COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology Palakkad
CS5016: Computational Methods and Applications

21 Mar, 2024

Coding Assignment 7

Partial Differential Equations and Finding Zeros

1. The following PDE governs heat conduction in a unit length rod with unit thermal diffusivity

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}$$

with boundary condition

$$u(0,t) = u(1,t) = 0 \quad \forall t > 0$$

Write a function to visualize (through an animation) heat conduction in the rod; initial condition $u(x,0) = e^{-x} \quad \forall x \in [0,1]$.

2. The following PDE governs heat conduction in a 2-D unit square sheet $\Omega = [0,1] \times [0,1]$ with unit thermal diffusivity in the presence of a heat source $f(x,y,t) = e^{-\sqrt{(x-x_c)^2+(y-y_c)^2}} \quad \forall (x,y) \in [0,1] \times [0,1]$

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial^2 u(x,t)}{\partial y^2} + f(x,y,t)$$

with boundary condition

$$u(x,y,t) = 0 \quad \forall (x,y) \in \partial\Omega, t > 0$$

and initial condition

$$u(x,y,0) = 0 \quad \forall (x,y) \in \Omega$$

Write a function to visualize (through an animation) heat conduction in the sheet.

NOTE: Your function is expected to take x_c and y_c as its arguments.

Use `matplotlib.pyplot.imshow`[1] to visualize the heat conduction.

3. Write a function that takes as its argument an integer n and two positive real numbers a and ϵ. The function should then compute the n^{th} root of a with an error tolerance of ϵ. Your function should have a worst-cast run-time complexity of $O(\log(1/\epsilon))$.

4. When an analytical expression for the derivative of function $f : \mathbb{R} \to \mathbb{R}$ is not available, one can compute the root of f using the *Secant method*[2] defined by the recurrence relation

$$x_{k+1} = x_k - f(x_k)\frac{(x_k - x_{k-1})}{(f(x_k) - f(x_{k-1}))}$$

However, it is known that *Secant method* exhibits slower convergence than *Newton-Raphson method*. Write a program to compare the convergence rate of both these methods for a function f of your choice.

[1] `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html`
[2] `https://en.wikipedia.org/wiki/Secant_method`

Continued on next page...

5. Write a function that uses the Newton-Raphson method to find a vector $\mathbf{x} = [x_1, x_2, x_3]$ such that

$$f_1(\boldsymbol{x}) = 3x_1 - cos(x_2x_3) - 3/2 = 0$$
$$f_2(\boldsymbol{x}) = 4x_1^2 - 625x_2^2 + 2x_3 - 1 = 0$$
$$f_3(\boldsymbol{x}) = 20x_3 + e^{-x_1x_2} + 9 = 0$$

   Also, plot the value of $\|\boldsymbol{f}(\boldsymbol{x}_k)\|$ against the number of iterations.

   NOTE: You can use routines from `scipy.linalg` in your program.

6. The *Aberth method*[3] is a root-finding algorithm that can simultaneously approximate all roots of a univariate polynomial. Write a function that takes as its arguments an array of real number $[a_1, a_2, \ldots, a_n]$, computes the polynomial $g(x) = \Pi_{i=1}^{n}(x - a_i)$, and outputs all roots of $g(x)$ computed (within an error is 10^{-3}) using the *Aberth method*.

   HINT: To the enhanced `Polynomial` class developed a few labs ago, add a method `printRoots` that outputs all roots of the polynomial computed within an error is 10^{-3} using the *Aberth method*.

7. Write a function that leverages the enhanced `Polynomial` class to compute *all zeros* of a continuous function f in the interval $[a, b]$ within an error is 10^{-3}. [20]

   NOTE: Your approach should work for any continuous function f that has at-least one zero in the interval $[a, b]$.

[3] `https://en.wikipedia.org/wiki/Aberth_method`