A direct solver for Laplace equation using spectral element discretization

CS598 Fall 2017 Project Report

Li Lu, UIUC

December 15, 2017

1 Motivation

The two primary goals in solving partial differential equations are accuracy and efficiency. High order methods are our way of achieving accuracy in the world of numerical algorithms. More considerations are needed in terms of efficiency: iterative solvers or direct solvers for example is a classic problem and should be determined on a case-by-case basis.

[Martinsson, 2013] describes a direct solver that combines high-order discretization as well as a hierarchical solver. Spectral collocation method was used as the high-order discretization scheme. The hierarchical solver assumes a tree structure in a square domain and divide the degrees of freedom into multi-level surface and interior points. The intrinsic tree structure enable the solving procedure to have a cost of order $O(N \log N)$.

This is of interest to me as my current research problem involves solving large quasi-linear systems in the context of high-order spectral element method. An investigation into this could potentially provide an alternative way for solving these systems.

2 Algorithm

In this project, efforts were put into developing a Laplace equation solver using spectral element method [Deville et al., 2002] (SEM) and the hierarchical solver framework from [Martinsson, 2013].

2.1 Introduction to SEM

Spectral element method([Deville et al., 2002]) is in effect a high order finite element method(FEM). Specifically, Gauss-Lobatto-Legendre points were chosen in the discretization. Basis functions are the Lagrange polynomials on these points. Since this belongs to the Galerkin method family, test functions are the same as the basis functions.

Numerical quadrature is used in constructing the operators. One difference from classical FEM is the choice of diagonal mass matrix instead of a full mass matrix. This choice originated from performance consideration.

Consider Laplace equation with inhomogeneous Dirichlet boundary data,

$$\nabla^2 u = 0, \quad u|_{\partial\Omega} = f \tag{1}$$

To formulate this in SEM, we construct the weak form by

$$\begin{aligned}\int_\Omega v\nabla^2 u\,\mathrm{d}V &= 0\\ \int_\Omega \nabla v\cdot\nabla u &= 0\end{aligned}\tag{2}$$

Expanding u and v in terms of basis functions l_i's

$$u=\sum_{i=1}^{n}u_i\phi_i(x,y)=\sum_{j=0}^{N}\sum_{k=0}^{N}u_{jk}l_j(x)l_k(y),\quad v=\sum_{i=1}^{n}v_i\phi_i(x,y)=\sum_{j=0}^{N}\sum_{k=0}^{N}v_{jk}l_j(x)l_k(y),\tag{3}$$

Notice that all basis functions here l_i are found by the linear mapping from a reference element. The left hand side could be expressed

$$-\sum_j\sum_k\sum_l\sum_m v_{jk}\int_{x_a}^{x_b}\int_{y_a}^{y_b} l_j'(x)l_l'(x)l_k(y)l_m(y)+l_j(x)l_l(x)l_k'(y)l_m'(y)\,\mathrm{d}V\,u_{lm}=0\tag{4}$$

Defining 1D mass matrices B and stiffness matrices K for each element

$$\begin{aligned}B_{y_{ij}}&=\int_{y_a}^{y_b}l_i(y)l_j(y)\,\mathrm{d}y,\quad B_{xij}=\int_{x_a}^{x_b}l_i(x)l_j(x)\,\mathrm{d}x\\ K_{y_{ij}}&=\int_{y_a}^{y_b}l_i'(y)l_j'(y)\,\mathrm{d}y,\quad K_{xij}=\int_{x_a}^{x_b}l_i'(x)l_j'(x)\,\mathrm{d}x\end{aligned}\tag{5}$$

Then the weak form can be written in matrix form as

$$(B_y\otimes K_x+K_y\otimes B_x)\underline{u}=A\underline{u}=0\tag{6}$$

Break the solution u into homogeneous part u_h such that the homogeneous Laplace equation is satisfied, and u_b such that the boundary conditions are satisfied $u=u_h+u_b$. u_b can be anything, as long as the Dirichlet boundary data are satisfied.

$$A\underline{u}_h=-A\underline{u}_b\tag{7}$$

Restricting the unknown to the points that are not on the boundary of the domain, one obtains the homogeneous solution in the interior of the domain. Lastly, the final solution is found by $u=u_h+u_b$. This provides a solution for points inside the domain from Dirichlet boundary data. For multiple elements in a 2D domain, classic SEM also requires global elimination of repeated points,

2.2 Methodology: one element case

Now we introduce the solver in a one element case. Discretize the square domain into a 2D tensor product form of GLL points, as shown in figure 1.

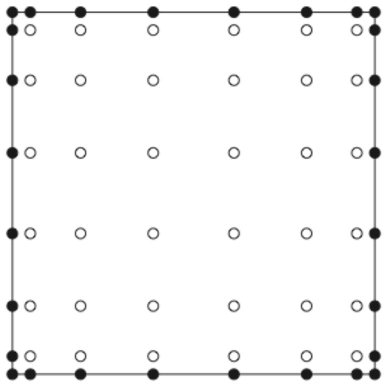

Figure 1: Degrees of freedom in one spectral element , source:[Martinsson, 2013]

The solid circles are the boundary points, and the empty circles are the interior points for this one element domain. From equation 6, instead of following the SEM approach, we seek another way to solve this equation. Denoting the exterior points by subscript e and the interior points by subscript i, the matrix A can be re-organized and rewritten by

$$A = \begin{bmatrix} A_{i,i} & A_{i,e} \\ A_{e,i} & A_{e,e} \end{bmatrix} \tag{8}$$

Recall that the goal is to solve for the interior points. We pick out the rows corresponding to i in the matrix and move the known data to the right hand side, then a solution is found by

$$u_i = -A_{i,i}^{-1} A_{i,e} u_e \tag{9}$$

For a one element case u_e has all the boundary data, and u_i are all the unknowns. Equation 9 is all that is required to solve this problem.

For later references, we define the Dirichlet-to-Neumann(DtN) operators. They find the partial derivatives given the Dirichlet data on the boundary of the domain. First we need the SEM derivative operators that apply to the 2D solution vector

$$\partial_x u = Du, \quad Eu = \partial_y u \tag{10}$$

Operators D and E can be found by finding the derivative values on the GLL points from the Lagrange polynomials. Organize the rows and columns of D and E in a similar way to obtain the DtN operators defined by

$$\begin{aligned} v_e &= (D_{e,e} + D_{e,i}U)u_e = Vu_e \approx \partial_x u_e \\ w_e &= (E_{e,e} + E_{e,i}U)u_e = Wu_e \approx \partial_y u_e \end{aligned} \tag{11}$$

v_e and w_e are the approximate partial derivatives of u_e.

2.3 Methodology: multiple element case

In multi-element case, a merging operation is critical to obtaining the needed operators for the combination of two boxes. Later we will see that in the tree structure, the parent box operators are built from its children boxes, through this merging operation. Consider the following two spectral element combining into one big box,

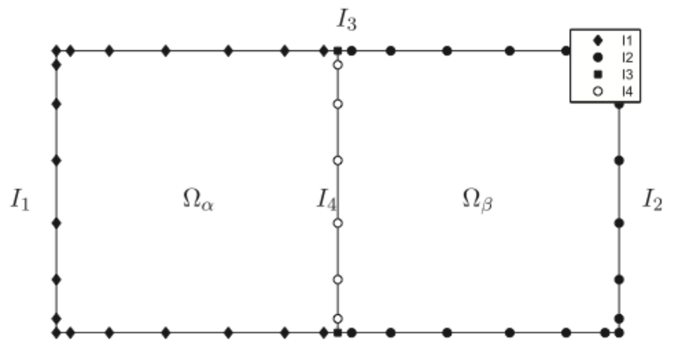

Figure 2: Index sets , source:[Martinsson, 2013]

we only need to consider the boundary points of the two small boxes. Divide them into four different index sets I_1 to I_4

- I_1: points that are on the boundary of box α but not on the boundary of box β
- I_2: points that are on the boundary of box β but not on the boundary of box α
- I_3: points that are shared by α and β, and are still on the boundary of the big box
- I_4: points that are shared by α and β, and are not on the boundary of the big box

Points in I_1, I_2, I_3 make up the boundary sets of the big box, and I_4 are defined as the interior points. Now we seek operators U, V, W for the big box such that we can find solution values on I_4, and derivative data on I_1, I_2 and I_3. We pick a specific indexing such that the new boundary points vector is in the order of I_1, I_2 and I_3. The operators U, V, W could be found after derivation detailed in [Martinsson, 2013]. Following are summarized equations,

- U: if the two small boxes are aligned horizontally

$$U = (V^\alpha_{4,4} - V^\beta_{4,4})^{-1} \left[-V^\alpha_{4,1} | V^\beta_{4,2} | V^\beta_{4,3} - V^\alpha_{4,3} \right] \tag{12}$$

- U: if the two small boxes are aligned vertically

$$U = (W^\alpha_{4,4} - W^\beta_{4,4})^{-1} \left[-W^\alpha_{4,1} | W^\beta_{4,2} | W^\beta_{4,3} - W^\alpha_{4,3} \right] \tag{13}$$

- V

$$V = \begin{bmatrix} V^\alpha_{1,1} & \mathbf{0} & V^\alpha_{1,3} \\ \mathbf{0} & V^\beta_{2,2} & V^\beta_{2,3} \\ \frac{1}{2}V^\alpha_{3,1} & \frac{1}{2}V^\beta_{3,2} & \frac{1}{2}V^\alpha_{3,3} + \frac{1}{2}V^\beta_{3,3} \end{bmatrix} + \begin{bmatrix} V^\alpha_{1,4} \\ V^\beta_{2,4} \\ \frac{1}{2}V^\alpha_{3,4} + \frac{1}{2}V^\beta_{3,4} \end{bmatrix} U \tag{14}$$

- W

$$W = \begin{bmatrix} W^\alpha_{1,1} & \mathbf{0} & W^\alpha_{1,3} \\ \mathbf{0} & W^\beta_{2,2} & W^\beta_{2,3} \\ \frac{1}{2}W^\alpha_{3,1} & \frac{1}{2}W^\beta_{3,2} & \frac{1}{2}W^\alpha_{3,3} + \frac{1}{2}W^\beta_{3,3} \end{bmatrix} + \begin{bmatrix} W^\alpha_{1,4} \\ W^\beta_{2,4} \\ \frac{1}{2}W^\alpha_{3,4} + \frac{1}{2}W^\beta_{3,4} \end{bmatrix} U \tag{15}$$

A few things to note

- Simple relations between the two small boxes are assumed, namely they can only be either horizontal or vertical to each other.

- The reason that the geometric relation between the two small boxes is important is due to the equilibrium requirement, which dictates that the normal derivative across the interface of the to small boxes has to be equal. This has similarities to the non-overlapping domain decomposition methods.
- Alternatively, if we were to apply this method to a more complex case the normal derivative requirements would become an equation involing both V and W.
- An order is implied such that operators on smaller boxes are always found before the big box.

2.4 Methodology: hierarchical scheme

Now we introduce the tree structure and the hierarchical scheme. Consider a square domain subdivided by half in the following way, and order the boxes first on higher level in the tree.

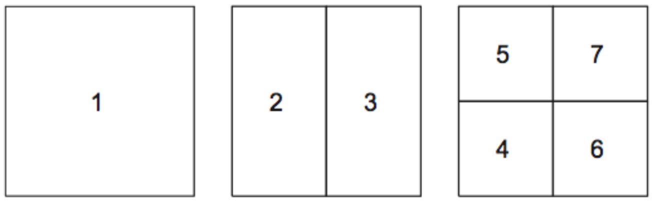

Figure 3: Box IDs, source:[Martinsson, 2013]

For simplicity, in this project we only consider the box structure as shown in this diagram. The lowest level boxes(leaf boxes) are the spectral elements. The hierarchical scheme is made up of two major steps: *building* and *solving*. In the building stage, we start from the lowest level, where operators in each leaf box is constructed by standard SEM procedure; then we are going up the tree levels, and we can always apply the merging process to find the new operators for the parent box using the children boxes' operators. The pseudo-code reads

Algorithm 1 Pre-computation(build)

```
1: for $\tau = N_{boxes}$ to 1 do
2:     if $\tau$ is a leaf then
3:         Eval $U^\tau, V^\tau, W^\tau$, Eqs: 10,11
4:     else
5:         Let $\sigma_1, \sigma_2$ be the children of $\tau$
6:         if $\sigma_1$ and $\sigma_2$ are horizontal then
7:             Eval $U^\tau$ using $V^{\alpha,\beta}$, Eq 12
8:         else
9:             Eval $U^\tau$ using $W^{\alpha,\beta}$, Eq 13
10:        end if
11:        Eval $V^\tau, W^\tau$, Eqs: 14,15
12:    end if
13: end for
```

The solving step then is fairly straightforward. Going down the tree structure and apply operator U to obtain the interior values, which then becomes boundary data for the children boxes on a lower level. Once the entire tree is traversed all of the data needed will be computed. The pseudo-code is

Algorithm 2 Forward solve

```
1: Find boundary data for box 1 $\mathbf{u} = f(\mathbf{x})$
2: for $\tau = 1$ to $N_{boxes}$ do
3:     $\mathbf{u}(I_i^\tau) = U^\tau \mathbf{u}(I_e^\tau)$
4: end for
```

3 Preliminary results

On domain $[0, 1]^2$, function $u = \cos kx \exp ky$ is an exact solution to the Laplace equation, and has nontrivial boundary values. Take $k = \pi/2$, the exact solution field is shown below

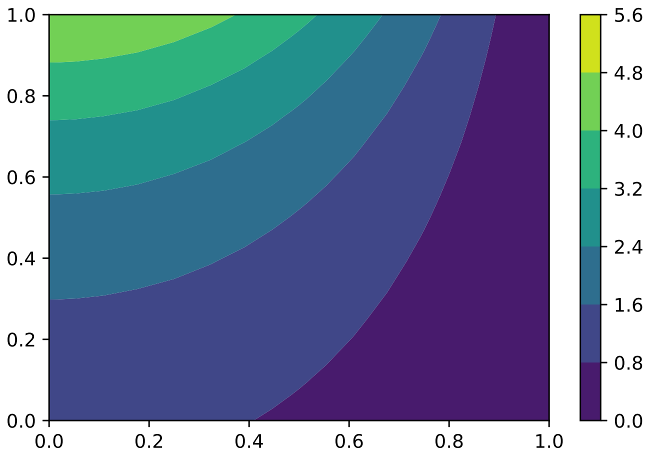

Figure 4: Exact solution

A 2×2 element mesh, with $N = 10$ was used to generate a series of solution. Here we plot the numerical solution after every box has found its interior values.

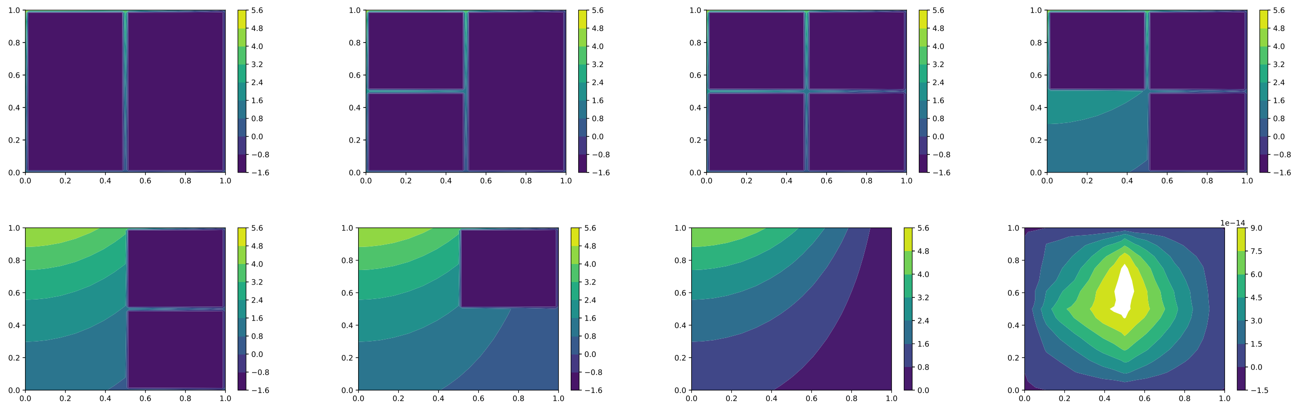

Figure 5: Procedural solution and error in the final solution

We can see the solver behaving as expected, producing an accurate solution.

Next, establish the convergence of this method. Since the simplistic implementation only allows for this one mesh (and tree structure), the parameter we can tune is polynomial order N. The L2 norm errors in the final solution of the entire field as N increases are shown in 1. A comparison against classic spectral element method is conducted.

N	Direct solve	SEM inverse	D.o.f.
N=4	7.241619e-05	4.607383e-06	81
N=6	9.484310e-07	2.703998e-09	169
N=8	1.976962e-10	1.138766e-12	289
N=10	8.856289e-13	3.308540e-13	441
N=12	1.160633e-13	1.927155e-13	625

Table 1: Convergence behavior

Solution time wise, this algorithm is fairly competitive to solving classic SEM using conjugate gradient solver(with no preconditioner). Choosing different tolerances for CG sovle such that the solution error is comparable with the direct solver, the timings are shown in table 2.

N	Direct(err)	SEM-Iter.(err)	Direct(time,s)	SEM-Iter.(time,s)
N=6	9.4843e-07	1.3211e-07	1.4010e-03	3.4709e-03
N=8	1.9770e-10	4.4944e-10	1.0770e-03	5.1010e-03
N=10	8.8563e-13	4.9215e-12	1.6313e-03	1.9790e-02
N=12	1.1606e-13	8.8005e-12	2.3076e-03	3.7499e-02

Table 2: Solution time

4 Conclusion and extension

An investigation into this hierarchical solver in the settings of spectral element method is conducted, and preliminary tests prove the feasibility of this idea. As [Martinsson, 2013] argue, their method uses more memory than an iterative method. However, this method is expected to be fast, with asymptotic complexity for the building stage to be $O(N^1.5)$, and $O(N \log N)$ for the solving stage.

A couple questions were raised and could be interesting to look into

- How to extend this method to other equations? The extension to Poisson's and Helmholtz is straightforward as it was already outlined in the original paper.
- How does this algorithm scale? Is it really as claimed? Further study and careful timings are needed.
- Compared to SEM direct-inverse, solution produced by this method loses about two digits for the same polynomial order. Why is this the case?

References

M.O. Deville, P.F. Fischer, and E.H. Mund. *High-order methods for incompressible fluid flow.* Cambridge University Press, Cambridge, 2002.

P.G. Martinsson. A direct solver for variable coefficient elliptic pdes discretized via a composite spectral collocation method. *Journal of Computational Physics*, 242:460 – 479, 2013. ISSN 0021-9991.

5 Appendix

Two Python notebook files are uploaded to the submission page.

- File `q` has my implementation of the fast solver as described. In the second cell the variable N controls the polynomial order and can be changed to an integer. The variable E denotes the element number in one direction of the 2D domain (e.g. $E = 2$ means there are in total 4 elements). This code has quite a bit hard coding specially for the structure of boxes defined above, so changing E will not work.
- File `s` has the SEM solver implementation. Similarly N in the second cell denotes the polynomial order. With this code element number can be changed arbitrarily, provided that there are same number of elements in the two directions and the domain has the same length in the two directions.
- In the two files the first cell has the setup for spectral element method operators.