Statistics & Numerical Method, Problem Set 2 (due Oct. 29, 2024)

1. Poisson solver (24 pts)

We have introduced the Poisson equation in the lecture

$$\nabla^2\Phi = \rho \ , \tag{1}$$

where we have taken $4\pi G = 1$. In this problem, let us solve the Poisson equation in two dimensions on a uniform, 20×20 rectangular grid whose coordinates span over $([-10, 10], [-10, 10])$. This means the size of each grid cell is 1×1, and the value of that cell is defined at cell centers [e.g., $x_i = -10 + (i - 0.5)$, $y_j = -10 + (j - 0.5)$].

Using finite difference, this system of equations (400 in total!) can be expressed as

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = \rho_{i,j} \ , \ (i, j = 1, .., 20) \ . \tag{2}$$

In this problem, we take

$$\rho_{i,j} = \exp\left[-(x_i^2 + y_j^2)\right] \ . \tag{3}$$

Additional boundary conditions must also be prescribed, and we set $\Phi = 0$ at $x, y = \pm 10$. To implement this boundary condition, you will need to include one additional "ghost cell" surrounding your grid, and enforce, e.g., $\Phi_{0,j} = -\Phi_{1,j}$, $\Phi_{N+1,j} = -\Phi_{N,j}$ (and similar in two other boundaries), so that one finds $\Phi = 0$ at the grid boundary after averaging the last grid cell and the ghost cell. Note that this boundary condition must be enforced after every numerical step.

To solve the above equations, we ask you to write your own solver in your favorite programming language using the following iterative methods: a). Jacobi; b). Gauss-Seidel; c). Successive over-relaxation with relaxation factor $\omega = 1.5$; d). Steepest descent; e). Conjugate gradient.

1). How many iterations are required for each of the methods to converge to within $\|R\|_2 < 10^{-6}$? Show the converged solution. To better present your results, we further ask you to calculate the residual vector R (i.e. a 20×20 matrix) for up to 100 steps, and plot $\|R\|_2$ as a function of number of iterations. Please attach the original code that should be properly documented. (4 pts for each)

2). Further discuss the computation cost per iteration for each method. (4pts)

2. Fitting 1D image of stars (11 pts)

I have taken an image of bright stars from an amateur telescope. Unfortunately, I made a mistake in data reduction, with the result that I was only able to obtain 1-dimensional (but extremely well-sampled) "picture" of the sky. Undaunted, I ask you to help analyze the data...

Attached is a file `stars.dat` from the observation (in ascii; I being a merciful teacher didn't write them in the arcane format that astronomers really use). There are three columns; the position of the measurement x, the brightness y, and y's standard deviation dy (i.e., noise). I happen to know that the effective point-spread function (PSF, ϕ) when data was taken was a Gaussian $N(0, 1)$, i.e., $\phi = \exp(-x^2/2)/\sqrt{2\pi}$.

1). Plot the line data including 1σ error bars, and identify how many stars (N) there are in the 1D image. (2 pts)

2). Each star j is characterized by its location $x_{s,j}$ and its luminosity A_j, which is typically found by doing a (non-linear) least mean square problem to minimize

$$\chi^2(\boldsymbol{A}, \boldsymbol{x}_s) = \sum_i \frac{[y_i - \sum_j A_j \phi(x_i - x_{s,j})]^2}{dy_i^2} \ , \tag{4}$$

where summation is taken over all pixels. Is this function a convex function? Why? (2 pts)

3). Write a computer program to obtain these parameters (2N in total), attach your code and overplot the result. (7 pts) Your grade will depend on how much effort you make in solving this problem, and the full score is given to those who write their own optimization solvers. It is also OK to blindly use matured optimization packages, and in this case the full score is reduced to 4 pts at maximum.

Hint: the better you know about the problem, the more likely you can find a better optimization scheme. Here, the parameters $x_{s,j}$ and A_j play completely different roles, which may allow you to reduce the complexity and design more efficient/robust solvers.