Math 21a: Multivariable Calculus, Fall 2000

Mathematica Assignment 1

Support

Welcome to this Mathematica computer assignment! In case of technical problems with this assignment please consult first the FAQ page at http://www.courses.fas.harvard.edu/~math21a/labs/faq.html

If the problem is not listed and you are still stuck, send e−mailto knill@math.harvard.edu. We will

come back to you as soon as possible. Please be aware that *Mathematica* is a pretty complex program

demanding a lot of resources from computers, whether it is a Unix workstation, a Mac or PS. Save

therefore the notebook frequently during your work. This paragraph had also to be written twice because the notebook suddenly disappeared. As everybody of you knows, we unfortunatly still have to live with segmentation faults, errors of type 1 etc.

Credit

In order to get credit for this notebook, you have to read through the notebook and answer the questions in magneta colered boxes. The completed notebook and printed has to be turned in to your CA until Friday October 13, 2000.

For printing information at the computer labs in the Science center, see http://www.fas.harvard.edu/computing/guides/printing.html

You are welcome to collaborate. However, every student has finally to fill out and print out his or her own notebook. At one point, your creativity will be challenged in finding (discovering, creating) new three dimensional curves. Printing out an original curve (one which does not appear in this notebook) gives already full credit for this assignment.The best curves will be posted with your name on the math21a website. This is a chance to have named a space curve after you! By the way, closed space curves are called **knots** and play an important role in biology (DNA of Bacteria), Physics (quantum field theory) or mathematics (three dimensional manifolds).

Introduction to Mathematica

Mathematica Cells

Mathematica notebooks are subdivided into cells, each of which is marked by a blue bracket at the right of the window. These cells come in several different types. For instance, the text you are currently reading is in a "text" cell, while the expression below is in an "input" cell.

```
In[9]:= 76 - 23
```

You can tell the different types of cells apart by the font they use: text cells use a normal weight proportional width font (Times), while input cells use a boldface typewriter−stylefont (**Courier**). Text cells are just for show, containing instructions, descriptions, or explanations. If you want *Mathematica* to evaluate a mathematical expression, you must put it in an input cell. To create a new cell, position the cursor between two existing cells or after the last cell. The cursor will then change from a vertical I−beamto a horizontal I−beam. If you click there, *Mathematica* will draw a horizontal line across the window. If you then start typing, *Mathematica* will create a new input cell for you. Any new cell you create will automatically be an input cell. (Creating a text cell takes an additional step, but you won't be needing to create any text cells in these labs.)

Try creating a few input cells and entering a few expressions in the space below.

Evaluating Expressions

All commands are entered into *Mathematica* as text. To evalute an expression, click anywhere in the expression and press the <Enter> key on the keypad (not the one on the main keyboard) or hold down the <Shift> key and press <Enter>. *Mathematica* will then think for a moment and write the answer below the expression. Your input and *Mathematica*'s output are labeled "In[#]:" and "Out[#]:" respectively, and *Mathematica* draws the input in **boldface** to distinguish it from the plain text output. For example, to add the following two numbers, click the sum below and press <Enter>.

In[10]:=
```
3 + 8
```

Throughout these *Mathematica* assignments, we will give you lots of expressions to evaluate, along with descriptive text. Just go through it one cell at a time, reading the text and evaluating each of the expressions one by one. In these notebooks, cells which require you to try out what you have learned will be boxed in blue.

Basic Operators

The basic operations are **+** (addition), **-** (subtraction), ***** (multiplication), **/** (division), and **^** (exponentiation). *Mathematica* obeys the usual operator precedence, first performing exponentiation, then multiplication and division, and finally addition and subtraction. If you want to group your operations differently, use parentheses **()**. Try evaluating the following expressions:

In[11]:=
```
12 - 3 * 5
```

In[12]:=
```
(12 - 3) * 5
```

In[13]:=
```
6 / 2 * 3
```

In[14]:=
```
6 / (2 * 3)
```

In[15]:=
```
-5^2
```

In[16]:=
```
(-5)^2
```

If you put two expressions next to each other separated only by a space, *Mathematica* multiplies them, just like in normal mathematical notation. This can be confusing if you just put two numbers next to each other, but it comes in handy with more complicated expressions. Here are some examples:

In[17]:=
```
5 2
```

In[18]:=
```
3 (8 - 4)
```

Functions and Constants

Mathematica comes preprogrammed with lots of functions and constants. Some of the functions we will be using are **Sin[x]**, **Cos[x]**, **Exp[x]** (the exponential e^x), **Log[x]** (the natural logarithm $\ln x$), **Sqrt[x]** (square root $\sqrt{x}$), and **Abs[x]** (the absolute value $|x|$). Note that the first letter of all built−infunctions are capitalized, and you use square brackets **[]** to apply them, as opposed to regular parentheses **()**. Some useful constants that Mathematica knows are **Pi** (or π) and **E** (or $e \approx 2.71828$, the base for natural logarithms). Again, be sure to capitalize the first letter. Alternatively, you can use the palette of templates and symbols on the right of your screen to type in some of these operators and constants. (If you do not see the palette, select from the menu File −Palettes−BasicInput.) Here are some examples.

In[19]:=
```
Sin[0.123]
```

Just like in standard math notation, if you write two things next to each other with only a space between them, *Mathematica* multiplies them.

In[20]:=
```
Cos[2 Pi]
```

When you evaluated this next expression, note that Mathematica is clever enough to give you the exact answer, rather than a decimal approximation.

In[21]:=
```
Sin[π / 4]
```

If you prefer a numeric approximation, add **//N** to the end of your expression.

In[1]:=
```
Sin[π / 4] // N
```

Defining Variables and Functions

Mathematica is more than just a fancy calculator. It lets you do both symbolic and numerical calculations using variables and functions. You can use any string of letters for a variable name. In addition to the usual variable names **x**, **y**, **t**, you can also use more descriptive names like **theta** or **radius**. You can assign variables a value and then use that value in further computations. Here are some examples:

In[22]:=
```
a = 5
```

In[23]:=
```
a^2 - 3 a + 4
```

In[24]:=
```
b = -3
```

In[25]:=
```
a b
```

Notice that if you write two variable names next to each other separated only by a space, *Mathematica* multiplies them, just like in normal mathematical notation. However, you must take care to put in the space between your variable names. Otherwise, *Mathematica* will think that you are talking about a single variable with a long name! For instance, in the following cell, instead of multiplying **a** by **b**, *Mathematica* assumes you are using a new variable **ab**:

In[26]:=
```
ab
```

To define a function for *Mathematica*, we write **f[x_] := value**, where **f** is the name of the function, **x** is the variable, and

To define a function for *Mathematica*, we write **f[x_] := value**, where **f** is the name of the function, **x** is the variable, and value is the expression for the function. Note that when defining a function, you again use square brackets **[]** instead of parentheses. In *Mathematica*, this distinction is very important; parentheses **()** are used to group expressions, while square brackets **[]** are used when applying a function. When defining a function, you use colon−equal **:=** instead of a normal equal sign, and you put an underscore _ immediately after the variable name. This tells *Mathematica* to evaluate the right side only after plugging in the value of the variable. Note that *Mathematica* does not give you an output when you define a function, but once a function is defined, you can apply it to any number you want, or even to a variable or expression. When you are finished typing out

your function definition in the input cell, you must type 'enter' on the keypad or hold down

'shift' and press 'enter' on the regular key board to tell *Mathematica* that your definition is ready.

(This tells Matheamtica to read the input cell). Give the following functions a try:

In[27]:=
```
f[x_] := x^2 - 3 x + 4
```

In[28]:=
```
f[5]
```

In[29]:=
```
f[t + 1]
```

Here we define a function of two variables. Note that your variable and function names can be as long or as descriptive as you like.

In[30]:=
```
DistanceFromOrigin[x_, y_] := Sqrt[x^2 + y^2]
```

In[31]:=
```
DistanceFromOrigin[3, 5]
```

Exercise: Quadratic Formula

Do you remember the quadratic formula from high school? It gives you the roots of a quadratic polynomial $a\,x^2 + b\,x + c$. In case you still don't remember, here it is: $\frac{-b \pm \sqrt{b^2 - 4\,a\,c}}{2\,a}$. Sure that's one formula, but it still takes a while to plug in the values and compute the roots by hand. However, using *Mathematica*, you can define functions to do all the hard work for you. Try creating a few new cells below and define two functions **PlusRoot[a,b,c]** and **MinusRoot[a,b,c]** to compute these two roots (corresponding to the plus−or−minus sign in the numerator). Then use your functions to find the roots of the polynomial $3\,x^2 + 2\,x - 5$. (Hint: the answers are 1 and $-\frac{5}{3}$)

Here are some things to remember:

- To create a new input cell, position the cursor below this gray box so that *Mathematica* switches the cursor to a horizontal I−beam, click the mouse, and then start typing. Make sure you are typing into a new input cell (with bold typewriter−styletext in the **Courier** font) and not into one of the pre−existingtext cells. If you are typing into a text cell, you will not be able to evaluate your expression.
- To take a square root, use the **Sqrt[]** function, and remember that the first letter is capitalized.
- Remember to use parentheses **()** to group expressions and square brackets **[]** when applying functions.
- To multiply two variables together, write them next to each other separated by a space. If you leave out the space, *Mathematica* thinks it is one variable with a two letter name.
- When you define a function, don't forget to put an underscore after your variable names **(a_, b_, c_)** on the left hand side, and use colon−equals **:=** instead of just equals **=** in the function definition. Otherwise, *Mathematica* will use the values of **a**, **b**, **c** that you defined earlier in this notebook.
- Remember that before you use your functions, you must first 'evaluate'your function definitions by pressing 'enter' on the key pad or holding down 'shift' and pressing 'enter' on the standard key board.
- If you get the wrong answer, make sure that the expressions are grouped together correctly with parentheses. Your functions will need to be of the form (numerator) / (denominator).

Graphing Functions

To graph a function in *Mathematica*, you need to tell *Mathematica* the function itself, the name of the dependent variable, and the domain of the function in the following format:

Plot[*function to be plotted***, {***dependent variable***, ** *lower range***, ** *upper range***}]**

Note that we use **[]** for the **Plot** function. To describe the domain of the function, we enclose the dependent variable and the bounds in curly braces **{}**. For instance, to plot the function $f(x) = e^{-\frac{x^2}{2}}$ (called a Gaussian distribution function) in the domain $-3 \le x \le 3$ you would type:

In[32]:=
```
f[x_] := Exp[-x^2 / 2]
```

In[33]:=
```
Plot[f[x], {x, -3, 3}]
```

In the Plot command, be sure to write **f[x]** including the variable name. If you just write **f**, *Mathematica* will think that **f** is a variable name, not a function. As a shorthand, you can skip the step where you define the function separately, and include that in the Plot command itself:

In[34]:=
```
Plot[Exp[-x^2 / 2], {x, -3, 3}]
```

Try defining and plotting the following functions using the **Plot** command:

$y = \ln x \qquad -5 \le x \le 5$

$g(t) = \sin t \qquad -2\pi \le t \le 2\pi$

$F(x) = \int_0^x g(t)\, dt \qquad -2\pi \le x \le 2\pi$

(Use the definite integral template ($\int_{\square}^{\square} \square\, d\square$) from the toolbox on the right.)

Parametric Curves

2D Vector−Valued Functions

Parameterizing a Circle

As we have seen, *Mathematica*'s **Plot[]** command lets us plot the graph of a function. However, not all curves in the plane can be described as the graph of a function. In particular, a function can have only one value of y for each value of x. Suppose we want to draw the unit circle, given by the equation $x^2 + y^2 = 1$. To draw it as a graph of a function, we first solve this equation for y to obtain $y = \pm\sqrt{1 - x^2}$. But since there are two possible values for the square root, we cannot draw it as a graph of a single function.

Instead, we can plot the circle by treating the two coordinates $x = \cos(t)$ and $y = \sin(t)$ as functions of a third variable t called a parameter. However, before getting to the circle, it is instructive to first plot the graphs of x and y as functions of t separately:

In[35]:=
```
x[t_] := Cos[t]
```

In[36]:=
```
Plot[x[t], {t, 0, 2 Pi} ]
```

In[37]:=
```
y[t_] := Sin[t]
```

In[38]:=
```
Plot[y[t], {t, 0, 2 Pi}]
```

Now, to plot the circle, combine them both into a single vector−valued function $\mathbf{r}(t) = x(t)\,\mathbf{i} + y(t)\,\mathbf{j}$. In general, *Mathematica* represents vectors (both vector−valued functions and vector values) by separating the vector components with comma (**,**) and enclosing them in curly braces **{}**. Thus, we can represent the vector−valued function in *Mathematica* as follows:

In[39]:=
```
r[t_] := {x[t], y[t]}
```

Use *Mathematica* to compute the vector $\mathbf{r}(t)$ corresponding to various values of t:

In[40]:=
```
r[0]
```

In[41]:=
```
r[Pi / 4]
```

In[42]:=
```
r[Pi / 2]
```

In[43]:=
```
r[t]
```

If you base these vectors at the origin, their endpoints describe a circle as t varies from 0 to 2π. In *Mathematica*, you plot this curve using the **ParametricPlot[]** command, which requires you to specify a vector−valued function, the dependent variable, and the domain in the following format:

ParametricPlot[*vector−valued function to be plotted***,**
{*dependent variable***,** *lower range***,** *upper range***},**
AspectRatio−> Automatic]

where **AspectRatio−> Automatic** tells *Mathematica* to use the same scale for both the x and y axes.

In[44]:=
```
ParametricPlot[r[t], {t, 0, 2 Pi}, AspectRatio -> Automatic]
```

If we plot a predefined vector−valuedfunction such as **r[t]**, *Mathematica* generates a warning but proceeds with the parametric plot. To prevent the warning message from being displayed, use the **Evaluate[]** version of the function **r[t]** to tell *Mathematica* to substitute the definition of **r[t]** into the function to be plotted, as shown in the example below.

In[45]:=
```
ParametricPlot[Evaluate[r[t]], {t, 0, 2 Pi}, AspectRatio -> Automatic]
```

In the above example, you used the domain $0 \le t \le 2\pi$. What happens if you take a larger domain? Try it in the input cell below. What happens if you take a smaller domain? Specify a domain in the cell below that draws only the bottom half of the circle.

In[46]:=
```
ParametricPlot[Evaluate[r[t]], {t, __, __}, AspectRatio -> Automatic]
```

Parameterizing Ellipses

To change the circle into an ellipse (making x the major axis), try the vector−valuedfunction $\mathbf{r}(t) = 2\cos(t)\,\mathbf{i} + \sin(t)\,\mathbf{j}$:

In[47]:=
```
r[t_] := {2 Cos[t], Sin[t]}
```

In[48]:=
```
ParametricPlot[Evaluate[r[t]], {t, 0, 2 Pi}, AspectRatio -> Automatic]
```

Find a vector−valued function parameterizing the ellipse with major axis of length a and minor axis of length b centered at the point (c, d). Write your function in terms of the constants a, b, c, d:

In[49]:=
```
r[t_] := {__, __}
```

Now try plotting the above curve to see if you got it right. Use the cell below to plot the ellipse centered at (2, 1) with major and minor axes of length 4 and 3, respectively. The first line assigns these values to a, b, c, d, while the second actually draws the graph. If the plot doesn't draw, make sure to evaluate your function definition for **r[t_]** above first.

In[50]:=
```
a = 4; b = 3; {c, d} = {2, 1};
ParametricPlot[Evaluate[r[t]], {t, 0, 2 Pi}, AspectRatio -> Automatic]
```

More Examples

Now, change the function to something more complicated, such as $\mathbf{r}(t) = (\, t - \cos(t^2)\,)\,\mathbf{i} + e^{\sin(\pi t)}\,\mathbf{j}$:

In[51]:=
```
r[t_] := {(t - Cos[t^2]) , Exp[Sin[Pi t]] }
```

In[52]:=
```
ParametricPlot[Evaluate[r[t]], {t, 0, 2 Pi}, AspectRatio -> Automatic]
```

Next, try graphing some other vector−valued functions by replacing the **i** and **j** components above with the functions below and re−evaluating them to get new graphs.

$$t^2 - t \qquad |t| \qquad t^3 - 1 \qquad t + |t|$$

Now, come up with some interesting functions of your own. (Recall the absolute value $|t|$ is given by *Mathematica*'s **Abs[]** function.) Having done so, vary the domain of t. For the trigonometric functions used above, it makes sense to use the domain $0 \le t \le 2\pi$, since $\sin(t)$ and $\cos(t)$ are periodic with period 2π. However, other functions might have interesting behavior outside of that domain. Below are three more functions for you to experiment with. Choose an appropriate domain for each (i.e., replace the __ with your minimum and maximum values for t) and graph them.

$\mathbf{r}(t) = 4\cos(\frac{t}{2})\,\mathbf{i} + 4\sin(\frac{t}{2})\,\mathbf{j}$

In[53]:=
```
r[t_] := {4 Cos[t / 2], 4 Sin[t / 2]}
```

In[54]:=
```
ParametricPlot[Evaluate[r[t]], {t, __, __}, AspectRatio -> Automatic]
```

$\mathbf{r}(t) = (\, t - \sin(t)\,)\,\mathbf{i} + (\, 1 - \cos(t)\,)\,\mathbf{j}$

In[55]:=
```
r[t_] := {(t - Sin[t]), (1 - Cos[t])}
```

In[56]:=
```
ParametricPlot[Evaluate[r[t]], {t, __, __}, AspectRatio -> Automatic]
```

$\mathbf{r}(t) = t\,\mathbf{i} + (\, t^2 + 1\,)\,\mathbf{j}$

In[57]:=
```
r[t_] := {t, (t^2 + 1)}
```

In[58]:=
```
ParametricPlot[Evaluate[r[t]], {t, __, __}, AspectRatio -> Automatic]
```

Polar Coordinates

Returning to the circle case, any circle centered at the origin can be parameterized by the vector−valued function $\mathbf{r}(\theta) = r\,(\cos(\theta)\,\mathbf{i} + \sin(\theta)\,\mathbf{j})$, where r is the radius and θ is the angle from the positive x axis. Here the radius r is constant, but what if we varied r with θ? In other words, what would happen if we changed the radius while the circle was being drawn? Such a function $r(\theta)$ is said to be in polar coordinates. Try setting $r(\theta) = \theta$ and see what happens.

In[59]:=
```
r[theta_] := theta
```

In[60]:=
```
ParametricPlot[Evaluate[r[theta] {Cos[theta], Sin[theta]}],
 {theta, 0, 2 Pi}, AspectRatio -> Automatic]
```

Now play with the cells above to graph a spiral that winds around the origin three times.

Next, change the radius to be a periodic function, such as $r(\theta) = \cos(2\,\theta)$.

In[61]:=
```
r[theta_] := Cos[2 theta]
```

In[62]:=
```
ParametricPlot[Evaluate[r[theta] {Cos[theta], Sin[theta]}],
 {theta, 0, 2 Pi}, AspectRatio -> Automatic]
```

Change the cells above to come up with an equation to make an 8−leafclover. Then draw a 3−leafclover. Do you need the whole domain $0 \le \theta \le 2\,\pi$ for the 3−leafclover? Try to find the smallest domain that draws a good 3−leafclover. Good luck! Experiment with other functions $r(\theta)$ in the cells below. You may be surprised at some of the exotic curves you get. Finally, draw a straight line segment using polar coordinates.

In[63]:=
```
r[theta_] := __
```

In[64]:=
```
ParametricPlot[Evaluate[r[theta] {Cos[theta], Sin[theta]}],
 {theta, 0, 2 Pi}, AspectRatio -> Automatic]
```

3D Vector−ValuedFunctions

Helix

In this section, we will experiment with three dimensional vector−valued functions. First, we will draw a helix, or a 3D spiral, using the function $\mathbf{r}(t) = \cos(t)\,\mathbf{i} + \sin(t)\,\mathbf{j} + t\mathbf{k}$. Modify the function given below to get a helix that wraps around four times.

In[65]:=
```
r[t_] := {Cos[t], Sin[t], t}
```

In[66]:=
```
ParametricPlot3D[Evaluate[r[t]], {t, 0, 2 Pi}]
```

Now change t to $\frac{t^2}{2\,\pi}$. Notice that the plotted curve is identical to the original. Try again by replacing t with $\frac{t^3}{4\,\pi^2}$ and then $2\,\pi \sin(\frac{t}{4})$. You should notice that changing the parameterization has no effect on the curve because all these replacements for t are functions of t that vanish at $t = 0$ and equal $2\,\pi$ at $t = 2\,\pi$. Try to come up with at least one reparameterization of your own.

In[67]:=
```
r[t_] := {Cos[__], Sin[__], __}
```

In[68]:=
```
ParametricPlot3D[Evaluate[r[t]], {t, 0, 2 Pi}]
```

Slinky

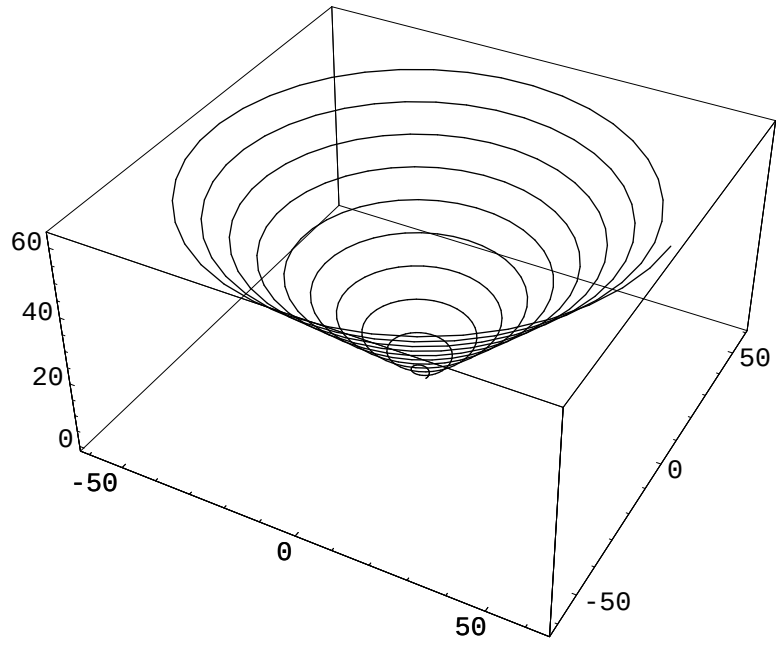
The next task is to draw a vector−valuedfunction describing a Slinky™, everyone's favorite spring toy. (We will add the option **PlotPoints->400** to the command to make *Mathematica* compute the value of the function at more points than usual, giving us better resolution.)

In[75]:=
```
r[t_] := {Cos[t] (2 + Cos[40 t]), Sin[40 t], Sin[t] (2 + Cos[40 t])}
```

In[76]:=
```
ParametricPlot3D[Evaluate[r[t]], {t, 0, Pi}, PlotPoints -> 400]
```

Spiralling Cone

Try drawing a spiral winding 10 times around a cone, as shown below. (Hint: Modify the **r[t]** used in the helix.)



In[71]:=
```
r[t_] := {__, __, __}
```

In[72]:=
```
ParametricPlot3D[Evaluate[r[t]], {t, __, __}, PlotPoints -> 400]
```

Exercise

Try changing the components of **r[t]** to get a feel for various curves. Take your favorite functions and draw at least 4 different curves. Print out your favorate curve from this

exercice. Turning in an original curve already gives **full** credit for this assignment. The prettiest

curves will be posted on the website.

Rotato Peeler™

If there is additional time, here is a task designed especially for fans of late−nightinfomercials. Rotato Peeler™ is an amazing gadget that can peel an orange, apple, potato, or Kiwi in one piece. It has a blade that sits on the skin of the fruit that slowly moves from the top to the bottom while the fruit turns (courtesy of a hand crank operated by various actors who probably have better things to do than peel fruit). Assuming the fruit is a perfect sphere, we will draw the spiral path of the blade on the fruit's surface, wrapping around 10 times, using a parametric curve.

In[73]:=
```
r[t_] := {Sin[t] Cos[20 t], Sin[t] Sin[20 t], Cos[t]}
```

In[74]:=
```
ParametricPlot3D[Evaluate[r[t]], {t, 0, Pi}, PlotPoints -> 400]
```

Note: This curve involves two separate circular motions. The first is the horizontal motion in the plane spanned by **i** and **j**. This is a

Note: This curve involves two separate circular motions. The first is the horizontal motion in the plane spanned by **i** and **j**. This is a circle that wraps around ten times. The second is the vertical motion involving the radius in the **i j** plane and the vertical height along the **k** axis. This is just half a circle going from the north pole to the south pole. Combining the two motions, we get the product of two $\sin t/\cos t$ with different periods for the horizontal and vertical circles.

Harvard University, Mathematics 21a Fall 2000, modified from a notebook by G. Landweber /P. Hsu (1998/1999)