Differential Equations

Your Name Goes Here

Introduction

The goal of this notebook is to introduce you to the process of solving differential equations in *Mathematica*. We will introduce you to the `DSolve` command which is useful for finding exact solutions. We will also see how *Mathematica* can be used to draw slope fields to visualize the solution to differential equations. Furthermore, we can visualize the steps of Euler's method and write an automated procedure for computing a numerical solution to a differential equation.

Execute All The Cells In This Section Before Beginning

Evaluate every cell below every time you begin working on this notebook. It initializes certain commands that you will be using. You do not need to understand the commands in this section.

```
SlopeField[F_, {var1_, a_, b_}, {var2_, c_, d_}] := VectorPlot[{1, F}, {var1, a, b},
    {var2, c, d}, VectorScale → {Min[Min[Abs[b - a], Abs[d - c]]/40, 0.03], Automatic, None},
    VectorStyle → "Segment", Frame → None, Axes → True, AxesLabel → {x, y}, VectorPoints → 20];

EulerTable[F_, {x0_, y0_}, h_, n_, row_: All] :=
  Module[{upperBound, RHS, ode, soln, X, Y, i, exact},
   upperBound = x0 + h n;
   RHS = F /. y → y[x];
   ode = y'[x] == RHS;
   soln = Flatten@NDSolve[{ode, y[x0] == y0}, y, {x, x0, upperBound}];

   X = {x0};
   Y = {y0};
   For[i = 1, i ≤ n, i++,
    AppendTo[X, X[[i]] + h];
    AppendTo[Y, Y[[i]] + h F /. {x → X[[i]], y → Y[[i]]}];
   ];

   exact = (y[#] & /@ X) /. soln;
   If[row === All,
    TableForm[{Range[0, n], X, Y, exact, Abs[exact - Y]}^T,
     TableHeadings → {None, {"i", "x_i", "y_i (Euler)", "y(x_i) (Exact)", "Error"}}],
    TableForm[{Transpose[{Range[0, n], X, Y, exact, Abs[exact - Y]}][[row + 1]]},
     TableHeadings → {None, {"i", "x_i", "y_i (Euler)", "y(x_i) (Exact)", "Error"}}]
   ]
  ];
```

```
EulerPlot[F_, {x0_, y0_}, h_, n_] :=
  Module[{Xupper, RHS, ode, soln, X, Y, i, exact, Ylower, Yupper},
   Xupper = x0 + h n;
   RHS = F /. y → y[x];
   ode = y'[x] == RHS;
   soln = Flatten@NDSolve[{ode, y[x0] == y0}, y, {x, x0, Xupper}];

   X = {x0};
   Y = {y0};
   For[i = 1, i ≤ n, i++,
    AppendTo[X, X[[i]] + h];
    AppendTo[Y, Y[[i]] + h F /. {x → X[[i]], y → Y[[i]]}];
   ];

   exact = (y[#] & /@ X) /. soln;
   Ylower = Min[{Y, exact}^T];
   Yupper = Max[{Y, exact}^T];
   Legended[Show[
     Plot[Evaluate[y[x] /. soln], {x, x0, Xupper}, PlotStyle → Blue],
     ListLinePlot[{X, Y}^T, PlotStyle → Red],
     VectorPlot[{1, F}, {x, x0, Xupper}, {y, Ylower, Yupper},
      VectorScale → {Min[Min[Abs[Xupper - x0], Abs[Yupper - Ylower]]/40, 0.03], Automatic, None},
      VectorStyle → {{Gray, "Segment"}}, VectorPoints → 20],
     Show[Graphics[{PointSize[0.01], Red, Point /@ Transpose[{X, Y}],
        Blue, Point /@ Transpose[{X, exact}]}]],
     AxesLabel → {x, y}, ImageSize → Large, PlotRange → All
    ], LineLegend[{Blue, Red}, {"Exact Solution y(x)", "Euler Approximation of y(x)"}]]
  ];

(* http://
 mathematica.stackexchange.com/questions/18012/label-area-on-plot/18079#18079 *)
braceLabel[{p1_, p2_}, lbl_, scale_: .02] :=
  {Arrowheads[{{scale, 0, {Graphics@Circle[{1, -1}, 1, {Pi/2, Pi}], -1}}, {scale, 1,
      {Graphics[{Circle[{-1, 1}, 1, {-Pi/2, 0}], Rotate[Inset[lbl, {0, 4}], 180 Degree]}],
       1}}}], Arrow[{p1, (p1 + p2)/2}],
   Arrowheads[{{scale, 0, {Graphics@Circle[{1, 1}, 1, {Pi, 3 Pi/2}], -1}},
     {scale, 1, {Graphics@Circle[{-1, -1}, 1, {0, Pi/2}], 1}}}], Arrow[{(p1 + p2)/2, p2}]};

EulerAnimate[F_, {x0_, y0_}, h_, n_] :=
 Module[{Xupper, RHS, ode, soln, X, Y, i, exact, Ylower, Yupper, veclength,
   exactplot, vecplot, vecplotlight, initialpoint, whitestepsize, frames, frame1,
   frame2, frame3, contourpieces, points, currentslope, frame, whitepointplot},
  Xupper = x0 + h n;
  RHS = F /. y → y[x];
  ode = y'[x] == RHS;
  soln = Flatten@NDSolve[{ode, y[x0] == y0}, y, {x, x0, Xupper}];

  X = {x0};
  Y = {y0};
  For[i = 1, i ≤ n, i++,
   AppendTo[X, X[[i]] + h];
```

```
 AppendTo[Y, Y[[i]] + h F /. {x → X[[i]], y → Y[[i]]}];
];

exact = (y[#] & /@ X) /. soln;
Ylower = Min[{Y, exact}^T];
Yupper = Max[{Y, exact}^T];

veclength = Min[ Min[Abs[Xupper - x0], Abs[Yupper - Ylower]] / 40, 0.03 ];

exactplot =
 Plot[Evaluate[y[x] /. soln], {x, x0, Xupper}, PlotStyle → Blue, AspectRatio → Automatic];
vecplot = VectorPlot[{1, F}, {x, x0, Xupper}, {y, Ylower, Yupper}, VectorScale →
    {veclength, Automatic, None}, VectorStyle → {{Gray, "Segment"}}, VectorPoints → 20];
vecplotlight = VectorPlot[{1, F}, {x, x0, Xupper}, {y, Ylower, Yupper},
  VectorScale → {veclength, Automatic, None},
  VectorStyle → {{LightGray, "Segment"}}, VectorPoints → 20];
initialpoint = Graphics[{Red, PointSize[Large], Point[{x0, y0}]}];
whitestepsize = With[ {startPoint = {h, Ylower - (Yupper - Ylower)/13},
    endPoint = {X[[1]], Ylower - (Yupper - Ylower)/13}, scale = .02}, Graphics[
    {White, braceLabel[{startPoint, endPoint}, Style["h (stepsize)", Larger], scale]}] ];
whitepointplot = ListPlot[{X, Y}^T, PlotStyle → White];

frames = {};

frame1 = Legended[Show[
    exactplot,
    whitepointplot,
    whitestepsize,
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
  ], Column[{LineLegend[{Blue}, {"Exact Solution y(x)"}]}]];
AppendTo[frames, frame1];

frame2 = Legended[Show[
    exactplot,
    whitepointplot,
    initialpoint,
    whitestepsize,
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
  ], Column[{LineLegend[{Blue}, {"Exact Solution y(x)"}], PointLegend[{Red},
      {"Starting Point (x0, y0)"}, LegendMarkers → {Graphics[Disk[]]}]}]];
AppendTo[frames, frame2];

frame3 = Legended[Show[
    exactplot,
    whitepointplot,
    initialpoint,
    vecplot,
    whitestepsize,
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
  ],
  Column[{LineLegend[{Gray, Blue}, {"Slope Field", "Exact Solution y(x)"}], PointLegend[
```

```
      {Red}, {"Starting Point (x0, y0)"}, LegendMarkers → {Graphics[Disk[]]}]}]];
AppendTo[frames, frame3];

contourpieces = {};
points = {initialpoint};

For[i = 1, i ≤ n, i++,
 currentslope = F /. {x → X[[i]], y → Y[[i]]};

 If[i > 1,
  frame = Legended[Show[
      exactplot,
      vecplotlight,
      contourpieces,
      points,
      whitestepsize,
      AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
     ], Column[{LineLegend[{Red, LightGray, Blue},
        {"Approximate Solution", "Slope Field", "Exact Solution y(x)"}],
       PointLegend[{Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <>
          ToString[i - 1] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
          ToString[i - 1] <> "\)]\))"}, LegendMarkers → {Graphics[Disk[]]}]}]];
  AppendTo[frames, frame];
 ];

 frame = Legended[Show[
    exactplot,
    vecplotlight,
    If[i > 1, contourpieces, {}],
    ContourPlot[y - Y[[i]] == currentslope (x - X[[i]]), {x, X[[i]], X[[i]] + 3 veclength},
     {y, Ylower, Yupper}, ContourStyle → Directive[Black, Thickness[0.02]]],
    points,
    whitestepsize,
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
   ],
   If[i > 1,
    Column[{LineLegend[{Black, Red, LightGray, Blue},
       {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
         "\)]\),\!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))",
        "Approximate Solution", "Slope Field", "Exact Solution y(x)"}], PointLegend[
       {Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
         "\)]\), \!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))"},
       LegendMarkers → {Graphics[Disk[]]}]}], Column[{LineLegend[
       {Black, LightGray, Blue}, {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <>
         ToString[i - 1] <> "\)]\),\!\(\*SubscriptBox[\(y\), \(" <>
         ToString[i - 1] <> "\)]\))", "Slope Field", "Exact Solution y(x)"}],
      PointLegend[{Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <>
         ToString[i - 1] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
         ToString[i - 1] <> "\)]\))"}, LegendMarkers → {Graphics[Disk[]]}]}]
   ]
  ];
 AppendTo[frames, frame];

 frame = Legended[Show[
```

```
    exactplot,
    vecplotlight,
    If[i > 1, contourpieces, {}],
    ContourPlot[y - Y[[i]] == currentslope (x - X[[i]]), {x, X[[i]], X[[i]] + h},
     {y, Ylower, Yupper}, ContourStyle → Directive[Black, Thickness[0.02]]],
    points,
    With[{startPoint = {X[[1]] + i h, Ylower - (Yupper - Ylower)/13},
      endPoint = {X[[1]] + (i - 1) h, Ylower - (Yupper - Ylower)/13}, scale = .02}, Graphics[
      braceLabel[{startPoint, endPoint}, Style["h (stepsize)", Larger], scale]]],
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
   ],
   If[i > 1,
    Column[{LineLegend[{Black, Red, LightGray, Blue},
       {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
         "\)]\),\!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))",
        "Approximate Solution", "Slope Field", "Exact Solution y(x)"}], PointLegend[
       {Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
         "\)]\), \!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))"},
       LegendMarkers → {Graphics[Disk[]]}]}], Column[{LineLegend[
       {Black, LightGray, Blue}, {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <>
         ToString[i - 1] <> "\)]\),\!\(\*SubscriptBox[\(y\), \(" <>
         ToString[i - 1] <> "\)]\))", "Slope Field", "Exact Solution y(x)"}],
      PointLegend[{Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <>
         ToString[i - 1] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
         ToString[i - 1] <> "\)]\))"}, LegendMarkers → {Graphics[Disk[]]}]}]
  ]
 ];
AppendTo[frames, frame];

frame = Legended[Show[
    exactplot,
    vecplotlight,
    If[i > 1, contourpieces, {}],
    ContourPlot[y - Y[[i]] == currentslope (x - X[[i]]), {x, X[[i]], X[[i]] + h},
     {y, Ylower, Yupper}, ContourStyle → Directive[Black, Thickness[0.02]]],
    points,
    With[{startPoint = {X[[1]] + i h, Ylower - (Yupper - Ylower)/13},
      endPoint = {X[[1]] + (i - 1) h, Ylower - (Yupper - Ylower)/13}, scale = .02}, Graphics[
      braceLabel[{startPoint, endPoint}, Style["h (stepsize)", Larger], scale]]],
    Graphics[{EdgeForm[{Red, Thick}], White, Disk[{X[[i + 1]], Y[[i + 1]]}, 0.01]}],
    AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
   ],
   If[i > 1,
    Column[{LineLegend[{Black, Red, LightGray, Blue},
       {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
         "\)]\),\!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))",
        "Approximate Solution", "Slope Field", "Exact Solution y(x)"}],
```

```
        PointLegend[{Red, Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <>
           ToString[i - 1] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
           ToString[i - 1] <> "\)]\))", "New Point (\!\(\*SubscriptBox[\(x\), \(" <>
           ToString[i] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
           ToString[i] <> "\)]\))"}, LegendMarkers → {Graphics[Disk[]],
           Graphics[{EdgeForm[{Red, Thickness[0.3]}], White, Disk[]}]}]}],
      Column[{LineLegend[{Black, LightGray, Blue},
         {"Line with slope F(\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
           "\)]\),\!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))",
          "Slope Field", "Exact Solution y(x)"}], PointLegend[{Red, Red},
         {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <> ToString[i - 1] <>
           "\)]\), \!\(\*SubscriptBox[\(y\), \(" <> ToString[i - 1] <> "\)]\))",
          "New Point (\!\(\*SubscriptBox[\(x\), \(" <> ToString[i] <>
           "\)]\), \!\(\*SubscriptBox[\(y\), \(" <> ToString[i] <> "\)]\))"}, LegendMarkers →
          {Graphics[Disk[]], Graphics[{EdgeForm[{Red, Thickness[0.3]}], White, Disk[]}]}]}]
     ]
    ];
   AppendTo[frames, frame];

   AppendTo[contourpieces, ContourPlot[y - Y[[i]] == currentslope (x - X[[i]]),
     {x, X[[i]], X[[i]] + h}, {y, Ylower, Yupper}, ContourStyle → Red]];
   AppendTo[points, Graphics[{Red, PointSize[Large], Point[{X[[i + 1]], Y[[i + 1]]}]}]];
   frame = Legended[Show[
      exactplot,
      initialpoint,
      vecplotlight,
      contourpieces,
      With[{startPoint = {X[[1]] + i h, Ylower - (Yupper - Ylower)/13},
        endPoint = {X[[1]] + (i - 1) h, Ylower - (Yupper - Ylower)/13}, scale = .02}, Graphics[
        braceLabel[{startPoint, endPoint}, Style["h (stepsize)", Larger], scale]]],
      points,
      Graphics[{EdgeForm[{Red, Thick}], White, Disk[{X[[i + 1]], Y[[i + 1]]}, 0.01]}],
      AxesLabel → {x, y}, ImageSize → Large, PlotRange → {{x0, Xupper}, All}
     ], Column[{LineLegend[{Red, LightGray, Blue},
        {"Approximate Solution", "Slope Field", "Exact Solution y(x)"}],
       PointLegend[{Red, Red}, {"Starting Point (\!\(\*SubscriptBox[\(x\), \(" <>
          ToString[i - 1] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
          ToString[i - 1] <> "\)]\))", "New Point (\!\(\*SubscriptBox[\(x\), \(" <>
          ToString[i] <> "\)]\), \!\(\*SubscriptBox[\(y\), \(" <>
          ToString[i] <> "\)]\))"}, LegendMarkers → {Graphics[Disk[]],
          Graphics[{EdgeForm[{Red, Thickness[0.3]}], White, Disk[]}]}]}]];
   AppendTo[frames, frame];
  ]; (* end for *)

  ListAnimate[frames, AnimationRunning → False]
 ](* end module *)
```

Mathematica Commands

Representing Differential Equations in *Mathematica*

Recall that an (ordinary) differential equation is a mathematical equation that relates some function of one variable with its derivatives. For example, here is a first-order non-homogeneous differential equation:

$$2\,y'(x) - 3\,y(x) = x^2$$

This differential equation has independent variable x and dependent variable y. Solving a differential equation consists essentially of finding the form of an unknown function. In *Mathematica*, unknown functions are represented by expressions like `y[x]` where `y` is the dependent variable and `x` is the independent variable. The derivatives of such functions are represented by `y'[x]`, `y''[x]`, and so on. To represent the previous example in *Mathematica* we would use the following notation:

```
2 y'[x] - 3 y[x] == x^2
```

It is very important to notice that there are two equal signs above. A single equal sign is used for assignment (storing values) while two equal signs is used for equality. Because of this difference, you can store differential equations in variables. Here, we store the differential equation above in the variable `myDifferentialEquation`:

```
myDifferentialEquation = 2 y'[x] - 3 y[x] == x^2;
```

Solving Differential Equations (DSolve)

To solve a differential equation in *Mathematica*, we use the **`DSolve`** command. **`DSolve[ode, y[x], x]`** solves the differential equation **`ode`** which has dependent variable **`y`** and independent variable **`x`**. For example, to solve the differential equation $2\,y'(x) - 3\,y(x) = x^2$ we do the following:

```
myDifferentialEquation = 2 y'[x] - 3 y[x] == x^2;
DSolve[myDifferentialEquation, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{1}{27}\left(-8 - 12\,x - 9\,x^2\right) + e^{3\,x/2}\,C[1]\right\}\right\}$$

From this output we can see that the general solution to the differential equation is

$$y(x) = \frac{1}{27}\left(-8 - 12\,x - 9\,x^2\right) + e^{3\,x/2}\,C[1].$$

The $C[1]$ term is how *Mathematica* represents arbitrary constants. There is one arbitrary constant in the solution because the order is one.

Initial Value Problems

An initial value problem is a differential equation that is given with one or multiple initial conditions. Initial conditions look like:

$$\begin{aligned} y(x_0) &= y_0 \\ y'(x_0) &= y_1 \\ &\vdots \end{aligned}$$

When you are given an initial condition (or multiple initial conditions) you can solve a differential equation to get a particular solution (one without arbitrary constants). To solve a differential equation with initial conditions, we again use the **`DSolve`** command. **`DSolve[{ode, initialCondition1, initialCondition2, ...}, y[x], x]`** solves the differential equation **`ode`** which has dependent variable **`y`** and independent variable **`x`** and has initial conditions **`initialCondition1, initialCondition2, ...`**. For example, suppose we want to solve the follow-

ing differential equation with one initial condition:

$$2y'(x) - 3y(x) = x^2$$
$$y(3) = 5$$

As before we represent the differential equation and initial condition as an equality using two equal signs and then use **DSolve**:

```
myDifferentialEquation = 2 y'[x] - 3 y[x] == x^2;
initialCondition1 = y[3] == 5;
DSolve[{myDifferentialEquation, initialCondition1}, y[x], x]
```

$$\left\{\left\{y[x] \to -\frac{8\,e^{9/2} - 260\,e^{3x/2} + 12\,e^{9/2}\,x + 9\,e^{9/2}\,x^2}{27\,e^{9/2}}\right\}\right\}$$

From this output we can see that the general solution to the differential equation is

$$y(x) = -\frac{8e^{9/2} - 260e^{3x/2} + 12e^{9/2}x + 9e^{9/2}x^2}{27e^{9/2}}.$$

As another example, here is a second-order differential equation with two initial conditions:

$$y''(x) + 2y'(x) - 3y(x) = e^x$$
$$y(0) = 1$$
$$y'(0) = 2$$

As before we represent the differential equation and initial conditions as an equality using two equal signs and then use **DSolve**:

```
myDifferentialEquation = y''[x] + 2 y'[x] - 3 y[x] == Exp[x];
initialCondition1 = y[0] == 1;
initialCondition2 = y'[0] == 2;
DSolve[{myDifferentialEquation, initialCondition1, initialCondition2}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{1}{16}\,e^{-3x}\left(-3 + 19\,e^{4x} + 4\,e^{4x}x\right)\right\}\right\}$$

From this output we can see that the general solution to the differential equation is

$$y(x) = \frac{1}{16}e^{-3x}\left(-3 + 19e^{4x} + 4e^{4x}x\right).$$

An initial value problem is a differential equation that is given with boundary conditions. Boundary conditions look like:

$$y(x_0) = y_0$$
$$y(x_1) = y_1$$
$$\vdots$$

Solving boundary value problems is just like solving initial value problems. **DSolve[{ode, boundaryCondition1, boundaryCondition2, ...}, y[x], x]** solves the differential equation **ode** which has dependent variable **y** and independent variable **x** and has boundary conditions **boundaryCondition1, boundaryCondition2, ...**. For example, suppose we want to solve the following differential equation with two boundary conditions:

$$y''(x) + 2y'(x) - 3y(x) = e^x$$
$$y(0) = 1$$
$$y(3) = 2$$

As before we represent the differential equation and boundary conditions as an equality using two equal signs and then use **DSolve**:

```
myDifferentialEquation = y''[x] + 2 y'[x] - 3 y[x] == Exp[x];
boundaryCondition1 = y[0] == 1;
boundaryCondition2 = y[3] == 2;
DSolve[{myDifferentialEquation, boundaryCondition1, boundaryCondition2}, y[x], x]
```

$$\left\{\left\{y[x] \to \frac{e^{-3x}\left(-8e^{9}+7e^{12}-4e^{4x}+8e^{9+4x}-3e^{12+4x}-e^{4x}x+e^{12+4x}x\right)}{4\left(-1+e^{12}\right)}\right\}\right\}$$

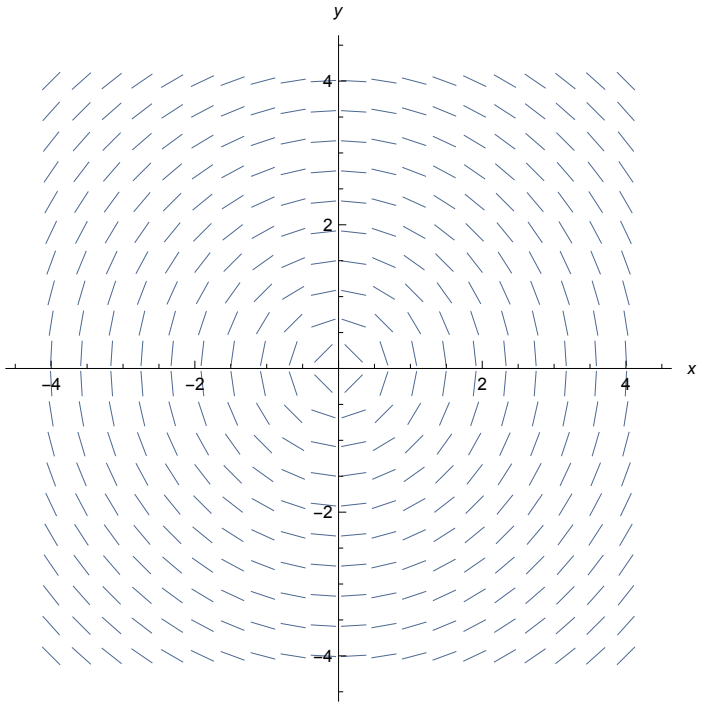
From this output we can see that the general solution to the differential equation is

$$y(x) = \frac{e^{-3x}\left(-8e^{9}+7e^{12}-4e^{4x}+8e^{9+4x}-3e^{12+4x}-e^{4x}x+e^{12+4x}x\right)}{4\left(-1+e^{12}\right)}.$$

Slope Fields

To draw a slope field in *Mathematica* we use the **SlopeField** command. **SlopeField[F, {x, a, b}, {y, c, d}]** allows us to visualize the solutions to the differential equation $y' = F(x, y)$ where $a \le x \le b$ and $c \le y \le d$. For example, we can visualize $y' = -\frac{x}{y}$ where $-4 \le x \le 4$ and $-4 \le y \le 4$ by doing the following:

```
SlopeField[-x / y, {x, -4, 4}, {y, -4, 4}]
```



Euler's Method

To get approximate solutions to ordinary differential equations, we can use Euler's method. Suppose our differential equation looks like: $y' = F(x, y)$. Euler's method is an algorithm that takes three inputs, and outputs a set of points

$$y(x_0) = y_0$$

with the following specification:

Input

$F(x, y)$, a function in the variables x and y,
(x_0, y_0), a point,
h, a stepsize
n, the number of steps

Output

$\{(x_0, y_0), \ldots, (x_n, y_n)\}$, a list of points such that $x_i = x_{i-1} + h$

$$y_i = y_{i-1} + h\,F(x_{i-1}, y_{i-1})$$

Euler's method is implemented three different ways. The first is using **EulerTable**. **EulerTable[F, {x0, y0}, h, n]** outputs a table of values with $n + 1$ rows, calculated during Euler's method, such as the approximate solution points (x_i, y_i), the actual solution points $(x_i, y(x_i))$, and the error, which is the absolute value of the difference between the actual solution and the approximate solution: $| y(x_i) - y_i |$. For example, suppose we wish to execute Euler's method using a stepsize of 0.1 for 20 steps for the differential equation

$y' = x + y$.
$y(0) = 1$

We would execute the following:

```
EulerTable[x + y, {0, 1}, 0.1, 20]
```

i	x_i	y_i (Euler)	$y(x_i)$ (Exact)	Error
0	0	1	1.	0.
1	0.1	1.1	1.11034	0.0103419
2	0.2	1.22	1.24281	0.0228056
3	0.3	1.362	1.39972	0.0377176
4	0.4	1.5282	1.58365	0.0554494
5	0.5	1.72102	1.79744	0.0764226
6	0.6	1.94312	2.04424	0.101116
7	0.7	2.19743	2.32751	0.130071
8	0.8	2.48718	2.65108	0.163904
9	0.9	2.8159	3.01921	0.203311
10	1.	3.18748	3.43656	0.249078
11	1.1	3.60623	3.90833	0.302098
12	1.2	4.07686	4.44023	0.363376
13	1.3	4.60454	5.03859	0.434047
14	1.4	5.195	5.7104	0.5154
15	1.5	5.8545	6.46338	0.608881
16	1.6	6.58995	7.30606	0.716119
17	1.7	7.40894	8.24789	0.83895
18	1.8	8.31983	9.29929	0.979453
19	1.9	9.33182	10.4718	1.13997
20	2.	10.455	11.7781	1.32311

If you execute Euler's method with a large number of steps, this table gets excessively big. The **EulerTable** command has an additional argument that allows you to pick out a specific row of the table. For instance, in the table above, we can use the following command to the get the row where $i = 10$:

```
EulerTable[x + y, {0, 1}, 0.1, 20, 10]
```

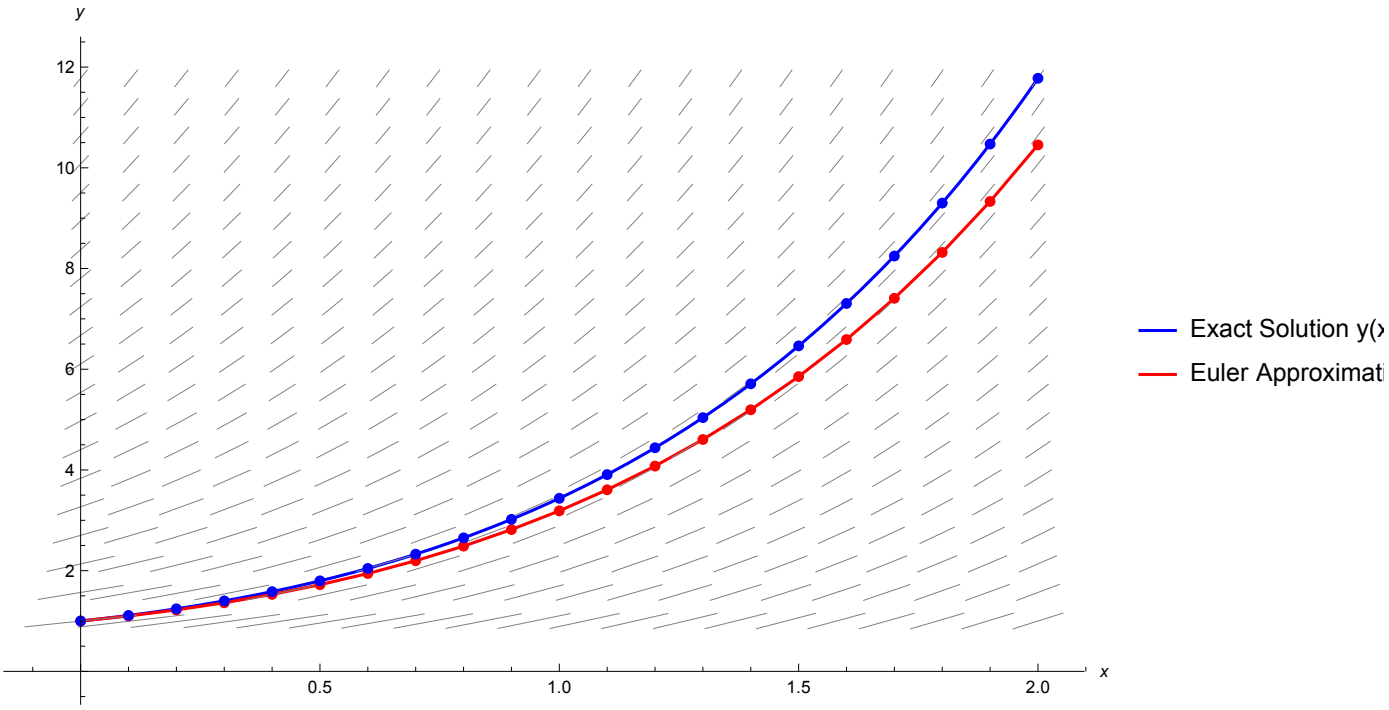
i	x_i	y_i (Euler)	$y(x_i)$ (Exact)	Error
10	1.	3.18748	3.43656	0.249078

To help visualize Euler's method, we use the **EulerPlot** command. **EulerPlot[F, {x0, y0}, h, n]** outputs the points calculated using Euler's method, connected by straight line segments, along with a plot of a high quality numerical solution (nearly exact solution). For example, suppose we wish to execute Euler's method using a step-size of 0.1 for 20 steps for the differential equation

$y' = x + y$.
$y(0) = 1$

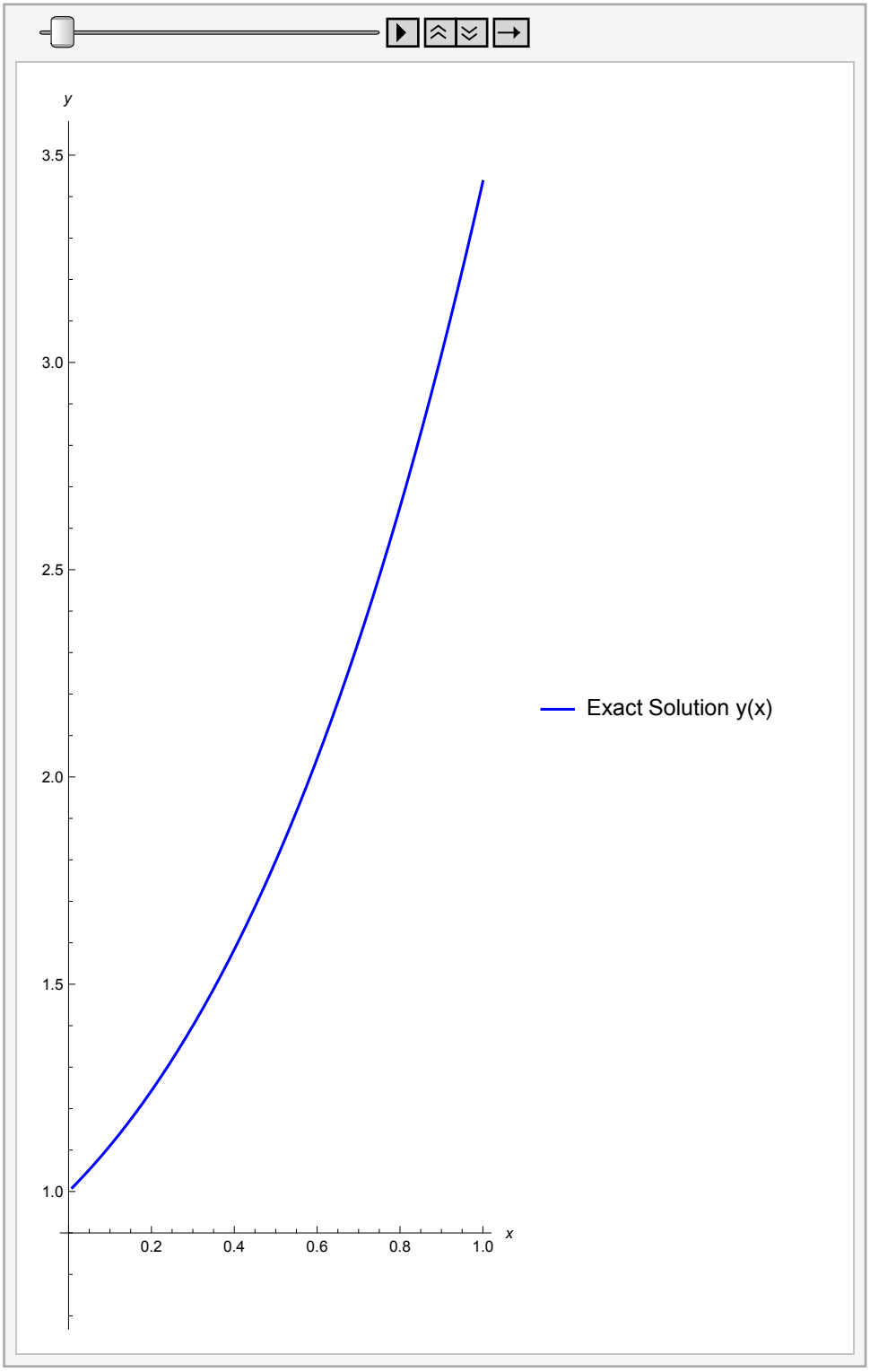
We could visualize this by executing the following:

```
EulerPlot[x + y, {0, 1}, 0.1, 20]
```



To give a step by step animation of Euler's method, we can use the **`EulerAnimate`** command. It has the same specification as the **`EulerPlot`** command. Here we illustrate Euler's method with a stepsize of 0.5 for 2 steps. Drag the slider to move through the frames of the animation:

```
EulerAnimate[x + y, {0, 1}, 0.5, 2]
```



Problems

(a) Compute the general solution of the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$.

You will notice that the solution is quite strange. This is because *Mathematica* is having difficulty finding an analytic solution.

(b) Compute the particular solution of the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$ with initial condition $y(1) = 0$.

Nothing particularly changes when we add an initial condition.

(c) Draw the slope field for the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$ for $-4 \le x \le 4$ and $-4 \le y \le 4$.

(e) Execute the code below. Click and drag the slider to see how Euler's method with a stepsize of 0.5 and 2 steps works when solving the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$.

$y(1) = 0$

```
EulerAnimate[y / Exp[x] + 3 Cos[x] + 2, {1, 0}, 0.5, 2]
```

(d) Execute the code below. Click and drag your mouse to change the initial point (red point). The red curve is a high accuracy numerical approximation to the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$. What interesting behavior do you observe as you change the initial point?

Answer:

```
Manipulate[
 Show[
  VectorPlot[{1, y/Exp[x] + 3 Cos[x] + 2}, {x, -4, 4},
   {y, -4, 4}, VectorScale → {0.03, Automatic, None}, VectorStyle → "Segment",
   Frame → None, Axes → True, AxesLabel → {x, y}, VectorPoints → 20],
  Plot[Evaluate[y[x] /. NDSolve[{y'[x] == y[x]/Exp[x] + 3 Cos[x] + 2, y[p[[1]]] == p[[2]]},
      y, {x, -4, 4}]], {x, -4, 4}, PlotRange → {-4, 4}, PlotStyle → Red]
 ],
 {{p, {1, 0}}, Locator, Appearance → Graphics[{Red, PointSize[0.02], Point[{1, 1}]}]},
 AppearanceElements → None, Paneled → False]
```

(e) Consider the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2$. Use the EulerPlot and

$$y(1) = 0$$

EulerTable command with a stepsize of 0.1 for 30 steps to help visualize the approximate solution to this differential equation. Compare the plot you get with the plot above when you move the initial condition to the point (1, 0). Answer the following questions.

- According to Euler's method, what is $y(4)$?
  Answer:
- What is the exact value of $y(4)$?

Answer:

- What is the error in the approximation?

  Answer:

(f) Consider the differential equation $y' = \frac{y}{e^x} + 3\cos(x) + 2.$

$$y(1) = 0$$

What would be an appropriate number of steps and an appropriate stepsize so that Euler's method gives an approximate value of $y(4)$ that is within 0.0003 away from the exact value of $y(4)$?

Hint: Use `EulerTable` *with a carefully chosen stepsize and and step count number.Then choose the correct row to display.*

Answer: