CS5016: Computational Methods and Applications

Partial Differential Equations and Finding Roots

Albert Sunny

Department of Computer Science and Engineering
Indian Institute of Technology Palakkad

21 March, 2024

What is an PDE?

An equation involving one or more derivatives of an unknown function.

If all derivatives are taken with respect to a several independent variable we get an **partial differential equation**.

The well-known 1-D heat equation

$$\frac{\partial u(x,t)}{\partial t} - \mu \frac{\partial^2 u(x,t)}{\partial x^2} = f(x,t), \quad x \in (a,b), t > 0$$

where $\mu > 0$ is the coefficient representing thermal diffusivity.

Boundary value problem

Differential equations in an open multidimensional region $\Omega \subset \mathbb{R}^d$ for which the value of the unknown solution (or its derivatives) is prescribed on the boundary $\partial\Omega$ of the multidimensional region.

$$\frac{\partial u(x,t)}{\partial t} - \mu \frac{\partial^2 u(x,t)}{\partial x^2} = f(x,t), \quad x \in (a,b), t > 0$$

with the initial condition

$$u(x,0) = g(x) \quad \forall x \in [a,b]$$

and boundary condition

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

Approximation by finite differences

Consider the following approximation for $h > 0$

$$\frac{\partial u(x,t)}{\partial x} \approx g(x,t) = \frac{u(x+h/2,t) - u(x-h/2,t)}{h}$$

Then, we have

$$\begin{aligned}
\frac{\partial^2 u(x,t)}{\partial x^2} &\approx \frac{g(x+h/2,t) - g(x-h/2,t)}{h} \\
&\approx \frac{u(x+h,t) - u(x,t)}{h^2} - \frac{u(x,t) - u(x-h,t)}{h^2} \\
&= \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2}
\end{aligned}$$

Approximation by finite differences

Let us partition the interval $[a, b]$ into interval $I_j = [x_j, x_{j+1}]$ of length h for $j = 0, 1, \ldots, N$ with $x_0 = a$ and $x_N = b$.

Let $u_j(t)$ denote an approximation of $u(x_j, t)$, $j \in \{0, \ldots N\}$. Then, for all $t > 0$, we should have $\forall j \in \{1, \ldots, N-1\}$

$$\frac{du_j(t)}{dt} - \frac{\mu}{h^2}(u_{j-1}(t) - 2u_j(t) + u_{j+1}(t)) = f_j(t)$$

with initial condition $u_0(t) = 0$ and $u_N(t) = 0$, $f_j(t) = f(x_j, t)$, and $u_j(0) = g(x_j)$

Giving us the following system of ODE

$$\frac{d\boldsymbol{u}(t)}{dt} = \frac{\mu}{h^2}\boldsymbol{A}\boldsymbol{u}(t) + \boldsymbol{f}(t)$$

with $\boldsymbol{u}(0) = \boldsymbol{g}$

Handling problem with 2 spatial dimensions

We have $u(\boldsymbol{x}, t)$, where $\boldsymbol{x} \in \mathbb{R}^2$. The heat equation is given as

$$\frac{\partial u(\boldsymbol{x}, t)}{\partial t} - \mu \frac{\partial^2 u(\boldsymbol{x}, t)}{\partial x_1^2} - \mu \frac{\partial^2 u(\boldsymbol{x}, t)}{\partial x_2^2} = f(\boldsymbol{x}, t) \quad \forall x \in \Omega$$

with initial condition $u(\boldsymbol{x}, 0) = g(\boldsymbol{x})$ and boundary condition $u(\boldsymbol{x}, t) = 0 \, \forall \boldsymbol{x} \in \partial\Omega, t \geq 0$

Approximate Ω as a grid of points such that $x_{1,i} = x_{1,0} + ih$ and $x_{2,j} = x_{2,0} + jh$

Figure out approximations for $\frac{\partial^2 u(\boldsymbol{x},t)}{\partial x_1^2}$ and $\frac{\partial^2 u(\boldsymbol{x},t)}{\partial x_2^2}$, and solve the resultant system of ODE.

Root-finding algorithms

In mathematics and computing, a root-finding algorithm is an algorithm for finding **zeroes**, also called "**roots**", of continuous functions[1].

A zero of a function $f : \mathbb{R} \to \mathbb{R}$, is a number x such that $f(x) = 0$. As, generally, the zeroes of a function cannot be computed exactly nor expressed in closed form, root-finding algorithms provide approximations to zeroes.

Most root-finding algorithms do not guarantee that they will find all the roots; in particular, if such an algorithm does not find any root, that does not mean that no root exists.

[1] `https://en.wikipedia.org/wiki/Root-finding_algorithms`

The bisection method

Consider a continuous function $f : \mathbb{R} \rightarrow \mathbb{R}$ and an interval $[a, b]$. If $f(a) \cdot f(b) <= 0$, then function f has at least one zero/root in the interval $[a, b]$, i.e., there exists a point $x^* \in [a, b]$ such that $f(x^*) = 0$.

Algorithm 1 Pesudocode

```
while |a − b| > ε do
    let c = (a + b)/2
    if sgn(f(c)) == sgn(f(a)) then
        a = c
    else
        b = c
    end if
end while
return (a + b)/2
```

What is the run-time complexity of the above algorithm?

Newton-Raphson method

Consider a differentiable function $f : \mathbb{R} \to \mathbb{R}$. If the f satisfies sufficient assumptions and the initial guess x_0 is close, a root can be found using the following iterative method

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

What are the conditions that f should satisfy?

If $F : \mathbb{R}^k \to \mathbb{R}^k$ multivariate vector-valued function, then the iteration is given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}(\boldsymbol{x}_k)^{-1}\boldsymbol{F}(x_k)$$

where $\boldsymbol{J}(\boldsymbol{x}_k)$ is the **Jacobian matrix** of $\boldsymbol{F}$.

More root-finding methods

The SciPy module `scipy.optimize` offers methods to find zeros/roots of function. To know more visit `https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html#root-finding`

Thank You