國立清華大學
NATIONAL TSING HUA UNIVERSITY

10820ASTR660000
Computational Astrophysics
計算天文物理

Release date: 2021.04.1
Due in class: 2021.04.8
(submit to google classroom)

Problem Set 5

Reading Assignments

1. Inverse square root is essential in many physical process. In video games, the performance is more important than the accuracy. Read the article "Fast inverse square root" http://www.lomont.org/papers/2003/InvSqrt.pdf
2. Read the article "Buffon's needle problem" http://mathworld.wolfram.com/BuffonsNeedleProb
html
3. Read the root-finding documentation in `scipy` https://docs.scipy.org/doc/scipy/reference/optimize.html

Programming Assignments

1. Modify the Fortran program we developed in lectures to evaluate the real root(s) of

$$x^3 + 1.5x^2 - 5.75x + 4.37 = 0 \tag{1}$$

Please try: a) bisection, b) false position, c) Newton's method, and d) secant method. compare the performance of methods (draw a plot). [30pts]

2. Write a Fortran program to evaluate

$$\int_0^\infty \frac{dx}{1 + x^2 + x^4} \tag{2}$$

by breaking the range at 2 and transforming the tail suitably onto a finite range. [30pts]

3. (a) Evaluate the Stefan-Boltzmann constant by writing a Fortran program to do Monte Carlo (MC) integration

$$\frac{\sigma_B T^4}{\pi} = \int_0^\infty B_\nu(T) d\nu, \tag{3}$$

where

$$B_\nu(T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{kT}} - 1} \tag{4}$$

is the Planck function. (b) Try your integration with different sample sizes, N, and check the $1/\sqrt{N}$ law in MC integration. [30pts]

4. Many methods for solving nonlinear equations in one-dimension do not generalize directly to $n-$dimensions. However, the most popular and powerful method, Newton's method, does generalize. For a differentiable function $\boldsymbol{f}$ is bases on the truncated Taylor series

$$\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{s}) \sim \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x})\boldsymbol{s}, \tag{5}$$

where $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x})$ is the Jacobian matrix of $\boldsymbol{f}$, $\left\{J_f(x)\right\}_{ij} = \partial f_i(\boldsymbol{x})/\partial x_j$. If $\boldsymbol{s}$ satisfies the linear system $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x})\boldsymbol{s} = -\boldsymbol{f}(\boldsymbol{x})$, then $\boldsymbol{x}+\boldsymbol{s}$ is taken as an approximate zero of $\boldsymbol{f}$. In this sense, Newton's method replaces a system of nonlinear equations with a system of linear equations. The algorithm can be written as

```
// Newton's method for a system of nonlinear equations
x_0 = initial guess
for k = 0, 1, 2, ... {
   solve J_f(x_k) s_k = - f(x_k) for s_k // Compute Newton step
   x_{k+1} = xk + sk                     // Update solution
}
```

Modify the Fortran codes we developed in this lecture and the previous lecture (linear system) to solve the nonlinear system

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 - 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{6}$$

Hint: The solution vector $\boldsymbol{x} = [0,1]^T$. [30pts]