PURDUE UNIVERSITY

Numerical Methods for Some Eigenvalue Problems

OVERVIEW

For this project, we attempted to calculate the first eigenvalue of the Monge-Ampère (MA) equation on the unit ball using numerical methods. The MA equation is a fully non-linear PDE where no eigenvalues are known. However, on a unit ball, symmetry reduces the equation to a non-linear ODE in which we can use iterative approaches to compute the eigenvalues for.

Introduction

The Monge Ampère (MA) operator is defined as the determinant of the Hessian of φ, or $\mathrm{MA}(\varphi) = \det\left(\nabla^2 \varphi\right)$. It has applications in many fields, such as convex geometry and optimal transport problem.

Setup

Using the radial symmetry of the ball, we reduce the eigenvalue equation into an ODE; we write MA_r for this radial reduction of the operator, which is a non-linear, second order ODE on [0, 1]. To implement numerical methods, we consider the linearization of $MA_r^{1/n}$ at a strictly convex function φ $-\ L_\varphi\colon C^\infty(B) \to C^\infty(B)$.

Using the linearization, we obtain: $MA_r^{1/n}(\varphi) = L_\varphi \varphi$. The eigenvalue problem is now

$$\begin{cases} L_\varphi^D \varphi = -\lambda\varphi \\ \varphi(0) = -1 \\ \varphi(1) = \varphi'(0) = 0 \end{cases}$$

MatLab BVP4c

BVP4c is a boundary value problem solver for ODEs that uses a 3-stage Lobatto IIIa Runge-Kutta method. Like many numerical methods, providing an accurate initial guess is crucial for its convergence. We select a mesh size m and define a set of points at $r^2 - 1$ where $r = 0, 1/m, 2/m, \dots, (m-1)/m,\ 1$ as a solution guess, $\lambda = 2$ for n = 1, and $\lambda_n \approx \lambda_{n-1}$. Determining the mesh size proves to be difficult, so we developed a program that continuously tries different mesh sizes until bvp4c converges to a solution.

After graphing the eigenvalues we obtained (Figure 1), the points appear to follow a four-parameter logistic regression with the equation

$$y = 3.994 - \frac{2.094}{1+(x/3.216)^{0.8487}}$$

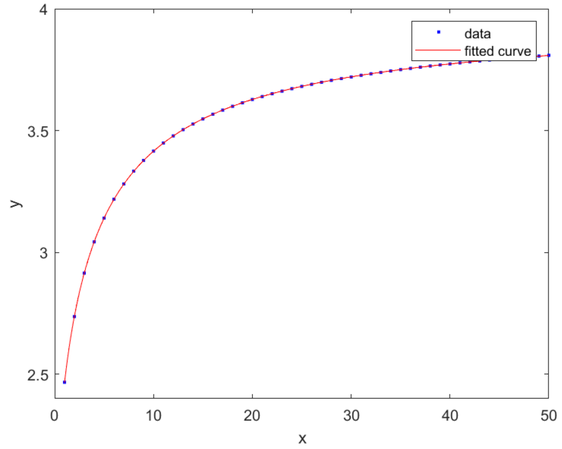
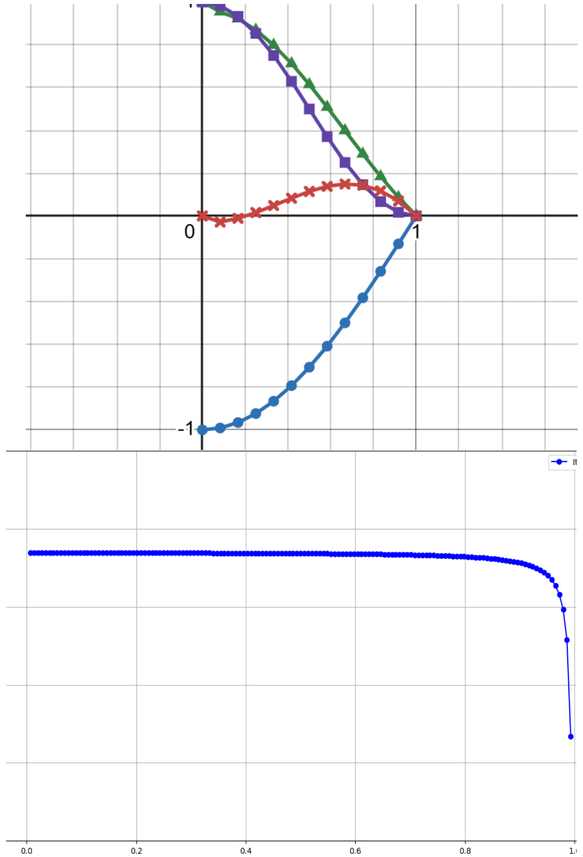


Figure 1: Eigenvalues from n=1 to n=50 with a fitted curve

Gradient Descent

Another approach taken was to solve for the first eigenvalue using gradient descent. By minimizing the distance between the left- and right-hand sides of the MA equation, we approximate a solution φ, and we can then extract an estimation for λ.



Convex Programming

A convex program is a type of optimization problem that consists of variables, convex constraints involving those variables, and a convex objective function. The objective function will minimize or maximize a function of the variables subject to all the constraints. There exist robust algorithms to solve convex programs. In this project, we tried to write our problem as a convex program. Theoretically, we found out it was possible to write it. Our eigenvalue problem looks like this

$$\left(\frac{u'}{r}\right)^{n-1} u'' = \lambda^n (-u)^n$$
$$u(0) = -1$$
$$u(1) = u'(0) = 0.$$

We create this convex program by creating a variable for u,u',u'',and λ. While this function is not convex since it involves odd powers and products of variables, it is convex in the logarithm of the variables,

$$(n-1)\log\left(\frac{u'}{r}\right) + \log(u'') = n\log\lambda + n\log(-u)$$

Even though the log isn't convex, we can create variables equal to the log of other ones by making use of our objective function and the exponential cone.

CONCLUSIONS

We have observed a pattern for the eigenvalues we have calculated. Further research is needed to verify the accuracy of the eigenvalues beyond n=50. Improvement on the algorithm may also be necessary to increase the speed of the program as n grows.

CONTRIBUTORS: Ronit Mehta, Rishika Ramakrishnan, Truman Mohr, Michelle Jyi

Special thanks to our mentor Professor Nick McCleerey and graduate mentor Shay Phagan for their support and guidance on this project