ME 2016 A- Computing Techniques
Fall 2018

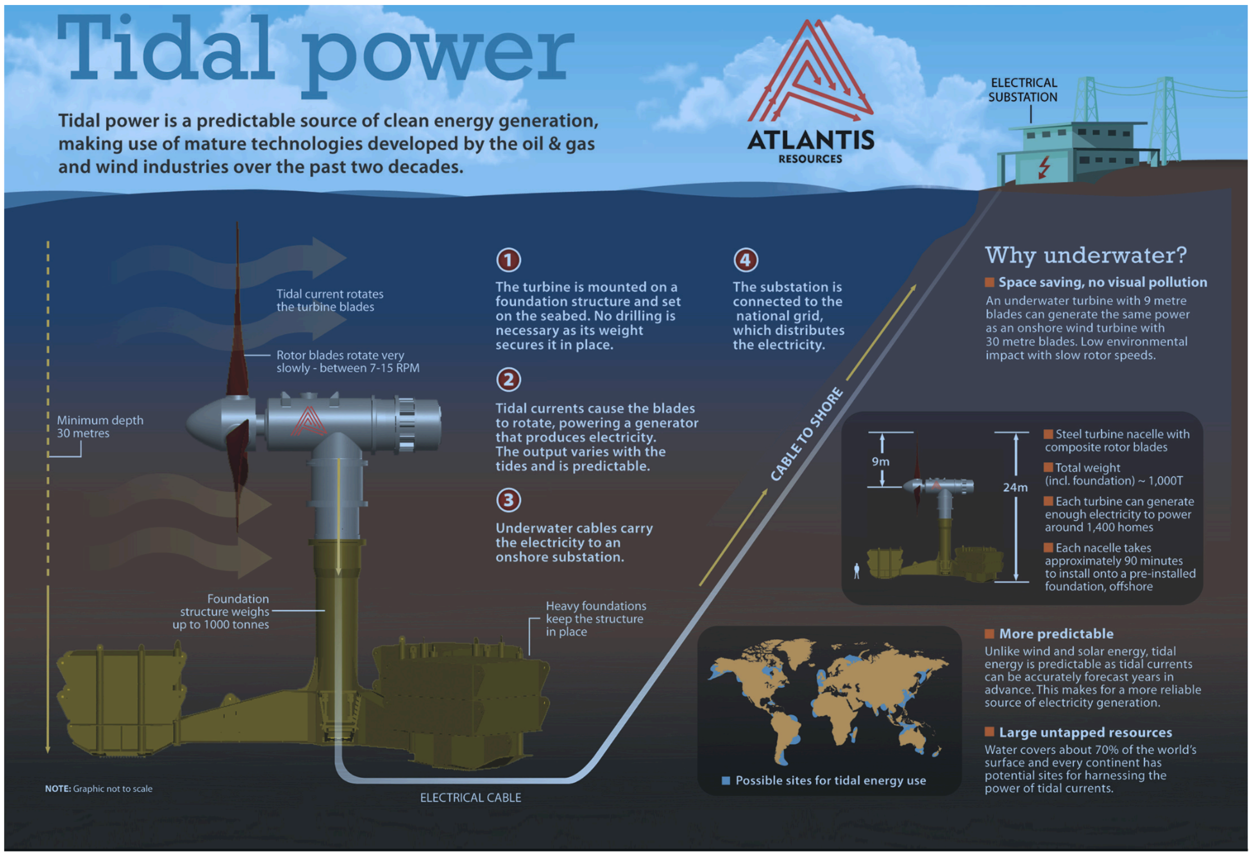
Computer Project 2
Due Thursday, November 1 at 2:55 pm (5 minutes before class time)

Root-Finding and Numerical Integration

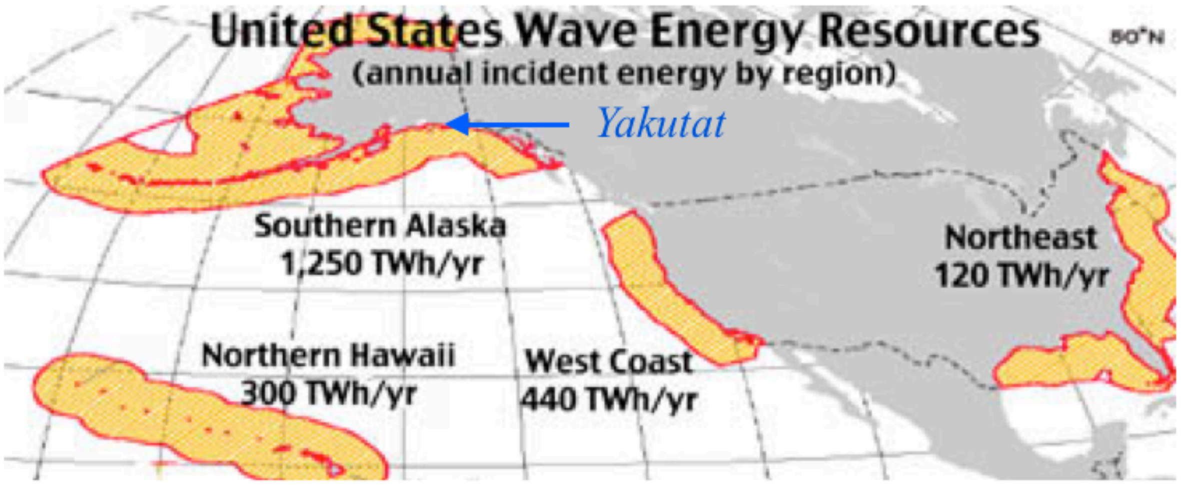
Your work at the Wind Energy Division of Georgia Power has made you well-known in the energy community, and you are now the go-to analyst in renewably energy. The State of Alaska has recently contacted you to look into a tidal system to supplement the state's energy needs.

<u>Background information on tidal energy:</u>

Tides are generated from the gravitational interactions between the earth, the moon, and the sun. As tides ebb and flow near shorelines, they create tidal currents. The energy in these currents can be captured with a tidal turbine, much like wind energy can be harvested with a wind turbine. Since water is considerably denser than air, a significantly larger amount of energy can be extracted from water than air under similar conditions. Tidal power is a predictable and inexhaustible energy source, and the ability to harness its immense potential allows a better power generation forecast than with any other renewable system, making it a reliable power generation technique to supplement other alternative energy sources.


Tidal power
Tidal power is a predictable source of clean energy generation, making use of mature technologies developed by the oil & gas and wind industries over the past two decades.
ATLANTIS RESOURCES
ELECTRICAL SUBSTATION
Tidal current rotates the turbine blades
Rotor blades rotate very slowly - between 7-15 RPM
Minimum depth 30 metres
Foundation structure weighs up to 1000 tonnes
1 The turbine is mounted on a foundation structure and set on the seabed. No drilling is necessary as its weight secures it in place.
2 Tidal currents cause the blades to rotate, powering a generator that produces electricity. The output varies with the tides and is predictable.
3 Underwater cables carry the electricity to an onshore substation.
4 The substation is connected to the national grid, which distributes the electricity.
Heavy foundations keep the structure in place
CABLE TO SHORE
ELECTRICAL CABLE
NOTE: Graphic not to scale
Why underwater?
Space saving, no visual pollution
An underwater turbine with 9 metre blades can generate the same power as an onshore wind turbine with 30 metre blades. Low environmental impact with slow rotor speeds.
9m
24m
Steel turbine nacelle with composite rotor blades
Total weight (incl. foundation) ~ 1,000T
Each turbine can generate enough electricity to power around 1,400 homes
Each nacelle takes approximately 90 minutes to install onto a pre-installed foundation, offshore
Possible sites for tidal energy use
More predictable
Unlike wind and solar energy, tidal energy is predictable as tidal currents can be accurately forecast years in advance. This makes for a more reliable source of electricity generation.
Large untapped resources
Water covers about 70% of the world's surface and every continent has potential sites for harnessing the power of tidal currents.

Power generation in isolated areas of the world is typically a difficult and costly process. Oftentimes, residents who are not connected to a traditional power grid are forced to use local power generation techniques that prove to be expensive and inefficient. For example, the residents of Yakutat, Alaska (population: ~620) use diesel-powered generators for their electricity needs; diesel fuel is shipped from distant locations, yielding a price per kilowatt hour of electricity that is substantially higher than the American average of 15 cents/kWh. The southern Alaskan coast has been tagged by many researchers as having great potential for tidal power generation, and you have determined that Yakutat is a perfect location to test a tidal turbine system to address the local demands.



<u>Tidal Power Generation:</u>

The total power P (W) generated by a tidal turbine is proportional to the cube of the current velocity v and can be calculated with the following integral

$$P = \frac{1}{2} A \rho_w \eta_{mech} \int_{v_{cutin}}^{v_{cutout}} C_p(v) v^3 p(v) dv \tag{1}$$

where ρ_w is the density of seawater (1026 kg.m^{-3}), v_{cutin} and v_{cutout} are the cut-in and cut-out speeds (same definitions as for a wind turbine), A is the swept area of the turbine blades, η_{mech} is the mechanical efficiency of the turbine and C_p is the turbine power coefficient, which is a function of v. Inside the integral, *p(v)* denotes the *probability density function*, which is a function that describes the relative likelihood for a random variable to take on a given value; specifically, in Eq. (1), this function relies on experimental velocity data to evaluate the probability that, for a given value comprised between v_{cutin} and v_{cutout}, the current flows at that particular velocity.

You are planning to use a new turbine prototype for which you have developed a numerical model that relates the power coefficient and the flow velocity according to

$$v = 6.433\times e^{-1.25C_p} - 6\times 10^{-16}\times e^{70C_p}. \qquad (2)$$

The other turbine parameters are summarized in the table below:

v_{cutin}	3.0 m.s^{-1}
v_{cutout}	5.0 m.s^{-1}
Blade Diameter	18 m
η_{mech}	0.9

Description of the data:

velocity.mat: Velocity data collected at a 50 m depth by the National Data Buoy Center at a Cape Suckling, near Yakutat, Alaska. Row 1 is average velocity (m.s^{-1}) for each month of the year, and row 2 is each month's respective standard deviation.

MATLAB programming:

1) Write a **generic** function to integrate $\int_{x_0}^{x_f} y(x)dx$ with the following format:

```
function I = integrator(x,y,method)
```

where `x` is a vector and `y` is a matrix whose columns `y(:,j)` are vectors of the same length as `x` and contain equally spaced data points. The output `I` is a **row** vector whose elements are the integrals of each column of `y` with respect to `x`. In other words, the jth element of `I` is $I(j) = \int_{x_0}^{x_f} y(:,j)dx$.

`method` is the technique used, and should either be the string 'trap' for the trapezoidal rule or 'simp' for Simpson's 1/3 rule. If the method argument is missing in the integrator function call, the default method should be set to the trapezoidal.

The integrator function should evaluate the number of intervals to integrate and should check that this number is even when Simpson's 1/3 rule is selected; if that's not the case, the function should stop and print an error message warning the user.

The integrator function should contain 2 subfunctions (local functions), one to implement the trapezoidal rule and one to implement Simpson's 1/3 rule. No loop should be used (use vector operations only). Check your subfunctions with simple problems for which you know the answer.

Hint: You may want to start writing this function for the (easier) special case of integrating a *vector* `y` and obtaining a *scalar* output `I`, and then, after checking your results, modifying your code to handle the general case of a matrix `y` yielding a vector `I`.

2) Write a **generic** function to solve a root-finding problem of the type $f(x) - V = 0$, where V is a vector; the function should have the following format:

```
function X = root_finder_vector(f,x0,V)
```

where `f` is a function handle that defines f, `V` is the vector V and `X` is a vector of the same length as V containing the roots: each element X(i) is the solution of $f(x) - V(i) = 0$. `x0` represents the initial guess(es), and is either a scalar (one initial guess for all V values) or a vector of the same size as V (each element of `x0` is a different initial guess for each element of `V`). Use the built-in function `fzero`.

3) Write a script called `CP2` that does the following:

- Defines all the necessary parameters and constants appearing in Equation (1).

Part A:

- Creates a COLUMN vector called `velocities` containing 100 equally spaced values ranging from v_{cutin} to v_{cutout}
- Calls your `root_finder_vector` function to calculate the values of the power coefficient for each value of the `velocities` vector by solving Equation (2); use C_p = 0.59 (this is the maximum possible value for the power coefficient, known as the *Bentz limit*) as your initial guess for all velocity values. This should produce a COLUMN vector of 100 C_p values.
- In Figure 1, plots C_p as a function of v from v_{cutin} to v_{cutout} , as a continuous line.

Part B:

- Loads the **velocity.mat** file
- Calculates the probability density function *p(v)* using the built-in command `normpdf`. This should a return a 100 by 12 matrix whose elements represent the probability that the actual velocity assumes the value of each element in the `velocities` vector, in a given month; `normpdf` calculates this probability based on the statistical information in the collected experimental data. See the documentation on this command for more detail.
- Formulates the integrand in Equation (1), using your `velocities` vector as the vector of independent variables, your *p(v)* matrix, and your C_p vector from Part A (if you have difficulties with Part A and are unable to generate the vector for C_p, or you think it is not correct, assume that C_p is constant and equal to 0.5).
- Call your integrator function to calculate the integral defined by Equation (1), and, in Figure 2, plot the projected turbine power output (in MW) as a function of the month, using discrete points.
- *This part of the project in meant to give you practice in exploring the Matlab help and documentation to use an unfamiliar command not covered in class*:
    - Use the `spline` command to fit a spline interpolating curve to your calculated monthly power data; plot the resulting curve as a smooth and continuous curve (use at least 100 points) in Figure 2 (with the discrete data still displayed).

- `spline` fits a cubic polynomial between each pair of successive points to form a piece-wise continuous curve; in some case, however, this is not adequate at the beginning and at the end of the data (first and last intervals). It is the case here, especially in the first interval. Read the command help to learn how to specify the slope at the first and last points; use the `spline` command again and use forward (first point) and backward (last point) difference approximations for the slopes. Plot the resulting smooth and continuous curve with a different color in Figure 2. Include a legend.

There is no report for CP2.

Instructions for Computer Project 2 submission:

Include all 3 m-files in a single zipped folder named **CP2_LASTNAME_FIRSTNAME** and upload to Canvas by the due date and time above. Please use the **.zip** format only to avoid problems with opening your folder zipped in other formats.

Please note the following, valid for all MATLAB assignments:

1. **Refrain from using the commands clear, clc and close (or any of their variations) in any of your m-files, as they interfere with the grading process.**

2. **When your code is run, it should produce only what is asked for in the assignment. Please make sure to suppress any unnecessary intermediate result in any form: no other figure, curve or value printed in the command window should appear.**

   Specifically, when CP2.m is run, we should only see, in that order:

   - **Figure 1 with a single continuous curve**
   - **Figure 2 with discrete points and 2 continuous curves**

Make sure your plots are correctly formatted, with titles,…You will lose points if you ignore these instructions.

Please note that Canvas will only accept a file with a ZIP extension: all of your M-files must be in that zipped folder. If we cannot open your files because you failed to follow these instructions, you will get a zero for this assignment.

You can upload your folder as often as you like, in case you find an error in your work. Older files will be kept in Canvas, but **only the last submitted folder will be graded**. It is your responsibility for this (and all other) electronic assignment to make sure we are grading the correct folder.

Finally, please keep in mind the policies about late assignments and collaboration outlined in the syllabus, and copied here for your convenience:

- Late assignments: homework and projects will be accepted up to 24 hours after the deadline with a 50% penalty. This policy will be strictly enforced and no submission will be accepted after 24 hours. Hard copies of homework will be due at the end of class. ***It is your responsibility to make sure that you have successfully uploaded and submitted all the required files in the required format to Canvas on time***: please double-check to avoid having to re-submit your work after the due date. The only acceptable proof that you have submitted your work on time is the time stamp of your Canvas submission. ***Files will not be accepted by email even if they show a "last modified date" that is before the due date.***

- Collaboration: students may *discuss* their assignments with each other, but homework and projects must be *completed individually* by each student. ***You must turn in your own work***. Copying someone else's work and submitting it as your own will not be tolerated. In particular, ***for Matlab assignments***, this policy means that students can discuss aspects such as the general approach to solve a problem or the syntax of a specific command, but ***they should not look at each other's codes***. If it is suspected that this has occurred, you will be reported to the Dean of Students for an honor code violation.