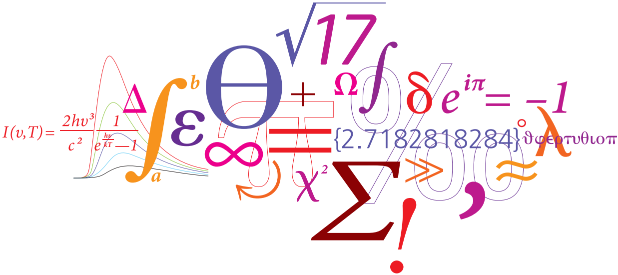
Gotta catch 'em all!

Root finding using matricies

Arun Suresh

Georgia State University



- The root finding problem
  - Issues with this
- A shift in perspective
  - Finding the A matrix
  - Pros
- 3.5 different applications
  - Polynomial functions
  - Non-Algebraic roots
  - An actual physics application
  - An extreme case

      may not be so extreme depending on the programming language and skill-set of the programmer

The root finding problem

- Used to obtain the roots of any given function
- Close to 8-10 different algorithms exist
  - Ranging from bisection to various interpolation schemes
- Motivation:
  - Find an interval where the given function changes sign
  - Choose two points such that function takes opposite signs at each of them
  - Approximate a root in the given interval
  - Change the end points to make the approximation better
  - Converge to the true root
- Generally fast convergence (Unless the given function is horrible)

Issues

- The issues with these methods are
  - Initially the user has to bracket a root for most algorithms to work. For example,
    - $x^4 - 102632x^3 - 115571937x^2 - 114853500x + 692815788$ has four roots $103746, 2, -3$ and -1113
    - Initial interval must be in the order of 100000s in order to obtain a root
    - This only gets worse for non-polynomial function (because long division is not a thing)
- For a given interval, the algorithm converges to only one root per run.
  - Sometimes it can oscillate between two given the same initial interval in two different runs.
- Forces the user to have an intuitive idea about the approximate distribution of the roots
- complex roots are not a thing in these algorithms

A shift in perspective

- In the final week of classes we discussed Eigenvalue solvers
- The Eigenvalue problem
  - Let $A = n \times n$ matrix
  - Find all λ such that $A\mathbf{x} = \lambda\mathbf{x} \implies (A - \lambda I)\mathbf{x} = \mathbf{0} \implies det(A - \lambda I) = 0$
  - This yields an n^{th} degree polynomial equation
- So any polynomial root finding can be implemented as an eigenvalue problem given we can obtain the corresponding A matrix.
- Assuming we do have such a matrix,
  - There are eigenvalue solvers such as
    - Householder QR transformations
    - Divide and Conquer iterations

    that use matrix methods to solve for λ

Finding the A matrix

- Suppose we have a polynomial $f = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0$
- For ease of writing, we let LC $= 1$
- Find a matrix A such that $det(A - xI) = f$
  - Quadratic polynomial:
    $x^2 + 3x + 4 = x(x+3) + 4 = (0 - x)(-3 - x) - (-4)(1)$.
  - $A_2 = \begin{bmatrix} 0 & -4 \\ 1 & -3 \end{bmatrix}$
  - Cubic polynomial: $x^3 + 3x^2 + 4x + 3 = x(x^2 + 3x + 4) - (-3)(1)$
  - A_2 for $x^2 + 3x + 4$ known! A_2 forms the first principal sub-matrix of A_3
  - new constant term $\implies$ last principal sub-matrix has determinant 1
  - This yields $A_3 = \begin{bmatrix} 0 & 0 & -3 \\ 1 & \mathbf{0} & \mathbf{-4} \\ 0 & \mathbf{1} & \mathbf{-3} \end{bmatrix}$

Finding the A matrix

- This can be easily generalized now for n^{th} degree polynomials!
- It would simply be

$$A_n = \begin{bmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_{n-1} \end{bmatrix}$$

- This matrix is often called a companion matrix

Benefits of this method

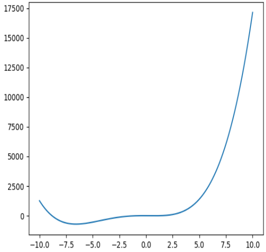
- Very robust. Since we use very sparse matrices, computationally not very costly
- identify all roots at the same time no matter how large!
- identify even the complex roots!
- computationally faster than root finding
    - Divide and Conquer : Quadratic convergence
    - QR algorithm: Cubic convergence

3.5 Different applications: Polynomial Functions

- Take $n \in \mathbb{Z}^+$
- Generates a random n^{th} degree polynomial with integer coefficients.
- Create A_n
- Solve for eigen-values of A_n using eigenvalue solver that comes built in with numpy (uses QR factorization for hessenberg matrices)

Results

- $n = 4$ and $f = x^4 + 8x^3 - 8x^2 - 6x + 4$



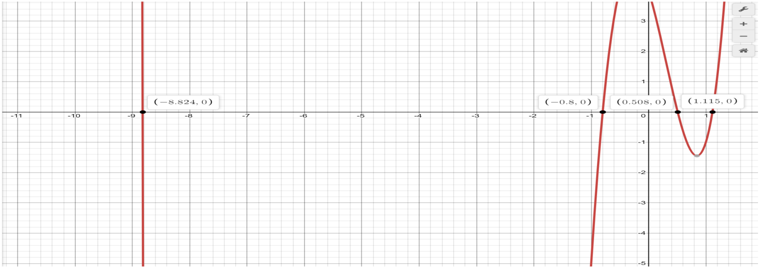
(a) Polynomial

- The output obtained was

```
[1, 8, -8, -6, 4]
[ 1.11505411  0.50838569 -0.79968166 -8.82375813]
```

(b) Coefficients + Zeroes

- The following is the function plotted on desmos and output from Wolfram



(c) Desmos plot

Roots: More digits Exact forms Step-by-step solution

$x \approx -8.8238$

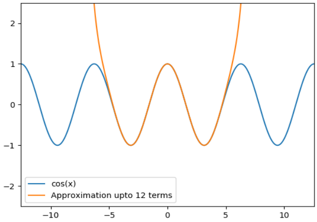
$x \approx -0.79968$

$x \approx 0.50839$

$x \approx 1.1151$

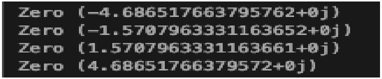
(d) Wolfram

- But you may ask - what about non-polynomial function?
- Answer: TAYLOR SERIES!
- Take $f(x) = \cos(x) \quad x \in [-2\pi, 2\pi]$
- Consider the first twelve terms of its Taylor polynomial!



(e) Cos(x) + Taylor

- Output obtained



```
Zero (-4.686517663795762+0j)
Zero (-1.5707963331163652+0j)
Zero (1.5707963331163661+0j)
Zero (4.68651766379572+0j)
```

(f) Output

Error Analysis:

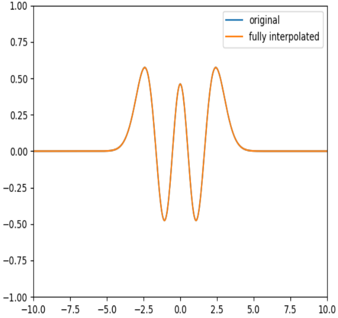
- From practice, we know that in this domain, $\cos(x) = 0 \iff x = -\frac{\pi}{2}, -\frac{3\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}$
- $\frac{\pi}{2} = 1.57079632679$ and $\frac{3\pi}{2} = 4.71238898038$
- So, absolute errors are:
  - $|1.57079632679 - 1.57079633311| = 6.32000008 \times 10^{-9}$
  - $|4.71238898038 - 4.68651766379| = 0.02587131659$
- So, absolute error is always less than 0.03 for $|x| < \frac{3\pi}{2}$
- For more accurate estimates: increase number of terms!

- But what are the explicit physics applications?
- Answer: Lots. Here's an example
- The quantum harmonic oscillator has solutions of the form

$$\Psi = \left(\frac{\alpha}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(y) e^{-y^2/2}$$

- H_n = Hermtie polynomials.
- So, $\Psi = 0 \implies H_n = 0$
  - This now becomes a root-finding problem for polynomials!

- Plotted below is the fourth eigenstate solution to the QHO wavefunction



(g) Ψ_4

- Setting $H_4(y) = 0$ the output was

```
[-1.65068012 -0.52464762  0.52464762  1.65068012]
```

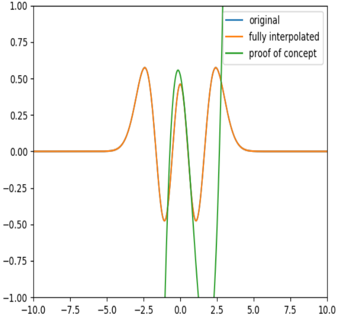
(h) Ψ_4 zeroes

- $H_4(y) = 16y^4 - 48y^2 + 12$
- H_4 has very nice closed form roots $\psi_i = \pm\sqrt{\frac{3}{2} \pm \sqrt{\frac{3}{2}}}$
- $\sqrt{\frac{3}{2} + \sqrt{\frac{3}{2}}} = 1.65068012388578$ and $\sqrt{\frac{3}{2} - \sqrt{\frac{3}{2}}} = 0.524647623275290$
- So the absolute errors are:
  - $|1.65068012388578 - 1.65068012000000| = 3.88 \times 10^{-9}$
  - $|0.524647623275290 - 0.52464762000000| = 3.27 \times 10^{-9}$
- So, absolute error is always less than 3.88×10^{-9}

A final "pseudo-example"

- Suppose we do not have an analytic function to work with - then this process gets a bit tedious.
- With only data points, one can
  - interpolate through the points
  - obtain a set of polynomials
    - $m \times n$ coefficients for m^{th} order interpolation with n data points
  - For each polynomial check if it changes sign in its respective domain
    - if it does, run the algorithm to find the root in that domain
    - if not, move to the next polynomial
- This is a painful process, and perhaps root-finding can do better here, if the function is well-behaved in the interested domain

- Presented below is a proof of concept that this would indeed work.
- Consider Ψ_4 again.



(i) Ψ_4 interpolation

- The interpolated polynomial returned 0.524647
- With an absolute error of $|0.524647623275290 - 0.524647| = 6.23 \times 10^{-7}$ which is still a very good approximation!

Work Cited:

- Algorithms for reducing matrix to condensed form
  http://www.cs.utexas.edu/users/flame/pubs/flawn53.pdf
- Horn, Roger A.; Charles R. Johnson (1985). Matrix Analysis. Cambridge, UK: Cambridge University Press. pp. 146–147. ISBN 0-521-30586-1. Retrieved 2010-02-10

Thank you!

Note: This is the end of the presentation. I will leave you alone now.

Arun Suresh

Georgia State University