MATH1110H-B-F01-lab-2023-10-17

November 21, 2023

[1]:

```
# MATH 1110H-B F02 lab 2023-10-17
#
# Our objective is to learn to solve (simple!) differential equations
# using SageMath.  We did the basics two weeks ago,  but here is a bit
# more.
#
# Recall that we need to be able to declare a generic function
# function to be able to take derivatives of functions,  generic or
# otherwise,  and solve equations including such derivatives.
#
# We first declare y to be a generic function of x:
#
y = function('y')(x)  # y = function('y',x) should work,  too.
#
# diff(y,x) represents the derivative of y with respect to x,  and the
# desolve command is optimized to solve equations involving derivatives.
# One can also apply it to higher-order derivatives;  for example, the
# second derivative can be denoted by diff())y,x,2).
#
# Note that as with the basic solve command,  desolve needs to be told
# explicitly what to solve for.
#
desolve(diff(y,x,2) == -y, y)
#
# Note that the answer has two unknown constants _K1 and _K2 .
```

[1]:

```
_K2*cos(x) + _K1*sin(x)
```

[2]:

```
# In most applications of differential equations we are also given
# requirements along the lines of "when x= we should have y= ".  These
# are called "initial conditions" and can be specified in desolve by
# adding ics[<x-value>,<y-value>].  For example,  if we were to specify
# that when x=0 we should have y=7 and dy/dx = 6 in the example above,
# we would type in:
#
desolve(diff(y,x,2) == -y, y, ics=[0,7,6])
#
```

```
# This basically pins down the formerly unknown constants:
```

```
[2]: 7*cos(x) + 6*sin(x)
```

```
[3]: # The other tricks that could have gone here were covered two weeks ago,
     # and again the week before,  so check them out,  too.
```