Letting $\mathbf{u_i}$ be the position vector, $\{x_i, y_i\}$, for P_i we can express d_{ij} as $\|\mathbf{u_j} - \mathbf{u_i}\|$. Thus our force becomes:

$$F_{1_y} = k(\|\mathbf{u_2} - \mathbf{u_1}\| - l)\frac{y_2 - y_1}{\|\mathbf{u_2} - \mathbf{u_1}\|}.$$

A similar process can be used to derive the equations for F_{1_x}, the x component of F_1.

We repeat this derivation to find the x and y components of F_3, the force of the spring connecting mass point P_2 with P_3. The last remaining force acting on P_2 is gravity, or $-mg$, which only acts in the y component.

The motion of P_2 is determined by the combination of forces, F_1, F_3 and gravity. We use this to write differential equations whose solutions, $x_2(t)$ and $y_2(t)$ describe the position function of P_2. Summing these forces and dividing by the mass of P_2 yields the following system of second order, nonlinear, differential equations,

$$\begin{aligned} x_2'' &= \frac{k}{m}(\|\mathbf{u_3}-\mathbf{u_2}\|-l)\frac{x_3-x_2}{\|\mathbf{u_3}-\mathbf{u_2}\|}+\frac{k}{m}(\|\mathbf{u_2}-\mathbf{u_1}\|-l)\frac{x_2-x_1}{\|\mathbf{u_2}-\mathbf{u_1}\|}\\ y_2'' &= \frac{k}{m}(\|\mathbf{u_3}-\mathbf{u_2}\|-l)\frac{y_3-y_2}{\|\mathbf{u_3}-\mathbf{u_2}\|}+\frac{k}{m}(\|\mathbf{u_2}-\mathbf{u_1}\|-l)\frac{y_2-y_1}{\|\mathbf{u_2}-\mathbf{u_1}\|}-g. \end{aligned}$$

The equations for the remaining points can be derived in a similar fashion, giving us a system of ten differential equations for the motion of the five mass points modeling the yarn. In general, the differential equations describing the motion of an unfixed mass point, P_i, that is connected to two adjacent points, P_{i-1} and P_{i+1}, by separate springs (P_2 and P_4 in our example) are:

$$\begin{aligned} x_i'' &= \frac{k}{m}(\|\mathbf{u_{i+1}}-\mathbf{u_i}\|-l)\frac{x_{i+1}-x_i}{\|\mathbf{u_{i+1}}-\mathbf{u_i}\|}+\frac{k}{m}(\|\mathbf{u_i}-\mathbf{u_{i-1}}\|-l)\frac{x_i-x_{i-1}}{\|\mathbf{u_i}-\mathbf{u_{i-1}}\|}\\ y_i'' &= \frac{k}{m}(\|\mathbf{u_{i+1}}-\mathbf{u_i}\|-l)\frac{y_{i+1}-y_i}{\|\mathbf{u_{i+1}}-\mathbf{u_i}\|}+\frac{k}{m}(\|\mathbf{u_i}-\mathbf{u_{i-1}}\|-l)\frac{y_i-y_{i-1}}{\|\mathbf{u_i}-\mathbf{u_{i-1}}\|}-g. \end{aligned}$$

Then, for our two end mass points, P_1 and P_5, which are each only connected to one mass, our equations are:

$$\begin{aligned} x_1'' &= \frac{k}{m}(\|\mathbf{u_2}-\mathbf{u_1}\|-l)\frac{x_2-x_1}{\|\mathbf{u_2}-\mathbf{u_1}\|}\\ y_1'' &= \frac{k}{m}(\|\mathbf{u_2}-\mathbf{u_1}\|-l)\frac{y_2-y_1}{\|\mathbf{u_2}-\mathbf{u_1}\|}-g \end{aligned}$$

and,

$$\begin{aligned} x_5'' &= \frac{k}{m}(\|\mathbf{u_5}-\mathbf{u_4}\|-l)\frac{x_5-x_4}{\|\mathbf{u_5}-\mathbf{u_4}\|}\\ y_5'' &= \frac{k}{m}(\|\mathbf{u_5}-\mathbf{u_4}\|-l)\frac{y_5-y_4}{\|\mathbf{u_5}-\mathbf{u_4}\|}-g. \end{aligned}$$

Since P_3 is fixed by the pin, $x_3'' = y_3'' = 0$. While we derived these equations for a system with five mass points, it is expandable to an arbitrary number of mass points.

3 NetLogo Yarn Model

3.1 Setup

The system describing the motion of the yarn is modeled using NetLogo in the file Yarn.nlogo. We encourage the reader to download NetLogo, open the file and work along. The user will interact with the model through the Interface tab, however, the Code tab is also accessible.

The Interface includes several buttons and sliders. When the *Setup* button is pressed, the user will see the total number of mass points, n, placed along a horizontal line, $y = 5$, equidistant from each other. These mass points are referred to as turtles in the NetLogo programing language. Through the `turtles-own` command, every turtle can be assigned additional variables. In our model we assign each turtle an initial velocity and acceleration of 0.

Consecutive turtles are connected by links, which represent our springs. Each turtle is numbered, starting with turtle 0, which is used to determine the starting x-coordinate by $x = i - \frac{n-1}{2}$. The user may choose n with the *Total* slider within the interface. We only allow odd integers to ensure that we have a middle point. To update the total number of mass points, change the value on the slider and click *Setup* again. Notice the amount of mass points does not change the length of the yarn.

3.2 Coding Euler's Numerical Method in NetLogo

Once the setup with the desired values is complete, the yarn, starting at rest, is displayed within the View window. Try moving the yarn by pressing the *Move* button. Notice the position updates on each time step, referred to in NetLogo as a tick. As an agent-based modeling environment, a series of commands are applied to each individual agent or turtle on each tick. The *Move* button calls the procedure `move`, found in Code tab, which is applied to every individual mass point. In our model we design the `move` procedure to compute Euler's method for the system of differential equations described in Section 2.2.

The `move` procedure is applied to a mass point P with coordinates (x, y). The procedure first calls `calc-forces`, which calculates all the forces acting on P. For every neighboring mass point, P_i, `calc-forces` calculates $x_i - x$ and $y_i - y$, the differences between the x and y coordinates of P and P_i; and d, the distance between the two mass points. Using these values and l, the natural length of the springs, the procedure calculates the force for each component for every neighbor:

$$\begin{aligned} F_{x_i} &= k(d-l)\frac{x_i - x}{d} \\ F_{y_i} &= k(d-l)\frac{y_i - y}{d}. \end{aligned}$$

The procedure sums forces acting on P as they are calculated. This sum is then divided by the mass, gravity is applied in the negative y direction, and the result is set equal to the acceleration in each component, a_x and a_y. To keep the yarn from unnaturally bouncing, friction, f, is applied as a damping term to the mass point's existing velocity. The calculated acceleration is then added to the mass point's velocity:

$$\begin{aligned} v_x(t) &= (1-f)v_x(t-1) + a_x \\ v_y(t) &= (1-f)v_y(t-1) + a_y. \end{aligned}$$

After all the velocities are computed for every agent, the procedure `calc-forces` ends. Another procedure,`calc-position`, adds the velocities to the x and y coordinates of P, moving the mass points to their new positions.

$$\begin{aligned} x(t) &= x(t-1) + v_x(t) \\ y(t) &= y(t-1) + v_y(t). \end{aligned}$$

This process is repeated for as long as the *Move* button is activated. Since it is set as a forever button, the procedure will repeat, moving on to the next tick until the button is deactivated.

3.3 Modification of the model

We have already seen that the number of mass points can be chosen by the user. Friction, gravity, mass and the spring constant can also be adjusted in the Interface tab. Try experimenting with different spring constants. (The Setup button will need to be pressed after every change.) Notice the yarn does not act as we would expect, stretching far beyond its natural length.

Most yarn can only stretch 10% beyond their natural length before breaking. Using mass spring systems to simulate fabric is convenient; however, as we have seen, springs stretch and bounce more than fabric. To ensure the yarn acts more naturally, we build in constraints for any springs that stretched beyond this point [6].

As greatest stretch occurs around the pinned point, the `constrain` procedure acts first on the two mass points closest to the pinned point. First, the length of the springs connecting these points and the pinned point is calculated. If the length is greater than 1.1 times their natural length, l, then the turtle turns towards the pin and moves in that direction until it is less than $1.1l$ units away, shrinking the spring to an appropriate stretched length. Next, the mass points directly connected to the points just constrained go through a similar process. The springs that connect them to those already constrained are measured, then the positions of the points are adjusted. The procedure continues to work outwards until it reaches the outer mass points. When all points have been constrained, the program moves on to the next time step.

Now return to the Interface and run the simulation using the *Constrained Move* button instead of *Move*. Notice the yarn in the constrained system falls quickly to rest without bouncing and preserves its natural length.

4 NetLogo Fabric Model

With the constrain procedure in place, we have all the elements necessary to expand the model from a yarn to a fabric model. We do this in NetLogo 3D, which is included in the NetLogo download and uses a similar interface and syntax. Again, the reader is encouraged to open the 3D model, Fabric.nlogo3d, and work along.

The biggest difference in the Code tab in the three dimensional version of NetLogo, is the addition the z component. Recall in Section 2.2, we derived two differential equations for every mass point, whose solutions $x(t)$ and $y(t)$ composed the position vector $\{x(t), y(t)\}$. To allow this point to move in the three dimensional environment, we add an equation for the forces in the new direction, z. The new component is also included in Euler's numerical method, solving for the acceleration, velocity and position in the z direction and leading to our final solution vector, $\{x(t), y(t), z(t)\}$.

Returning to the Interface, notice there are now three setup options. Click the *Setup Quadrangular* button, which will place the turtles in a horizontal plane. The series of mass points creates a mesh model, which is two dimensional. We see that many mass points are now connected to more than two neighbors. This will increase the number of forces acting on each mass point, as each spring applies a restoring force.

The springs of the quadrangular mesh connects any mass point, $P_{i,j}$, as shown in Figure 4, to at most four others, $P_{i+1,j}$, $P_{i,j+1}$, $P_{i-1,j}$, and $P_{i,j-1}$. To approximate the circular fabric we only use the rectangular lattice points which fall inside of the circumference. The size of the circle is fixed, but the lattice resolution can be changed using the *Scale* slider. Try a few different values and notice how the mesh is affected.

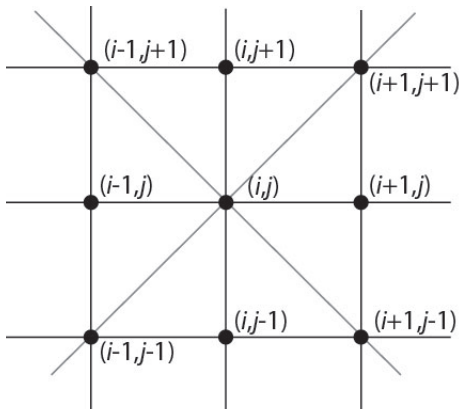

Figure 4: Traditional Meshes: dot at (i, j) represents mass point $P_{i,j}$

The next mesh is the triangular mesh, which is also controlled by the *Scale* slider. When we press the *Setup Triangular* button, we can see this mesh is similar to the quadrangular. The difference is the addition of diagonal springs, which control stretch in the bias directions. Again, in Figure 4, any mass point, $P_{i,j}$ is now connected to eight others: $P_{i+1,j}$, $P_{i+1,j+1}$, $P_{i,j+1}$, $P_{i-1,j+1}$, $P_{i-1,j}$, $P_{i-1,j-1}$, $P_{i,j-1}$, and $P_{i+1,j-1}$. This also complicates our equations slightly. Notice the quadrangular mesh included springs of equal natural length, while the triangular mesh's added diagonal springs have another natural spring length, $\sqrt{2l^2}$. This second length is added to the Code tab within the `calc-forces` procedure to ensure each force calculated has the correct magnitude.

Both of these meshes are traditionally used as they mimic the woven structure of fabric. An arc mesh was designed to mimic for the shape of the drape. We were interested in seeing whether a shape-based mesh geometry would be more efficient at producing an accurate drape than a traditional mesh. Try the *Setup Arc* button and notice that although the springs are straight, the mesh better approximates the shape of the circle. Inner, concentric circles are constructed with linearly increasing radius, r_i, with $i = 0, 1, \ldots, c$ until we reach the radius of the fabric. On each of these circles there are a number of equally spaced mass points. Each mass point falls on a straight radial line from the center of the circle to the circumference of our sheet. The resolution of the arc mesh is also controlled by the *Scale* and *Arc Scale* sliders.

While the first controls the number of concentric circles, the second controls how many points lie on each circle.

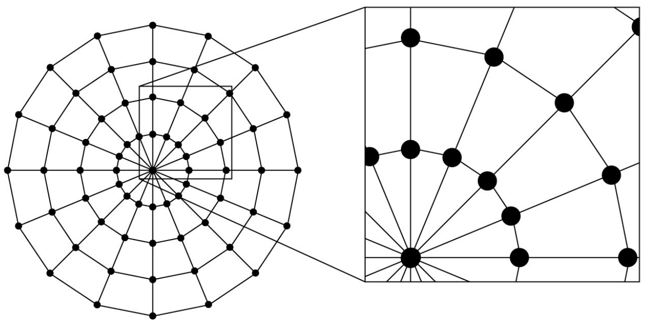

Figure 5: Arc Mesh

5 Comparison of Mesh Models

Section 4 outlines three potential mesh geometries, but which mesh produces the best results? To determine this, we compare each simulation to a real fabric using the drape coefficient, a standard measurement in the textile industry. We will also examine the efficiency of each simulation by finding the computational time.

We ran a simulation of each mesh geometry on a low resolution mesh (*Scale* = 3 and *Arc Scale*= 32), and fixed all other constants. These resolutions use a similar number of mass points.The computation time was measured from the time the *Move* button was placed until the fabric reached a resting position. To calculate the drape coefficient we took screen

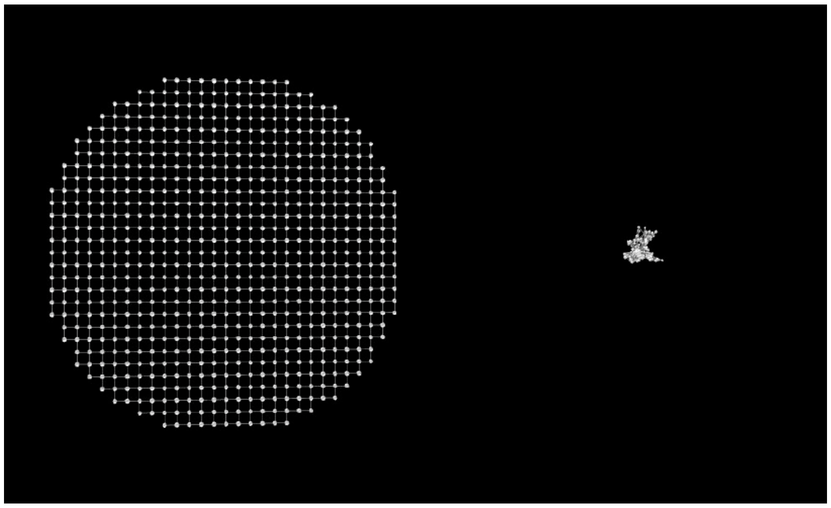

Figure 6: Quadrangular Mesh Drape

captures of both starting and ending positions, which can be seen in Figures 6, 7, and 8.

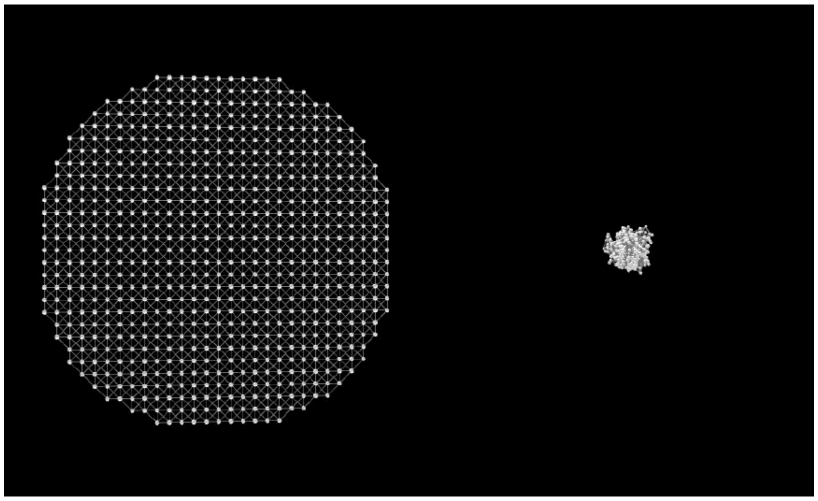

Figure 7: Triangular Mesh Drape

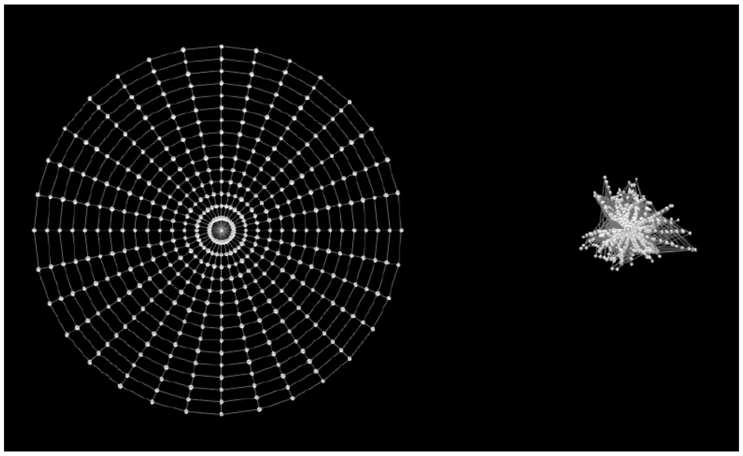

Figure 8: Arc Mesh Drape

We used *Adobe Photoshop*®, to find the area, in pixels, for both shapes. The areas are used to calculate the drape coefficient, DC,

$$DC = \frac{\text{Area of final drape shape}}{\text{Area of undraped circle}},$$

which we express as a percentage. The drape coefficient can also be calculated for an actual circular fabric piece. Using a camera and keeping the same distance between the lens and the fixed point, we captured images of the circle lying flat on a table and final draped shape [1]. Our results can be found in Figure 9. Clearly, our arc mesh produced a drape with a drape coefficient closest to the actual fabric drape. Notice, also, that it was able to do this in a fraction of the computational time as the traditional meshes.

	Drape Coefficients (%)		Average Computational
	Range	Average	Time (seconds)
Quadrangular	.77 – 1.00	.91	279.6
Triangular	1.93 – 2.14	2.01	107.5
Arc	5.79 – 6.90	6.32	27.9
Actual	10.57 – 15.14	13.03	

Figure 9: Results

6 Conclusion

This experiment could be replicated as a classroom activity, both to help students understand harmonic motion and to see a new, interesting application of the material. The models could also be used in the classroom in other ways. The yarn model can illustrate the behavior of multiple mass spring systems and the effect of the forces composing the differential equations. Students can experiment with different spring constants and mass in the Yarn.nlogo file, and they will quickly see how the system is affected. This application is also a way to generate interest among groups typically underrepresented in Mathematics.

Other possibilities are more involved. Interested students may try coding a different numerical methods, like the Runge-Kutta method, to solve the differential equations. This would require understanding and analyzing the method in order to find the best way to write the procedure in the given syntax. They may also try to construct their own mesh geometry and explore how to use the existing equations for the new mesh.

Students may want to explore a different problem using differential equations in NetLogo. A monitor within our model shows the number of differential equations numerically solved at every time step. We can see that even a low resolution mesh has thousands of equations. Since NetLogo is clearly capable of handling large systems, and has many options for visual representation of these system, students would benefit greatly from using this tool to create their own model for any problem involving ordinary differential equations such as climate and weather systems, interacting species, or mixing problems. The NetLogo tutorials allow students with limited or no programing experience to independently develop the necessary skills in a few hours. As such this software could be an excellent addition to the differential equations classroom.

References

[1] Narahari Kenkare and Traci May-Plumlee. Fabric drape measurement: A modified method using digital image processing. *Journal of Textile and Apparel, Technology and Management*, 4, 2005.

[2] Ben Kenwright, Rich Davison, and Graham Morgan. Real-time deformable soft-body simulation using distributed mass-spring approximations. In *CONTENT 2011, The Third International Conference on Creative Content Technologies*, pages 29–33, 2011.

[3] Oliver Knill. The Tacoma bridge, 2003. URL http://www.math.harvard.edu/archive/21b_fall_03/tacoma/index.html.

[4] Mark Meyer, Gilles Debunne, Mathieu Desbrun, and Alan H. Barr. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation*, 12(1):1–12, 2001. ISSN 1099-1778. doi: 10.1002/vis.244. URL http://dx.doi.org/10.1002/vis.244.

[5] J.C. Polking, A. Boggess, and D. Arnold. *Differential Equations With Boundary Value Problems*. Pearson/Prentice Hall, 2006. ISBN 9780131862364. URL http://books.google.com/books?id=JnlwQgAACAAJ.

[6] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *IN GRAPHICS INTERFACE*, pages 147–154, 1995.

[7] Andrew Selle, Jonathan Su, Geoffrey Irving, and Ronald Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics*, 15:339–350, 2009. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.79.

[8] Richard Sorger and Jenny Udale. *The Fundamentals of Fashion Design*. Ava Publishingl, 2006. ISBN 2940373396.

[9] Thomas Wangler. Using technology in a differential equations course: Lessons learned implementing a new paradigm. *CODEE Journal*, October 2011. URL http://www.codee.org/ref/CJ11-0583.

[10] U. Wilensky. Netlogo, 1999. URL http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.