Solving Differential Equations by Computer

R. Herman

Summer 2015, 7/2/2015

1 Maple

Direction fields

Enter the differential equation, being careful to write the dependent variable as a function. DEplot can be used to provide a direction field. Particular solutions can be added using a set of initial conditions. If the direction field is not desired, then set **arrows = none**.

```
> restart: with(DEtools):
> ode := diff(y(t),t) = 1-y(t);
> DEplot(ode,y(t),t=0..10,y=0..5);
> ics:=[y(0)=1,y(0)=3,y(0)=5];
> DEplot(ode,y(t),t=0..10,y=0..5,ics,arrows=medium,linecolor =black);
```

Solutions

One can also seek analytic solutions using the differential equation in dsolve. Initial value problems are solved by including initial conditions including braces.

```
> ode := diff(y(t),t) = 1-y(t);
> dsolve(ode,y(t));
> dsolve({ode,y(0)=1},y(t));
```

Second order differential equations can also be solved. Initial values can be entered as well.

```
> dsolve(x^2*diff(y(x),x$2)+3*x*diff(y(x),x)-3*y(x)=x^2,y(x));
> dsolve({x^2*diff(y(x),x$2)+3*x*diff(y(x),x)-3*y(x) = x^2, y(0)=1,
  D(y)(0)=0},y(x));
```

Numerical Solution

Numerical solutions can be obtained using **type = numeric**

```
> with(plots):
> dsolve({EQ, ICs});
> p:=dsolve({EQ, ICs},type=numeric, range=0..1):
> odeplot(p);
```

One can choose a numerical method from a list and plot the solution.

```
> with(plots):
> ode:={x^2*diff(y(x),x$2)+3*x*diff(y(x),x)-3*y(x)=x^2}:
> ics:={y(0)=1,D(y)(0)=0}:
> dsol := dsolve(ode union ics, numeric, method=rkf45, relerr=Float(1,
  -8), abserr=Float(1,-8),maxfun=0, output=procedurelist):
> odeplot(dsol, [x,y(x)], 1..4);
```

Systems of Differential Equations

Systems of first order equations can be solved and the solutions displayed using DEplot.

```
> restart: with(DEtools):
> EQ1:=diff(x(t),t) = -x(t)+6*y(t);
> EQ2:=diff(y(t),t) = x(t)-2*y(t);
> DEplot( [EQ1,EQ2], [x(t),y(t)], t=0..5, x=-5..10, y=-5..5,
  [[x(0)=1,y(0)=1], [x(0)=1,y(0)=3], [x(0)=1,y(0)=-2], [x(0)=1,
  y(0)=-3]], arrows=none,linecolor=blue);
```

2 MATLAB

2.1 Direction Fields

One can produce direction fields in MATLAB. A sample code is given by

```
> [x,y]=meshgrid(0:.1:2,0:.1:1.5);
> dy=1-y;
> dx=ones(size(dy));
> quiver(x,y,dx,dy)
> axis([0,2,0,1.5])
> xlabel('x')
> ylabel('y')
```

The mesh command sets up the xy-grid. In this case x is in $[0, 2]$ and y is in $[0, 1.5]$. In each case the grid spacing is 0.1.

We let dy = 1-y and dx =1. Thus,

$$\frac{dy}{dx} = \frac{1-y}{1} = 1-y.$$

The quiver command produces a vector (dx,dy) at (x,y). The slope of each vector isdy/dx. The other commands label the axes and provides a window with xmin=0, xmax=2, ymin=0, ymax=1.5.

dsolve.

One can use MATLAB to obtain solutions and plots of solutions. The function dsolve obtains the symbolic solution and ezplot is used to quickly plot the symbolic solution.

```
> sol = dsolve('Dx=2*sin(t)-4*x','x(0)=0','t');
> ezplot(sol,[0 10])
> xlabel('t'),ylabel('x'), grid
```

ODE45 and other solvers.

There are several ODE solvers in MATLAB, implementing Runge-Kutta and other numerical schemes. Examples of its use are in the text. For example, one can implement ode45 using

```
> [t y]=ode45('func',[0 5],1); plot(t,y)
```

One can define func in a file func.m such as

```
function f=func(t,y)
f=-t*y/sqrt(2-y^2);
```

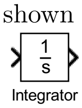
See MATLAB help for other examples.

3 SIMULINK

Simulink is a graphical environment for designing simulations of systems. Let's use Simulink to solve the differential equation

$$\frac{dx}{dt} = 2\sin 3t - 4x.$$

The simulation in Simulink takes the form shown in Figure 1.

This system uses the integrator block 

to integrate $\frac{dx}{dt}$, producing $x(t)$. The Scope is used to plot the output of the integrator block, $x(t)$.

The input of the integrator is the right side of the differential equation, $2\sin 3t - 4x$. The sine function can by input using the Sine Wave Function, whose parameters are set in the component. In order to get $4x(t)$, we grab the output of the integrator and boost it by the Gain value. Then, using the adder component,

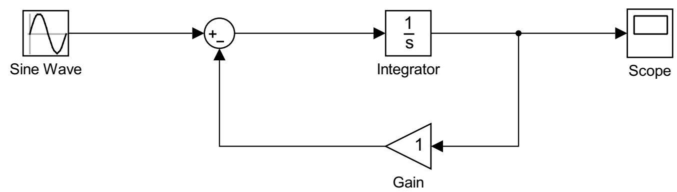


Figure 1: System for solving first order ODE.

these terms are added, or subtracted, and fed into the integrator.That is the main idea behind solving this system.

In MATLAB, type simulink to bring up Simulink Library Browser and then click the yellow plus to bring up new model. [You can also click the Simulink Library icon in MATLAB.] We build the model by dragging and connecting the needed components from sections such as Continuous, Math Operations, Sinks, or Sources.

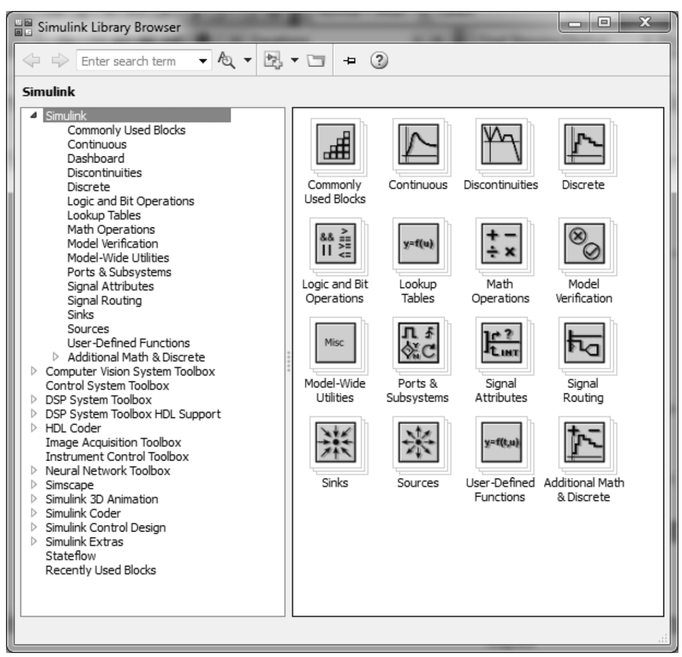


Figure 2: The Simulink Library Browser. [As seen in MATLAB 2015a.]

For this example, follow the following steps:

- Drag the Integrator from Continuous group; Sum, Gain, Sine Wave from Math Operations; and, a Scope from the Sink group, onto the model area.
- Connect the Integrator to the Scope by clicking on the Integrator output

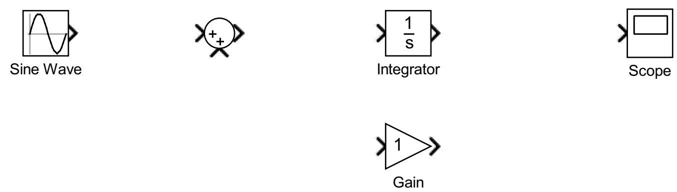


Figure 3: Add needed components to the model window.

and dragging to the Scope until they are connected. [Figure 3.]

- Connect the output of Sum to the input of the Integrator. [Figure 4.]

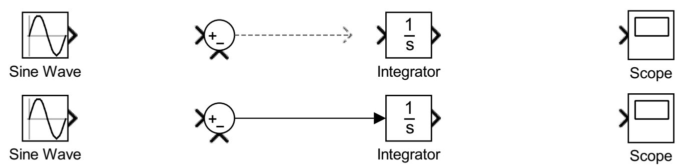


Figure 4: Example of connecting two components: Align the components, Click on output of one and drag to another. Then, release to finalize connection

- Right-click the Gain control and choose Flip Block under Rotate & Flip. Double-click the Gain and change the Gain value from 1 to 4. It should change on the control.

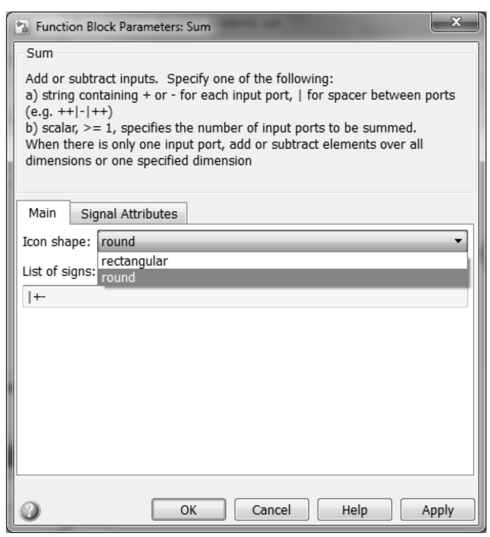


Figure 5: Block Parameters for the Sum control.

- Double-click the Sum control to bring up Block Parameters and change from |++ to |+- in order to set addition/subtraction nodes. [Note that the | is a blank node. Also, you can change the block to rectangular form. See Figure 5]

- Double-click the Sine Wave function and change the frequency to 3 rad/s and the amplitude to 2. Set the time dropdown menu to Use Simulation Time.

- Connect the Gain output to the negative input of Sum and the Sine Wave output to the positive input on the Sum control.

- To add a node to route an x value to the Gain, hold the CTRL key and click on the Output line of the Integrator and drag towards the input of the Gain. You can also Right-Click the line where you want the node and drag from there. See Figure 6.

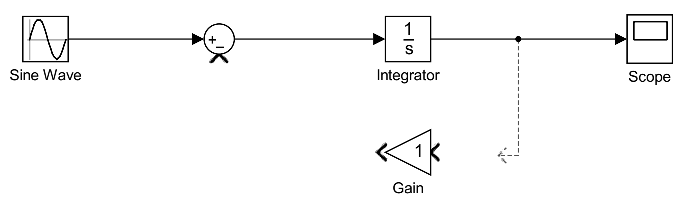


Figure 6: Add a node by right-clicking one the line and dragging to the input of a block.

- The initial value of x is inserted by double-clicking the Integrator and setting the value.

- One can annotate the diagram by clicking near where labels are needed and typing in the text box. This leads to the model in Figure 7.

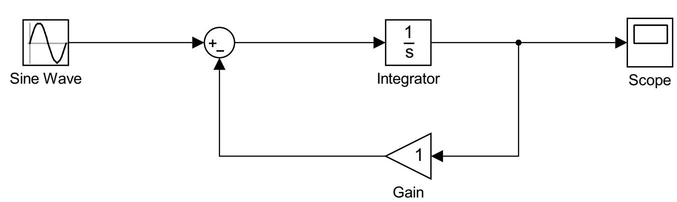


Figure 7: Connections for First Order ODE.

- Save the file under a useable file name. This file can be called in MATLAB, or one can use the run button to run the simulation.

- Double-click the scope to see the solution. One can use autoscale to rescale the scope view.

- Also, one can make further changes to the system by checking the Configuration Parameters under the Simulation menu item. See Figure 8.

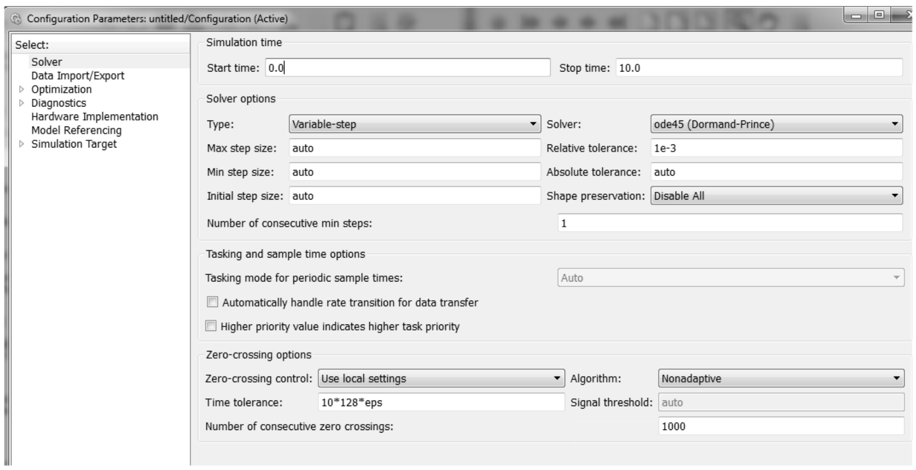


Figure 8: System Configuration Parameters.

In Figure 9 are shown a separable and a first order linear differential equation. The time dependent functions are obtained using the Clock block and the Function block. These were done as independents systems in the same model window.

Often we might want to access the solutions in MATLAB. Using the first model in Figure 8, add the To Workspace block. Double-click and rename the variable as y and change the output type to array. Run the simulation. This will put tout and y data into the MATLAB workspace.

In MATLAB you can plot the data using plot(tout,y). You can add labels with xlabel('t'), ylabel('y'), title('y vs t'). Adding the command set(gcf,'Color',[1,1,1]) makes the plot background white.

We can solve second order constant coefficient differential equations using a pair of integrators. This is displayed in Figure 11.

4 Exercises

1. Sketch by hand the direction field for $y' = y(y+1)$. Based on your sketch, draw solutions satisfying the initial conditions $y(0) = -1.5, 1, 0, , 0.5$.

2. Use a computer package to draw the direction field for the given equations for $(x, y) \in [-5, 5] \times [-5, 5]$ and plot solutions throughout the region.

   (a) $y' = 1 + x + y$.

   (b) $y' = xy$.

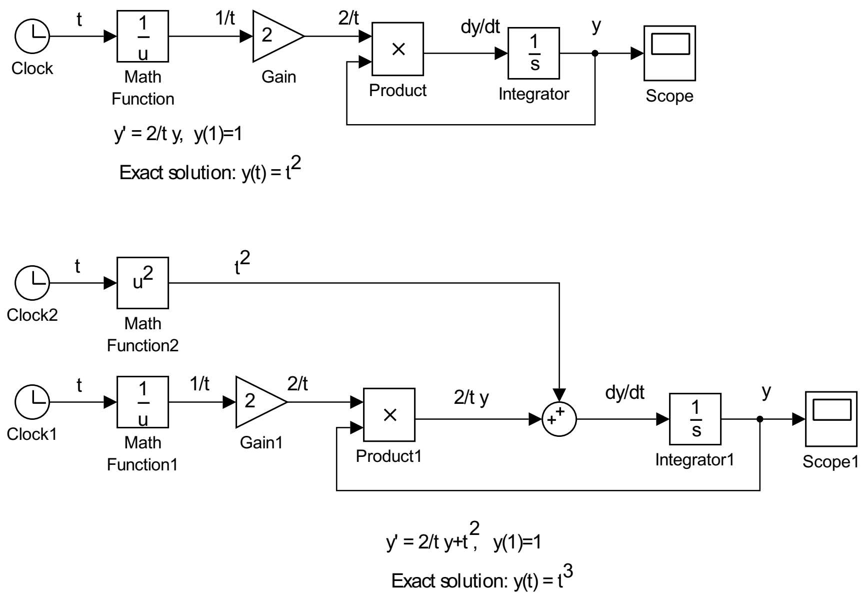


Figure 9: Two first order differential equations simulated in Simulink.

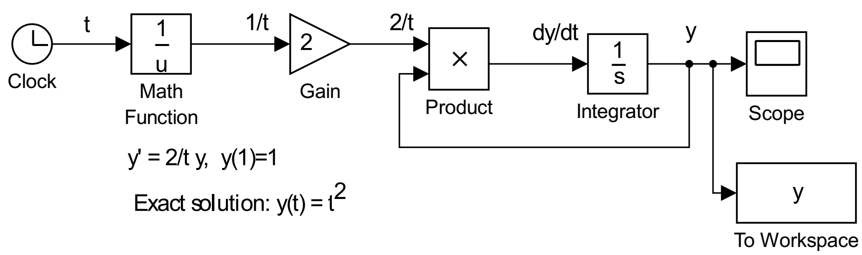


Figure 10: Adding To Workspace block for sending output to MATLAB.

(c) $y' = 2y(3-y)$.

(d) $y' = (y^2-4)(y-4)$.

3. Construct the model in Figure 1 and produce a plot of the solution.

4. Modify the model in the last problem to solve $\dfrac{dx}{dt} = f(t) - 2x$ for different function, $f(t)$.

5. Solve the following initial value problem using Maple, MATLAB, and Simulink. Are the solutions the same? Provide plots of the solutions. Are these consistent with your answers in Problem 2?

   (a) $y' = 2y(3-y)$, $y(0) = 0.5$.

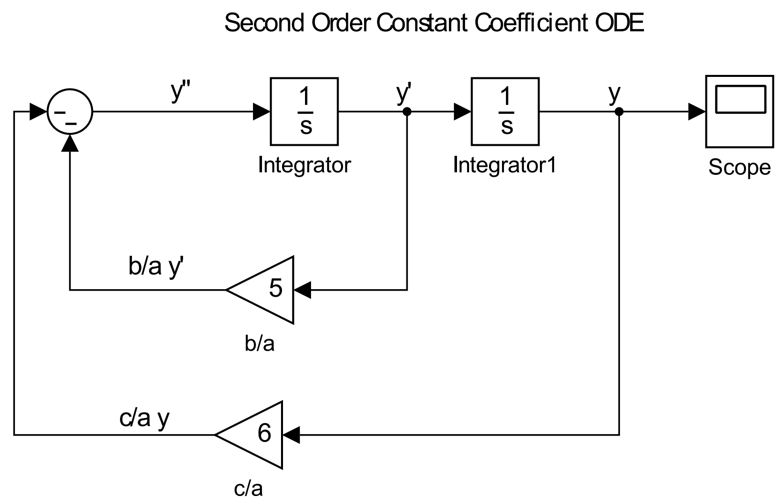


Figure 11: Second Order Constant Coefficient ODE.

(b) $y' = 1 + x + y$.

6. Consider the model in Figure 12. Fill in the question marks with the correct expression at that point in the computation. What differential equation is solved by this simulation?

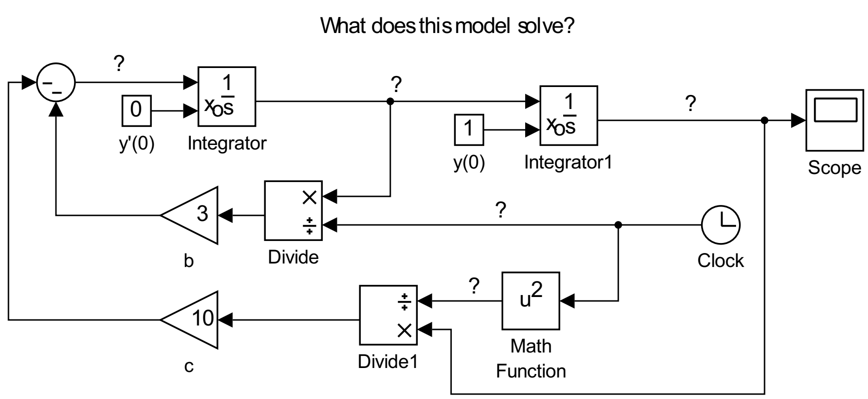


Figure 12: Mystery model for Problem 6.

5 Other Models

Linear sytems -

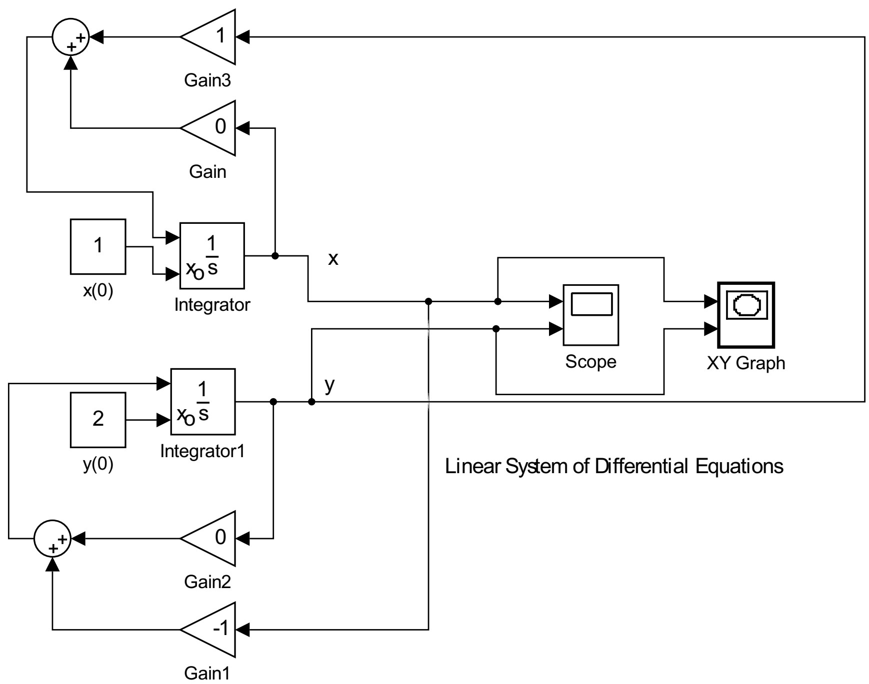


Figure 13: Linear system using two integrators.

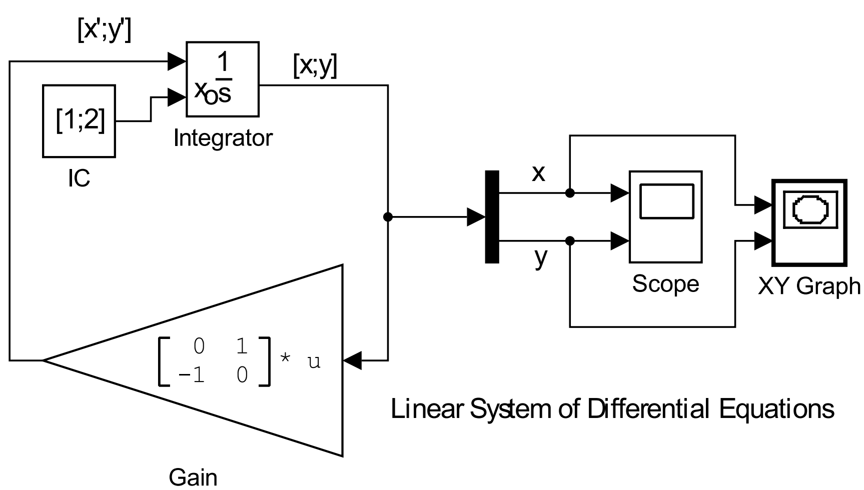


Figure 14: Linear system using matrix operation.

Here are simulations of a logistic equation, a forced, damped oscillator, projectile motion in the plane2, and a nonlinear system of two first order differential equations.

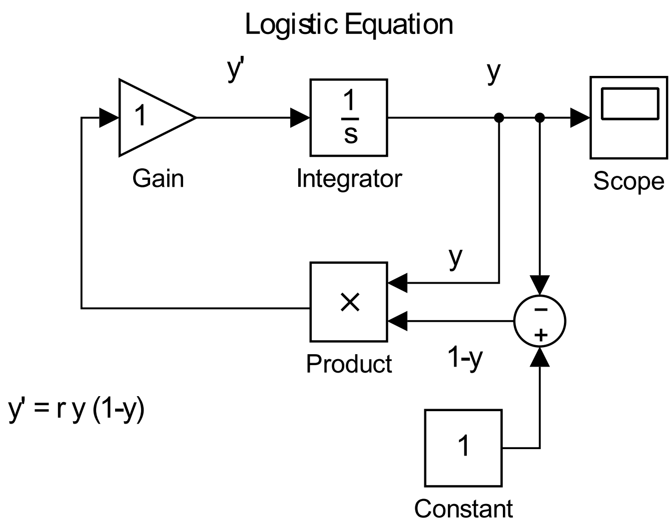


Figure 15: Logistic equation, $y' = ry(1-y)l$.

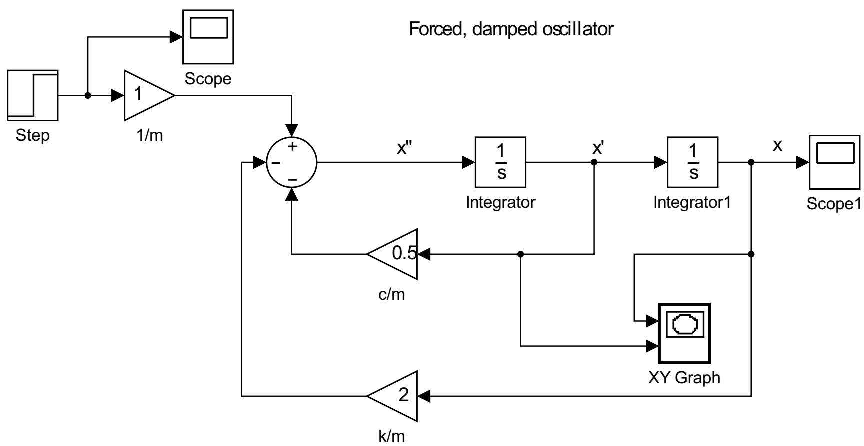


Figure 16: Forced, damped oscillator.

6 Stern-Gerlach Experiment

As described in Townsend's *A Modern Approach to Quantum Mechanics* and Schwinger's *Quantum Mechanics: Symbolism of Atomic Measurements,* based upon Section 5.1 of *Feynman's Lectures III,* the Stern-Gerlach Experiments can

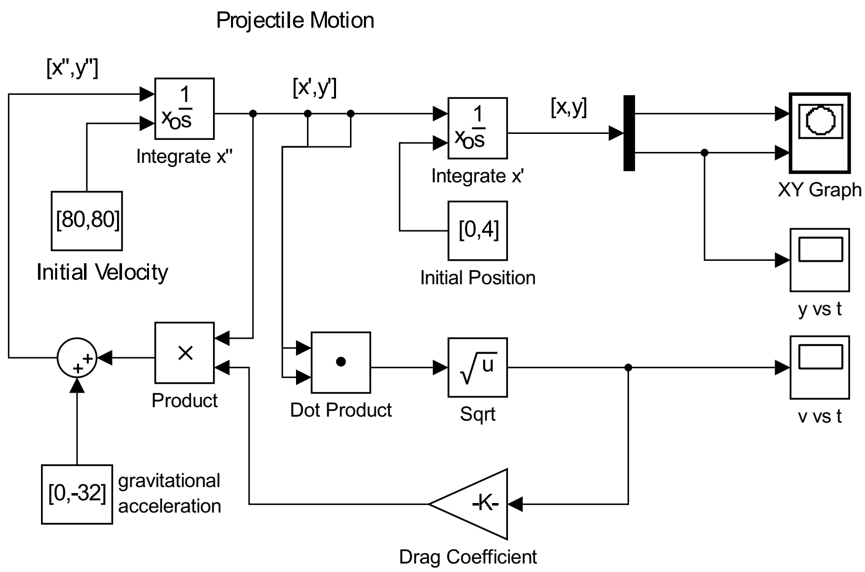


Figure 17: Projectile motion.

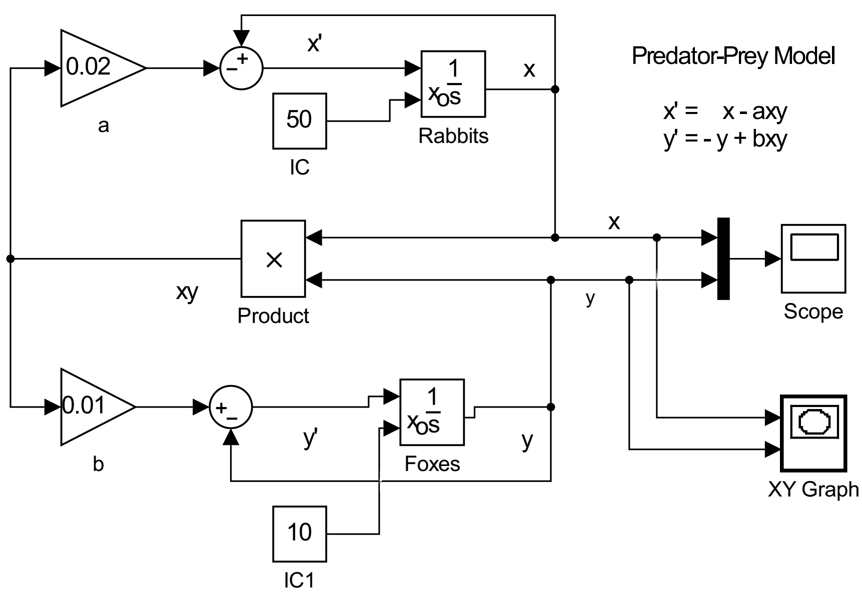


Figure 18: Predator-Prey model.

be described by following the paths of Ag atoms as a series of constant acceleration problems. The modified Stern-Gerlach setup can be used to elaborate

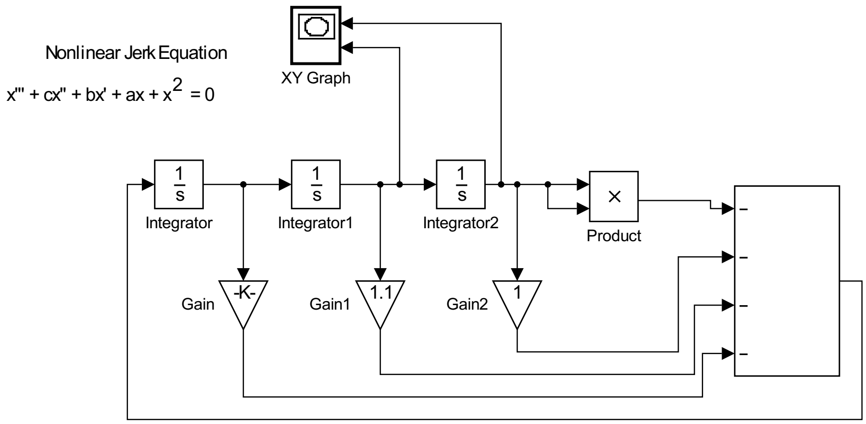


Figure 19: Nonlinear jerk model.

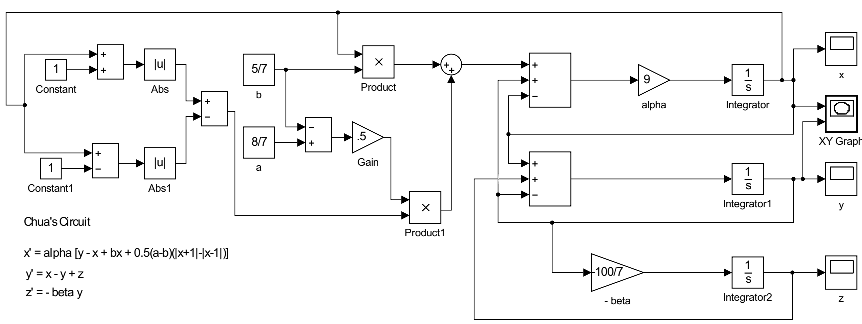


Figure 20: Nonlinear Chua model.

on the behavior of simple spin particles. The following diagrams are useful in carrying out the computations needed to show these paths.

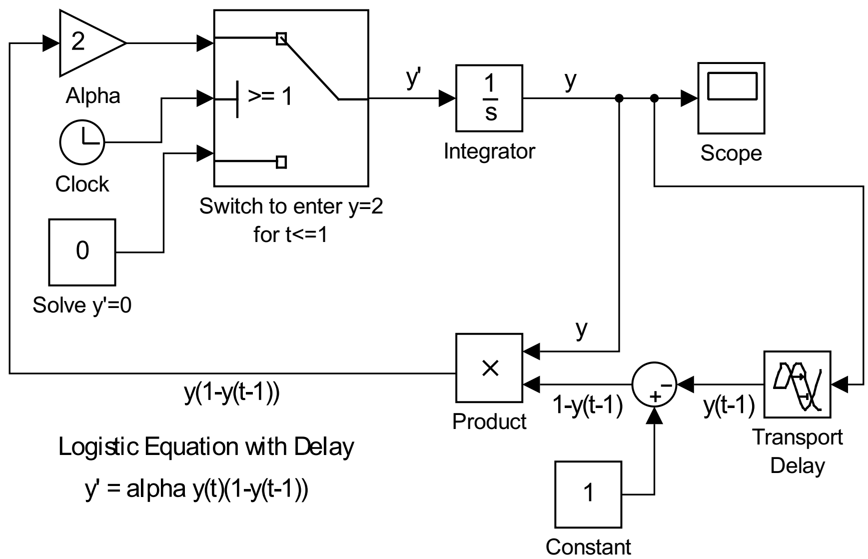


Figure 21: Logistic equation with delay, $y' = ry(t)(1 - y(t - \tau))$

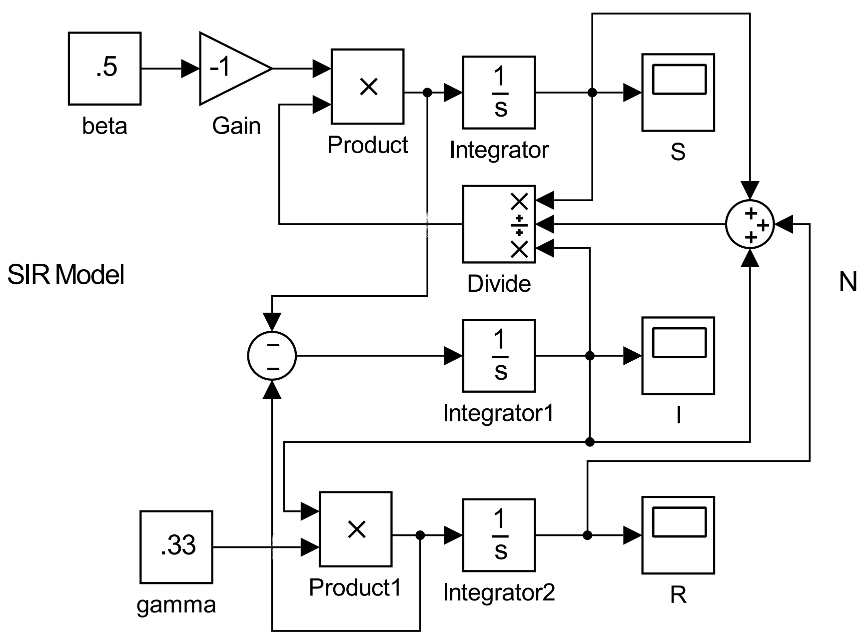


Figure 22: SIR epidemic model.

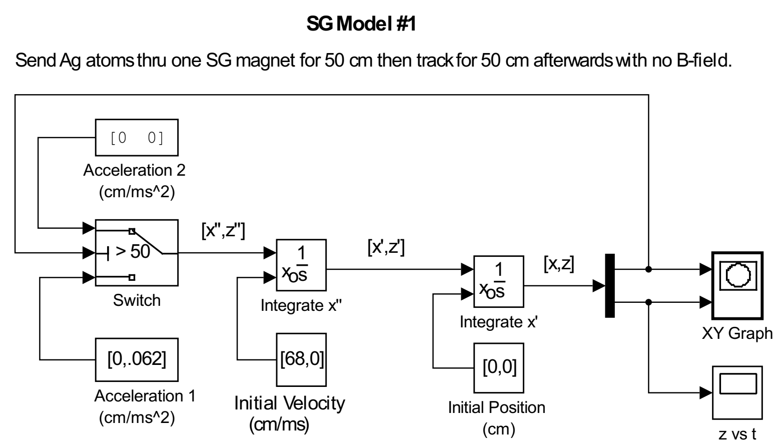


Figure 23: SG Experiment 1.

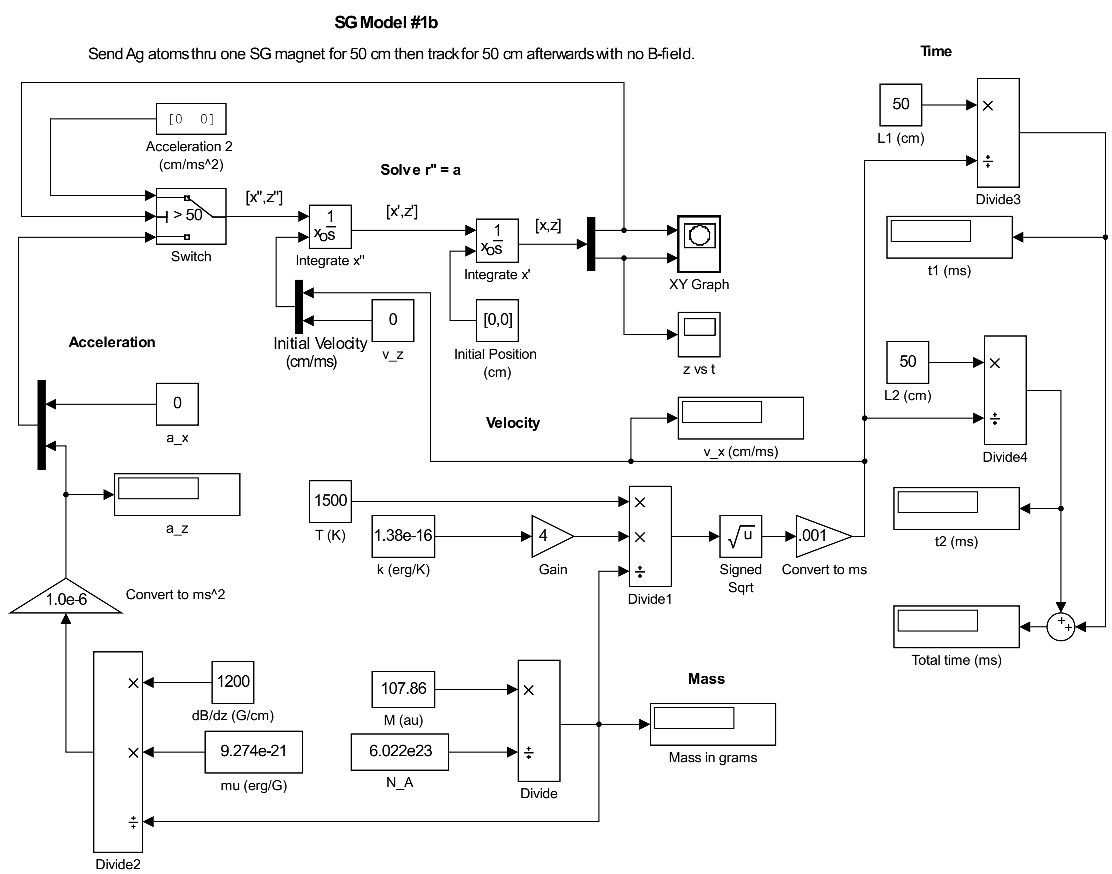


Figure 24: SG Experiment 1b.

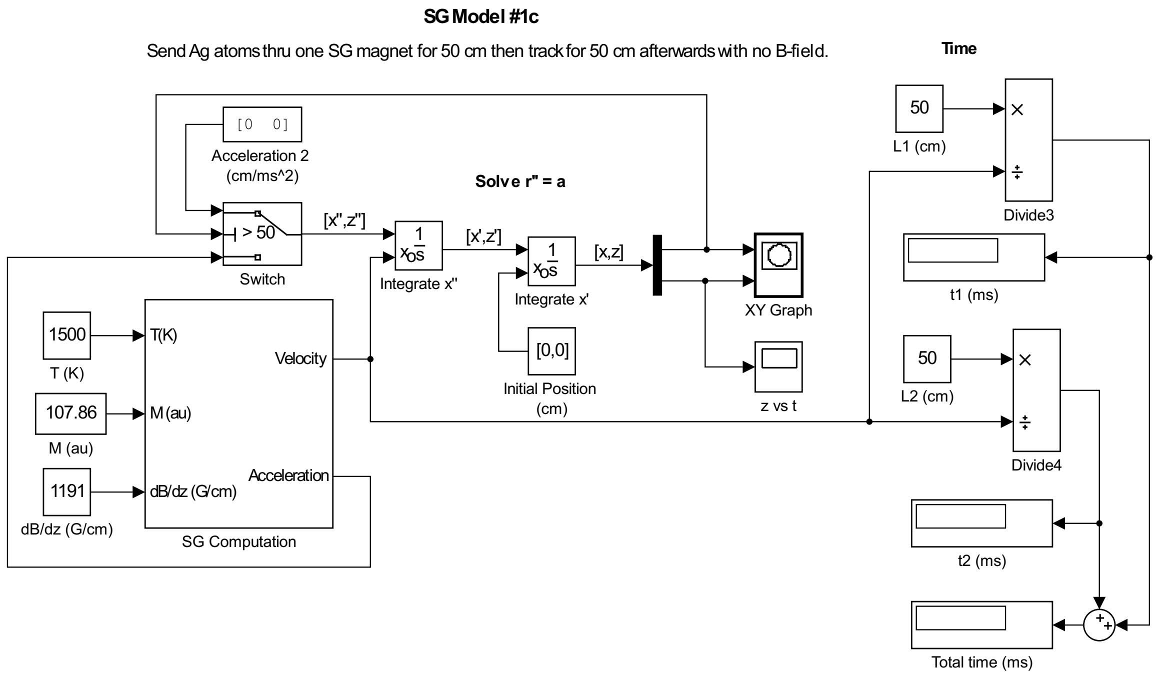


Figure 25: SG Experiment 1c.

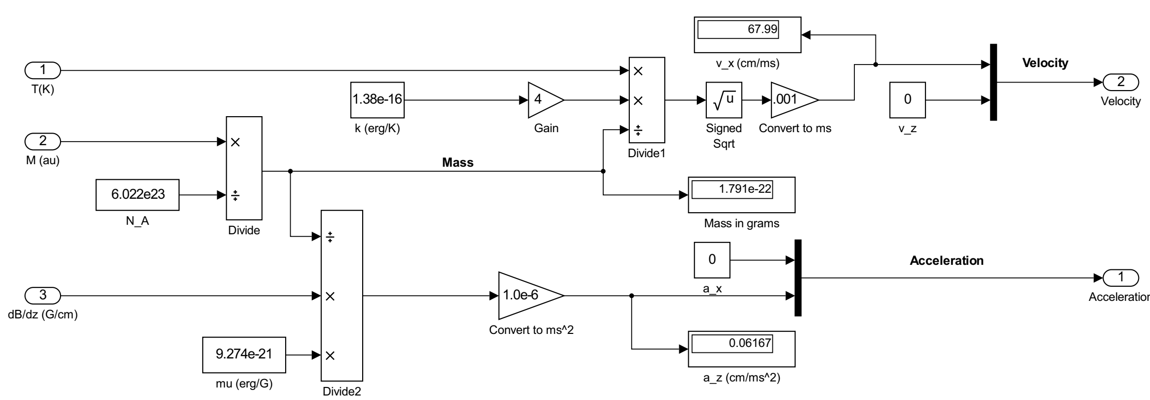


Figure 26: SG Experiment - Computational block.

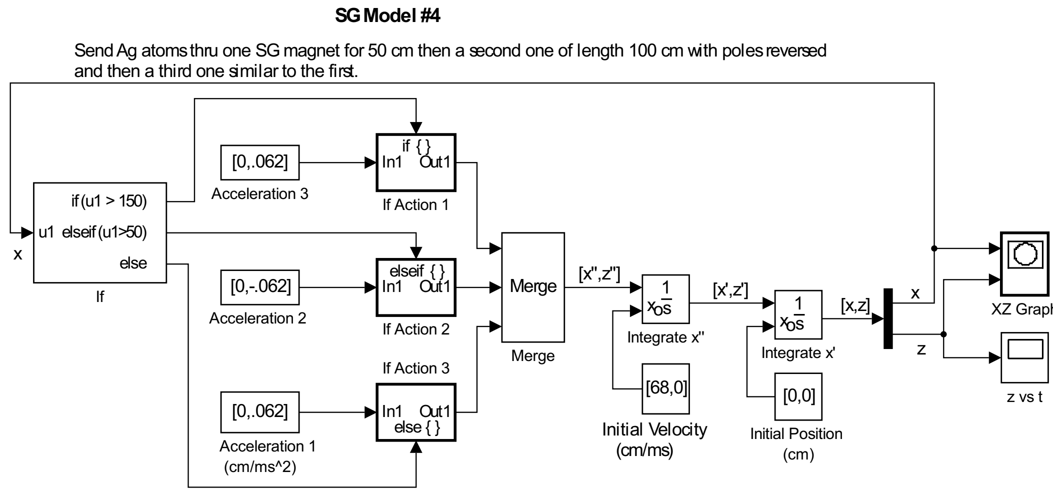


Figure 27: SG Experiment 4.

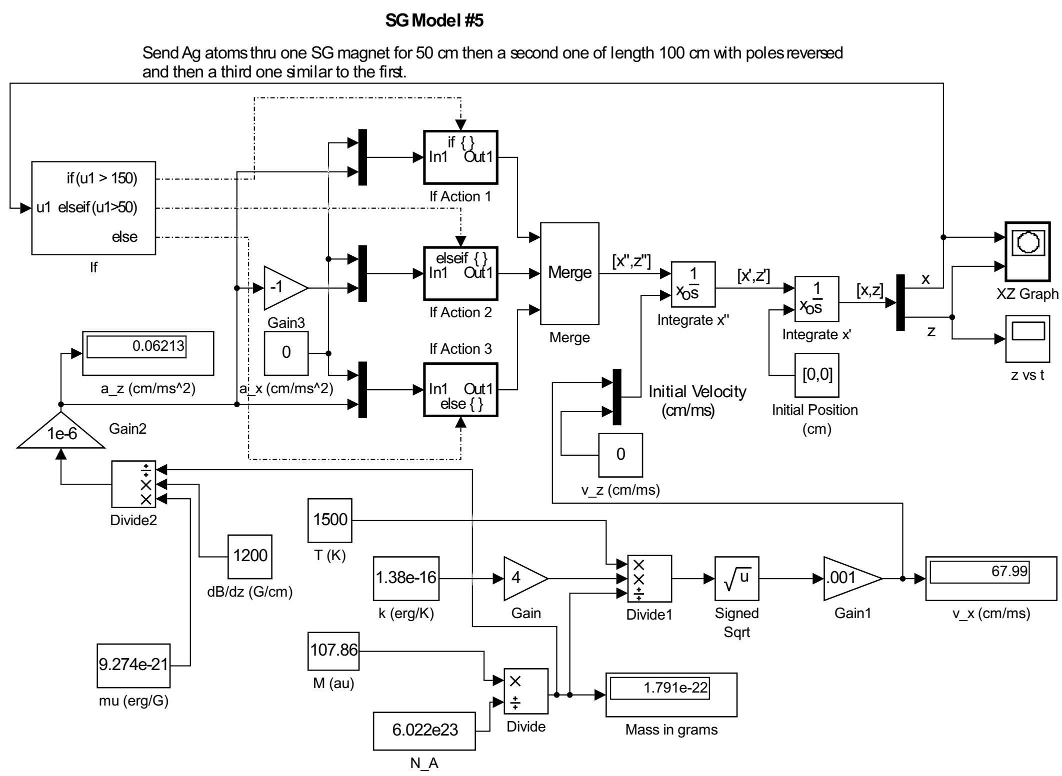


Figure 28: SG Experiment 5a.

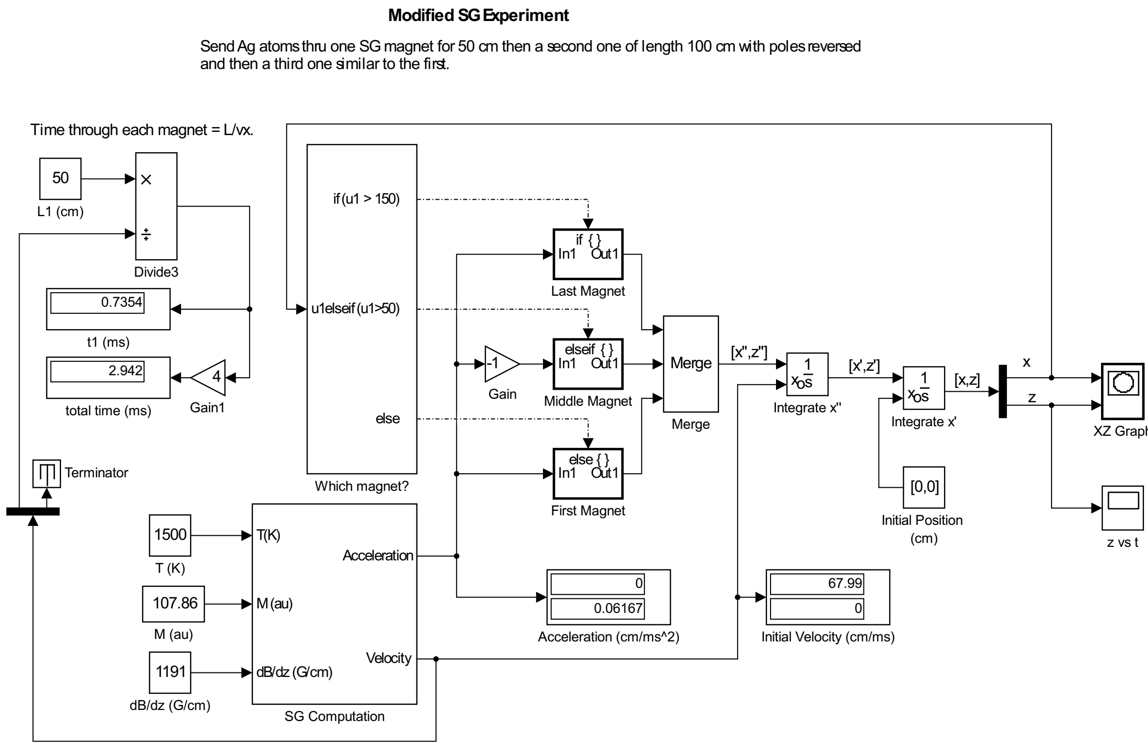


Figure 29: SG Experiment 5b.

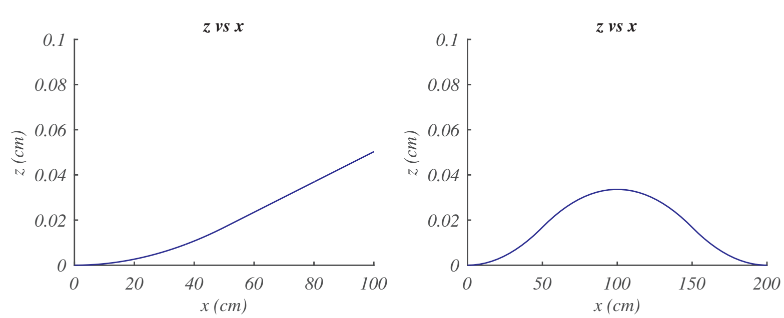


Figure 30: Results from SG Experiments 1 and 5.