Writing, Programming, and Proof

Robert M. Corless[1] & Eunice Y.S. Chan[2]

[1] The Ontario Research Centre for Computer Algebra, The Rotman Institute of Philosophy, and The School of Mathematical and Statistical Sciences, Western University, Canada
The David R. Cheriton School of Computer Science, University of Waterloo, Canada

[2] Department of Anesthesia and Perioperative Medicine, MEDICI Centre, Schulich School of Medicine and Dentistry, Western University, Canada

Context

- Teaching *Computational Mathematics* is increasingly important (Data Science, Visualization, Machine Learning, . . .)
- This is *difficult* because computational mathematics involves several things at once: mathematics, programming, complexity, and numerical stability, because of the *compromises* needed for efficiency.
- The chief compromise (for memory efficiency) is to use *floating-point* arithmetic instead of exact arithmetic.

Why not compute exactly?

Consider the simple question of solving the following linear system of equations.

$$\begin{bmatrix} -81 & -98 & -76 & -4 & 29 \\ -38 & -77 & -72 & 27 & 44 \\ -18 & 57 & -2 & 8 & 92 \\ 87 & 27 & -32 & 69 & -31 \\ 33 & -93 & -74 & 99 & 67 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The students are taught in their first course to find the *inverse*. [We try to cure them of this.] For this five by five matrix, Maple or other computer algebra systems work fine.

The inverse

$$A^{-1} = \begin{bmatrix} -\frac{5396765}{31934714} & \frac{7836137}{31934714} & -\frac{541124}{15967357} & -\frac{813681}{31934714} & -\frac{850307}{15967357} \\ \frac{3136013}{79836785} & -\frac{851997}{15967357} & \frac{1144771}{79836785} & \frac{1144922}{79836785} & \frac{398039}{79836785} \\ \frac{6211662}{79836785} & -\frac{2183568}{15967357} & \frac{1004049}{79836785} & \frac{77321}{159673570} & \frac{6240977}{159673570} \\ \frac{2291472}{11405255} & -\frac{695571}{2281051} & \frac{449899}{11405255} & \frac{384233}{11405255} & \frac{852141}{11405255} \\ -\frac{11684919}{159673570} & \frac{3340737}{31934714} & -\frac{622854}{79836785} & -\frac{2680781}{159673570} & -\frac{1529291}{79836785} \end{bmatrix}$$

However, for even such a modest example, *the answer is nearly unintelligible*, and the *memory usage during the computation is nearly unpredictable*.

So we use IEEE floats instead

- IEEE 754/854 Floating-Point Standard means portable code
- Predictable memory usage implies *efficiency*
- Thousands of times faster

Challenges from IEEE floats

- "Admit, for instance, the existence of a minimum magnitude, and you will find that the minimum which you have introduced, small as it is, causes the greatest truths of mathematics to totter." — Aristotle
- Floats are *not associative*: $a + (b + c) \neq (a + b) + c$ necessarily. For instance $-M + (M + 1) = 0$ while $(-M + M) + 1 = 1$ if $M = 3.14 \cdot 10^{17}$.
- This (and other features) break students' models of how the world works.
- **Needs special algorithms** e.g. **QR** algorithm for eigenvalues

A Challenging Example

Consider matrices of the form

$$\begin{bmatrix} 0 & U_1 & 0 & 0 & 0 & 0 & 0 \\ -U_1 & 0 & U_2 & 0 & 0 & 0 & 0 \\ 0 & -U_2 & 0 & U_3 & 0 & 0 & 0 \\ 0 & 0 & -U_3 & 0 & U_4 & 0 & 0 \\ 0 & 0 & 0 & -U_4 & 0 & U_5 & 0 \\ 0 & 0 & 0 & 0 & -U_5 & 0 & U_6 \\ 0 & 0 & 0 & 0 & 0 & -U_6 & 0 \end{bmatrix}$$

where each U_i can be one of $\{1, i, 1+i, 1-i\}$ (here $i^2 = -1$). Thus there are exactly $4^6 = 4096$ such matrices. We will compute all of the eigenvalues of all of these matrices and plot them all on the same graph.

Eigenvalue Pictures (Maple)

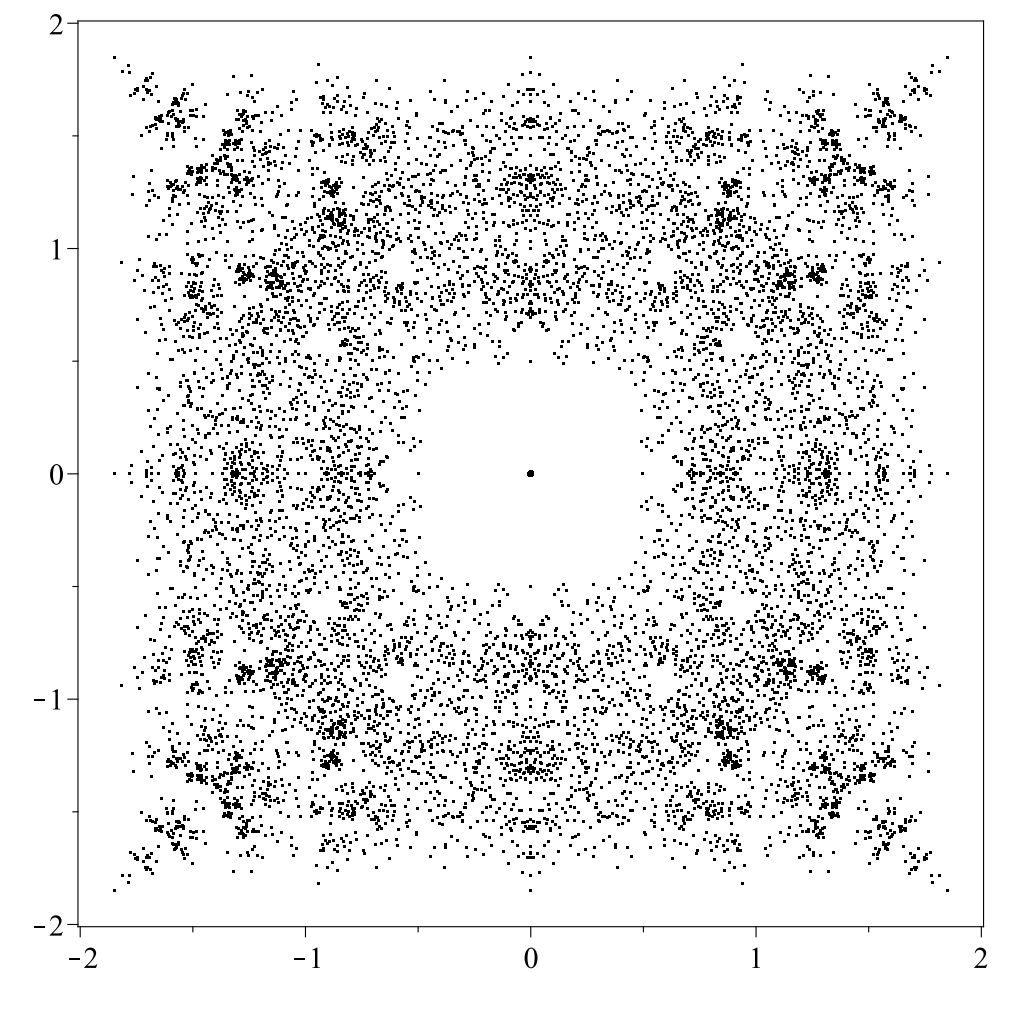


Figure 1: All eigenvalues computed numerically by Maple of all 4096 seven by seven skew-symmetric bidiagonal matrices with entries from $\{1, i, 1+i, 1-i\}$.

Zooming In

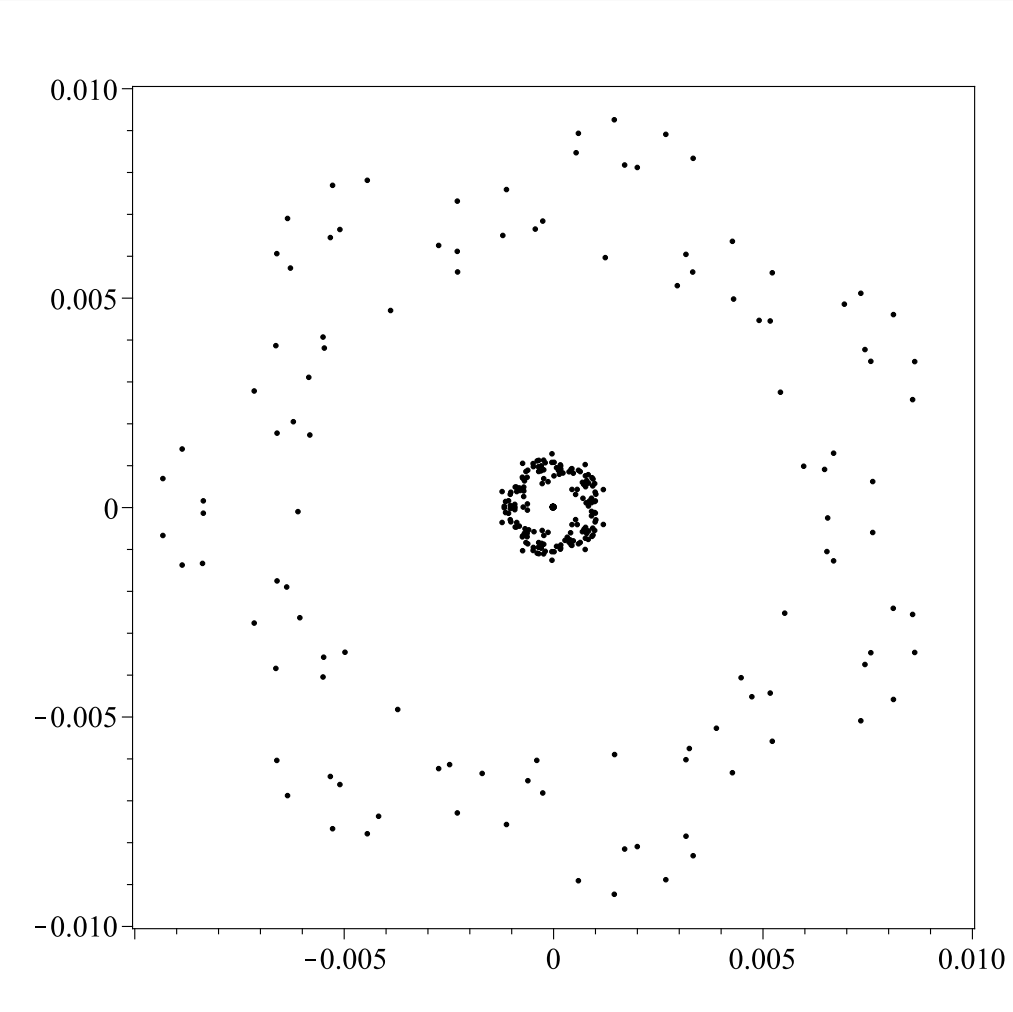


Figure 2: Zooming in on eigenvalues computed numerically by Maple of all 4096 seven by seven skew-symmetric bidiagonal matrices with entries from $\{1, i, 1+i, 1-i\}$.

Zooming In More

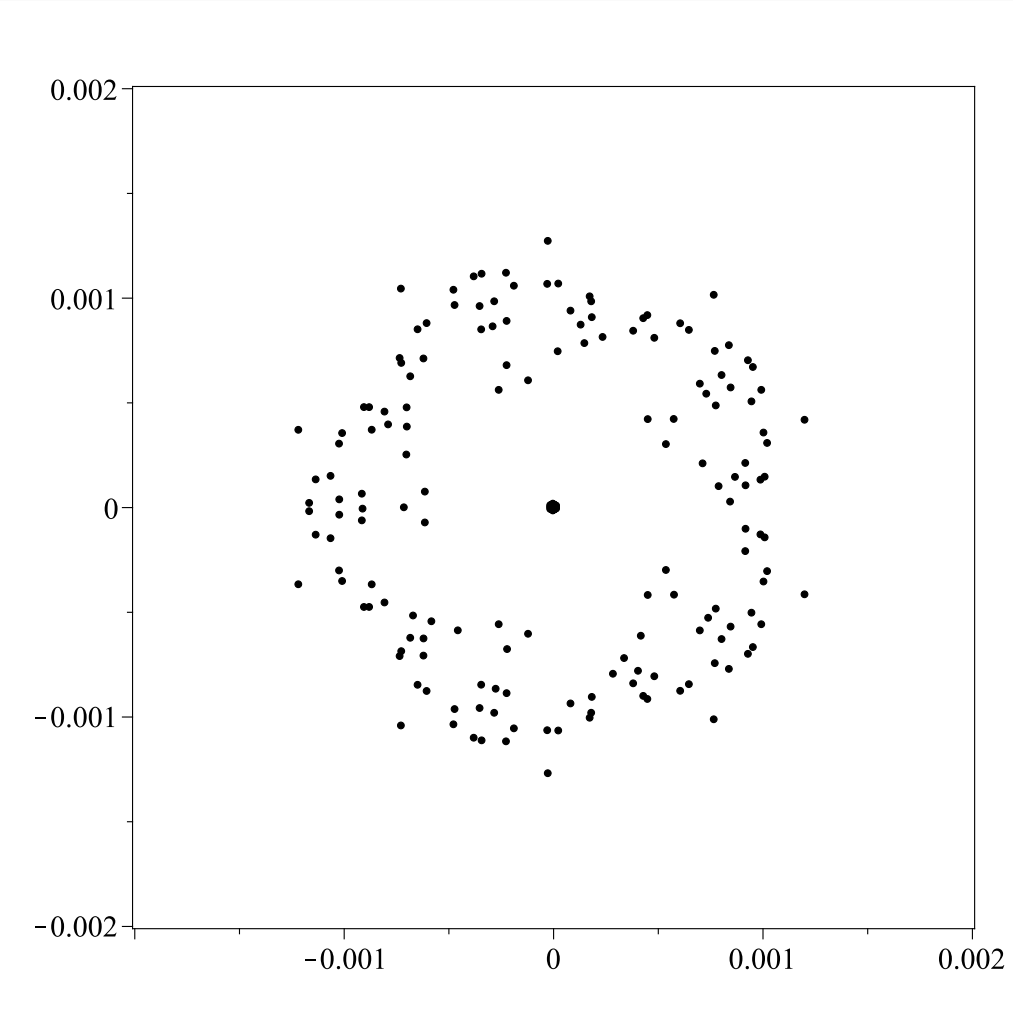


Figure 3: Zooming in more on eigenvalues computed numerically by Maple of all 4096 seven by seven skew-symmetric bidiagonal matrices with entries from $\{1, i, 1+i, 1-i\}$.

Zooming In Even More

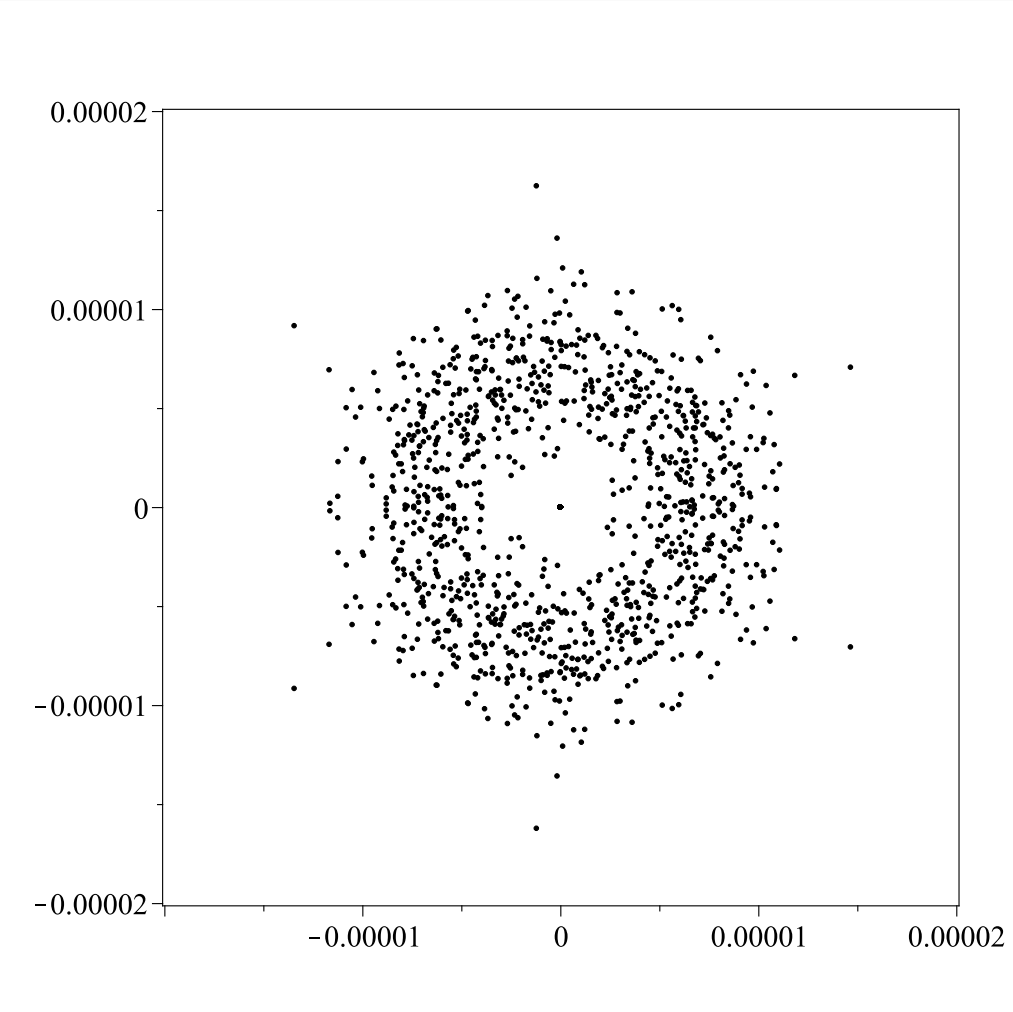


Figure 4: Zooming in even more on eigenvalues computed numerically by Maple of all 4096 seven by seven skew-symmetric bidiagonal matrices with entries from $\{1, i, 1+i, 1-i\}$.

Are they right?

- When we compare the results to those done in Matlab, they seem *exactly the same*.
- The eigenvalues have a *pleasing symmetry*: seven-fold at the 0.01 scale, five-fold at the 0.001 scale, and three-fold at the 10^{-5} scale.

Are they right? [symbolic check with Maple and its CodeGeneration[Matlab]]
http://publish.uwo.ca/∼rcorless/Maple2019/

These small eigenvalues are Rounding Errors

- One can prove mathematically that the eigenvalues of this set of matrices has D_4 symmetry. This means that all these small eigenvalues with non-D_4 symmetry must really be 0.
- **It is a great surprise to students that rounding errors are not random.** Such surprises are didactically useful.
- The seven-symmetric roots come from matrices with λ^7 as their characteristic polynomials; the five-fold from matrices with a factor λ^5; the three-fold from matrices with a factor of λ^3. Multiple roots are infinitely ill-conditioned.
- Roots of $\lambda^7 = 10^{-14}$ are symmetric about 0 and of magnitude 0.01. “Exact eigenvalues of nearby matrices”

Upsetting the Quadratic Formula

Solve $x^2 - 2bx + 1 = 0$ for many large values of b using the quadratic formula: $x_{1,2} = b \pm \sqrt{b^2 - 1}$. The product of these roots is $b^2 - (b^2 - 1) = 1$. **But the quadratic formula is unstable.**

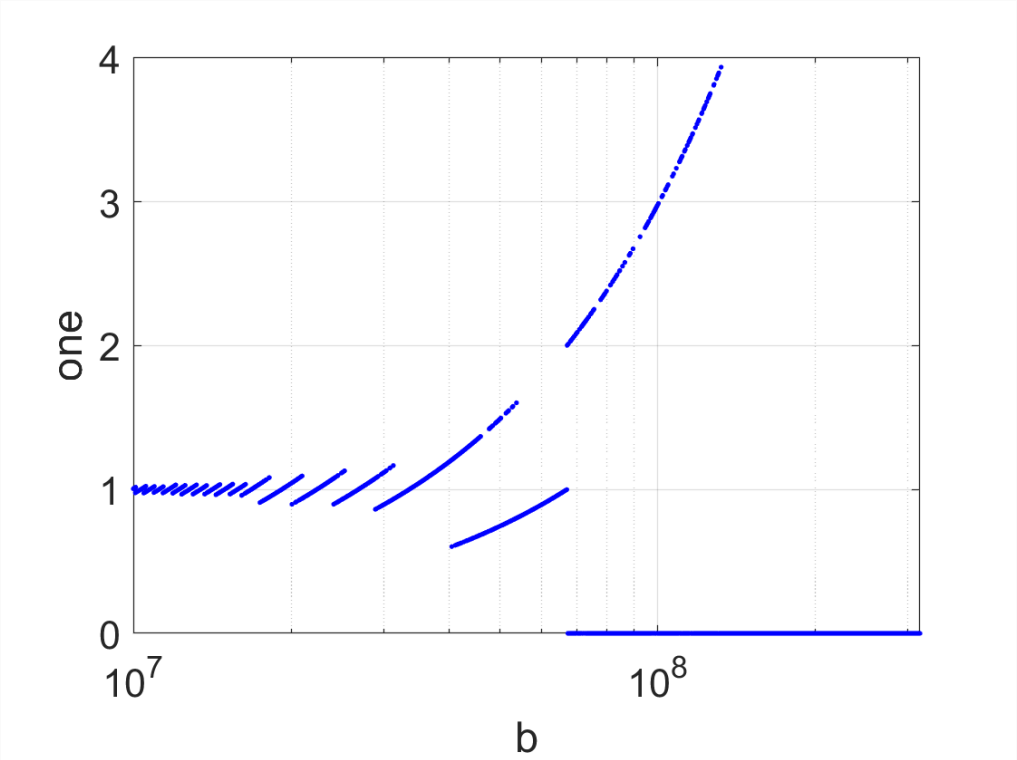


Figure 5: The product of two computed (unstable) quadratic formula roots, for different values of the parameter b.

The importance of Backward Error

- The students must learn to *reverse their perspective*. (Painful, but necessary)
- One useful tool for this is that a **numerically stable** algorithm gives (nearly) **the exact answer to a nearby question**.
- We use this uniformly on all topics: polynomial evaluation, rootfinding, solving linear systems, solving differential equations, etc.
- The students also need to learn that *some problems are sensitive to changes* or **ill-conditioned**.

Principles of Scientific Programming

- The Programmer is Responsible for the Result
- The Modeller should know what is going on in the program
- The Person Should be Able to Explain (and Reproduce) the Result

Principles of Numerical Analysis

- A good numerical method gives (nearly) the exact solution to nearly the right problem
- Some problems are sensitive to changes in their data or models.

The role of writing

"Good writing is clear thinking made visible."—Ambrose Bierce

1. Promotes precision
2. Enhances error-checking
3. A good essay is like a good program is like a good proof:
    - 3.1 defines its terms
    - 3.2 includes enough detail for a person "skilled in the art" to fill in the rest
    - 3.3 makes its conclusions clear

Forcing students to *create a narrative* forces them to think about what the results *mean*, not just what they *are*.

An example problem

Campus Circus

Your TA, Eunice Chan, also works as a part-time stunt performer. She is planning on being catapulted from the entrance of Middlesex college to the main doors of University Hospital, a distance of 605m. A 55kg undergraduate student is catapulted first to test the catapult to make sure Eunice will land close to the hospital.

Measurements of the speed resolved into x and y coordinates are taken from the undergraduate student's smartwatch. The file `speed2018.mat`, which can be downloaded from OWL, contains the speed data. To load this data into MATLAB use the following command

```
load speed2018.mat
```

This loads three variables:

- `t` The time points at which the measurements were taken, in seconds.
- `speedx` The speed in the x direction, in $\frac{\text{m}}{\text{s}}$
- `speedy` The speed in the y direction, in $\frac{\text{m}}{\text{s}}$

Figure 6: Example of lab problem from the quadrature unit for our course AM 2814 (Numerical Analysis)

An example problem

Table 1: Table of variables provided by *speed2018.mat* data file

index	1	2	···	39	40
t	0	0.2769	···	10.5231	10.8000
speedx	64.1547	63.1874	···	52.0130	47.0933
speedy	55.6829	55.3540	···	−52.8820	−53.8300

An example problem

- Numerically integrate the speed data in each direction using the Trapezoid rule to get the distance in each direction. That is compute

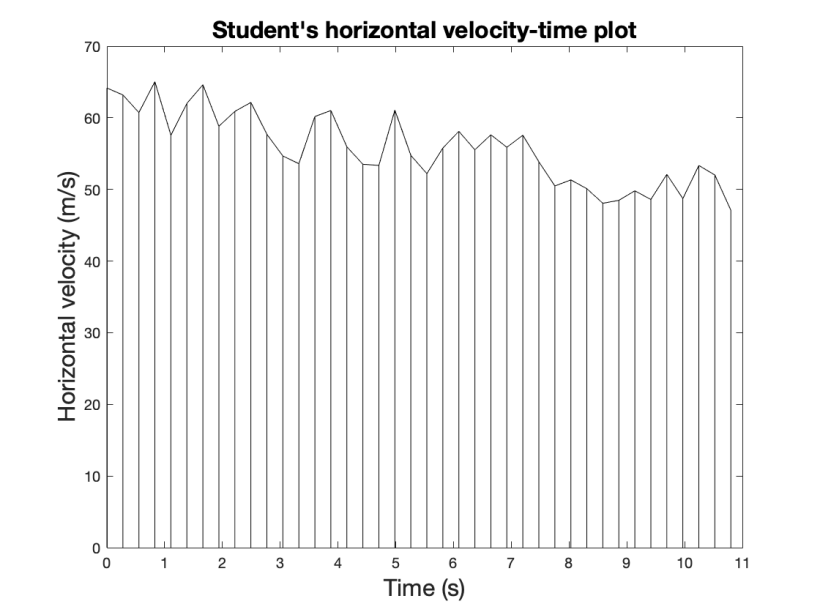
$$d_x(t_i) = \int_{t=0}^{t_i} v_x(t)\,\mathrm{d}t$$
$$d_y(t_i) = \int_{t=0}^{t_i} v_y(t)\,\mathrm{d}t$$

  where $d_x(t)$ is the distance in the x direction at time t and $d_y(t)$ is the distance in the y direction. You must compute this for all values of i. (Note: you will need to write a `trapezoidRule` function that works with vector input). You can use the built-in trapezoid rule function `trapz` in MATLAB if you would like.

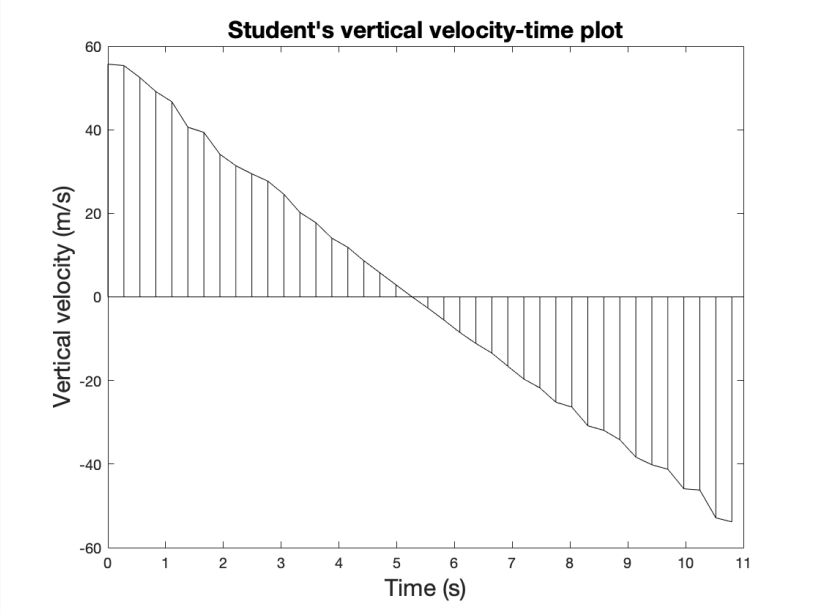
- Make a plot of the trajectory of the undergraduate student (i.e. $d_y(t)$ versus $d_x(t)$)
- Based on your analysis, does the undergraduate student land close to the hospital entrance (within 10m). What is the total distance travelled in the x direction?

Figure 7: Second part of the example of lab problem from the quadrature unit for our course AM 2814 (Numerical Analysis)

An example problem



(a) Horizontal velocity-time plot



(b) Vertical velocity-time plot

Figure 8: Velocity-time plots for example problem

An example problem

```
function T = vecTrapz(v, t)
    if (length(v) ~= length(t))
        error(["The lengths of the vectors"...
        "should be the same."]);
    end
    n = length(v);
    h = t(2) - t(1);
    T = 0;
    for i = 1:(n-1)
        T = T + ((v(i) + v(i+1))/2)*h;
    end
end
```

An example problem

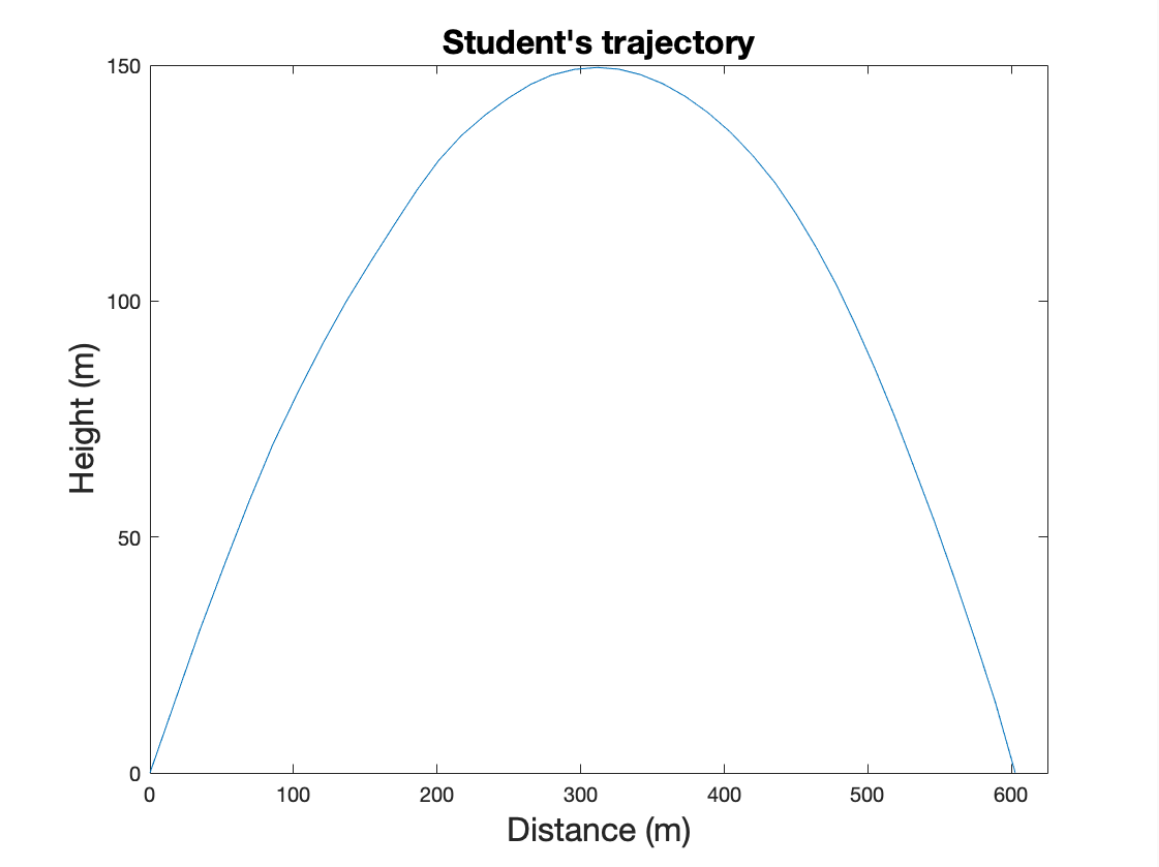
```
load speed2018.mat

dx = zeros(size(t));
dy = zeros(size(t));

for i = 2:length(t)
    dx(i) = vecTrapz(speedx(1:i), t(1:i));
    dy(i) = vecTrapz(speedy(1:i), t(1:i));
end

plot(dx, dy);
axis([0, 625, 0, 150])
xlabel('Distance (m)')
ylabel('Height (m)')
title("Student's trajectory")
```

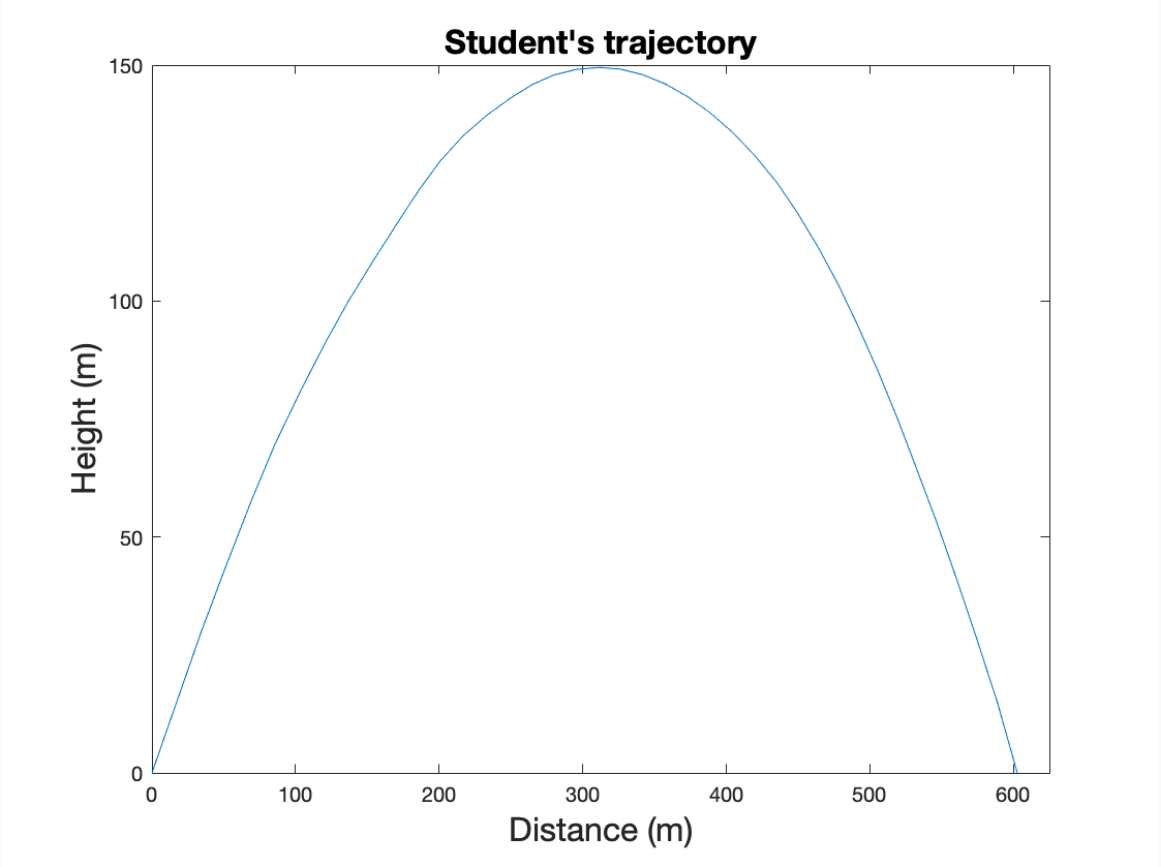
An example problem



(a) Solution to the example problem

Figure 9: Solution to example problem

An example problem



(a) Solution to the example problem

```
>> max(dy)

ans =

   149.5001

>> dx(end)

ans =

   602.5728
```

(b) Maximum height and horizontal distance

Figure 9: Solution to example problem

Reference Papers

Chauvenet prize-winning papers

1. The Perfidious Polynomial — J. H. Wilkinson
2. The Simple Continued fraction for e — C. D. Olds
3. Leonhard Euler's Integral — Philip J. Davis

Reference Texts and Videos

1. Numerical Computing with Matlab — Cleve Moler
2. Matlab Guide — Des Higham and Nick Higham
3. The Elements of Matlab Style — Richard K. Johnson
4. MIT OpenCourseWare: Gil Strang and Cleve Moler

Acknowledgements

Piers Lawrence wrote the first versions of the course labs, inspired by Bob Broughton's labs at the University of Canterbury. Steven Thornton developed them further. This research was supported in part by Western University through various sources including a Teaching Fellowship Award and in part by the Rotman Institute of Philosophy. Connection with the University of Bath group was made possible through an invitation from the Isaac Newton Institute during the Complex Analysis, Tools, Techniques and Applications programme when work on this project was undertaken. This work was supported by EPSRC Grant # EP/R014604/1

Thank You

¡Time for Questions!