3-060-T-Mma-DataToDifferentialEquation-TeacherVersion

Analyses for Data Set #2

Brian Winkel, Director SIMIODE
Chardon OH 44024 USA
BrianWinkel@simiode.org

All page numbers refer to those in Sullivan’s Modeling Scenario pdf. I moved all data sets into Mathematica from the Excel spreadsheet and I performed analyses on Data Set # 2 here.

Using the parameters for k and c offered on page 5 I was able to replicate the shapes of the total distance traversed.

Moreover, my position function compares favorably - VERY - to the toy position function from the parameters yo used to generate the data.

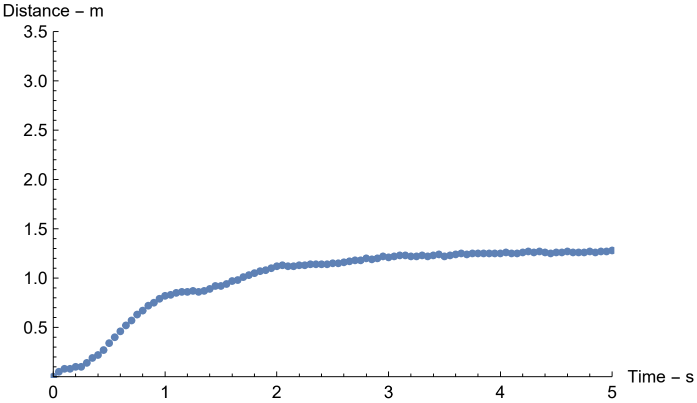
Analysis of Data from Data Set #2. Using discrete sum function to determine parameters in least squares sense.

We enter data for Data Set #2 from the spreadsheet.

```
data2 = {{0, 0.00}, {0.05, 0.05}, {0.1, 0.08}, {0.15, 0.08}, {0.2, 0.10},
   {0.25, 0.10}, {0.3, 0.14}, {0.35, 0.19}, {0.4, 0.22}, {0.45, 0.27}, {0.5, 0.34},
   {0.55, 0.40}, {0.6, 0.46}, {0.65, 0.52}, {0.7, 0.57}, {0.75, 0.63}, {0.8, 0.67},
   {0.85, 0.72}, {0.9, 0.75}, {0.95, 0.79}, {1, 0.82}, {1.05, 0.83}, {1.1, 0.85},
   {1.15, 0.86}, {1.2, 0.86}, {1.25, 0.87}, {1.3, 0.86}, {1.35, 0.87}, {1.4, 0.89},
   {1.45, 0.92}, {1.5, 0.92}, {1.55, 0.94}, {1.6, 0.97}, {1.65, 0.98}, {1.7, 1.01},
   {1.75, 1.03}, {1.8, 1.05}, {1.85, 1.07}, {1.9, 1.08}, {1.95, 1.10}, {2, 1.12},
   {2.05, 1.13}, {2.1, 1.12}, {2.15, 1.12}, {2.2, 1.13}, {2.25, 1.13}, {2.3, 1.14},
   {2.35, 1.14}, {2.4, 1.14}, {2.45, 1.14}, {2.5, 1.15}, {2.55, 1.15}, {2.6, 1.16},
   {2.65, 1.17}, {2.7, 1.18}, {2.75, 1.18}, {2.8, 1.20}, {2.85, 1.19}, {2.9, 1.20},
   {2.95, 1.22}, {3, 1.21}, {3.05, 1.22}, {3.1, 1.23}, {3.15, 1.23}, {3.2, 1.22},
   {3.25, 1.22}, {3.3, 1.23}, {3.35, 1.22}, {3.4, 1.23}, {3.45, 1.24}, {3.5, 1.22},
   {3.55, 1.23}, {3.6, 1.24}, {3.65, 1.25}, {3.7, 1.24}, {3.75, 1.25}, {3.8, 1.25},
   {3.85, 1.25}, {3.9, 1.25}, {3.95, 1.25}, {4, 1.25}, {4.05, 1.26}, {4.1, 1.25},
   {4.15, 1.25}, {4.2, 1.26}, {4.25, 1.27}, {4.3, 1.26}, {4.35, 1.27}, {4.4, 1.26},
   {4.45, 1.25}, {4.5, 1.26}, {4.55, 1.26}, {4.6, 1.27}, {4.65, 1.26}, {4.7, 1.26},
   {4.75, 1.26}, {4.8, 1.27}, {4.85, 1.26}, {4.9, 1.27}, {4.95, 1.27}, {5, 1.28}};
```

We plot the data. This plot matches the open circle data plot in Figure 1 on page 2.

```
data2Plot =
 ListPlot[data2, AxesLabel → {"Time - s", "Distance - m"}, PlotRange → {{0, 5}, {0, 3.5}}]
```



Here we offer differential equation in model (1) on page 3 of Sullivan' s Modeling Scenario. We use c = δ and k = κ .

```
eqMass2 = y''[t] + c y'[t] + k y[t] == 0
```

```
k y[t] + c y′[t] + y′′[t] == 0
```

We now offer a model with the parameters found on page 5 in Sullivan' s Modeling Scenario. It is our interest in doing a least squares estimate approach, but we are finding it hard to do with the Absolute value function!

Here is the solution with parameters k and c using the initial conditions given on page 1 for data set #2.

```
y2sol[t_, k_, c_] = y[t] /. DSolve[{eqMass2, y[0] == 0.5, y'[0] == 1}, y[t],
    t][[1]]
```

$$\frac{1}{\sqrt{c^2-4.\,k}}\,0.25\times\left(-4.\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}-1.\,c\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}+4.\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}+1.\,c\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}+1.\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}\sqrt{c^2-4.\,k}+1.\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}\sqrt{c^2-4.\,k}\right)$$

We compute the velocity, still with parameters k and c.

```
v2sol[t_, k_, c_] = D[y2sol[t, k, c], t]
```

$$\frac{1}{\sqrt{c^2-4.\,k}}\,0.25\times\left(-2.\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}\left(-c-\sqrt{c^2-4k}\right)-0.5\,c\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}\left(-c-\sqrt{c^2-4k}\right)+2.\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}\left(-c+\sqrt{c^2-4k}\right)+0.5\,c\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}\left(-c+\sqrt{c^2-4k}\right)+0.5\,e^{\frac{1}{2}\left(-c-\sqrt{c^2-4k}\right)t}\left(-c-\sqrt{c^2-4k}\right)\sqrt{c^2-4.\,k}+0.5\,e^{\frac{1}{2}\left(-c+\sqrt{c^2-4k}\right)t}\left(-c+\sqrt{c^2-4k}\right)\sqrt{c^2-4.\,k}\right)$$

If for Data Set #2 we use the model, call it Model2[t, k, c], we obtain the following model for total distance traversed using the integral offered in the second bullet to the students on page 2.

```
Model2[t_, k_, c_] := NIntegrate[Abs[v2sol[u, k, c]], {u, 0, t}, MaxRecursion → 12]
```

Here we plot the total distance traversed from Sullivan' s model. We use the parameters as explained on page 3 which supposedly generated the data - with three different initial conditions for data sets #1, #2, and #3, top of the page, namely c = δ = 2 and k = κ = 10.

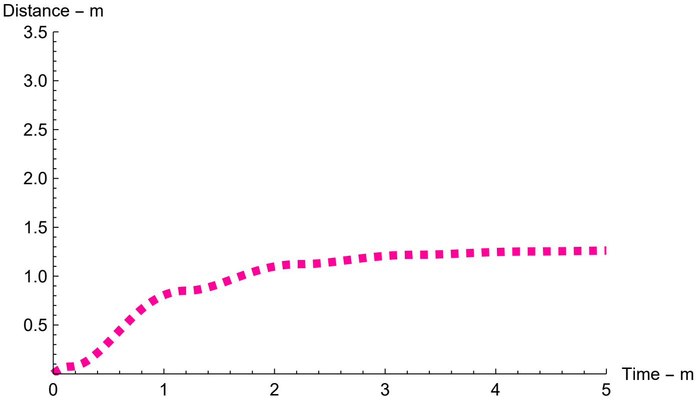
```
Model2[5, 10, 2]
```

```
1.26194
```

```
Model2Plot[k_, c_] :=
 Plot[Model2[t, k, c] , {t, 0, 5}, AxesLabel → {"Time - m", "Distance - m"},
  PlotStyle → {Dashed, Hue[.9], Thickness[.015]}, PlotRange → {{0, 5}, {0, 3.5}}]
```

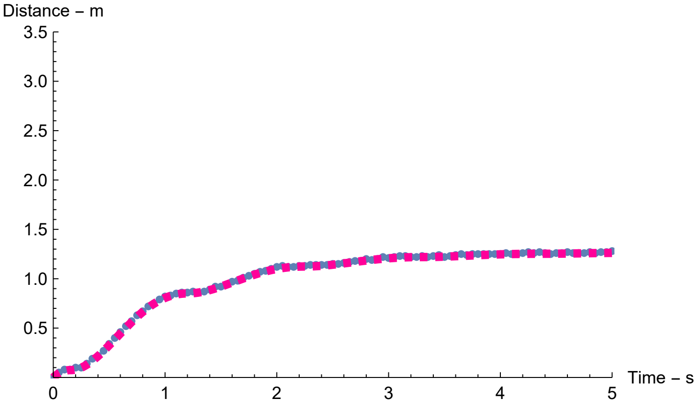
Here we plot the total distance traversed from Sullivan' s model. We use the generating parameters as explained on page 3, top of the page, namely c = δ = 2 and k = κ = 10.

```
Trial2 = Model2Plot[10, 2]
```



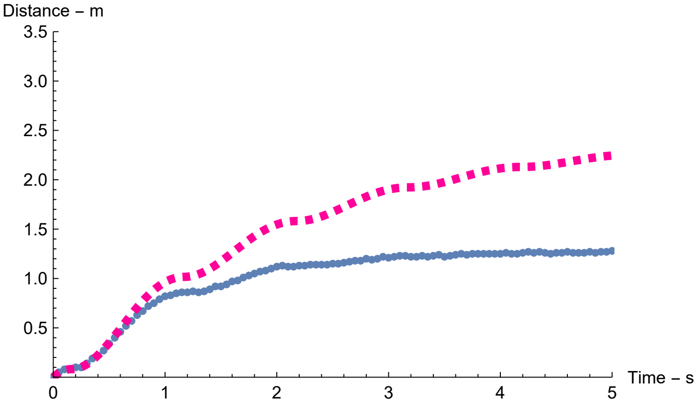
Here we show this model Model2(t, k, c) with parameters used to generate the data (c = δ = 2 and k = κ = 10) from Sullivan's page 5 of Modeling Scenario. This shows the model with Parameter values c = δ = 2 and k = κ = 10 predicts the same total distance traversed as the data. This is good!

```
Show[data2Plot, Trial2, PlotRange → {{0, 5}, {0, 3.5}}]
```



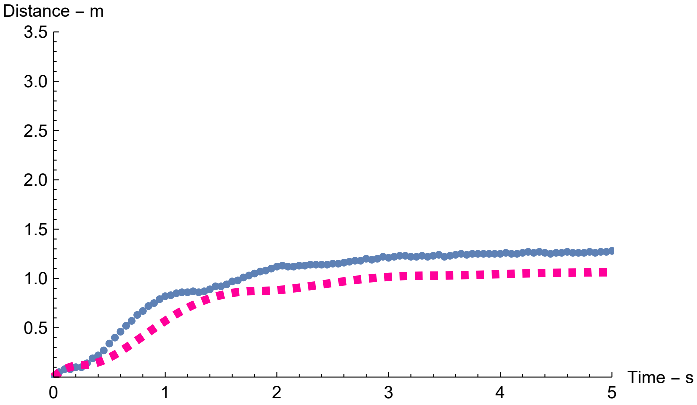
We take several values of c = δ and k = κ to show how we can be “off” in our estimates for the model (dashed line) does not predict the data well. Here we use c = δ = 1 and k = κ = 10. This means we have less resistance than used in the data generator and as expected the spring would traverse more distance over time, and the dashed plot of our model compared to that of the data shows just that.

```
Show[data2Plot, Model2Plot[10, 1], PlotRange → {{0, 5}, {0, 3.5}}]
```



Here we use c = δ = 2 and k = κ = 5. This means we have a smaller spring constant than used in the data generator and as expected the spring would traverse less distance over time for the spring is not pulled with as great a force for a given displacement than in the data generator and the dashed plot shows just that.

```
Show[data2Plot, Model2Plot[5, 2], PlotRange → {{0, 5}, {0, 3.5}}]
```



Here we discretize the total distance model using the solution to the differential equation with **discrete** estimates of distance traversed in each 0.05 s interval, using D = R*T for Abs[v2sol[u,k,c]]*.05. This yields a sum for total distance traversed, instead of an integral, which is quicker to compute.

```
Model2[t_, k_, c_] := NIntegrate[Abs[v2sol[u, k, c]], {u, 0, t}, MaxRecursion → 12]
```

```
ModelDiscrete2[k_, c_, m_] := Sum[Abs[v2sol[data2⟦i, 1⟧, k, c]] * .05, {i, 1, m}]
```

We form the sum of square errors (SSE(k, c)) between our model for a given parameter set (k, c) and the data set as a function of the parameters (k, c) in anticipation of finding the parameters k and c which minimize this SSE.

```
SSE[k_, c_] := Sum[(data2⟦i, 2⟧ - ModelDiscrete2[k, c, i])^2, {i, 1, Length[data2]}]
```

```
SSE[10, 1]
```

```
39.9427
```

```
SSE[5, 2]
```

```
2.64885
```

```
SSE[10, 2]
```

```
0.0615905
```

We compute values for SSE(k, c) over a grid to estimate the parameter which renders the smallest value of SSE(k, c).

```
Timing[datakcSSE = {};
 Do[
  Do[
   AppendTo[datakcSSE, {k, c, SSE[k, c]}],
   {k, 9.5, 10.5, .05}],
  {c, 1.5, 2.5, .05}]]
```

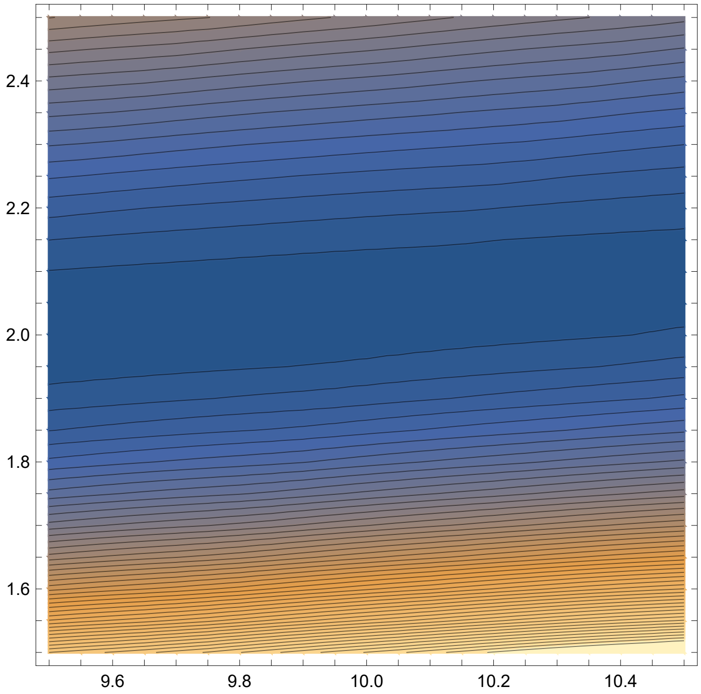
```
{111.727916, Null}
```

```
Length[datakcSSE] (*** Number of data points (k, c, SSE). ***)
```

```
441
```

Here we plot contours in the parameter space (k, c, SSE(k, c)).

```
ListContourPlot[datakcSSE, Contours → 50]
```



Of those SSE(k,c) we have computed over our grid we determine the values of k and c which give the smallest SSE(k, c).

```
sol = Sort[datakcSSE, #1[[3]] < #2[[3]] &][[1]]
{9.5, 2., 0.0218147}
```

Thus with parameters c = 2 and k = 9.5 we have a SSE(k,c) = 0.021847.

From the parameters obtained on page 5, namely c = 1.706 and k = 9.67963 we find a SSE of 1.8238. So our parameters c = 2.0 and k = 9.5 offer smaller SSE(k, c).

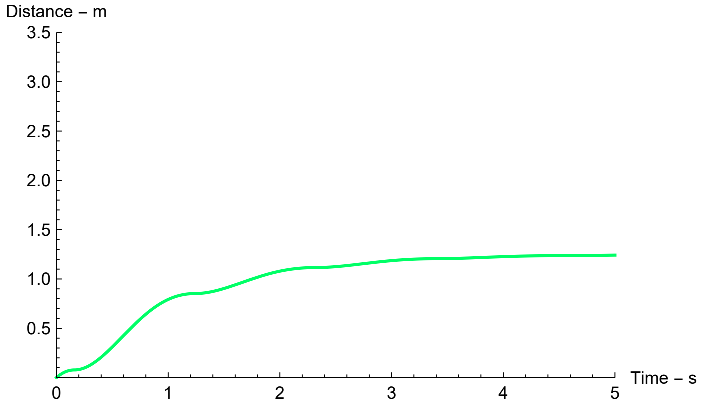
```
SSE[9.67963, 1.706]
1.8238
```

We build the integral for traversed distance as a function of the parameters k and c at time t.

```
Model2[t_, k_, c_] :=
 NIntegrate[Abs[v2sol[u, k, c]], {u, 0, t}, PrecisionGoal → 12, MaxRecursion → 40]
```

When we use the parameters from our least sum of squares approach we get a plot (green below) which is almost identical to that obtained from the data. This suggests that our parameters are quite good.

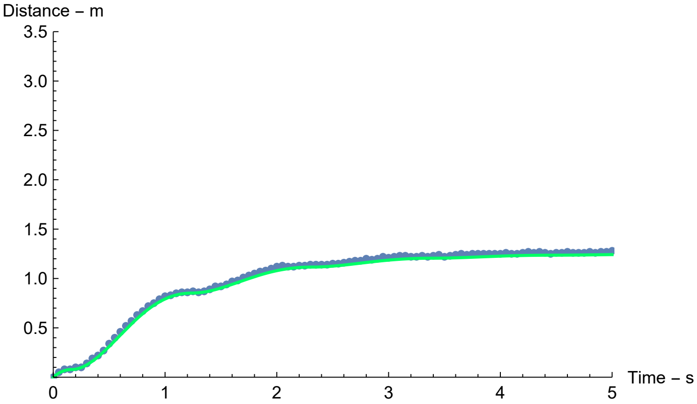
```
Model2Plot =
 Plot[Model2[t, sol[[1]], sol[[2]]] , {t, 0, 5}, AxesLabel → {"Time - s", "Distance - m"},
  PlotStyle → Hue[.4], PlotRange → {{0, 5}, {0, 3.5}}]
```



Indeed there is no visible difference in the results; still higher than the observed distance traversed, though.

Finally we show our two models results (green- discrete model and blue - data).

```
Show[data2Plot, Model2Plot, PlotRange → {{0, 5}, {0, 3.5}}]
```



As a final check we note that on page 3 we see the "toy problem", ytoy(t), had k = 10 and c = 2 as generator parameters and our model, ymod(t), we found plotted on the same axes. Nice match!

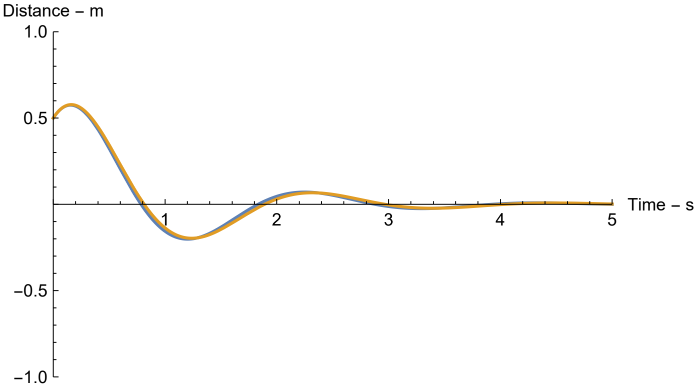
```
ytoy[t_] =
 y[t] /. DSolve[{y''[t] + 2 y'[t] + 10 y[t] == 0, y[0] == 0.5, y'[0] == 1}, y[t], t][[1]]
```

$0.5\, e^{-t}\, (1.\, \mathrm{Cos}[3\, t] + 1.\, \mathrm{Sin}[3\, t])$

```
ygen[t_] = y[t] /.
  DSolve[{y''[t] + sol[[2]] × y'[t] + sol[[1]] × y[t] == 0, y[0] == 0.5, y'[0] == 1}, y[t], t][[1]]
```

$0.5\, e^{-1.\, t}\, (1.\, \mathrm{Cos}[2.91548\, t] + 1.02899\, \mathrm{Sin}[2.91548\, t])$

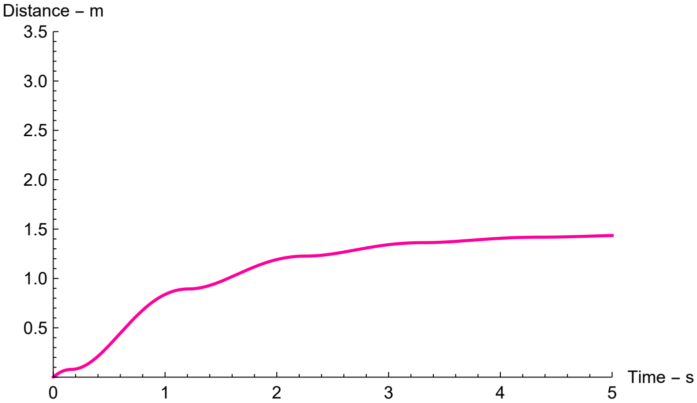
```
Plot[{ytoy[t], ygen[t]}, {t, 0, 5},
 PlotRange → {{0, 5}, {-1, 1}}, AxesLabel → {"Time - s", "Distance - m"}]
```



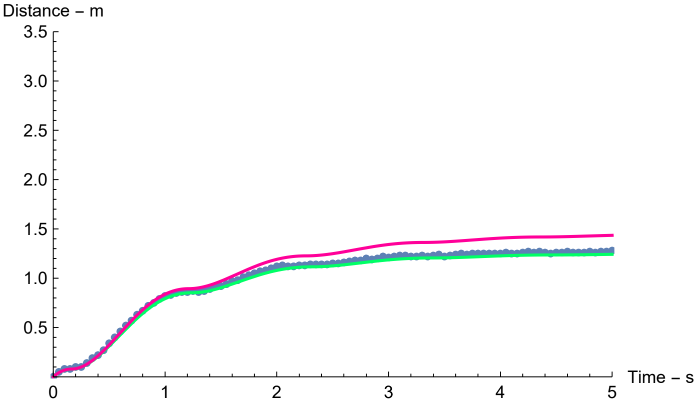
This looks like the Figure 3 plot of Data Set #2. So we are pleased with our SSE parameter estimating analysis.

If we use the parameter values found for Data Set # 1 and offered on page 5 we see that our model is not as good as with the SSE approach.

```
ModelSullivanPlot =
 Plot[Model2[t, 9.67963, 1.706] , {t, 0, 5}, AxesLabel → {"Time - s", "Distance - m"},
  PlotStyle → Hue[.9], PlotRange → {{0, 5}, {0, 3.5}}]
```



```
Show[data2Plot, Model2Plot, ModelSullivanPlot, PlotRange → {{0, 5}, {0, 3.5}}]
```



We consider a finer grid for our parameters k and c.

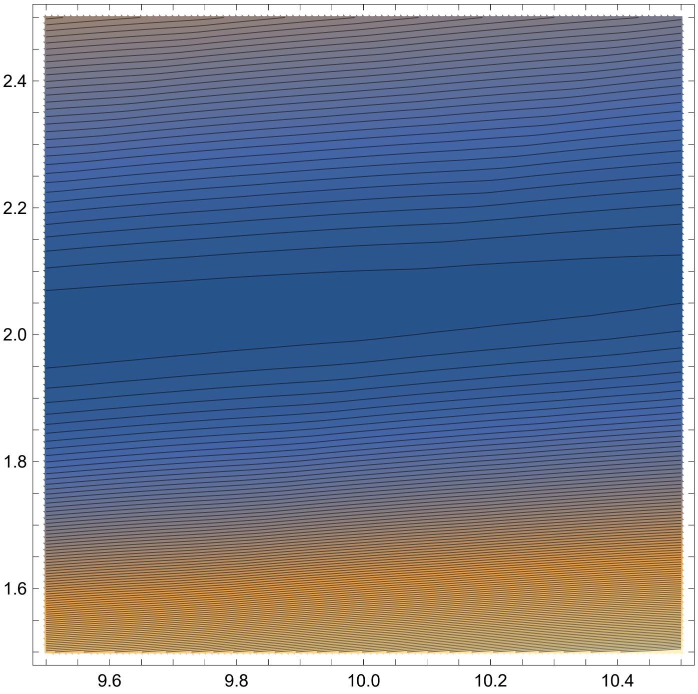
```
Timing[datakcSSE = {};
 Do[
  Do[
   AppendTo[datakcSSE, {k, c, SSE[k, c]}],
   {k, 9.5, 10.5, .01}],
  {c, 1.5, 2.5, .01}]]
```

```
{2642.142137, Null}
```

```
Length[datakcSSE] (*** Numb er of data points (k, c, SSE). ***)
```

```
10 201
```

```
ListContourPlot[datakcSSE, Contours → 100]
```



Of those SSE(k,c) we have computed over our grid we determine the values of k and c which give the smallest SSE(k, c).

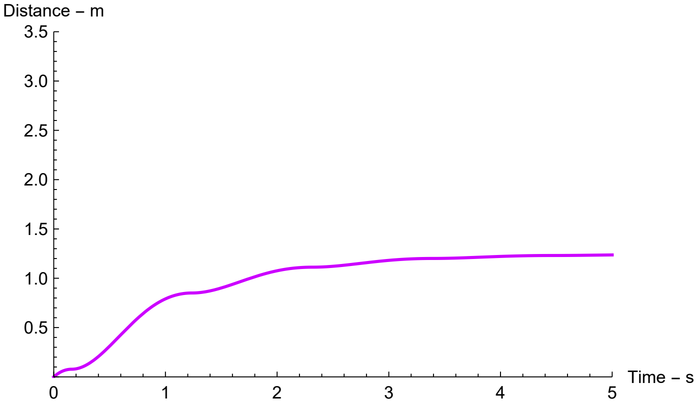
```
sol = Sort[datakcSSE, #1⟦3⟧ < #2⟦3⟧ &] ⟦1⟧
```

```
{9.5, 2.01, 0.0212246}
```

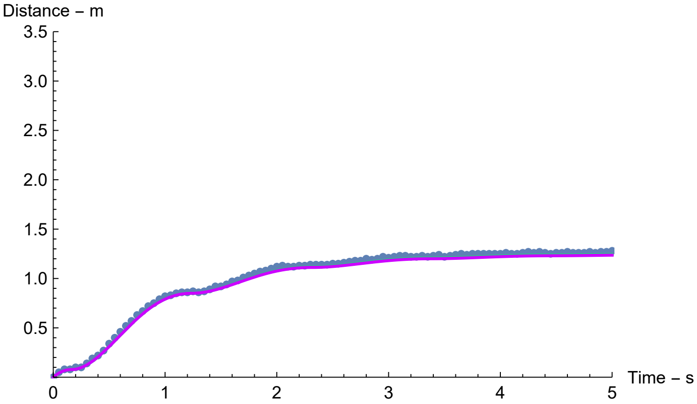
We see a slight diminution in SSE(kc) from when k = 9.5, c = 2.0, and SSE(9.5,2.01, 0.021815).

But not much different in the plot of distance traversed.

```
Model2aPlot =
 Plot[Model2[t, sol⟦1⟧, sol⟦2⟧] , {t, 0, 5}, AxesLabel → {"Time - s", "Distance - m"},
  PlotStyle → Hue[.8], PlotRange → {{0, 5}, {0, 3.5}}]
```



```
Show[data2Plot, Model2Plot, Model2aPlot, PlotRange → {{0, 5}, {0, 3.5}}]
```


Distance – m
Time – s