Numerical Solution of Elliptic Partial Differential Equations Using Agent-Based Simulation

Joe Dinius
University of Arizona
ECE508

November 1, 2009

Abstract

A technique for solving elliptic partial differential equations using the Netlogo programming language is developed. Netlogo provides a unique, flexible approach to solving problems in complex systems. However, the flexibility and design methodology of Netlogo make it an ideal choice for numerical evaluation of partial differential equations (PDEs) using finite differences. Both the Laplace and Poisson equations, two classic examples of elliptic partial differential equations, are solved using this technique. Results and verification of the method are presented.

Introduction

Partial differential equations (PDEs) are an important part of mathematical physics. These equations describe fundamental laws of nature in electromagnetics, mechanics, and fluid flow. Typically, closed-form solutions to PDEs either do not exist or are practically intractable. Therefore, it becomes important to develop numerical schemes for finding the solution.

A family of methods known as "finite differences" is most often employed to evaluate PDEs numerically. These methods all share a common thread: the value of the approximate solution at a grid point on a discretized domain is determined by the values of the approximate solution at neighboring points. How exactly this process works will be covered in subsequent sections. This solution method can be interpreted as an interaction between nearest neighbors, which makes the problem naturally extendable to agent-based simulation with Netlogo.

Netlogo provides an easy-to-use platform for investigating many problems in complex systems; however, the tool's flexibility provides the capability to solve deterministic PDEs numerically. This paper focuses on using this tool to approximate the solution of elliptic partial differential equations; specifically the Laplace and Poisson equations with constant boundary value.

Theory

Consider the PDE (a, b, c, d, e, f, g are real constants, u a twice-differentiable function in both x and y)

$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0.$$

The equation is called

- elliptic if $b^2 - 4ac < 0$
- parabolic if $b^2 - 4ac = 0$
- hyperbolic if $b^2 - 4ac > 0$.

The focus will be on elliptic PDEs, such as the Laplace equation or the Poisson equation. For generality, consider the slightly more complex Poisson equation

$$\begin{aligned} u_{xx} + u_{yy} &= f(x,y) \\ u(x,y) &= g(x,y) \text{ on } \partial\Omega \end{aligned}$$

where Ω represents the domain on which u is defined and $\partial\Omega$ is the boundary. For this paper, the discussion will be limited to $\Omega = [0,1] \times [0,1]$, the square of side length 1 originating from the origin, and

$$g(x,y) = \begin{cases} V_1 & (x,y) \in \{(0,y),(x,1),(1,y) | x,y \in [0,1]\} \\ V_2 & \text{otherwise} \end{cases}$$

. The finite differences approach gives the approximating solution to the above problem as [1]

$$u_{k,l} = \frac{1}{4}(u_{k-1,l} + u_{k,l+1} + u_{k+1,l} + u_{k,l-1}) - \frac{h^2}{4} f_{k,l}$$

where h is the length and width of each grid point. It should be clear that the approximate solution to the Laplace equation will come from $f_{k,l} = 0$ for all grid points. This simple nearest-neighbor interaction rule was incorporated into Netlogo. The pseudocode listed below demonstrates the process to solve the Poisson equation.

Pseudocode

1. Initialize boundary patches with values of g (from above) and all patches with values of f (any user-definable function)
2. If patches are not along the boundary, ask patches to update the value of u according to the relation given for $u_{k,l}$ above.
3. Run until solution converges

Results

Only a few results will be presented, as the finite difference method has been used for many years. Consider first the Laplace equation (where $f = 0$) with $V_1 = 0$ and $V_2 = 10$. The exact solution (see Appendix A) is

$$\begin{aligned} u(x,y) &= \sum_{n=1}^{\infty} A_n \sin(n\pi x)(\sinh(n\pi y) - \tanh(n\pi)\cosh(n\pi y)) \\ A_n &= 2\int_0^1 V_2 \sin(n\pi x)dx = \frac{2V_2}{n\pi}(1 - (-1)^n) \end{aligned}$$

Obviously, numerically we can only approximate this solution on a computer since the sum is over infinitely many terms. To view the solution, truncate to only the first 150 terms. The surface plot is seen in Figure 1.

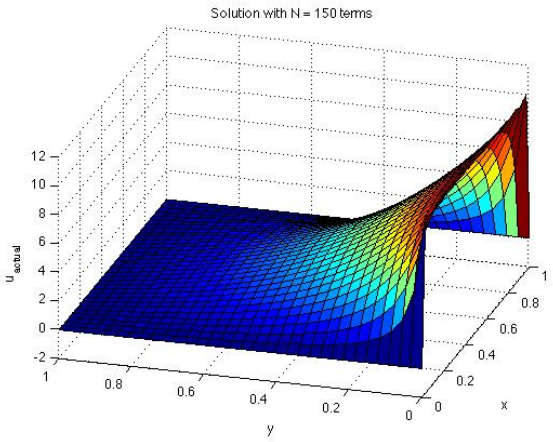

Figure 1: Analytical Solution with $N = 150$ terms

As a comparison, the absolute value of the difference between the numerical and analytical solutions is shown in Figure 2.

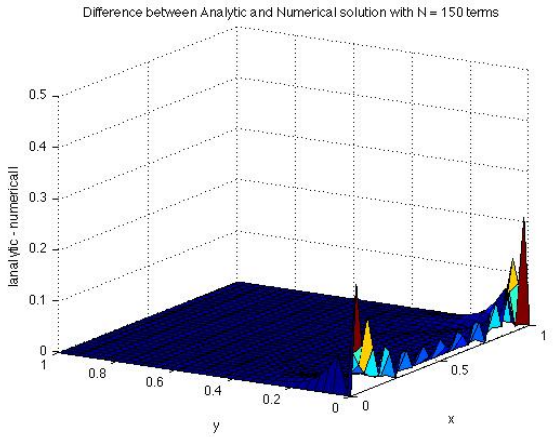

Figure 2: Difference Between Two Solutions- Numerical and Analytical

Figure 2 shows that the numerical solution is very close to the analytical one (less than 2 percent relative difference at maximum) and therefore the method is accurate. The "ringing" (oscillations) seen along the $y = 0$ line are due to the finite number of terms taken in the series representation of the actual solution. This exercise shows that the Netlogo technique is valid for solving PDEs of this type.

Now, consider a more complicated example. Let g be as defined above with the exception that $V_1 = 25$, but now let

$$f(x, y) = e^{x^2+y^2} \sin(8\pi x) \sin(30\pi y).$$

Finding a particular solution to this problem would be quite difficult, (and rather messy). However, using Netlogo, an approximation is found and shown in Figure 3.

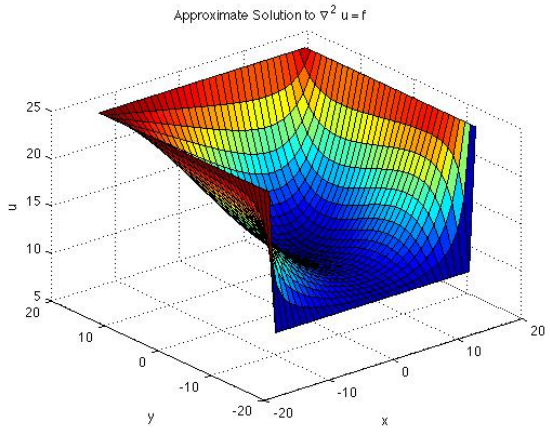

Figure 3: Approximate Solution to $u_{xx} + u_{yy} = f$

Conclusion

A new method for solving elliptic PDEs using agent-based simulation techniques and Netlogo was presented. Verification results were presented showing accuracy of the method versus analytical solutions. Additionally, results were presented showing the numerical solution for a particular example of the Poisson equation.

Appendix A: Exact Solution of Laplace's Equation

Consider Laplace's equation

$$\begin{aligned} u_{xx} + u_{yy} &= 0 \\ u &= g \text{ on } \partial\Omega \end{aligned}$$

where $\Omega = [0,1] \times [0,1]$ and

$$g(x,y) = \begin{cases} V_1 & (x,y) \in \{(0,y)|y \in [0,1]\} \\ V_2 & (x,y) \in \{(x,1)|x \in [0,1]\} \\ V_3 & (x,y) \in \{(1,y)|y \in [0,1]\} \\ V_4 & (x,y) \in \{(x,0)|x \in [0,1]\} \end{cases}$$

The equation is solved using separation-of-variables. After which, the problem reduces to

$$\begin{aligned} u(x,y) &= X(x)Y(y) \\ X'' &= \lambda^2 X \\ Y'' &= -\lambda^2 Y \end{aligned}$$

where λ is the separation constant. The negative sign can be interchanged between the equations for X and Y to match the boundary conditions. Due to the principle of superposition for linear problems, the problem can be broken up into four pieces; where $g = V_1, V_2, V_3$ and V_4, all other boundary conditions will be 0. Call these solutions u_1, u_2, u_3 and u_4 respectively. In the end, $u = u_1 + u_2 + u_3 + u_4$. Begin with u_1. The solutions to the above ordinary differential equations with this boundary condition are

$$\begin{aligned} X(x) &= A\cosh(\lambda x) + B\sinh(\lambda x) \\ Y(y) &= C\sin(\lambda y) + D\cos(\lambda y) \end{aligned}$$

Apply the boundary conditions

$$\begin{aligned} u_1(0,y) &= V_1 = A(C\sin(\lambda y) + D\cos(\lambda y)) \\ u_1(x,0) &= 0 = D(A\cosh(\lambda x) + B\sinh(\lambda x)) \\ u_1(1,y) &= 0 = (A\cosh(\lambda) + B\sinh(\lambda))(C\sin(\lambda y) + D\cos(\lambda y)) \\ u_1(x,1) &= 0 = (A\cosh(\lambda x) + B\sinh(\lambda x))(C\sin(\lambda) + D\cos(\lambda)) \end{aligned}$$

From these conditions, it follows that

$$\begin{aligned} \lambda &= n\pi \\ D &= 0 \\ A &= -B\tanh(\lambda) \\ V_1 &= AC\sin(\lambda y) \end{aligned}$$

This last relationship can not hold true for any single value of n. Therefore, a series solution must be applied to match the boundary conditions. The solution to u_1 is then

$$\begin{aligned} u_1(x,y) &= \sum_{n=1}^{\infty} \alpha_n \sin(n\pi y)(\sinh(n\pi x) - \tanh(n\pi)\cosh(n\pi x)) \\ \alpha_n &= 2\int_0^1 V_1 \sin(n\pi y)dy = \frac{2V_1}{n\pi}(1-(-1)^n) \end{aligned}$$

Apply the same methods for u_2, u_3, u_4 and the entire solution is

$$\begin{aligned} u(x,y) = u_1(x,y) + u_2(x,y) + u_3(x,y) + u_4(x,y) = & \sum_{n=1}^{\infty} \alpha_n \sin(n\pi y)(\sinh(n\pi x) - \tanh(n\pi)\cosh(n\pi x)) \\ & + \beta_n \sin(n\pi x)\sinh(n\pi y) \\ & + \gamma_n \sin(n\pi y)\sinh(n\pi x) \\ & + \delta_n \sin(n\pi x)(\sinh(n\pi y) - \tanh(n\pi)\cosh(n\pi y)) \end{aligned}$$

where the coefficients are as follows

$$\begin{aligned}
\alpha_n &= \frac{2V_1}{n\pi}(1-(-1)^n) \\
\beta_n &= \frac{2V_2}{n\pi}(1-(-1)^n) \\
\gamma_n &= \frac{2V_3}{n\pi}(1-(-1)^n) \\
\delta_n &= \frac{2V_4}{n\pi}(1-(-1)^n)
\end{aligned}$$

References

[1] Restrepo, Juan M. *Introduction to Scientific Computing Part II-Draft.* July 2001. Available http://www.physics.arizona.edu/ restrepo/475B/Notes/source.pdf