Imperial College London

Solving Monge-Ampere Equation for mesh movement under Hessain Informed Monitor Function

Trial and analysis on different gradient recovery techniques

Author

Yongqi Li

CID: 02261717

Supervised by

Prof. Matthew Piggott

Dr Stephan Kramer

A Thesis submitted in fulfillment of requirements for the degree of
Master of Science in Applied Computational Science and Engineering

Department of Earth Science and Engineering
Imperial College London

2023

Abstract

0.1 Motivation

Mesh adaptivity is a general technique used in computational physics to adapt the mesh during a computational simulation, especially in the context of solving PDEs over complex geometries or evolving domains.

The aim of mesh adaptivity is to focus resolution where it is needed. A feasible approach to adapting a mesh is the so-called "mesh movement" (r adaptivity). It takes a fixed number of mesh points and the mesh connectivity remains unchanged. [1].

In this paper, specific focus is given to mesh movment methods based on a scalar monitor function through equidistribution. In combination with an optimal transport condition, this results in a Monge–Ampère equation for a scalar mesh potential.**McRaeCotterBudd:16**. A foundational concept in this method is distinguishing between two domains: the physical domain, represented as Ω_P, and the computational domain, Ω_C. In essence, the algorithm tries to establish a connection between these two domains, formulated as:

$$\mathbf{x} : \Omega_C \to \Omega_P.$$

0.2 Aim of the project

It is difficult to solve a (fully nonlinear, elliptic) Monge–Ampère equation. Therefore, in this paper, 2 specific algorithmic approaches for solving the Monge-Ampère equation has been investigated: `Pseudo-Time Relaxation Method` and `quasi-Newton method`. Furthermore, in order to compare `Pseudo-Time Relaxation Method` and `quasi-Newton method` and have better understandings on their mathematical properties, numerical experiments under different conditions have been conducted. It is worth extending current mesh movments methods and explore new approaches.

Fortunately, a combination of the two algorithms was found and it proves itself with some nice properties.

As a crucial part for solving the monge-ampere equation, gradient recovery methods are often used to calculate the gradient of scalar field, which is represented in a discrete sense, especially in the context of finite element discretisations. In this parper, two most popular methods have been introduced: L2 gradient recovery and patch-based recovery by Zienkiewicz and Zhu. Two methods were also analyzed quantitatively and both methods have their own application scenarios.

All finite element related experiment are conducted using the automated system, **Firedrake**.

Acknowledgments

0.3 Statement of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced.

0.4 Appreciateions

The author would like to thank Prof. Matthew Piggott and Dr Stephan Kramer for their patience, support and ideas. The author also appreciate Joe Wallwork for his previous work towards this topic.

Contents

1

Mathematical background and Numerical Methods for Solving the Monge-Ampère Equation

Contents

1

1.1 Equidistribution principles

As mentioned in the abstract, mesh movement will create a map from the computational space (origin) to physical space (new). According to the optimal transport constraint [2]:

$$\mathbf{x} = \boldsymbol{\xi} + \nabla_{\boldsymbol{\xi}}\phi,$$

Moreover the coordinate transformation will lead to changes in cell volume, which is provided by the following equation:

$$\det(J(\xi)) = \det(\nabla \mathbf{x}(\boldsymbol{\xi})) = \det(I + H(\phi))$$

Consequently, to achieve equidistribution in line with a monitor function, the updated mesh positions have to follow the given Monge-Ampère equation:

$$m(x)\det(I + H(\phi)) = \theta$$

1.2 Define an Appropriate Monitor Function

Based on detailed theorem in **[McRae et al., 2018]**[3], selecting an appropriate monitor function M is challenging and problem-specific. The monitor function in this project is as:

$$m(x) = 1 + \beta \frac{H_{\text{norm}}}{H_{\text{norm\_max}}}$$

where $H_{\text{norm}} = \sqrt{(H(u) \cdot H(u))}$ is the Frobenius norm $\|H(u)\|_F$ of the Hessian matrix. This monitor function can move the mesh and clusters points in locations of high changes in the gradient of the solution. (This will be justified in the following parts)

1.3 Solving Monge-Ampère Equation with Pseudo-Time Relaxation Method

1.3.1 Discretizing the Monge-Ampère equation

The pseudo-time relaxation method transforms the elliptic equation into a parabolic form by introducing a pseudo-time variable τ

$$-\frac{\partial}{\partial \tau}\Delta\phi = m(x)\det(I + H(\phi)) - \theta$$

Here, $\Delta\phi$ is the Laplacian of ϕ, and Forward Euler is used for the pseudo-time integration. After explicit Euler discretization, the final expression is derived:

$$\Delta\phi^{n+1} - \Delta\phi^{n} = -\Delta\tau\left(m(x)\det(I + H(\phi^{n})) - \theta\right)$$

This equation indicates a linear convergence. For each iteration, change in the solution has a direct relaxation to the residual on the R.H.S.

1.3.2 Mixed Finite Element Method

To derive the weak form, a practical way, Mixed Finite Element Method is initiated by **[McRae et al., 2018]**[3]. The core idea is to substitute $H(\phi))$ with σ, the discrete Hessian [4], such that:

$$\int \Delta\phi^{n+1}\psi\, dx - \int \Delta\phi^{n}\psi\, dx = -\Delta\tau \int \left(m(x)\det(I + \sigma^{n})) - \theta\right)\psi\, dx$$

with L_2 projected σ:

$$\int_{\Omega} \tau : \sigma\, dx = -\int_{\Omega} \nabla\cdot\tau\cdot\nabla\phi\, dx + \int_{\partial\Omega} (\tau_{01}n_1\phi_{,x} + \tau_{10}n_0\phi_{,y})\, ds$$

With the expression of normalisation constant:

$$\theta = \frac{\int_{\Omega} m(x)\det(I + H(\phi))\, dx}{\int_{\Omega} dx}$$

1

The above variational problems will lead to the convergence of the nonlinear problem if $\tau << 1$. However, lacking a good a-priori estimate for 'optimal' τ is a key problem and will cause the collapse of experiment on high resolution meshes.

1.4 Quasi-Newton method

Another way to solve nonlinear elliptic Monge-Ampère equation is quasi-Newton method, which aims to iteratively find a solution to the nonlinear equation by approximating the the Jacobian matrix.

To start with the problem, residual term R will be expressed as the following weak form problem:

$$F(\phi, \sigma) = \int \tau : \sigma \, \mathrm{d}x + \int \nabla \cdot \tau \cdot \nabla \phi \, \mathrm{d}x - \int \psi(m(\mathbf{x}) \det(\underline{\mathbf{I}} + \sigma) - \theta) \, \mathrm{d}x.$$

The next step is to implement a Newton's method to address this issue. However, given the Newton steps might not preserve convexity[5] , quasi-Newton method is introduced.

1.4.1 Approximation of Jacobian Matrix

The Jacobian is derived by partially linearizing equation and represented by:

$$J_p = \int_\Omega \left(\tau : \sigma + \phi\psi + \nabla\phi \cdot \nabla\psi\right) \, d\Omega$$

This bilinear form ignores the dependence of m on ϕ.

Consequently, the whole process will end up with:

$$\begin{aligned} &\int_\Omega \tau \cdot \delta\sigma \, d\Omega + \int_\Omega \nabla \cdot \tau \cdot \nabla \delta\phi \, d\Omega \\ &- \int_\Omega v \, m(x) \left(\delta\sigma_{11} \left(1 + \sigma_{22}^n\right) + \left(1 + \sigma_{11}^n\right) \delta\sigma_{22} - \delta\sigma_{12}\sigma_{21}^n - \sigma_{12}^n \delta\sigma_{21}\right) \, d\Omega. \end{aligned}$$

Indeed, with the help of firedrake, we can directly set the nonlinear problem as:

$$F = \tau \cdot \sigma + \nabla\tau \cdot \nabla\phi - (\tau_{01} n_1 \partial_{x_1}\phi + \tau_{10} n_0 \partial_{x_0}\phi) - \psi(m(x) \det(\mathbf{I} + \sigma) - \theta)$$

1.Diffusion Term: $\tau \cdot \sigma$, which spreads out or "diffuses" changes in the mesh.

2. Gradient Term: $\nabla\tau \cdot \nabla\phi$ couples changes in the mesh (via ϕ) to the test function's gradient, ensuring continuity in the mesh motion.

3. Boundary Term: $\tau_{01}n_1\partial_{x_1}\phi + \tau_{10}n_0\partial_{x_0}\phi$ applies boundary conditions to the mesh motion. Essentially, it constrains mesh movement at the domain boundaries.

Code Demo is provided below:

```
#Initial settings
...
F = ufl.inner(tau, sigma)*self.dx \
    + ufl.dot(ufl.div(tau), ufl.grad(phi))*self.dx \
    - (tau[0, 1]*n[1]*phi.dx(0) + tau[1, 0]*n[0]*phi.dx(1))*self.ds \
    - psi*(self.monitor*ufl.det(I + sigma) - self.theta)*self.dx
...
# Custom preconditioner
Jp = ufl.inner(tau, sigma)*self.dx \
    + phi*psi*self.dx \
    + ufl.inner(ufl.grad(phi), ufl.grad(psi))*self.dx
...
# Setup the variational problem
problem = NonlinearVariationalProblem(F, self.phisigma, Jp=Jp)
nullspace = MixedVectorSpaceBasis(self.V, [VectorSpaceBasis(constant=True), self.V.sub(1)])
sp = ... #settings for solver para
self._equidistributor = NonlinearVariationalSolver(problem, ...)
```

1.5 Experiments under Different Context

To start the experiment, the following PDE is considered:

$$-\Delta u = f \quad \text{in } \Omega,$$

$$u = 2 \text{ on } x = 0,$$

where Ω is the unit square domain $[0,1] \times [0,1]$, and $f = \sin(\pi x)\sin(2\pi y)$.

1

The reason for poisson equations is that they are simple and basic: Laplacian of the solution u can be derived correctly, which can be used a measure of the curvature or concavity. Also, the Laplacian can be seen as a sum of the diagonal elements of the Hessian: $\Delta u = \text{trace}(H(u))$.

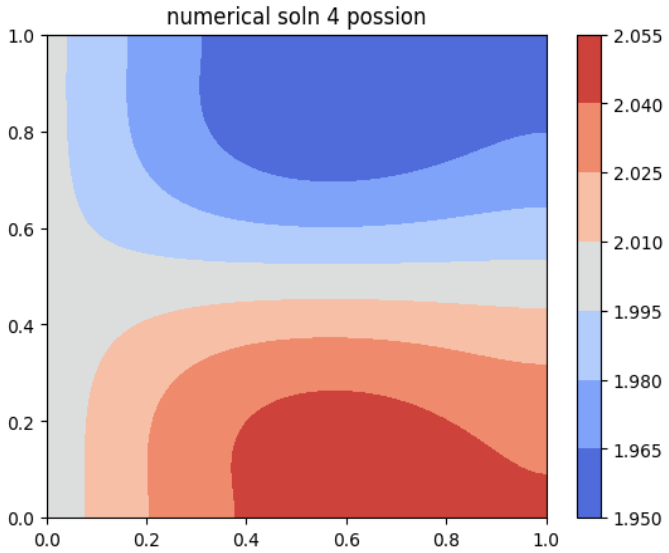


Figure 1.1: Visualization for the solution, high curvature regions will appear with closely spaced contour lines

1.5.1 Trial on Relaxation method

[Ramani and Shkoller, 2023][6] pointed out that the goal of integrating the parabolized Monge-Ampère equation is to make the system to reach a steady state. Hence, pseudotimestep size Δt is of importance in determining the convergence efficiency and stability of the iterations. Improper timestep can cause unexpected issues: smaller time steps indicates more iterations and run-time while a larger timestep could overshoot the solution, causing oscillations or divergence. This can be especially problematic in finer meshes.

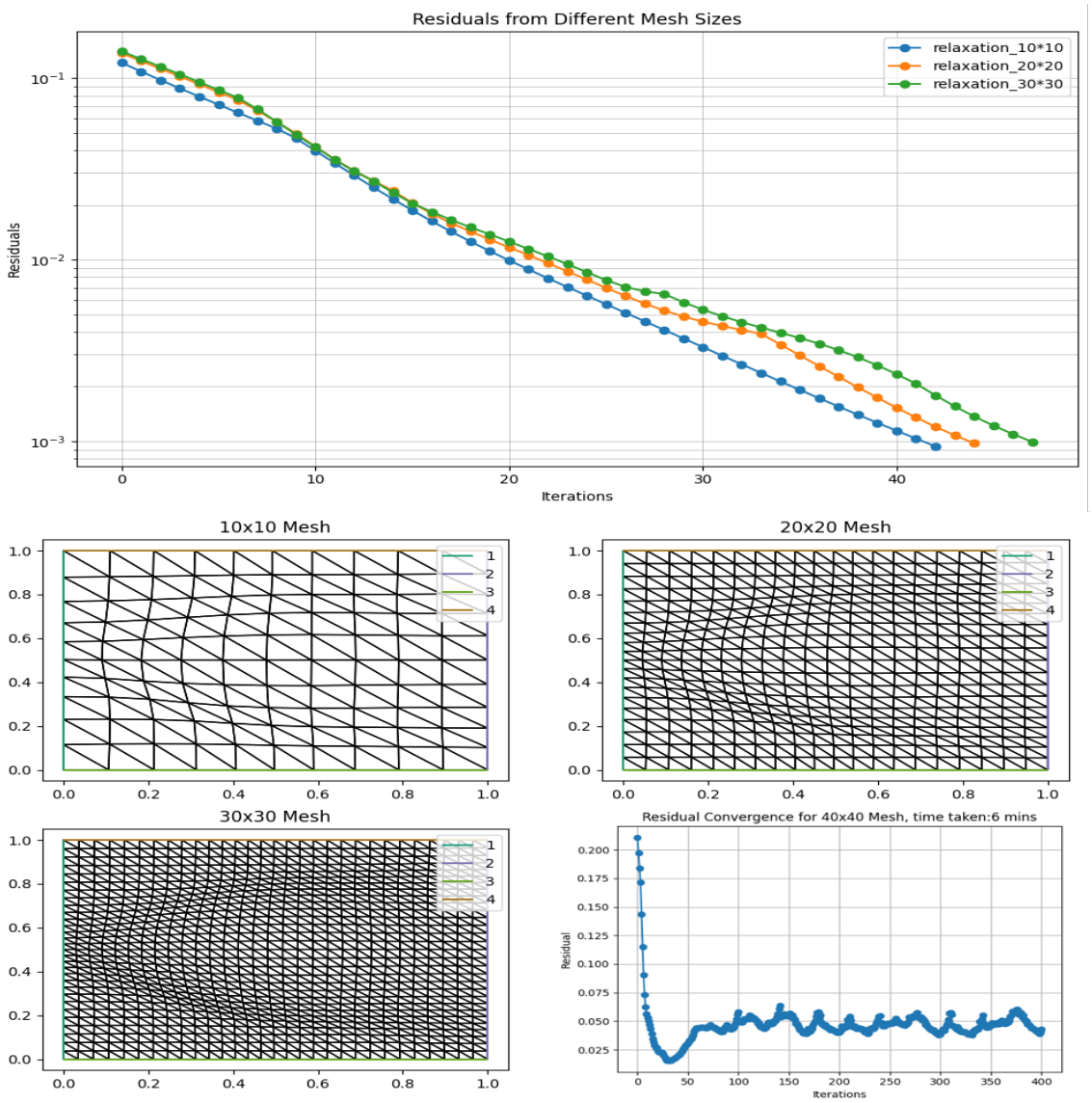


Figure 1.2: Visualization for adapted meshes and residuals with mesh size 10×10 to 40×40 with $\Delta t = 0.1$

The forward Euler method is explicit and has known stability constraints. The CFL (Courant–Friedrichs–Lewy) condition is necessary for convergence [7]. According to **[Wallwork, 2021]**'s finding [8], it is challenging to decide the optimal value for Δt due to $m(x)$. Nevertheless, absence of techniques to predict it beforehand forces us to select $\Delta t = 0.1$ for practical applications. Experiments were first conducted on meshes with sizes ranging from 10×10 to 40×40.

Refined meshes after movement justify the proper choice of $m(x)$ since it captures useful information from Hessian and reflects the curvature of the solution field. However, the divergence on the

1

40x40 mesh means that the pseudotimestep $\Delta\tau = 0.1$ may be too large for such a refined mesh, especially if it violates the CFL condition. After adjusting the pseudotimestep to $\Delta t = 0.04$, the problem is solved and a convincing adapted mesh is produced. However, $\Delta t = 0.04$ will be invalid again when mesh gets denser.)

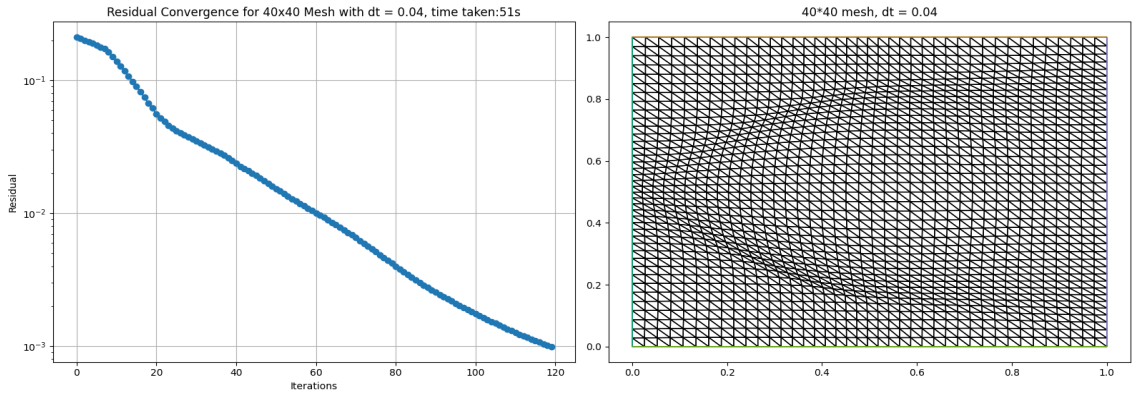


Figure 1.3: Visualization for the residual and adapted mesh with $\Delta t = 0.04$

Furthermore, the convergence of residual of relaxation method, as shown in the line plots(y axes log scaled), is basically linear, which follows previous expectation.

1.5.2 Trial on Quasi-Newton method

The conclusion is that quasi-Newton method would be a robust choice. Firstly, according to the statment of **[Feng and Neilan, 2009]** [9], the determinant in the Monge-Ampère equation introduces heavy nonlinearity and the cost of directly calculating the Jacobian is high. Quasi-Newton method uses an approximation to the Jacobian, updated iteratively based on the search direction and the changes in the residual. **Firedrake** and **PETSc** provide solvers like **NonlinearVariationalSolver** to employ a quasi-Newton method with line search, determine search direction and update. The documentation from Firedrake[10] assists us to have a deeper insight: line search algorithm will reduce the L2-norm of the residual during each nonlinear step. This property offers an adaptive approach to stepping through the solution space, especially crucial given the inherent nonlinearity of the Monge-Ampère equation.

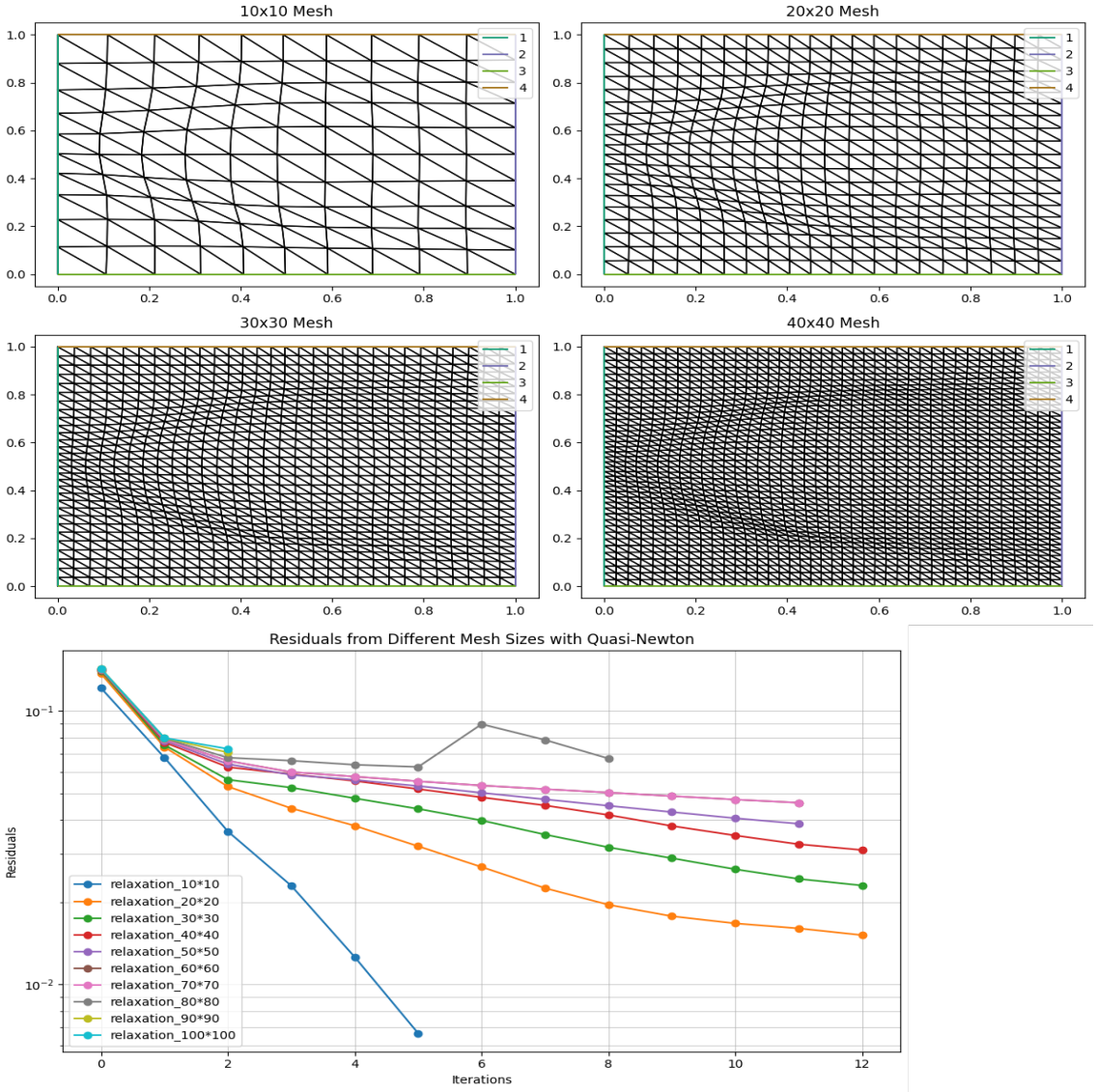


Figure 1.4: Visualization for the residual and part of adapted mesh for quasi-newton method

After a number of repeated experiments, final results have shown the robustness of Quasi-Newton method. In a computational study involving the **UnitSquareMesh**, an interesting observation was made regarding its convergence behavior. Specifically, for mesh resolutions spanning from 20×20 to 70×70, the typical convergence required is 11 to 12 iterations. Remarkably, as the mesh resolution is increased to 80×80 and beyond, the required iterations for convergence notably decrease. For resolutions of 90×90, 100×100, and even 200×200, convergence is achieved in merely 1 to 2 iterations. When the mesh becomes finer to some extent, the number of iterations required to converge drastically reduces. Several reasons may explain this phenomenon:

1

1. **Jacobian Approximations**:

   As metioned, quasi-Newton methods do not compute the full Jacobian but rather approximate it. On a coarse mesh, the complexity of the problem and the nonlinearities could be such that the Jacobian approximation needs several iterations to become 'useful'. After refining the mesh, the discrete representation of the Monge-Ampère equation gets closer to the true continuous version of the equation, and the problem could become more 'linear-like' on a local scale, meaning the initial Jacobian approximation might already be decent, and the quasi-Newton method can make better progress.

2. **Effect of Monitor Function**:

   As the mesh is refined, the values of H_{norm} across the domain may become more uniform or smooth. This could lead to the monitor function having less variability, which in turn might make the Monge-Ampère problem easier to solve iteratively.

1.5.3 Comparison between Relaxation Method and Quasi-Newton Method

The relaxation method and quasi-Newton methods are both iterative techniques, however, they approach the problem in different manners and come with their own strengths and weaknesses. (All comparisons are made under the assumption that pseudotimestep = 0.1)

Quasi-Newton method use fewer iterations to converge than the relaxation method, while each iteration of the quasi-Newton method is more computational expensive than an iteration of the relaxation method.

1. **Relaxation**: The relaxation method sets up two linear variational solvers: pseudo-timestepper (to update ϕ) and another for the equidistributor (to calculate σ).

   (a) **Pseudo-timestepper**: The weak formulation:

$$a(\phi, \psi) = \int_\Omega \nabla\psi \cdot \nabla\phi \, dx$$

$$L(\psi) = \int_\Omega \nabla\psi \cdot \nabla\phi^{old} \, dx + \Delta t\psi \times \text{residual} \, dx$$

   Here, the linear system Ax = b is solved using Conjugate Gradient method with the GAMG preconditioner.

(b) **Equidistributor**: For σ, the weak form is:

$$a(\sigma, \tau) = \int_\Omega \tau : \sigma \, dx$$

$$L(\tau) = -\int_\Omega \nabla \cdot \tau \cdot \nabla\phi \, dx + \int_{\partial\Omega} (\tau_{01} n_1 \phi_{,x} + \tau_{10} n_0 \phi_{,y}) \, ds$$

This system is solved using the Conjugate Gradient method with the block Jacobi preconditioner and the ILU preconditioner for subproblems.

Solving for ϕ (Pseudo-timestepper) involves solving a Poisson-like equation, which can be done in $O(N \log N)$ operations, where N is the number of nodes, using the CG method. The L2 projection for σ (Equidistributor) also involves solving a similar equation, hence similar complexity.

2. **Quasi-Newton**:

The "equidistributor" sets up the core of the quasi-Newton optimization.:

$$F = \int \tau : \sigma \, dx + \int \nabla\cdot\tau\nabla\phi \, dx - \int (\tau_{01} n_1 \phi_{,x} + \tau_{10} n_0 \phi_{,y}) \, ds - \int \psi \, (monitor \, det(I + \sigma) - \theta) \, dx$$

(a) **Jacobian Approximation**: The approximation of Jacobian:

$$Jp = \tau \cdot \sigma + \phi\psi + \nabla\phi \cdot \nabla\psi$$

represents the linearization of the system around the current state. This is the main reason for the effectiveness of the quasi-Newton method: continuously updating approximation to the Jacobian allows quasi-Newton to quickly correct its direction based on the current residual, potentially converging to the solution in fewer iterations than the relaxation method.

(b) **Solver**: The 'NonlinearVariationalSolver' uses F as the main residual form and J_p as the preconditioner under given convergence criteria. This solver internally uses the quasi-Newton method with the provided residual and Jacobian approximation.

Each of these iterations, however, involves more computational work, especially due to the handling of the Jacobian approximation and the associated linear system solves, making each iteration potentially more expensive than simpler iterative methods. Each iteration of the Quasi-Newton method might involve either an $O(N^2)$ or even $O(N^3)$ operation.

1

(a) **Jacobian Inversion and Linear Solve [3]**: Generalized Minimum RESidual(GMRES) method is used to solve the linear system, which is an iterative method and does not have the cubic complexity of direct methods like Gaussian elimination.

The preconditioner is "fieldsplit", and further settings for fieldsplit indicate the use of GAMG (Geometric Algebraic Multigrid) for one of the fields and ILU (Incomplete LU decomposition) for both fields. The complexity of ILU depends on the level of fill-in but can be $O(N^2)$ for sparse matrices. Multigrid methods, when efficient, are near $O(N)$ complexity for each V-cycle, but the constant factors can be significant.

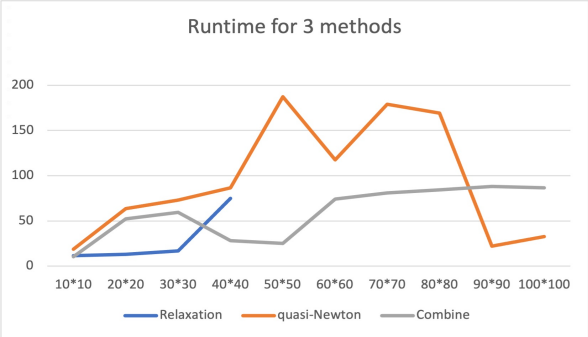
(b) **Matrix Inversion or Linear System Solution**: Linear systems of the form $Jx = b$ are usually solved in practice . Depending on the method used (e.g., Gaussian elimination, LU decomposition, etc.), this can also be an $O(N^3)$ operation for dense matrices.

Criteria	**Relaxation Method**	**Quasi-Newton Method**
Computational Cost	Lower per iteration	Higher per iteration due to matrix operations
Convergence Speed	Typically has linear convergence	Has superlinear convergence, which can be faster than the relaxation method, especially for well-behaved functions.
Nature of Iteration	Uses a pseudo-time approach to evolve the solution iteratively towards the steady state.	Directly iterates on the solution using approximations to the Jacobian matrix
Robustness	sensitive to the choice for suitable pseudotime step size.	A more general method that can be applied to a wide variety of nonlinear equations and optimization problems.

Table 1.1: Comparison between Relaxation method and Quasi-Newton method

1.6 Combination of relaxation and quasi-newton

Recognizing the strengths and weaknesses of both approaches, the combined method is proposed. This hybrid strategy begins with the relaxation method, leveraging its simplicity and broad applicability. If convergence is achieved smoothly, the process ends. However, should the relaxation method show signs of divergence, the algorithm will switch to the quasi-Newton method. By doing so, it capitalizes on the quasi-Newton's capability to handle challenging scenarios and its rapid convergence properties. (The combination of two methods has been implemented. Please refer to the repository for more details.) Several numerical experiment has been conducted on this new framework and the result we have achieved is quite promising.



Mesh Resolution	Relaxation	quasi-Newton	Combine
10*10	11.5303812	18.65015769	10.30899072
20*20	13.19458127	63.46197081	52.27323437
30*30	16.99916911	73.01738167	59.28918099
40*40	74.99526191	86.66376925	28.2884469
50*50	#NAN	187.3901703	25.05173016
60*60	#NAN	117.60198	74.25024176
70*70	#NAN	179.0556746	80.80761123
80*80	#NAN	169.0873756	84.21907568
90*90	#NAN	22.27098703	88.15374207
100*100	#NAN	32.51768875	86.59054542

Figure 1.5: runtime for different methods, improved efficiency of combined methods can be seen from the plot and table

In essence, this combined approach offers a dynamic solution pathway, adaptable to the problem's complexity. Starting with the straightforward relaxation method, it ensures computational efficiency. But when faced with potential divergence, the switch to the Quasi-Newton method acts as a safety net, ensuring robustness. This marriage of methods aims to offer the best of both worlds, presenting a promising avenue for tackling the intricacies of nonlinear equations like the Monge-Ampère.

Another potential improvement on this combination is that when the value of "Residual" is approaching the "rtol", the convergence speed would slow down when using pure relaxation method. Therefore, it is feasible to create a metric to measure the divergence speed and activate "switch method" when convergence speed is slow. Moreover, pure quasi-Newton can achieve dramatic convergence speed on dense meshes. According to this property, we may also add mesh size detection function to use pure quasi-Newton given dense meshes.

1

2

Gradient Recovery Techniques

Contents

Solving Monge-Ampère equation poses set of challenges. The evaluation of gradients and higher-order derivatives are pivotal to the equation's structure and solution methods. Gradient recovery methods enables efficient and accurate extraction of gradient information from discrete solutions.

The relaxation method uses a mixed finite element approach, which introduces additional variables σ to discrete Hessian and update iteratively.

1. **Gradient Recovery in the Process**: The gradient of potential ϕ are critical for updating the mesh coordinates and its corresponding Hessian in each iteration[11]. Before relaxation iterations begin, the gradient of the initial ϕ field is recovered. The `pseudotimestepper` will update ϕ based on the parabolized equation involving its recovered gradient and the `equidistributor` uses the recoverd gradient of ϕ to update the tensor σ. If these gradients are obtained directly from the FEM (which is typical), they might be discontinuous across elements. Gradient recovery techniques can provide a smoothed, post-processed version of these gradients.

Meanwhile, in the process of the quasi-Newton method, gradient recovery provides necessary direction and magnitude for mesh node movements.

1. **Initialization and Update of Monitor Function**: Before the quasi-Newton solver starts, the gradient of ϕ is recovered the `update_monitor` function is called as a `pre_function_callback`. In this callback, the L2 projector is employed to compute the gradient of ϕ. The resultant gradient gives the mesh movement.

2. **Updating Solution and Monitor**: After each iteration of the quasi-Newton method, the solution is updated, which in turn affects the potential ϕ.As the solution and ϕ change, it is natural to re-compute its recovered gradient to get the latest mesh movement direction.

In this chapter two gradient recovery methods have been introduced: L2 gradient recovery method and Zienkiewicz–Zhu Error Recovery Technique (ZZ recovery). As a tool for solving Monge–Ampère equation, several properties of recovery methods should be considered: the accuracy, the efficiency and what kind of scenario do they work.

As for the error analysis, `'errornorm'`[12] function in **'Firedrake'** is used in this chapter, which will compute the error:`e = u - u_h`in the specified norm. Here, `norm_type` was set to default value `"l2"`.

$$||\nabla u_{exact} - \nabla u_{rec}||_{l_2} = \sqrt{\sum_i (\nabla u_{exact} - \nabla u_{rec})^2}$$

2.1 L2 gradient recovery

The key idea of L2 gradient recovery is to use finite element methods to obtain a piecewise polynomial approximation of a function's gradient. Furthermore, L2 projection is trying to find an approximation g_h in a finite dimensional space W_h that is closest to the gradient ∇u in the L2 sense [13].

Given $g = \nabla u$, the L2 projection can be defined as:

$$\int_\Omega \psi \cdot g_h \, dx = -\int_\Omega \mathrm{div}(\psi) u_h \, dx + \int_{\partial\Omega} (\psi u_h) \cdot \hat{n} \, ds, \quad \forall \psi \in W_h$$

Here, g_h is the approximation of the gradient, ψ is a test function, and $\hat{n}$ is the outward unit normal to the boundary.

Several numerical experiments have been done to examine how well the L2 projection recovers the gradient of a function and the results do provide some interesting results. For simplicity, the paper

uses helmholtz equation:

$$-\nabla^2 u + u = f \text{ on } \Omega = [0,1] \times [0,1], \nabla u \cdot \vec{n} = 0 \text{ on } \Gamma$$

with:

$$f(x,y) = (1.0 + 8.0\pi^2)\cos(2\pi x)\cos(2\pi y).$$

The analytical solution is:$u(x,y) = \cos(2\pi x)\cos(2\pi y)$.

Experiments were conducted on meshes from 10×10 to 100×100 and comparisons were made between analytical gradient filed and L2 recovered version. The figure shows proves that L2 recovery method achieves good results.

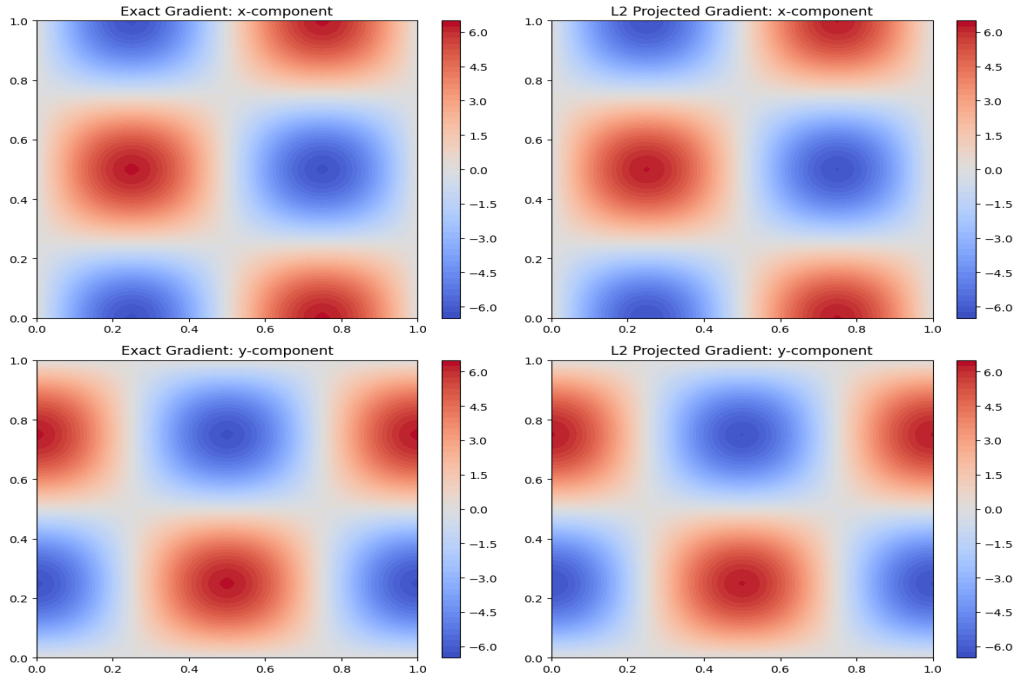


Figure 2.1: comparison between exact gradient and L2 recovered gradient on x and y components

Controlled experiments were made among CG(1), CG(2), CG(3), CG(4) to analyze the results of error data. Data suggests that the error is decreasing when the mesh is refined in both cases. The log-log line plots demonstrate that the mesh size (h) and error (e) has linear relationship:

$$\log(e(h)) = m\log(h) + \log(C)$$

2

This expression can be further written as:

$$e(h) \propto h^m$$

Analytical expressions mesh size (h) and error (e) were obtained by employing linear regression technique:

1. **CG1**: $\log(e(h)) = -1.8199 \times \log(h) + 3.9657$
2. **CG2**: $\log(e(h)) = -1.9115 \times \log(h) + 2.3012$
3. **CG3**: $\log(e(h)) = -3.0820 \times \log(h) + 2.3039$
4. **CG4**: $\log(e(h)) = -3.5453 \times \log(h) + 0.6650$

Both CG1 and CG2 cases seem to exhibit quadratic convergence behavior with mesh refinement. For CG3 and CG4, the rates of convergence are faster than CG1 and CG2. These outcome may need further analysis. However, in general CG1 is usually selected as a trade-off between accuracy and run-time. (Detailed experiment data can be found in notebooks from the repository)

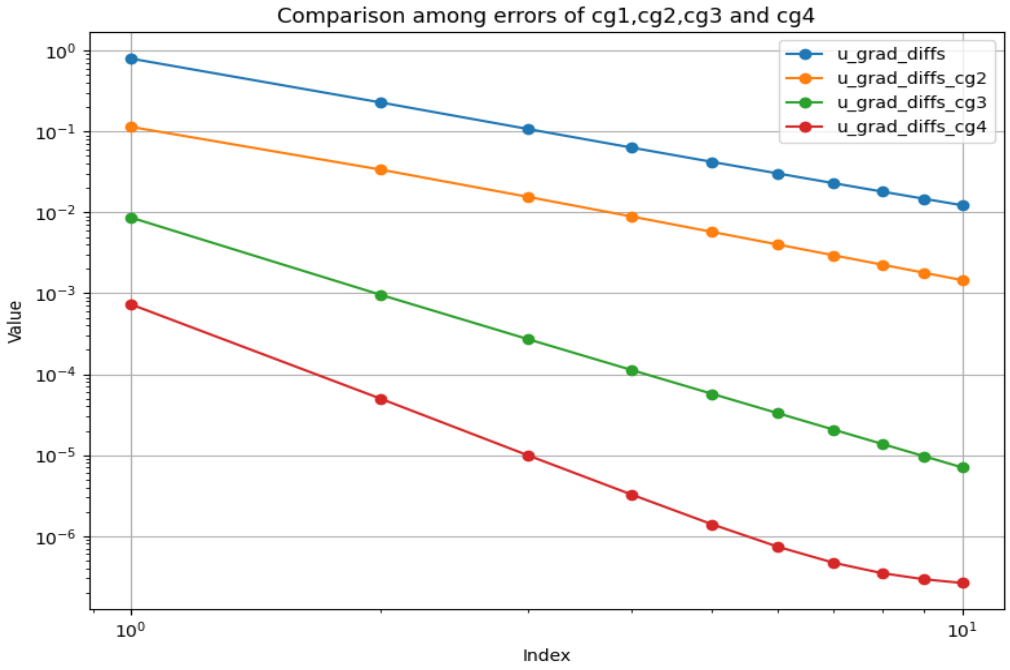


Figure 2.2: log-log plot for error data of L2 recovery

2.2 Zienkiewicz–Zhu Gradient Recovery

[Wallwork,2021] also proposed Z^2 gradient recovery technique to the gradient recovery process when using relaxation or quais-Newton mehods. Z^2 gradient recovery method is an extension of the traditional patch recovery technique [14]. Contrary to the L2 projection method that uses FEM to solve the PDEs over the globally, Z^2 technique focuses on a local, 'patch-based' approach, the "patch" refers to a set of elements surrounding a vertex (or node). To achieve a linear gradient dependent on its finite element gradient, ∇u_{hat}, Z^2 method will evaluate the finite element gradient at the centroids of the elements within the patch. To have a deeper understanding of this algorithm, we will look into the mathematical details of it.

Given a finite element approximation u_h of a scalar filed u, the goal is to obtain a more accurate representation of its the gradient.

For every vertex v in the mesh, consider a local patch of elements which shares that vertex. This patch is denoted as ω^v. The centroid of each element in the patch is given by x_i^v, for $i = 1, \ldots, k$.

The recovered gradient g^v at vertex v is the gradient that minimizes the local least-squares error over the patch:

$$F(g^v) = \sum_{i=1}^{k} \left(P(x_i^k)g^v - \nabla u_h(x_i^v)\right) \cdot \left(P(x_i^v)g^v - \nabla u_h(x_i^v)\right)$$

Where:

- $P(x) = [1, x]$ is the monomial vector.

- $\nabla u_h(x_i^v)$ is the gradient of the finite element solution at the centroid x_i^v.

To find the g^v that minimizes $F(g^v)$, we differentiate $F(g^v)$ with respect to g^v and set the resulting expression to zero. This yields a matrix-vector system of the form:

$$A^v g^v = [b_1^v, \ldots, b_N^v]$$

Where:

$$A^v := \sum_{i=1}^{k} P(x_i^v)^T P(x_i^v)$$

$$b_j^v := \sum_{i=1}^{k} P(x_i^v)^T \frac{\partial u_h(x_i^v)}{\partial x_j}$$

The matrix A^v is symmetric and, under mild assumptions, positive definite. Thus, the system has a unique solution. Once the system is solved for g^v, this provides a recovered gradient at vertex v.

In current code implementation, there is no built-in function to build up a "patch" in **Firedrake** to represent such kind of "graph data structure" that generate an element patch around a vertex. Therefore, self-defined `` `get_patch` `` function is designed to obtain a localized patch of elements surrounding a specified vertex.

`DMPlex` is a part of the `PETSc` library. It provides a flexible way to represent and manipulate unstructured meshes. The official documentation from `PETSc` claims that DMPLEX's performs well in dealing different mesh components like cells, faces, edges, and vertices. Every mesh components is treated as a point and is indexed with a unique identifier. The construction of mesh is implemented through interrelating a "covering" connection among all the points.

Althogh **[Zienkiewicz and Zhu, 1992]** [15] claimed that experiments demonstrated that the convergence of $l2$ errors at interior vertices is $O(h^4)$ for the derivatives with quadratic elements, the current code implementation so not support accurate simulation results(eg. CG(2)). As a result, numerical methods and quantitative analysis will mainly conducted with ZZ gradient recovery on P1 spaces. Same helmholtz pde experiments were also conducted using ZZ method and same techniques were applyed: the log-log errorline plot do exhibits linear relationship between mesh sizes and `errornorm`.

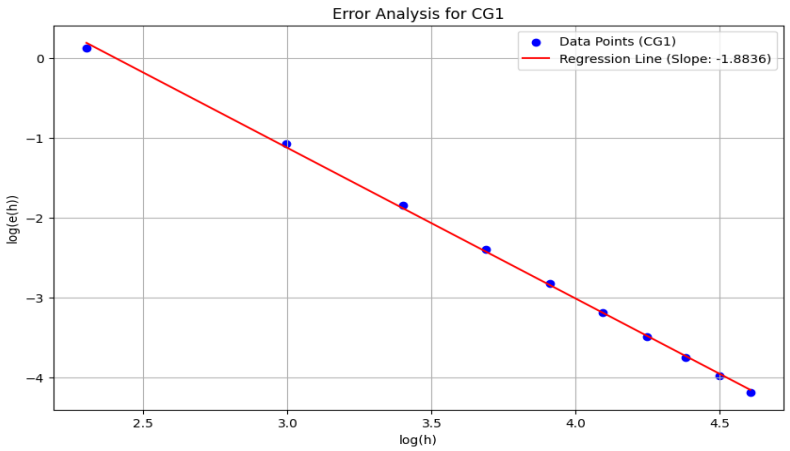


Figure 2.3: log-log plot for error data of ZZ recovery

The expression $log(e(h)) = -1.8836289317494033 \cdot log(h) + 4.522662578214993$ indicates that ZZ

recovery method with CG(1) would follow an approximate ”quadratic” rate of convergence instead, not $O(h^4)$.

Trial on CG(2) method collapsed due to low computational efficiency, unexpected trend of increasing error with mesh size and insufficient development of code. Some targeted actions like using Numba's JIT to accelerate computationally heavy loops, matrix calculations and tensor operations. Nevertheless, Numba might not support all features or functions in the `recover_z2` function since many variables or built-in function have access to many external libraries, like Firedrake or PETSc, which are not directly supported by Numba.

2.3 Performance under Singularities

Generally speaking, L2 method is more popular for its balance between efficiency and accuracy. However, when conducting numerical simulations, one of the most challenging problem is to deal with singularities. Singularities may arise for several reasons, both natural and artificial, and their existence brings complexities that can impact the accuracy and efficiency of a computational method, including gradient recovery techniques discussed so far.[16]

L shaped mesh is one of the representative test cases for examining the behavior of finite element methods. The domain will create a re-entrant corner and induce singularity. For simplicity, this project will study the Poisson equation with source term `f = Constant(1)`. (L shaped mesh is created by Gmsh, please refer to the repository for details).

It is impossible to get an analytical solution because at the re-entrant corner, the behavior of the solution is inherently singular, which shall be expressed in polar coordinates. In order to simulate the analytical gradient field, the gradient is computed using numerical solutions on high-resolution and high-order FEM methods (CG(6) in this case). In practice, the approximated 'exact gradient' is computed with the `'grad'` function.

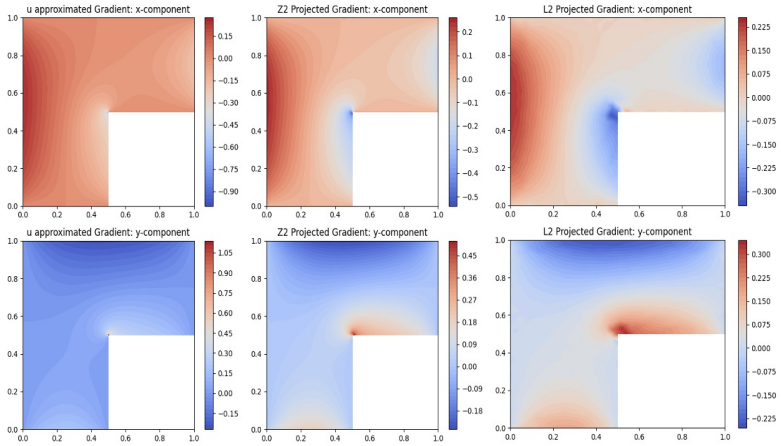


Figure 2.4: Comparison between L2 and ZZ Recovered Gradient

The figure turns out ZZ method performs better on a L-shaped domain . The essence of the ZZ method is its locality. The gradient at a node is recovered based on the solution values in the local patch around that node, which means that ZZ method can more flexibly adjust to sharp variations in the solution, which are present near singularities. Meanwhile, L2 recovery method typically tries to minimize the global L2 error. Although this is desirable in smooth regions, it can lead to unwanted inaccuracies in regions with sharp variations since the global fit may dilute the local singularity's effects, leading to less accurate gradient recovery near the re-entrant corner.[17]

Moreover, the paper also showcases a natural singularity is given by:

$$f(x,y) = \sin\left(\frac{1}{(x-0.5)^{\frac{1}{3}}}\right) \cdot \cos\left(\frac{1}{(y-0.5)^{\frac{1}{3}}}\right)$$

in practice a very tricky method was used to skip the singular area: meshes were refined with an odd number of cells such that the points are not a node in the mesh. Unfortunately, the `errornorm` for L2 = 23.0958177283 and ZZ = 21.0965294181 method do not have too much difference. The most challenging part is that the rapid oscillations near the singularity demand an extremely fine mesh to accurately capture the function's behavior[18]and further study and research in this part is worthy.

Conclusions

In this project, two straightforward numerical techniques, relaxation method and quasi-Newton Method, were presented to solve the Monge-Ampere Equations. Quantitative analysis and assessment of these methods were conducted, focusing on their rate of convergence, robustness, computational costs and some mathematical detailes. The result shows the fast runtime for single iteration of relaxation method and its sensitiveness to pseudo timestep. As for quasi-Newton Method, the result justified its robustness and efficiency. The combination of these two methods utilized the advantages of them and provided promising mesh movement results on meshes with different sizes.

As a key component of the mentioned methods, different gradient recovery techniques have also analyzed. Theoretically, L2 gradient recovery is preferred for its higher efficiency and highly highly involved packages and functions from 'Firedrake'. ZZ recovery method, althrough requires more optimization and improvement in the code, has its own advantage in singularity problems.

In the future works, we hope to analysis the choice of monitor function and tuning the scaling parameterβ to achieve a balance between speeding up the calculation of solving mong-ampere and informing the information of curvature and gradient to the mesh for movement. The improvement of ZZ recovery is also important. In fact, there are also researches using machine learning techniques to replace the complex solver, which also deserves more time and study upon this topic.

Bibliography

[1] Y. R. WEIZHANG HUANG and R. D. RUSSELL, "Moving mesh partial differential equations (mmpdes) based on the equidistribution principle," *Society for Industrial and Applied Mathematics*, vol. 31, pp. 709–730, 1994.

[2] C. B. Hilary Weller Philip Browne and M. Cullen, "Mesh adaptation on the sphere using optimal transport and the numerical solution of a monge–ampère type equation," *Journal of Computational Physics*, vol. 308, pp. 102–123, 2016.

[3] C. J. C. ANDREW T. T. MCRAE and C. J. BUDD, "Optimal-transport-based mesh adaptivity on the plane and sphere using finite elements," pp. 1–36, 2018.

[4] O. Lakkis and T. Pryer, "An adaptive finite element method for the infinity laplacian," in *Lecture Notes in Computational Science and Engineering*, Springer International Publishing, Oct. 2014, pp. 283–291. DOI: 10.1007/978-3-319-10705-9_28. [Online]. Available: https://doi.org/10.1007%2F978-3-319-10705-9_28.

[5] J. P. Milaszewicz, "On Brown's and Newton's methods with convexity hypotheses," *Journal of Computational and Applied Mathematics*, vol. 150, no. 1, pp. 1–24, Jan. 2003. DOI: 10.1016/S0377-0427(02)00489-2.

[6] R. Ramani and S. Shkoller, "A fast dynamic smooth adaptive meshing scheme with applications to compressible flow," *Journal of Computational Physics*, vol. 490, p. 112 280, Oct. 2023. DOI: 10.1016/j.jcp.2023.112280. [Online]. Available: https://doi.org/10.1016%2Fj.jcp.2023.112280.

[7] N. Y. Gnedin, V. A. Semenov, and A. V. Kravtsov, "Enforcing the courant–friedrichs–lewy condition in explicitly conservative local time stepping schemes," *Journal of Computational Physics*, vol. 359, pp. 93–105, Oct. 2018. [Online]. Available: https://doi.org/10.1016/j.jcp.2018.01.008.

[8] J. G. Wallwork, "Mesh adaptation and adjoint methods for finite element coastal ocean modelling," 2021. DOI: https://doi.org/10.25560/92820.

[9] M. N. XIAOBING FENG, "Mixed finite element methods for the fully nonlinear monge–ampère equation based on the vanishing moment method," 2009.

[10] *Solving pdes - firedrake 0+unknown documentation*. [Online]. Available: https://www.firedrakeproject.org/solving-interface.html.

[11] E. Kawecki, O. Lakkis, and T. Pryer, *A finite element method for the monge-ampère equation with transport boundary conditions*, 2018. arXiv: 1807.03535 [math.NA].

[12] *Errornorm - fenics project*. [Online]. Available: https://fenicsproject.org/olddocs/dolfin/1.5.0/python/programmers-reference/fem/norms/errornorm.html.

[13] D. Seibel1 and S. Weißer1, "Gradient recovery for the bem-based fem and vem," 2019.

[14] Zienkiewicz and 1. Zhu, "A simple error estimator and adaptive procedure for practical engineerng analysis. int. j. numer. meth. engng., 24," in *Numerical Methods in Engineering*, 1987, pp. 337–357.

[15] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique," *International Journal for Numerical Methods in Engineering*, vol. 33, no. 7, pp. 1331–1364, 1992. DOI: https://doi.org/10.1002/nme.1620330702. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620330702. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620330702.

[16] D. D. Alic *et al.*, "Theoretical issues in numerical relativity simulations," 2009.

[17] H. Wu and Z. Zhang, "Can we have superconvergent gradient recovery under adaptive meshes?" *SIAM Journal on Numerical Analysis*, vol. 45, pp. 1701–1722, 2007.

[18] L. B. Wahlbin, "Local behavior in finite element methods," *HANDBOOK OF NUMERICAL ANALYSIS, Finite Element Methods (Part 1)*, vol. 2, pp. 1701–1722, 1991.