CENTRAL FINITE DIFFERENCE FOR HOMOGENEOUS POISSON EQUATION IN 1D

ORSAN KILICER, TEXAS A&M UNIVERSITY

10/17/2019

Theoretical Part of the Problem

Poisson equation with homogeneous Dirichlet Boundary Condition can be defined as follows: Find $u \in C^2(\Omega)$ such that

$$\begin{cases} -\Delta u = f, & \text{in } \Omega, \\ 0, & u\big|_{\Gamma}. \end{cases}$$

where $f \in C(\Omega)$ and domain in our case is open interval. Also, in our case, $\Delta u = u_{xx}$. So, I am going to use central difference for the second derivative which can be defined by:

$$u''_m = \frac{u_{m+1} - 2u_m + u_{m-1}}{h^2} + O(h^2)$$

Also, since we have a homogeneous boundary condition, $u_0 = u_n = 0$ where n is the point number. The above formula contains three elements, we have to utilize artificial elements to be able to use f_0 and f_n. They will be u_{-1} and u_{n+1}. So, we obtain these equations:

$$\begin{aligned} -h^2 f_0 &= u_1 - 2u_0 + u_{-1} = u_1 + u_{-1} \\ -h^2 f_1 &= u_2 - 2u_1 + u_0 \ = u_2 - 2u_1 \\ -h^2 f_2 &= u_3 - 2u_2 + u_1 \\ &\vdots \\ -h^2 f_{n-2} &= u_{n-1} - 2u_{n-2} + u_{n-3} \\ -h^2 f_{n-1} &= u_n - 2u_{n-1} + u_{n-2} = -2u_{n-1} + u_{n-2} \\ -h^2 f_n &= u_{n+1} - 2u_n + u_{n-1} = -u_{n+1} + u_{n-1}. \end{aligned}$$

As you can notice between f_2 and f_{n-2}, the formula is same. So, we have this matrix-vector equation.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} u_{-1} \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \\ u_{n+1} \end{pmatrix} = -h^2 \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{pmatrix}$$

We take the inverse of the matrix and we obtain a solution for this matrix-vector equation. However, we also find u_{-1} and u_{n+1}. Instead of them, plug u_0 and u_n, respectively. So, we have the solution.
Generally, finding the inverse of a matrix is costly, for this situation iterative methods are generally used. However, classical iterative methods (Jacobi or Gauss Seidel) may not be good choices here because this matrix is not strictly diagonally dominant.

Theorem 1. *If the matrix is strictly diagonally dominant, i.e.*

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|$$

then Jacobi and Gauss-Seidel methods converge.

As we can notice the theorem's condition is not satisfied. Actually, I tried the Jacobi method, and the iteration did not converge. Modern iterative methods can be used here. However, for simplicity, I am using the inverse. Since it is a tridiagonal matrix, finding the inverse is easier. Also, I haven't used sparsity pattern for this matrix, again for the sake of simplicity.

Example

I choose 1001 points, $a = 0$, and $b = 1$. For the test function, I choose

$$u(x) = (x - a)(b - x)e^x.$$

So, f is forced to be

$$f(x) = -(-x^2 + (b + a - 4)x + (2 - a)b + 2a - 2)e^x.$$

Absolute max error is 2.16623755156e-07. So, as expected, it is $O(h^2)$ (Since $h = 10^{-3}$). Also, I draw the point-absolute error graph. It is here:

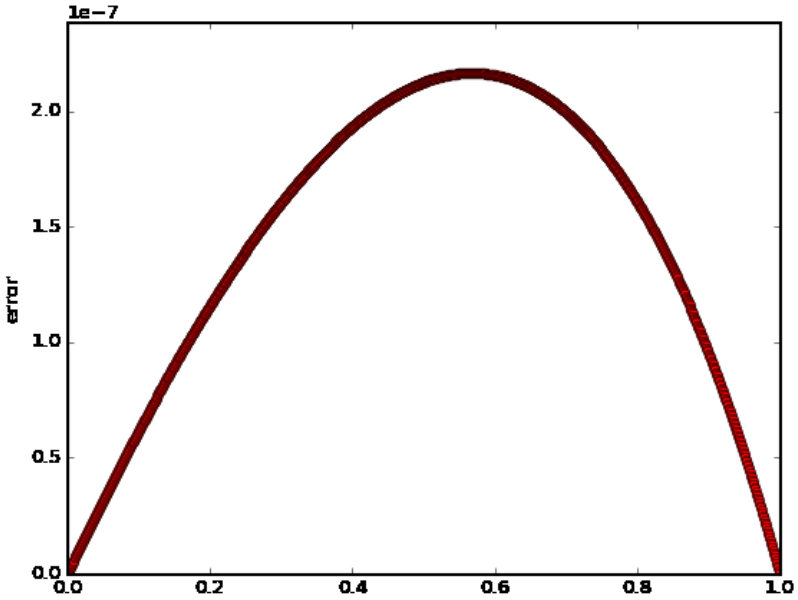

Figure 1: Point-Error Graph

As we can notice, error is increasing towards to the middle. Since we only have information at the boundary for the solution, this situation forces error to be getting bigger. This code is working, but when you take bigger intervals (i.e. $[0, 1000]$), the same test function did not work, because the numbers are getting bigger, so it fails. To avoid this problem, we have to scale the interval to $[0, 1]$. Now, it will work.