Solving a Simplified Monge-Ampere Equation with a Weak Galerkin Approach

Math 652

Colin Jensen

12/7/2020

Optimal Transport with Monge-Ampere Equation

Refresher:

In order to solve the optimal transport problem of finding an optimal map between two probability densities, we wish to solve the Monge-Ampere Equation

$$\begin{cases} \det(H_\varphi(x)) = \frac{\mu(x)}{\nu(\nabla\varphi(x))} & \forall x \in U \\ \nabla\varphi(\partial U) = \partial V \end{cases}$$

Optimal Transport with Monge-Ampere Equation

Refresher:

In order to solve the optimal transport problem of finding an optimal map between two probability densities, we wish to solve the Monge-Ampere Equation

$$\begin{cases} \det(H_\varphi(x)) = \frac{\mu(x)}{\nu(\nabla\varphi(x))} & \forall x \in U \\ \nabla\varphi(\partial U) = \partial V \end{cases}$$

NONLINEAR NIGHTMARE!

Optimal Transport with Monge-Ampere Equation

We impose some aggressive assumptions and end up with the much nicer, but still non-linear

$$\begin{cases} \Delta\psi + \det(H_\psi) = u - 1 \quad \forall \mathrm{x} \in U \\ \nabla\varphi \cdot \mathrm{n} = 0 \end{cases}$$

Optimal Transport with Monge-Ampere Equation

We impose some aggressive assumptions and end up with the much nicer, but still non-linear

$$\begin{cases} \Delta\psi + \det(H_\psi) = u - 1 \quad \forall \mathrm{x} \in U \\ \nabla\varphi \cdot \mathrm{n} = 0 \end{cases}$$

We attempt to overcome the nonlinearity with fixed point iteration:

$$\Delta\psi_k = u - 1 - \det(H_{\psi_{k-1}})$$

Optimal Transport with Monge-Ampere Equation

We impose some aggressive assumptions and end up with the much nicer, but still non-linear

$$\begin{cases} \Delta\psi + \det(H_\psi) = u - 1 \quad \forall \mathrm{x} \in U \\ \nabla\varphi \cdot \mathrm{n} = 0 \end{cases}$$

We attempt to overcome the nonlinearity with fixed point iteration:

$$\Delta\psi_k = u - 1 - \det(H_{\psi_{k-1}})$$

1. Initialize ψ_0 as the solution to $\Delta\psi_0 = u - 1$

Optimal Transport with Monge-Ampere Equation

We impose some aggressive assumptions and end up with the much nicer, but still non-linear

$$\begin{cases} \Delta\psi + \det(H_\psi) = u - 1 \quad \forall \mathrm{x} \in U \\ \nabla\varphi \cdot \mathrm{n} = 0 \end{cases}$$

We attempt to overcome the nonlinearity with fixed point iteration:

$$\Delta\psi_k = u - 1 - \det(H_{\psi_{k-1}})$$

1. Initialize ψ_0 as the solution to $\Delta\psi_0 = u - 1$
2. Approximate $\det(H_{\psi_0})$.

Optimal Transport with Monge-Ampere Equation

We impose some aggressive assumptions and end up with the much nicer, but still non-linear

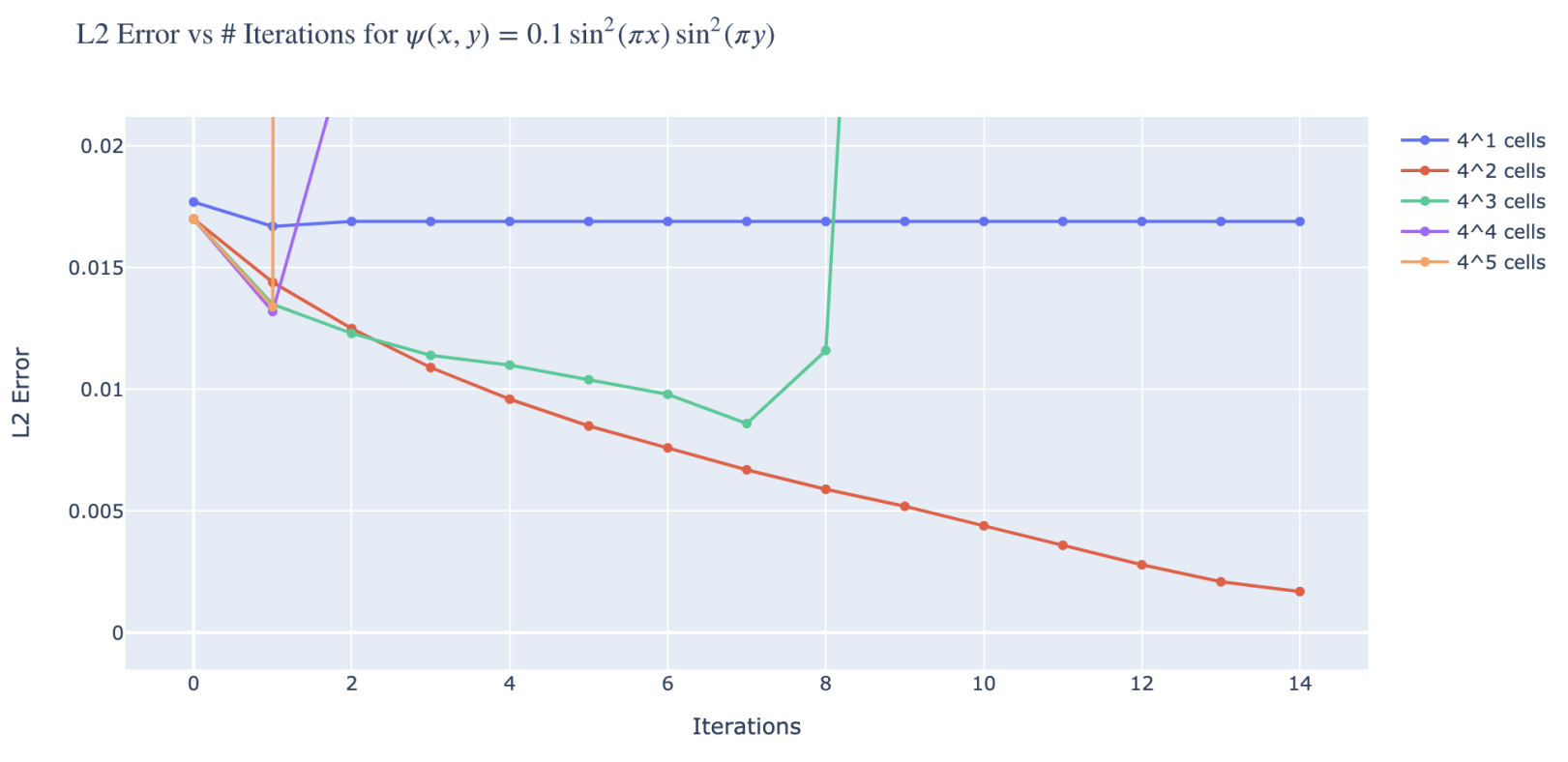
$$\begin{cases} \Delta\psi + \det(H_\psi) = u - 1 \quad \forall \mathrm{x} \in U \\ \nabla\varphi \cdot \mathrm{n} = 0 \end{cases}$$

We attempt to overcome the nonlinearity with fixed point iteration:

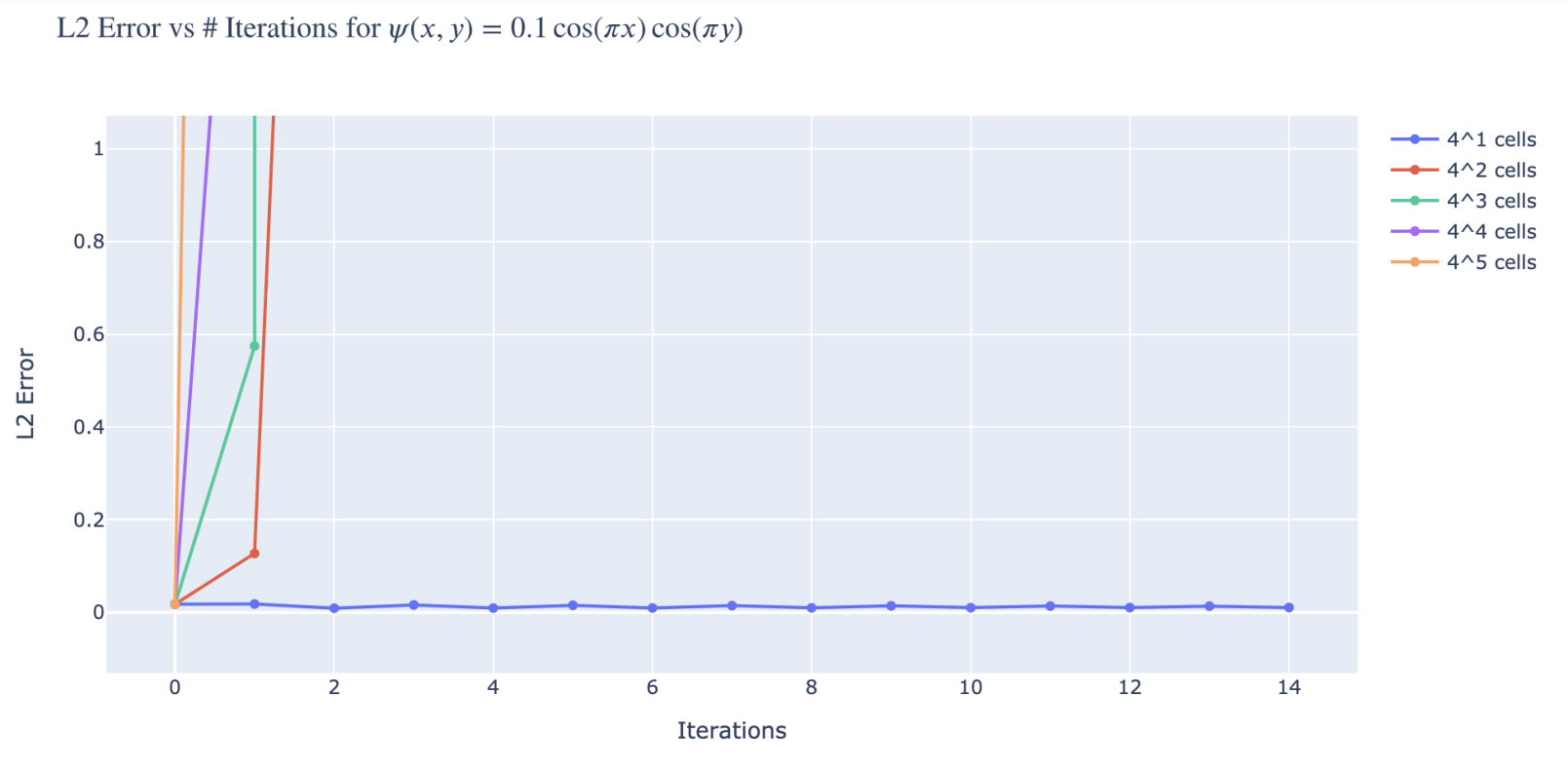
$$\Delta\psi_k = u - 1 - \det(H_{\psi_{k-1}})$$

1. Initialize ψ_0 as the solution to $\Delta\psi_0 = u - 1$
2. Approximate $\det(H_{\psi_0})$.
3. Iteratively solve $\Delta\psi_{k+1} = u - 1 - \det(H_{\psi_k})$ until the error falls below some pre-determined tolerance.

Results Using Continuous Galerkin Method



Results Using Continuous Galerkin Method



Weak Galerkin Methodology

To mitigate the poor performance of continuous Galerkin, I tried a weak Galerkin method

Weak Galerkin Methodology

To mitigate the poor performance of continuous Galerkin, I tried a weak Galerkin method

I assume the solution is in the form of a weak function $\psi_i = (\psi_i^o, \psi_i^\partial)$ and solve the Poisson problem using a weak Galerkin finite element

```
FESystem<dim> fe(FE_DGQ<dim>(degree), 1, FE_FaceQ<dim>(degree), 1);
```

Weak Galerkin Methodology

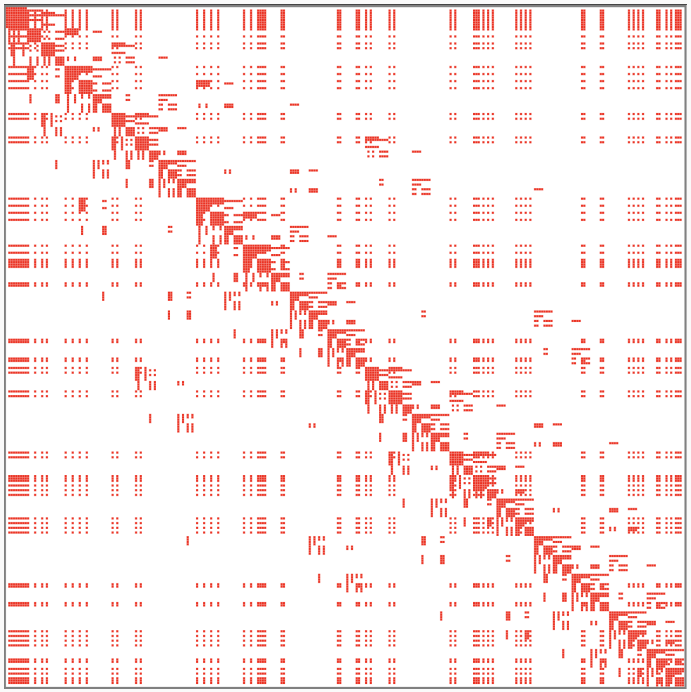


Figure 1: Sparsity pattern for continuous Galerkin system matrix with 64 cells

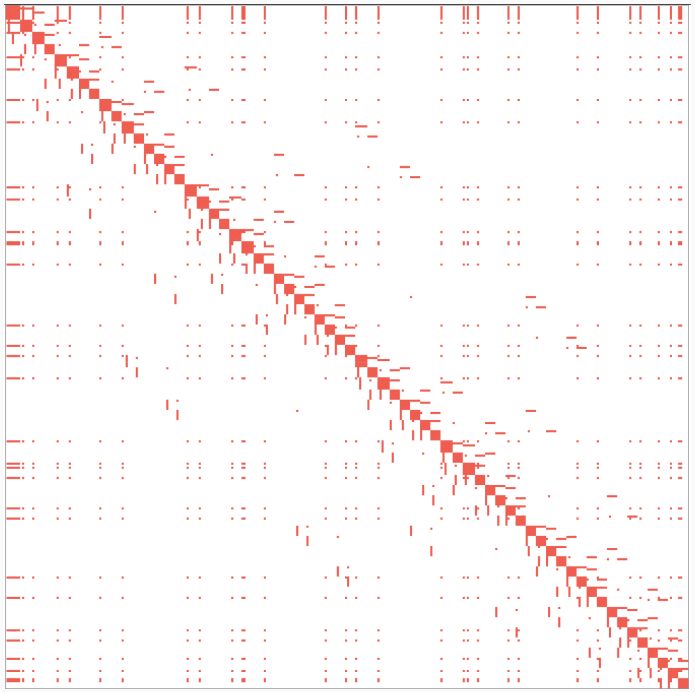


Figure 2: Sparsity pattern for weak Galerkin system matrix with 64 cells

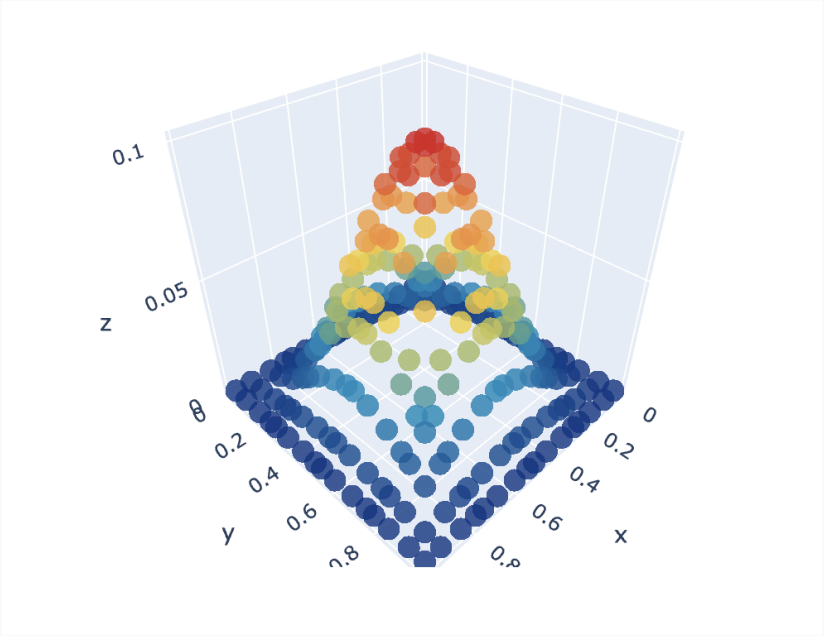


Figure 3: Interior solution for $\psi(x, y) = 0.1 \sin^2(\pi x) \sin^2(\pi y)$

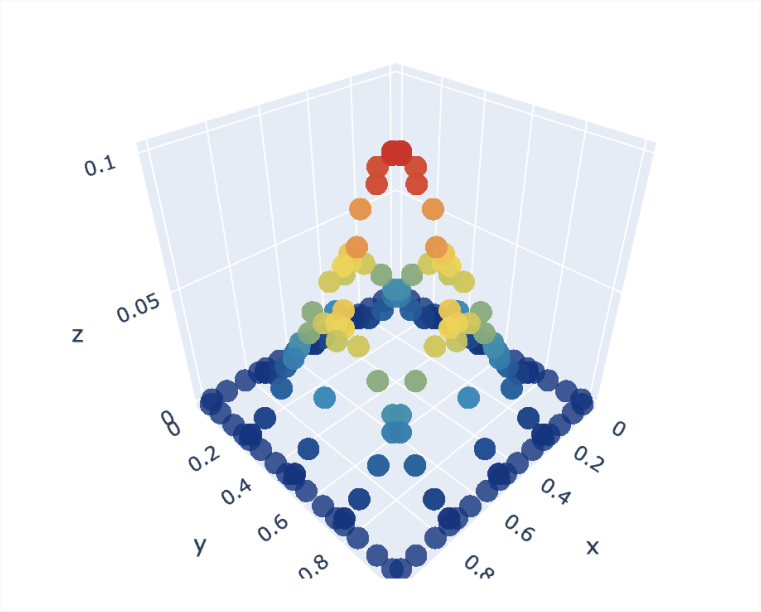


Figure 4: Cell interface solution for $\psi(x, y) = 0.1 \sin^2(\pi x) \sin^2(\pi y)$

Weak Galerkin Methodology

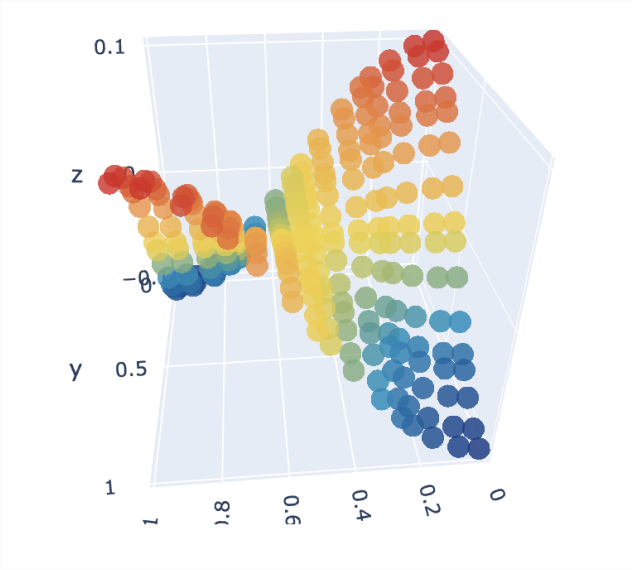


Figure 5: Interior solution for $\psi(x, y) = 0.1 \cos(\pi x) \cos(\pi y)$

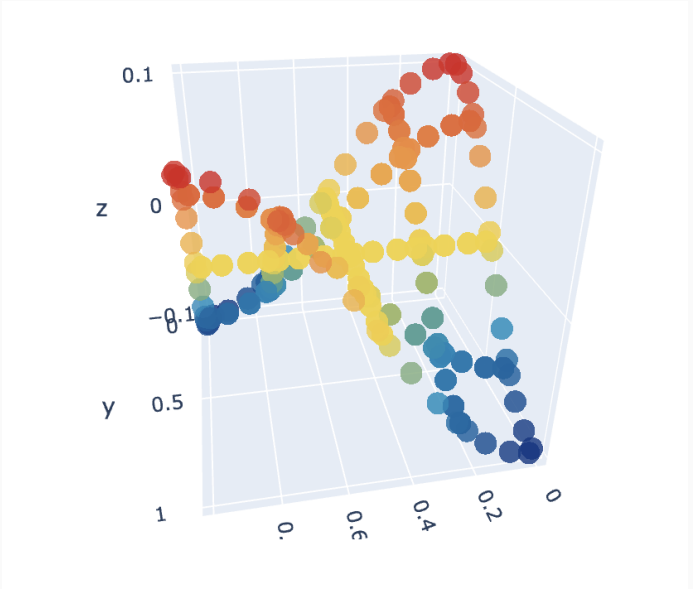


Figure 6: Cell interface solution for $\psi(x, y) = 0.1 \cos(\pi x) \cos(\pi y)$

Weak Derivatives

We cannot use the conventional definition of derivatives on weak functions. Therefore, we use the following formulas to define weak gradients and weak 2nd-order partial derivatives.

$$\int_K \mathrm{w} \cdot (\nabla_w \psi) = \int_{K^\partial} (\mathrm{w} \cdot \mathrm{n}) \psi^\partial - \int_{K^o} (\nabla \cdot \mathrm{w}) \psi^o, \qquad \forall w \in RT_{[n]}(K)$$

$$\int_K \phi \left(\partial^2_{w,ij} \psi \right) = \int_K \left(\partial_{ji} \phi \right) \psi^0 - \int_{K^\partial} \left(\partial_j \phi \right) \psi^\partial n_i + \int_{K^\partial} \phi \psi_i^g n_j \qquad \forall \phi \in Q_r(K).$$

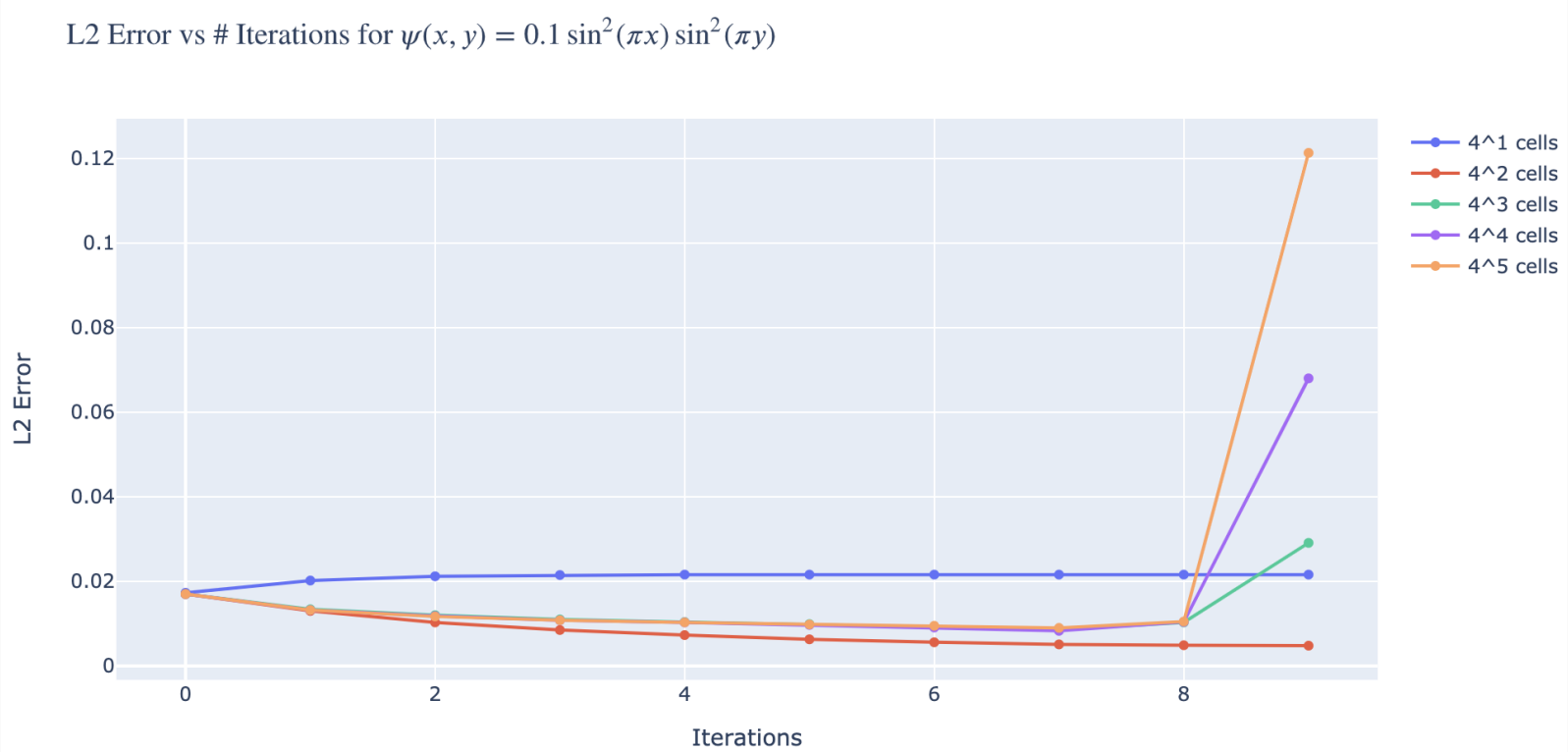
Weak Derivatives

We cannot use the conventional definition of derivatives on weak functions. Therefore, we use the following formulas to define weak gradients and weak 2nd-order partial derivatives.

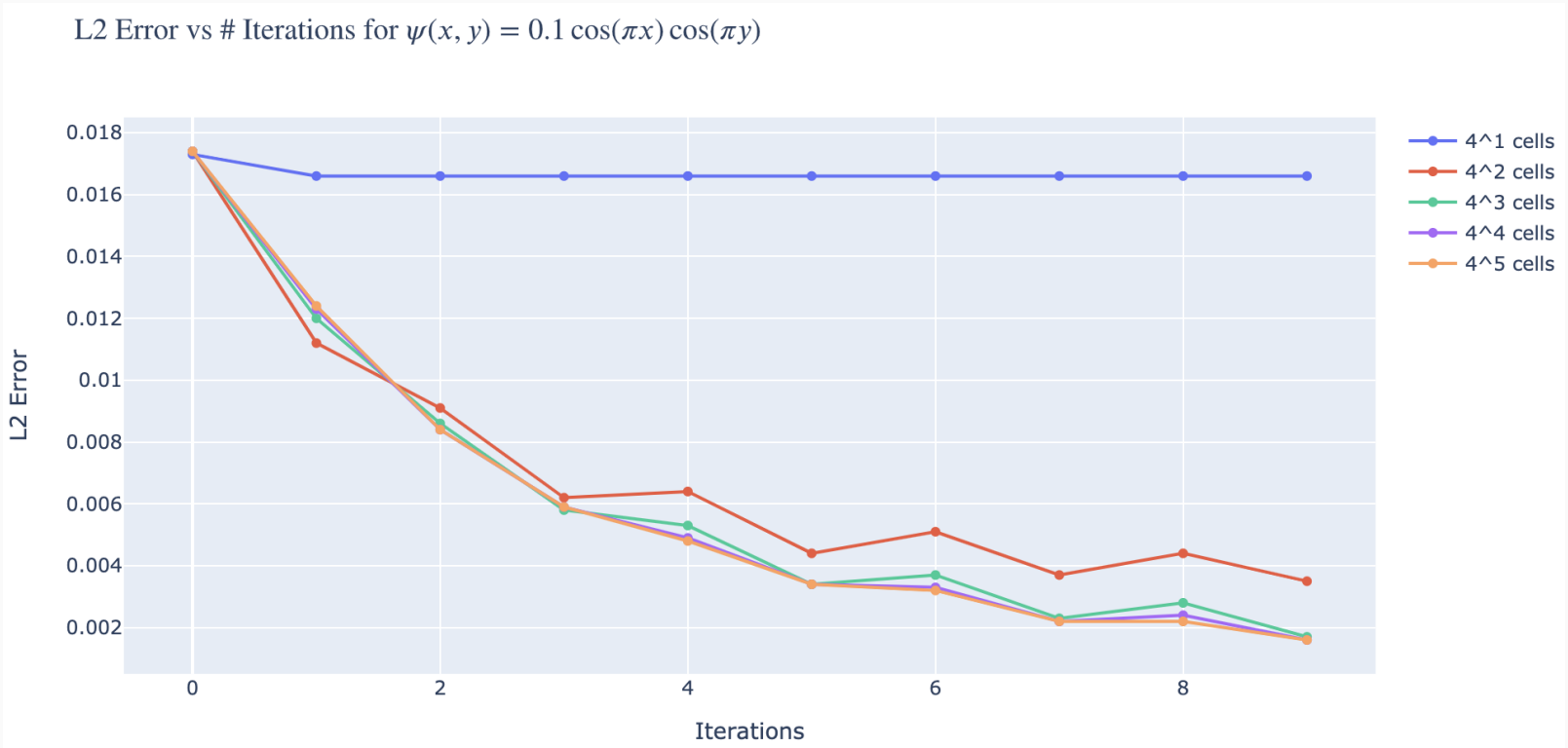
$$\int_K \mathrm{w} \cdot (\nabla_w \psi) = \int_{K^\partial} (\mathrm{w} \cdot \mathrm{n}) \psi^\partial - \int_{K^o} (\nabla \cdot \mathrm{w}) \psi^o, \qquad \forall w \in RT_{[n]}(K)$$

$$\int_K \phi \left(\partial^2_{w,ij} \psi \right) = \int_K \left(\partial_{ji} \phi \right) \psi^0 - \int_{K^\partial} \left(\partial_j \phi \right) \psi^\partial n_i + \int_{K^\partial} \phi \psi_i^g n_j \qquad \forall \phi \in Q_r(K).$$

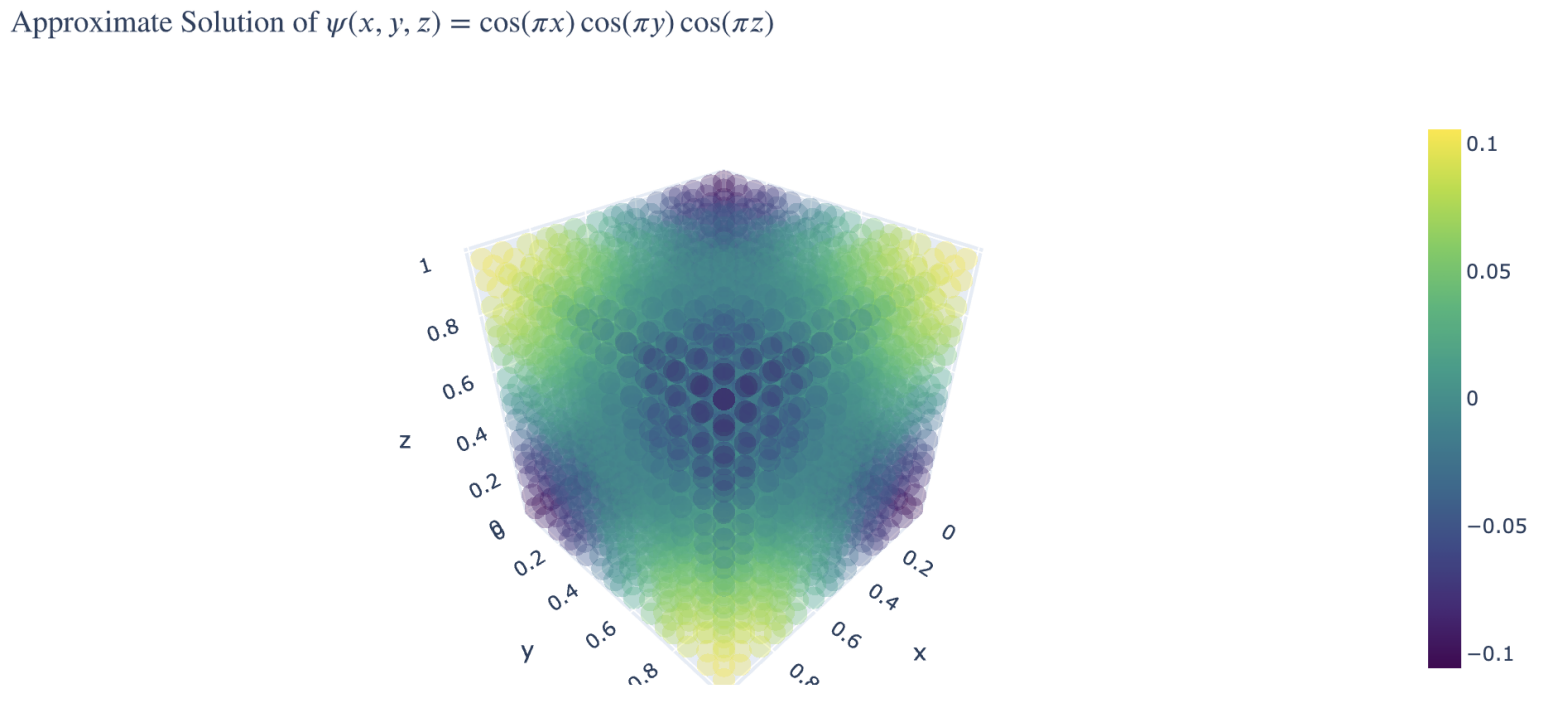
Results Using Weak Galerkin



Results Using Weak Galerkin



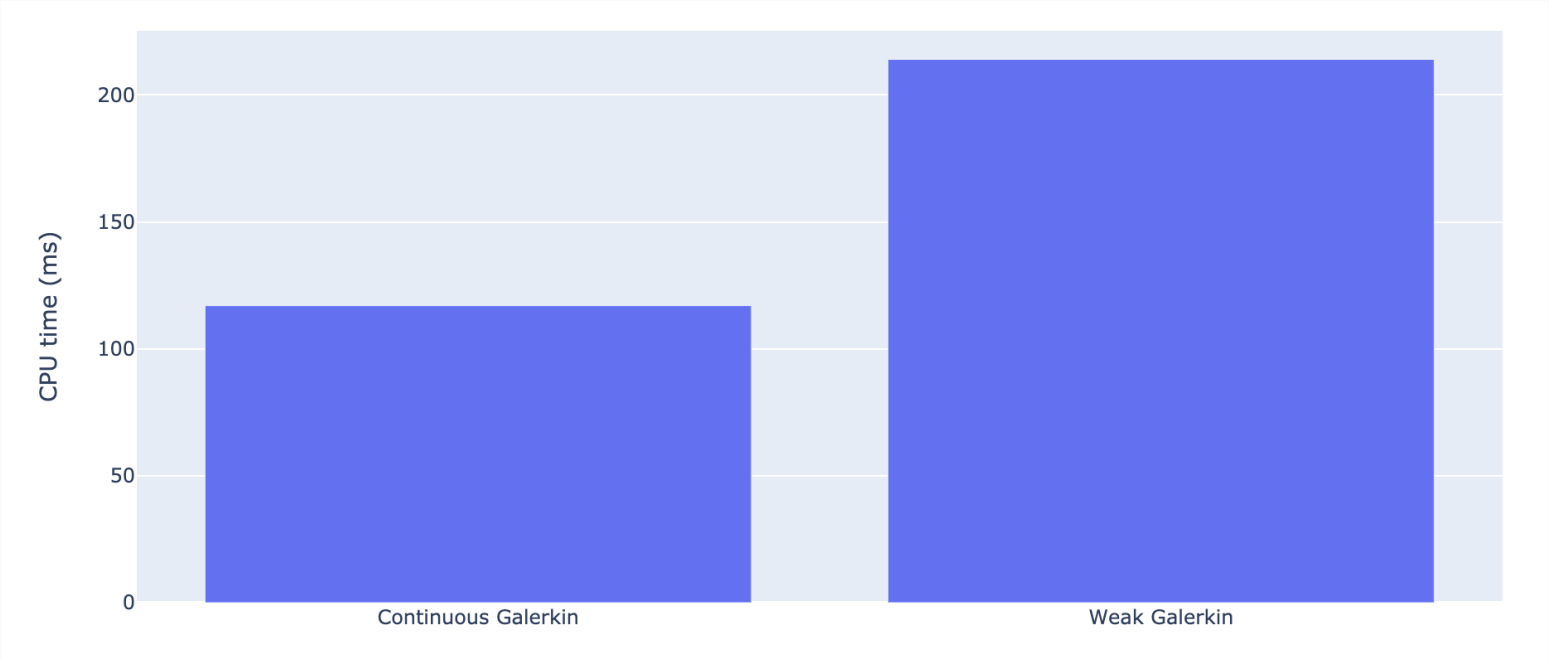
3D, Because We Can



Comparing Continuous and Weak Galerkin Methods

I ran an experiment on the following test case:

- Exact solution $\psi(x, y) = \sin^2(\pi x) \sin^2(\pi y)$
- 2nd-degree finite elements
- 16 cells
- Recording CPU time taken for L2 error to fall below 0.005



Comparing Continuous and Weak Galerkin Methods

Advantages of Weak Galerkin:

- Overall stability
- More efficient on high-order finite elements

Advantages of Continuous Galerkin:

- Faster, especially when considering assembling the system matrix

Future Work

- Try other methods of dealing with the non-linearity (Newton's Method, etc)
- Try on other domains, general quadrilaterals etc
- Figure out how to speed the process up by using a degree 1 finite element in the Poisson solver
- Work on the theoretical details