

COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology, Palakkad

CS5016: Computational Methods and Applications

Coding Assignment 7

Partial Differential Equations and Finding Zeros

15 Apr, 2021

Max points: 150

A few instructions

- Codes should be compatible with *Python3* and should run on Ubuntu.
- Code for each question should be placed in a separate stand-alone files.
- Codes should be well-commented.
- Appropriate exceptions should be raised and handled.
- 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 $u(x,0) = e^{-x} \quad \forall x \in [0,1].$

[20]

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.

[30]

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

Use matplotlib.pyplot.imshow¹ 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 $O(\log(1/\epsilon))$.

[20]

¹https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html

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 method² 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.

[20]

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

$$f_1(\mathbf{x}) = 3x_1 - \cos(x_2 x_3) - 3/2 = 0$$

$$f_2(\mathbf{x}) = 4x_1^2 - 625x_2^2 + 2x_3 - 1 = 0$$

$$f_3(\mathbf{x}) = 20x_3 + e^{-x_1 x_2} + 9 = 0$$

Also, plot the value of $||f(x_k)||$ against the number of iterations.

[20]

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

6. The Aberth method³ 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

[20]

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*.

number $[a_1, a_2, \ldots, a_n]$, computes the polynomial $g(x) = \prod_{i=1}^n (x - a_i)$, and outputs all roots

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].

²https://en.wikipedia.org/wiki/Secant_method

³https://en.wikipedia.org/wiki/Aberth_method