January 16th

Vectorization, Visualization and Numerical Differentiation/Integration

Tutorial info & Contact info

Hugh Podmore
podmore@yorku.ca

Website
nanosatellite.lab.yorku.ca/teaching/phys2030

Tutorials will consist of

- Some lecturing / review of course material.
- A series of small worked examples and one "big" one based on recent course material.
- A Q&A / help period.

- Tutorials are 3 hours.

Starting problem – Matrix Multiplication

- Compute the product of two matrices, C = AB where

- $A = \begin{pmatrix} 3 & 1 & 4 \\ 0 & 2 & 1 \\ 5 & 1 & 0 \end{pmatrix}$
- B$= \begin{pmatrix} 1 & 0 & 3 \\ 0 & 0 & 3 \\ 0 & 2 & 4 \end{pmatrix}$

Vectorization

- Matlab is built around PROCESSING VECTORS
- Consider these two programs:

```
function [results] =
two_powers_iterate(N)
    tic

    results = zeros(1, N);
    for i = 1:N
        results(i) = 2^i;
    end

    toc
end
```

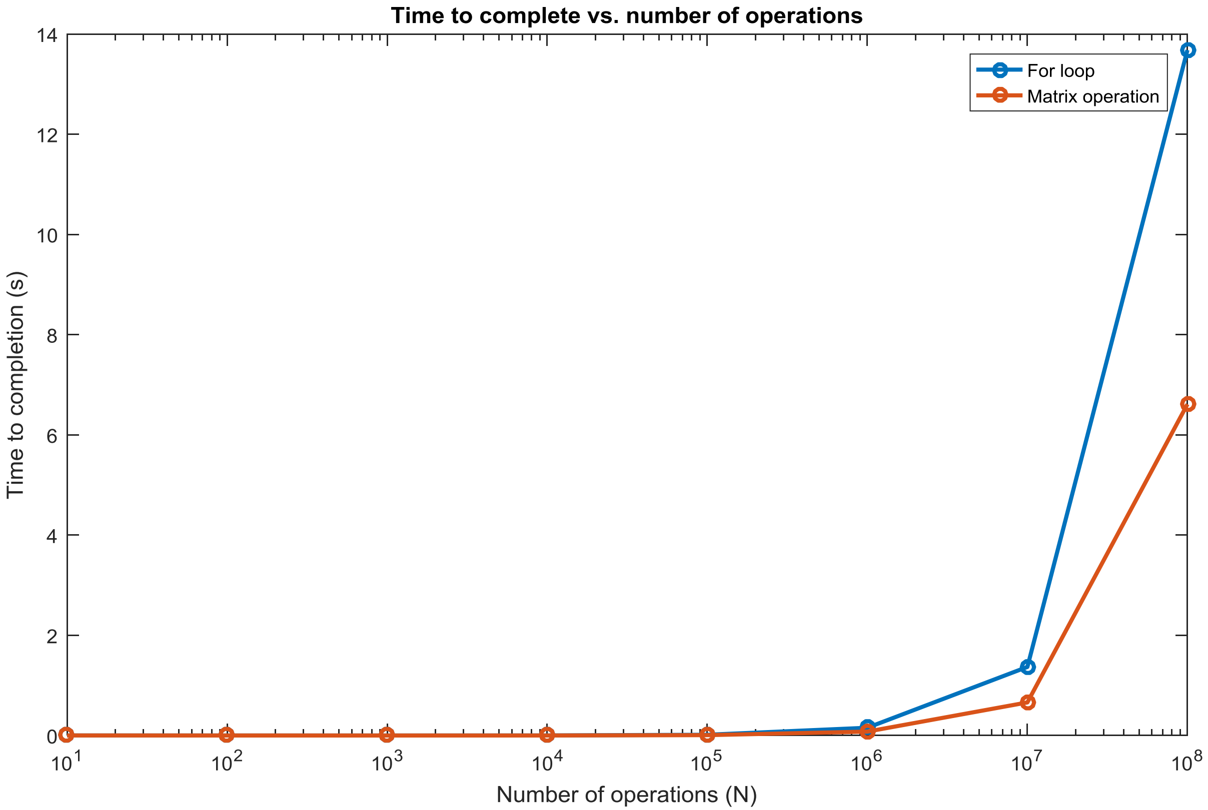
```
function [results] =
two_powers_vectorized(N)
    tic

    powers = 1:1:N;
    base = zeros(size(powers));
    base(1:end) = 2;

    results = base.^powers;

    toc
end
```

- They do the same thing (which is…?). Question: what's the point?

Vectorization

- For very large operations vectorization can be more efficient
  (think N = 10^9 *doubles* * 4 bytes / double = 4 Gb of data)

- Worth thinking about when processing large quantities of data.
- A "full" image taken by JWST would be about 250Mb.
  -Something like SKA or ALMA…?



Special Vector Syntax

- Suppose we have two vectors x_vec, y_vec

`*  :`   `x_vec*y_vec`   → vector/matrix multiplication (watch dimensions!)

`.*:`   `x_vec.*y_vec`   → per element multiplication (ie: [x1*y1, x2*y2…])

`‘  :`   `x_vec’`   → x_vec^T ie: transpose

Many built-in Matlab functions support vector input. Ex: cos() & sin().

Vectorization Exercise

- Exercise: Write a VECTORIZED function (no loops of any kind!) that does the following:

Take a vector of numbers (in degrees),

1. convert them to radians
2. compute their sine values.
3. get the absolute values and
4. sum all these numbers together and return the result.

Challenge: do it without using the Matlab functions abs() and sum()

Function handles

- We can quickly define generic functions using function handles “@(x)”

```
xpts = -5:0.1:5;
F1 = @(x) 5 .* exp(-x.^2);
ypts = F1(x);
```

- And Plot them using the fplot function

```
plot(xpts,ypts,’.’);
hold on
fplot(F1,[-5 5],’r’);
```

Important: This is just a fast way to define functions! Matlab treats these the same as functions made using the `function` command.

- Using a function handle, make a function which displays “hello world” regardless of the input

Visualization

- There are a number of built-in functions for visualization in Matlab
- Use the help function to see how each of the following is useful
  - plot
  - fplot
  - hist
  - rose
  - surf
  - contour
  - quiver

- The `peaks` function generates a sample 2D scalar function

- Try plotting `peaks(50)` with the surf and contour functions

<u>Exercise</u>: Generate two 1x1000 vectors using `rand(1,1000)` and `randn(1,1000)`.

- Using one of the visualization functions from the previous slide, show the difference between the data generated by each.
- Think of cases where `rand` and `randn` would be useful

Numerical Differentiation

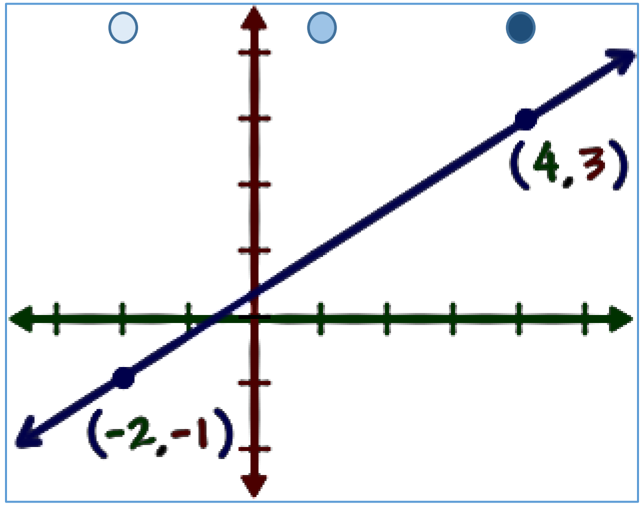
- Question: can we take the derivative of a set of discrete data (without knowing the function beforehand)?

- First guess, simple “rise over run”

- $\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

- This approximation is more accurate for closer points.
- How do we know that?
- Compare to the definition of a derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



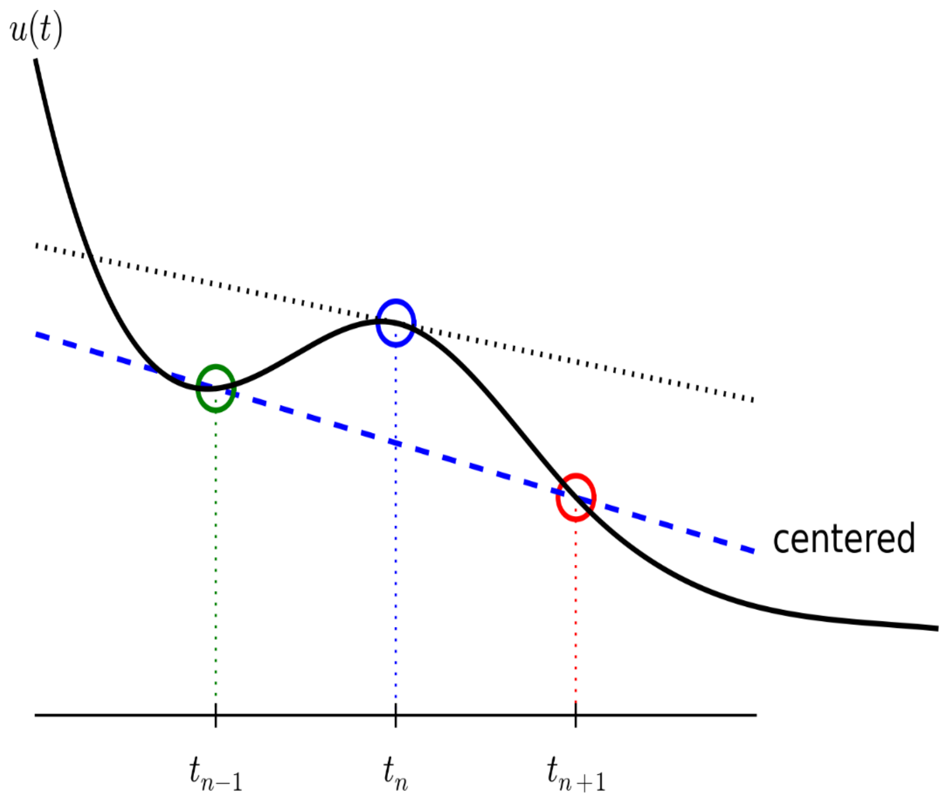
- We need to choose which x-values we associate with the derivative; the left value, right value or midpoint?

Exercise: Generate a set of points using the `sin()` function.

Use the `diff` function to calculate the derivative. Compare your results to the expected analytical value (which is?...)

Centered differences

- We can improve our estimate by considering points to the left and right of the point in question
- Why would this improve our estimate?



Taylor expansion

$$f(t \pm \Delta t) = f(t) \pm \Delta t \frac{df(t)}{dt} + \frac{\Delta t^2}{2!}\frac{d^2 f(t)}{\mathrm{dt}^2} \pm \frac{\Delta t^3}{3!}\frac{d^3 f\left(c_{1,2}\right)}{\mathrm{dt}^3}$$

- If we “truncate” the series after two terms we can rearrange to get the “rise over run” approximation, the remaining terms represent the error in our approximation.

- Now if we taylor expand the difference between the “+” and “-” versions…

$$f(t + \Delta t) - f(t - \Delta t) = 2\Delta t \frac{df(t)}{dt} + \frac{\Delta t^3}{3!}\left(\frac{d^3 f(c_1)}{\mathrm{dt}^3} + \frac{d^3 f(c_1)}{\mathrm{dt}^3}\right)$$

Two points to take away from the Taylor expansion

1. It shows that the error in the centered difference equations is reduced/can give us an idea of what that error is
2. It introduces us to the idea that we can improve our approximations by interpolating power series expansions

Exercise: write a function which takes the derivative of a generic set of x and y values using central differences. Test your function with values from the `sin()` function

For an extra challenge, try to do this without any loops

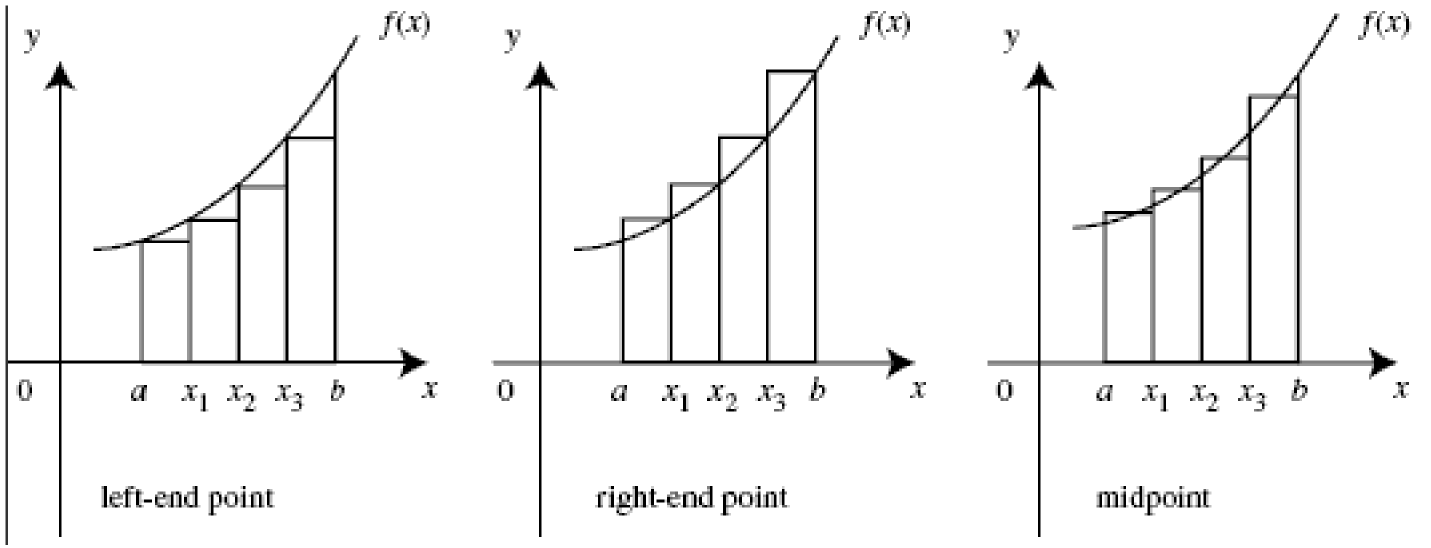
How can we expand differentiation to a function with x,y and z values? What is this operation called?

Numerical integration

- How can we determine the integral of a discrete set of data (even when we don’t have a function to describe it)?

- An integral represents the area under the curve

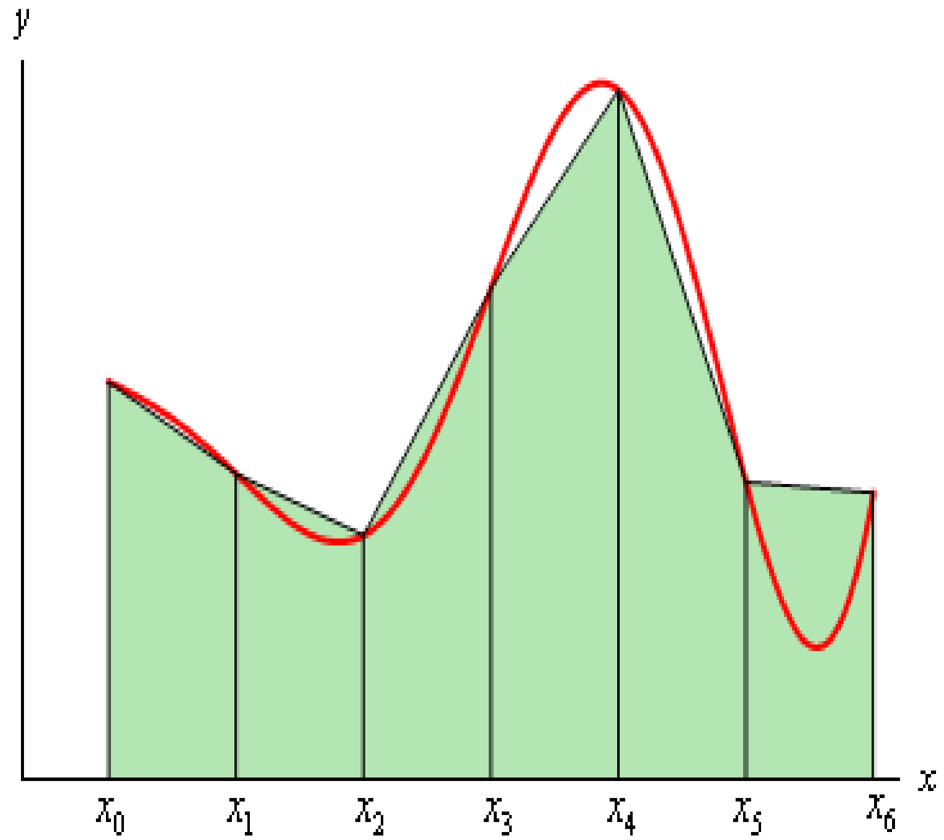
- Riemann sums approximate this area with rectangles (Area = height * width), we can define the height of each rectangle by matching the function value with the left side, right side or center of a rectangle.



Improving the approximation

We can divide the area up into trapezoids

- This amounts to approximating the curve with straight line segments
- What is the area of a trapezoid?



Exercise: Write a function which computes the integral of a function using Riemann sums (left, right). Use this function to compute

$$\int_{-10}^{10} \sin^2(x)\, dx$$

Simpson's Rule

- Going one step further, we can approximate the curve using quadratic segments.
- The lecture slides describe Simpson's rule using two different formulas
- $SIMP(n) = \frac{2*MID(n)+TRAP(n)}{3}$ and $\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}(f_0 + 4(f_1) + f_2)$

Applied Exercise

- Final exercise: Using data provided on course website, determine power produced by a solar cell of unit-area $1m^2$ at angle 30° when coated with a double-layer antireflective coating and with no coating.

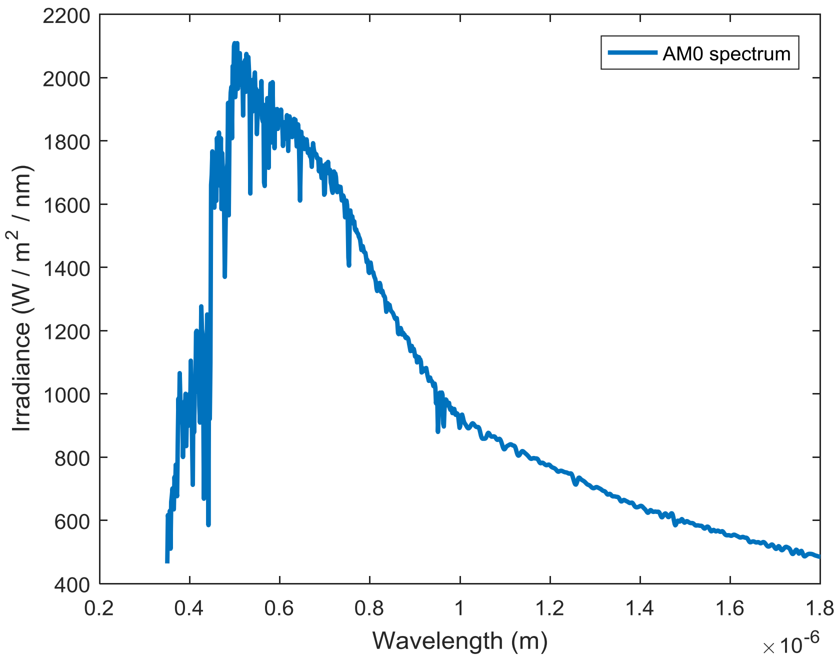
- You may not use any of MATLAB's built-in interpolation functions, derivative functions, or integration functions (*spline(), interp1(), trapz(), gradient()* etc.)

- Recall that the area of a cell will scale with cos(theta)

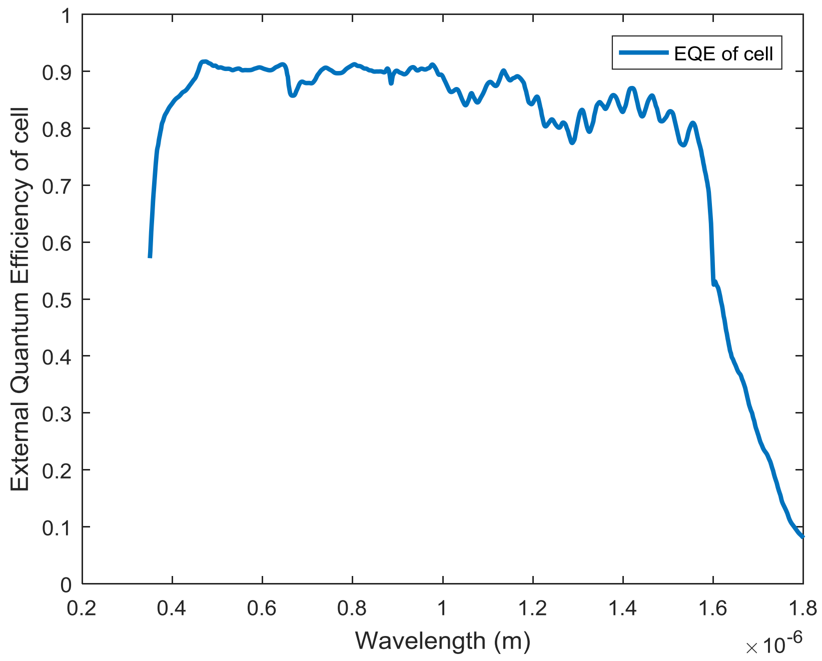
- use the formula:
  Power(lambda) = Irradiance(lambda) * Surface area * External quantum efficiency(lambda) * 27% efficiency * Transmittance (lambda)

Applied Exercise

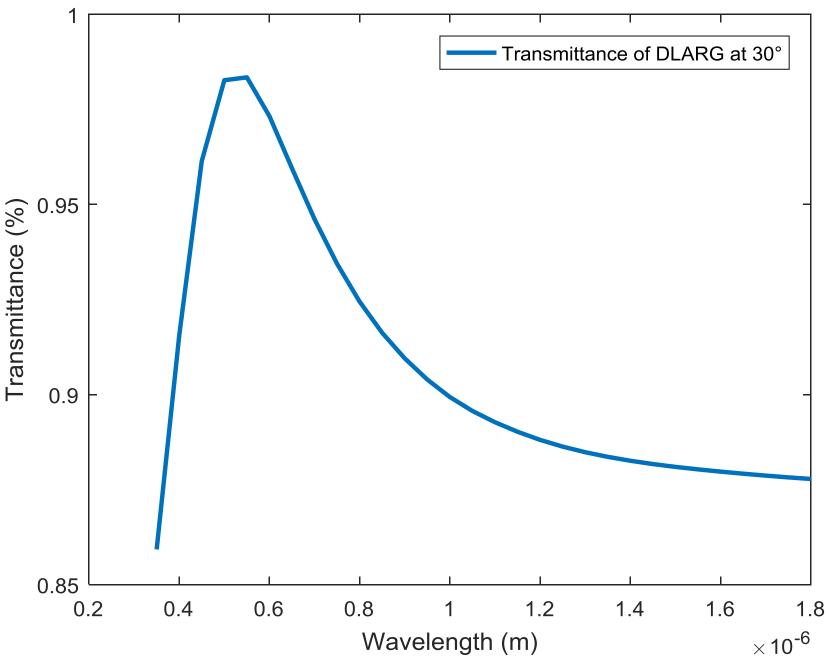
- First problem: our data doesn't match up!



AM0vec = 1 x 1000 double



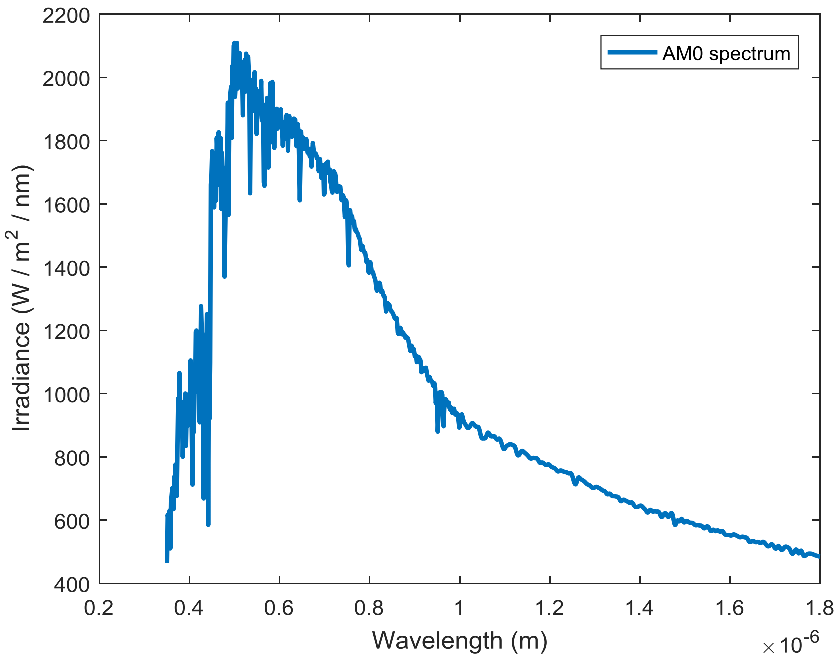
EQE_cell = 1 x 1451 double

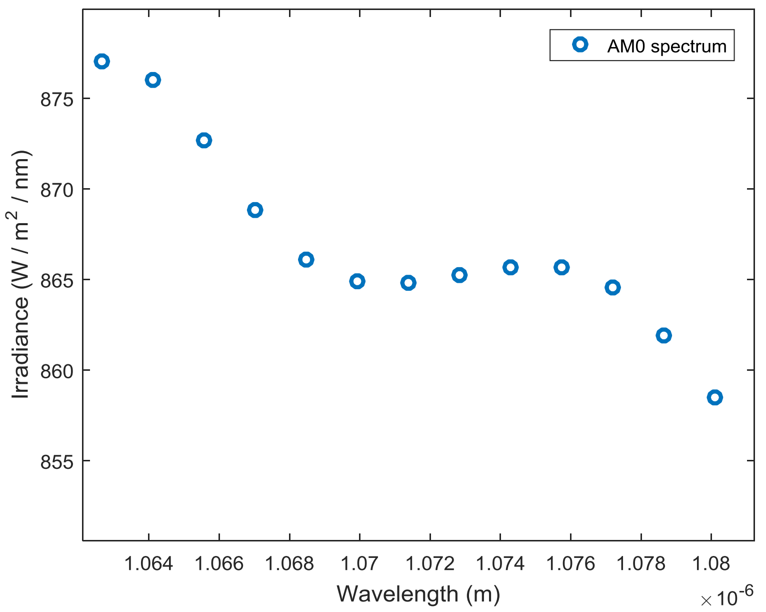


DLARG(: , 7) = 30 x 1 double

Applied Exercise

- Let’s say we want all of our data to have 1 x 1451 points. How can we improve our data to get there?

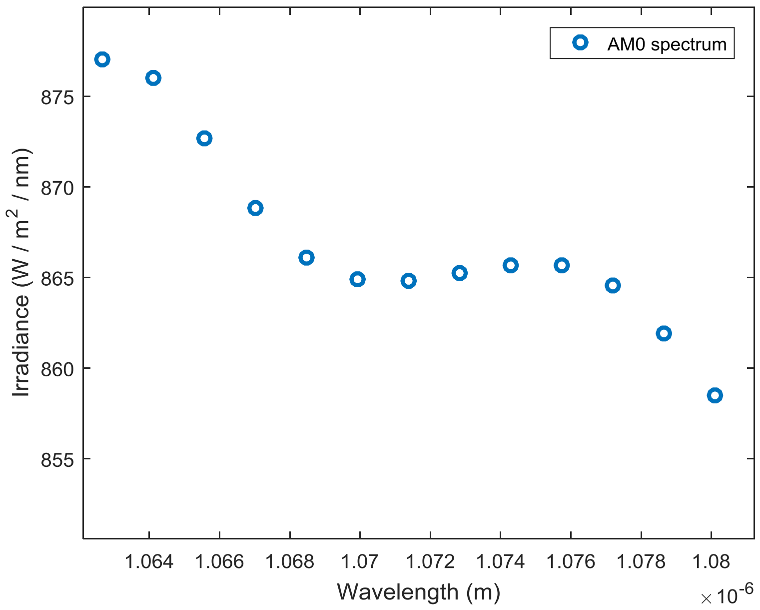




AM0vec = 1 x 1000 double

Applied Exercise

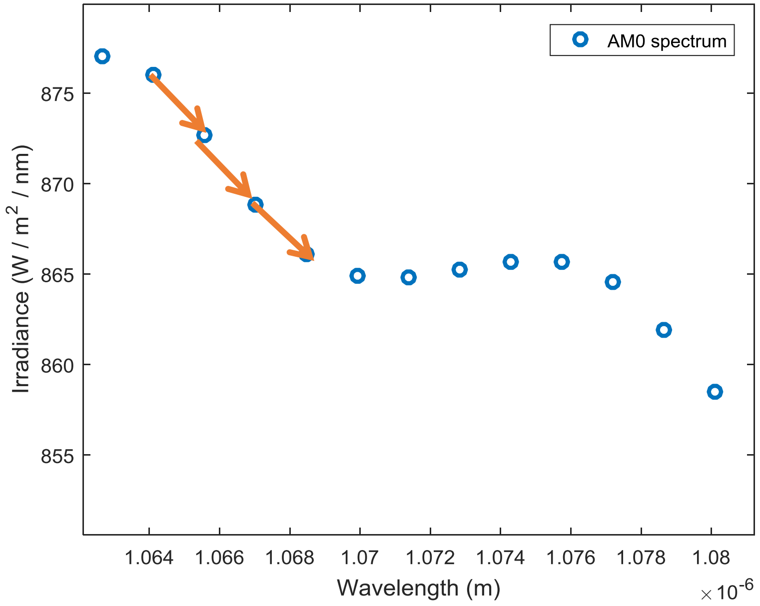
- This is the process of *linear interpolation* (one way, at least)



Applied Exercise

- This is the process of *linear interpolation* (one way, at least)

1) Approximate first derivative of function

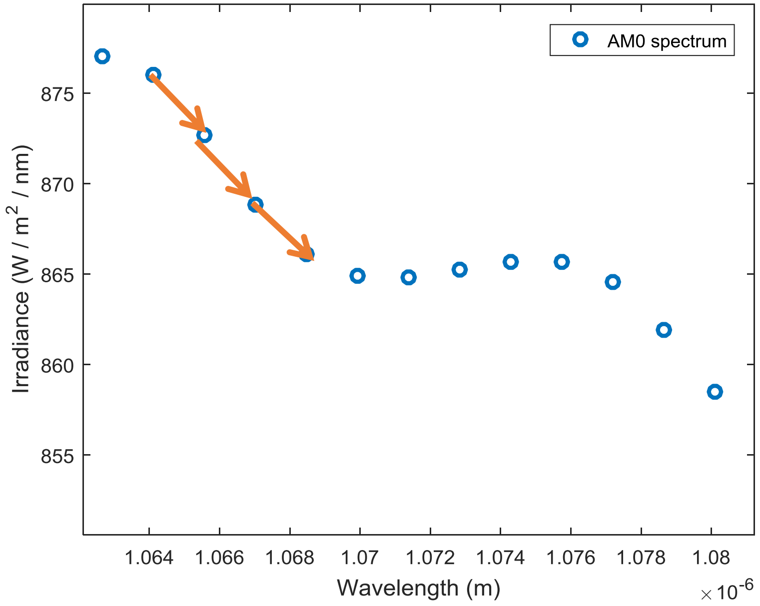


Applied Exercise

- This is the process of *linear interpolation* (one way, at least)

1) Approximate first derivative of function

2) For desired new point $x(n+\delta)$ find nearest neighbour $x(n)$ or $x(n+1)$?



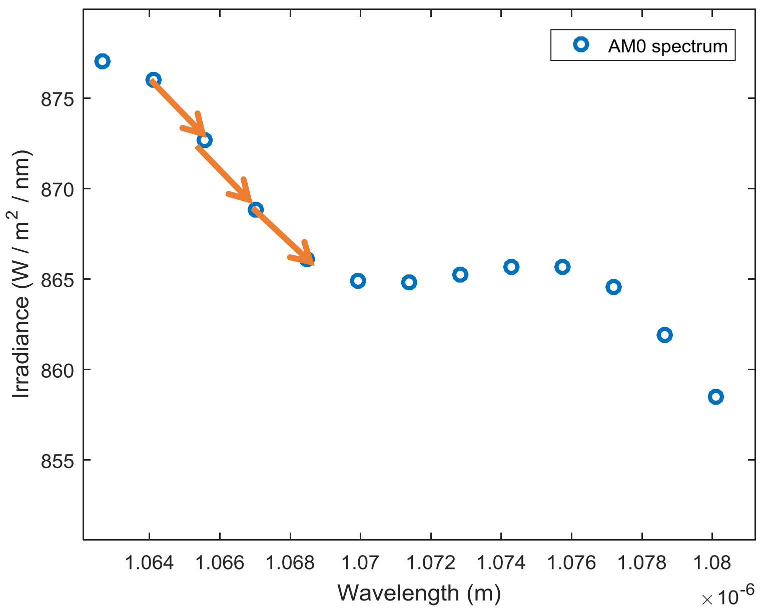
Applied Exercise

- This is the process of *linear interpolation* (one way, at least)

1) Approximate first derivative of function

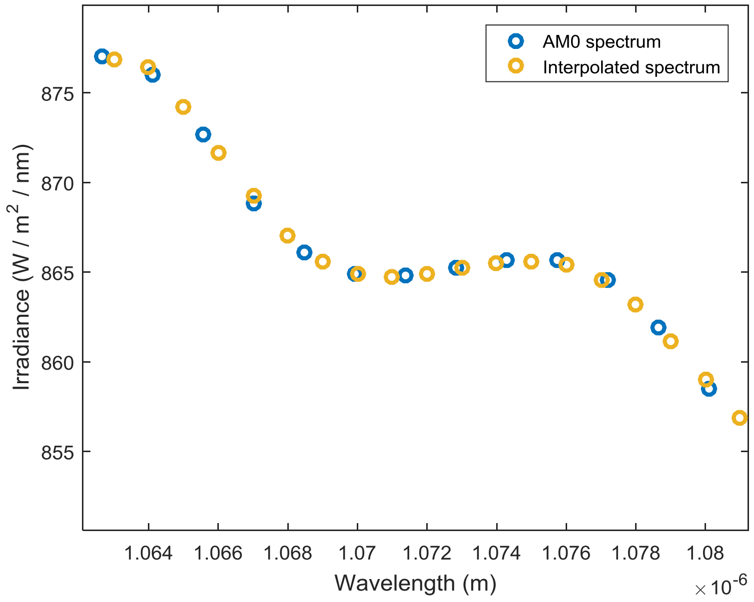
2) For desired new point $x(n+\delta)$ find nearest neighbour $x(n)$ or $x(n+1)$?

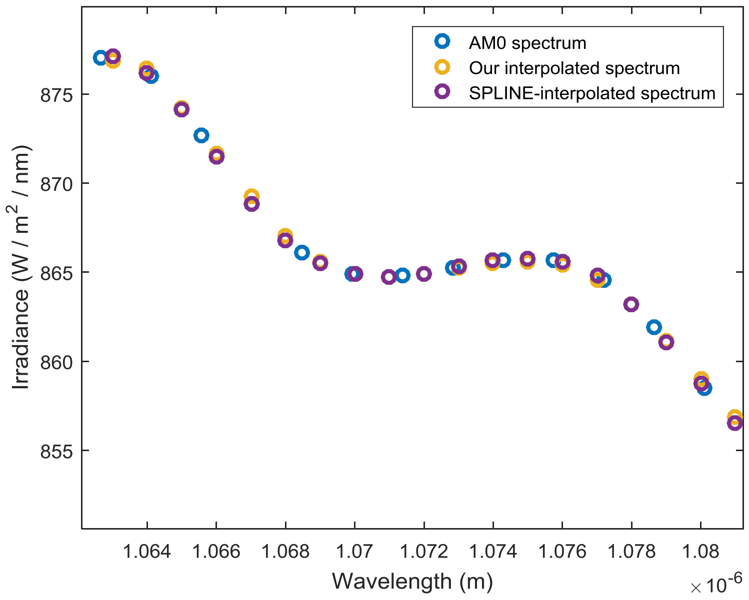
3) Project 1st derivative at $x(n)$ into "dead space" to retrieve $x(n+\delta)$



Applied Exercise

- We can perform this very useful procedure to very good approximation using the derivative methods we just did! (see *centered_diff_interp.m*)

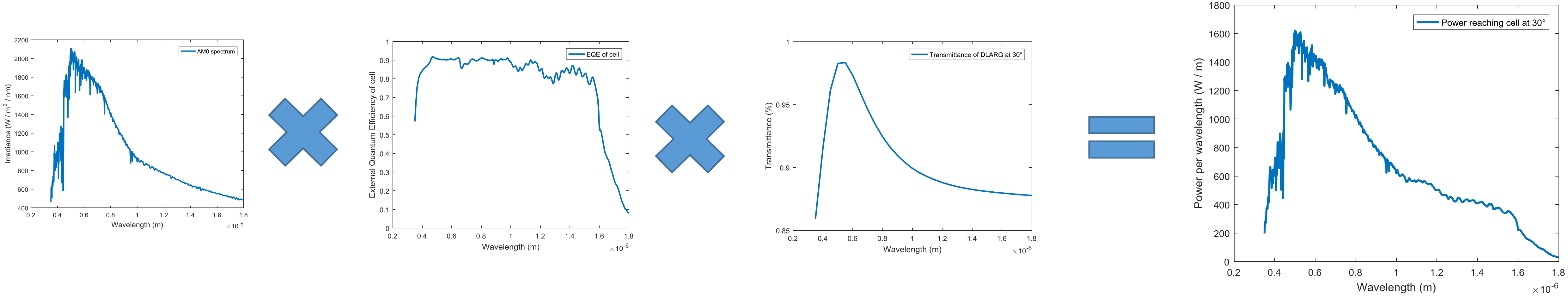




- NRMSE = 0.07% between this method and more sophisticated function!

Applied exercise

- Now our data is fixed, can multiply curves together to get true power per wavelength:



Integrating (recall that Irradiance in W / m^2 / um) gives us ~983W.
Is this reasonable?

- 983W * 0.27 * 2.277cm^2 / (100cm/m ^2) / 1.5 / cosd(30) = 0.046 W

Datasheet says we can expect 0.06W so pretty close!