Lecture 14: From integrals to derivatives

Recap

Hello everyone! So, second week of lectures.

I think the previous lecture is one of the most fascinating, and useful, lectures of the *entire course*.

So to try my best to communicate that, I'm just going to reiterate what we did last week:

- We wanted to calculate integrals. We did this with linear and quadratic approximations.
- *Experimentally*, we observed 2nd and 4th order convergence, *usually*, except
  periodic functions had *exponential* convergence
  other functions had *worse* convergence
- Using **integration by parts**, we derived the **Euler-Maclaurin** formula (and Bernoulli polynomials)
- The Euler-Maclaurin formula relates integrals to values of the function and derivatives on the boundary.
  This explains the convergences: *More regular functions converge faster.*
- But the convergence depends on the *grid spacing* - which gets *huge* for higher dimensions.
  This is the **Curse of dimensionality**: *Higher dimension functions converge slower*.
- *Amazingly, random guesses avoid this curse!*
- The Law of Large Numbers tells us random sampling (**Monte Carlo methods**) converges at $N^{-1/2}$ - *no matter the dimension!*

The fact we can get around this curse is absolutely crucial to fields throughout science, mathematics, finance, medicine, etc. Remember this!

Now on to the lecture:

An integral integral

Suppose we want to compute the following integral

$$T = \frac{1}{\sqrt{2}} \int_a^x \frac{dy}{\sqrt{E + \cos y}}$$

In this case, E and a are a parameters that vary over some appropriate ranges. This integral has a long and distinguished history. It's called an **Elliptic Integral of the First Kind (https://en.wikipedia.org/wiki/Elliptic_integral)**. There is a lot to say about it. There are a lot of good reasons for wanting to solve this and similar integrals. For example, the (deceptively) simple question of "what's the perimeter of an ellipse"? It's even related (through Elliptic functions (https://en.wikipedia.org/wiki/Elliptic_function), Elliptic curves (https://en.wikipedia.org/wiki/Elliptic_curve), and Modular forms (https://en.wikipedia.org/wiki/Modular_form)) to the proof of Fermat's Last Theorem (https://en.wikipedia.org/wiki/Fermat%27s_Last_Theorem).

But for right now, let's just say that you need to compute it in some way. You could start with one of the methods we discussed; or some more advanced version of what we discussed. But what if there's a different way of thinking about it?

Rather than thinking about $T = T(x)$, what if we were able to solve $x = x(T)$? If we could do either of these, we could reverse the axes on a plot and have the other.

An inverse problem

For example, what if we want to plot

$$y = \sqrt{x}$$

But we don't "know" how to take square roots. Simple, I know. But it illustrates the point. Rather than plotting the square root, we could plot

$$x = y^2$$

and reverse the axes on the plot. Let's see what I mean:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

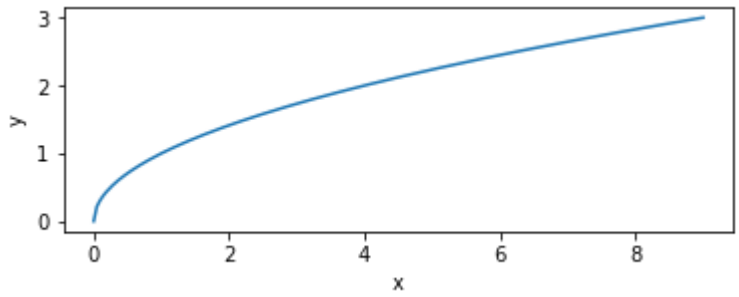
In [2]:

```
def square(x):
    return x*x

def root_2(x):
    return np.sqrt(x)

x = np.linspace(0,9,200)
y = np.linspace(0,3,200)
```

Here's what we would like to see:

In [3]:

```
fig, ax = plt.subplots()
ax.plot(x,root_2(x))
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_aspect(1)
```

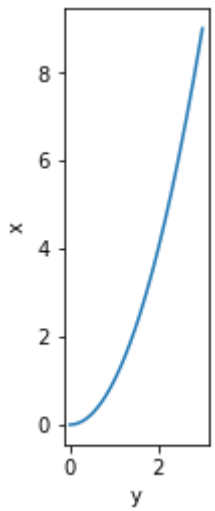


But we needed to make a `root_2` to get it.

Here's another way

In [4]:

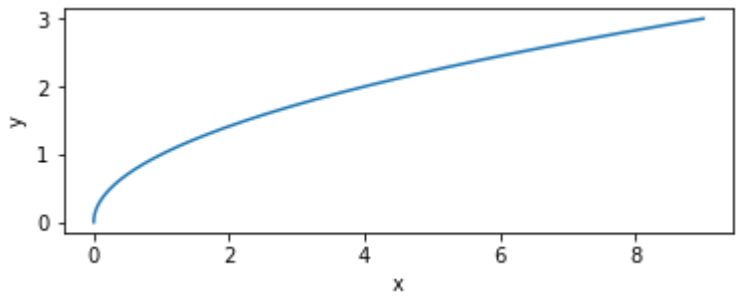
```
fig, ax = plt.subplots()
ax.plot(y,square(y))
ax.set_xlabel('y')
ax.set_ylabel('x')
ax.set_aspect(1)
```



This only required `square`. But the plots don't look right yet. We need to reverse the axes.

In [5]:

```
fig, ax = plt.subplots()
ax.plot(square(y),y)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_aspect(1)
```



Inverting an integral

Back to the integral problem...

If we want $T(x)$, the $x(T)$ will be a good second option. In fact, for this problem the thing we really want *is* $x(T)$. The integral is an elaborate way people constructed to get it. Maybe there's an easy way to get just what we really want. How?

We think of the integral as an *implicit* equation for $x(t)$. That is,

$$t \;=\; \frac{1}{\sqrt{2}} \int_a^{x(t)} \frac{dy}{\sqrt{E+\cos y}}$$

We can differentiate this with respect to t. I've changed $T \to t$ because it's going to be in independent variable now, and it's less clunky to carry around a t.

$$\frac{dt}{dt} \;=\; 1 \;=\; \frac{1}{\sqrt{2}\sqrt{E+\cos x(t)}} \frac{dx(t)}{dt} \;=\; \frac{x'(t)}{\sqrt{2}\sqrt{E+\cos x(t)}}$$

We can rewrite this slightly

$$x'(t) \;=\; \sqrt{2}\sqrt{E+\cos x(t)}$$

This is now an *ordinary differential equation* (ODE). Let's rearrange again:

$$\frac{x'(t)^2}{2} - \cos x(t) \;=\; E$$

This says that everything on the left-hand side is equal to E, which is a constant independent of t. So let's get rid of it by differentiating

$$\frac{d}{dt}\left[\frac{x'(t)^2}{2} - \cos x(t)\right] \;=\; \big(x''(t) + \sin x(t)\big)\, x'(t) \;=\; 0$$

And now (***spoiler alert***), we find our Elliptic integral is a reformulation of the equation for a simple pendulum:

$$x''(t) + \sin x(t) \;=\; 0$$

The differential form of the pendulum is a *second-order* differential equation. This means it requires two initial conditions. But we have two parameters. E and a.

Making $t = 0$ in the integral implies

$$x(0) = a$$

We also have an equation for $x'(t)$. At $t = 0$,

$$x'(0) \;=\; \sqrt{2}\sqrt{E + \cos a}$$

Integrals to Differential equations

We have the following general principle:

```
Integrals and differential equations are inverses of each other.
```

$$x''(t) + \sin x(t) \;=\; 0 \qquad \Longleftrightarrow \qquad t \;=\; \frac{1}{\sqrt{2}} \int_a^{x(t)} \frac{dy}{\sqrt{E + \cos y}}$$

(Obligatory The Simpsons derivative joke (https://www.youtube.com/watch?v=gSFd_2oJgak). It gets meta)

In this particular case, it's possible to start with the differential equation and get the integral. In many situations this is not possible. It much more common to be able to start with an integral and get a differential equation. In this sense, it's easier to model behaviour in the real world with differential equations. It's very useful to be able to solve differential equations without resorting to integration.

First order ODEs

The go-to way to solve ODEs is in **first-order form**. For the pendulum equation, this means defining

$$y(t) = x'(t)$$

Then as a first-order system of equations

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ -\sin x(t) \end{bmatrix}; \quad \begin{bmatrix} x(0) \\ y(0) \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

The "energy" is

$$E = \frac{b^2}{2} - \cos a$$

The fact that this doesn't change is a special property of our system that is not true in general. In fact, it is partially what allowed us to write the solution in terms of integrals. But we don't need this information to solve the 2nd-order system of ODEs.

After formulating the pendulum integral as a 2nd-order ODE, we can also abstract the whole idea.

We could assume the

$$x,\, f(x) \in \mathbb{R}^n, \quad \text{and} \quad x'(t) = f(x(t)), \quad \text{with} \quad x(0) = x_0$$

This is in n-dimensional continuous dynamical system. But it's straightforward to generalise this to a lot of other situations. To be able to solve ODEs we first need to understand how to compute derivatives numerically.

Break time!

This quiz has two questions:

- **How do you convert an integral equation to a differential equation?**
- **How do you convert an nth order ODE to a first order ODE?**

So get out the pen and paper, and start doing some calculus!

Just ask if you're stuck.

Differentation

For the trapezoidal rule,

$$\int_{x-h}^{x+h} F(y)\,\mathrm{d}y \;\approx\; h\,(\,F(x+h)\;+\;F(x-h)\,)\;+\;\mathcal{O}(h^2)$$

What if we assume that our integrated function is a derivative of another function,

$$F(x) \;=\; f'(x)$$

In this case we can integrate the left-hand side exactly, and rewrite the formula

$$\frac{f(x+h)-f(x-h)}{h} \;\approx\; (\,f'(x+h)\;+\;f'(x-h)\,)\;+\;\mathcal{O}(h^2)$$

This shifts x around a bit to make the expression look more symmetric. The formula is not very useful at this point. The left-hand side is very close the definition of a derivative. But the right-hand side combines derivative at different points. But the implication is the following

```
Just as we could approximate integrals as finite sums of function evaluati
ons,
we can approximate derivatives as finite differences of function evaluatio
n.
```

But we can model a guess after this expression. What about

$$f'(x) \;\approx\; \frac{f(x+h)-f(x-h)}{2h}$$

How good is this at approximating derivatives?

Taylor's Theorem

Taylor's theorem is about the best tool for relating derivatives of a function to its values at different points. It's a more familiar cousin of the Euler-Maclaurin formula.

$$f(x+h) \;=\; f(x)\;+\;hf'(x)\;+\;\frac{h^2}{2}f''(x)\;+\;\frac{h^3}{3!}f'''(x)\;+\;\ldots$$

If we flip $h \to -h$ and subtract (after a bit of rearranging),

$$f'(x) \;\approx\; \frac{f(x+h)-f(x-h)}{2h}\;-\;\frac{h^2}{6}f'''(x)\;+\;\mathcal{O}(h^4)$$

This is our first **finite-difference formula**; symmetric centered 2nd order. This is a workhorse very much in the spirit of the trapezoidal rule. Use it as your first option whenever you can.

Let's see this in action

In [6]:

```
def D(f,x,h=None):
    if h == None: h = x[1]-x[0]
    return (f(x+h)-f(x-h))/(2*h)
```

In [7]:

```
def grid(n): return np.linspace(0,2*np.pi,n)

def s(x): return np.sin(x)
def c(x): return np.cos(x)
```

In [8]:

```
x = grid(101)
y  = s(x)
Dy = D(s,x)
```

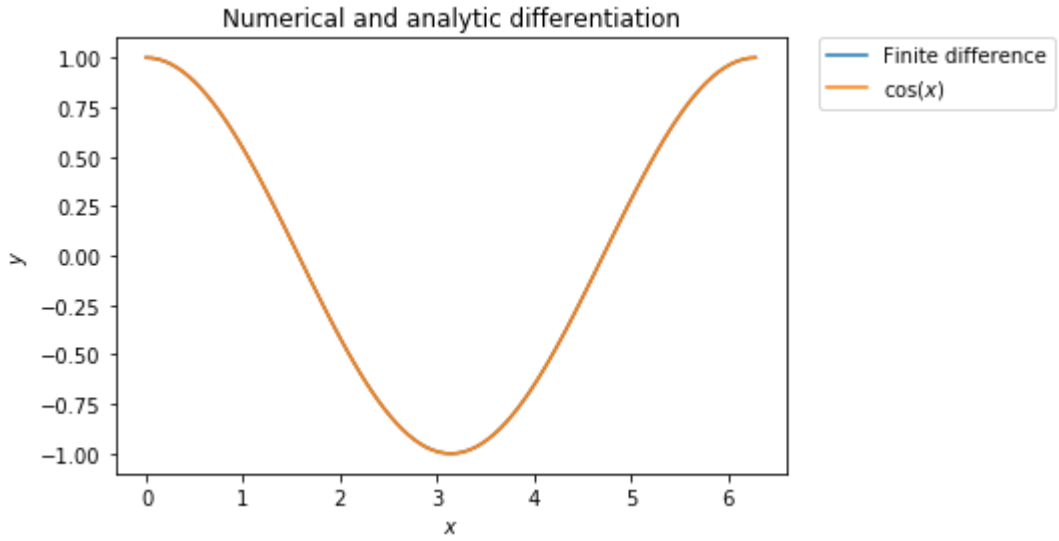
Let's plot this difference:

In [9]:

```
fig, ax = plt.subplots()
ax.plot(x,Dy,label='Finite difference')
ax.plot(x,c(x), label='$\cos(x)$')
ax.set(xlabel='$x$',ylabel='$y$',title='Numerical and analytic differentiation')
ax.legend(bbox_to_anchor=(1.05,1.),loc="upper left",borderaxespad=0.)
```

Out[9]:

```
<matplotlib.legend.Legend at 0x7f631b7152e8>
```



Ok, well what does the maximum error look like as a function of n?

In [10]:

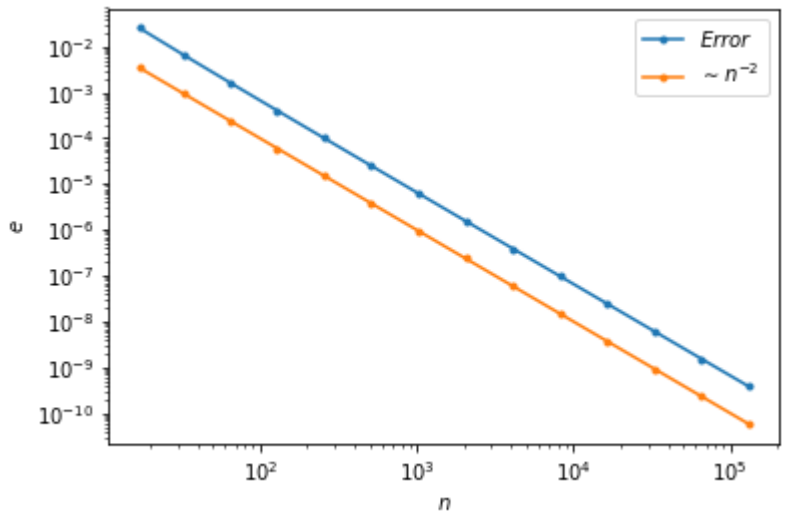
```
n = []
e = []
for k in (2**i+1 for i in range(4, int(np.log2(5e5)))):
    x = grid(k)
    e += [np.max(np.abs(c(x)-D(s,x)))]
    n += [k]

n, e = np.array(n), np.array(e)
```

In [11]:

```
fig, ax = plt.subplots()
ax.loglog(n,e,'.-')
ax.loglog(n,n**(-2.),'.-')
ax.legend(['$Error$', '$\sim n^{-2}$'])
ax.set_xlabel("$n$")
ax.set_ylabel("$e$")
```

Out[11]:

```
Text(0,0.5,'$e$')
```



Higher derivatives

We could compute a 2nd derivative by applying the first-order formula two times. This would produce

$$f''(x) \approx \frac{f(x+2h) - 2f(x) + f(x-2h)}{4h^2}$$

This has the disadvantage of sampling the function at spacing of $2h$. Higher-order derivatives would sample the function at wider and wider spacing. As we take higher derivatives, it's not possible to avoid sampling more points. But it can be much less that simple estimates would imply.

We can use Taylor's theorem to find simpler formulae for higher derivatives. For example

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

$$f'''(x) = \frac{f(x+2h) - 2f(h+x) + 2f(x-h) - f(x-2h)}{2h^3} - \frac{h^2}{4} f^{(5)}(x) + \ldots$$

Higher order

We can also use more sampling points to obtain higher accuracy (better error)

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + \frac{h^4}{30} f^{(5)}(x) + \ldots$$

We can do this *recursively* for different spacings.

In [12]:

```
def D(f,x,order=2,h=None):
    if h == None: h = x[1]-x[0]
    if order == 2: return (f(x+h)-f(x-h))/(2*h)
    elif order == 4: return (4/3)*D(f,x,h=h) - (1/3)*D(f,x,h=2*h)
```

It has some of the feel of Simpsons rule compared to the trapezoidal rule.

In [13]:

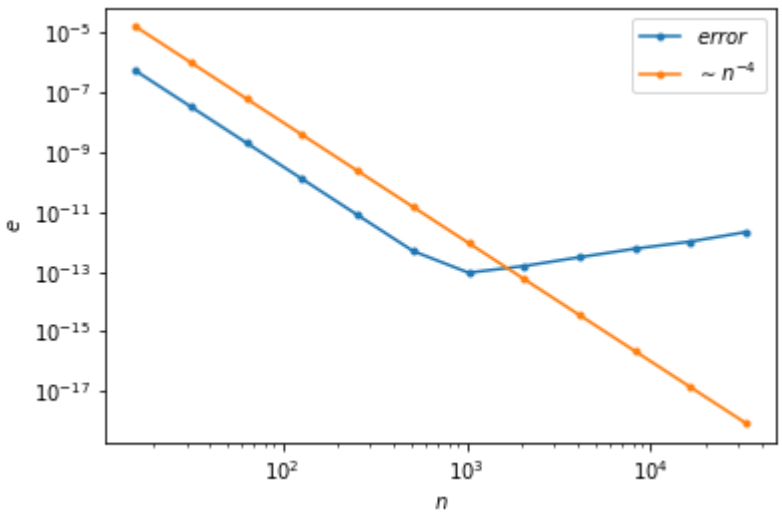
```
n = []
e = []
for k in (2**i for i in range(4,int(np.log2(1e5)))):
    x = grid(101)
    e += [np.max(np.abs(c(x)-D(s,x,order=4,h=1/k)))]
    n += [k]

n, e = np.array(n), np.array(e)
```

In [14]:

```
fig, ax = plt.subplots()
ax.loglog(n,e,'.-')
ax.loglog(n,n**(-4.),'.-')
ax.legend(['$error$', '$\sim n^{-4}$'])
ax.set_xlabel("$n$")
ax.set_ylabel("$e$")
```

Out[14]:

```
Text(0,0.5,'$e$')
```



We can also compute the 2nd derivative at the same accuracy with the same points away from x.

$$f''(x) = \frac{-f(x+2h) + 16f(x+h) + 30f(x) + 16f(x-h) - f(x-2h)}{12\,h^2} + \frac{h^4}{90} f^{(6)}(x) + \ldots$$

Next time, we'll see more about how to derive these formulae systematically. This will allow us to produce results that we can use on ODEs.