Efficient Dynamically Adaptive Mesh

Aaron Ellis

Undergraduate Research Project: Final Report
Under the Supervision of Moysey Brio and Dustin Ditchen
Summer 2004

Introduction

Numerical solutions of partial differential equations are found by approximating the solution at a discrete set of points. Without using a sufficiently large set of data points we may not discover all features of the solution. In order to avoid this problem it may seem clear, add more points. However, in doing so another problem arises, computer runtime and memory may exceed practical limitations. Changing from a hundred points to cover a thousand points over the same interval could add minutes even hours to computation time. We know many solutions to PDEs only contain certain areas of interests while in the rest of the domain it varies little. There have been many attempts to exploit this feature to increase computer efficiency. We have begun to study one of these methods involving dynamically adaptive meshes.

This method involves moving points within the mesh, while doing this we are not adding more data points but instead simply focusing more points to the area of interest. To determine these areas of interests we have used a monitor function. A monitor function considers various aspects of a solution, such as the first and second derivatives using this information to change the location of points within a given domain. In order find these points we first solved for x and y values, such as in the example below [1]:

$$x_\tau = \nabla' \cdot (w \nabla' x),$$
$$y_\tau = \nabla' \cdot (w \nabla' y),$$

Where our adaptive mesh monitor function is

w=√((1+(β1*u)^2+(β2*u')^2+ (β3*u'')^2))

Goal

It is our goal to continue the ongoing research to find the most efficient and accurate way to implementing this method to problems related to optical devices. While Dustin Ditchen configured such an equation using MATLAB to find areas of interest I have begun the task of running the monitor function for efficiency and accuracy. It has been be my job to test and discover the most efficient configuration of the monitor function for these given equations in order to create an adaptive mesh that interprets a solution using as little points as possible.

Monitor in single dimension

The first testing was performed over a simple Gaussian spike. Our monitor function depends on three quantities: amplitude, first derivative, and second derivative given weights of β1, β2, and β3, respectively. My next task was running the monitor against time, testing how fast we could push the equation to run while keeping the number of necessary points at a maximum. Past work [1] had suggested using only the weight of β2 (gradient variable) within the monitor function. We felt placing the gradient, laplacian and PDE solution in equilibrium would give the most accurate and efficient dynamically adaptive mesh to the solution curve. From our results we determined placing values of equal weight for all three β values is far more efficient and faster than β2 alone at 1. We also determined that while equal weighting beta values seemed better, it was also faster and more efficient when the reference was taken from β1 and it being 10. See Table 1.

β1	β2	β3	numpts	comprat	iter
1	0	0	52	0.0234	340
0	1	0	94	0.0222	454
0	0	1	100	0.0094	262
0.01	0.0054	0	28	0.0905	155
0.1	0.05	0.01	46	0.0468	95
1	0.54	0.11	82	0.0243	141
10	5.4	1.1	96	0.021	129
10	5.9	0.9	88	0.0255	83
100	54	11	108	0.019	270

Table 1: Monitor in 1-D

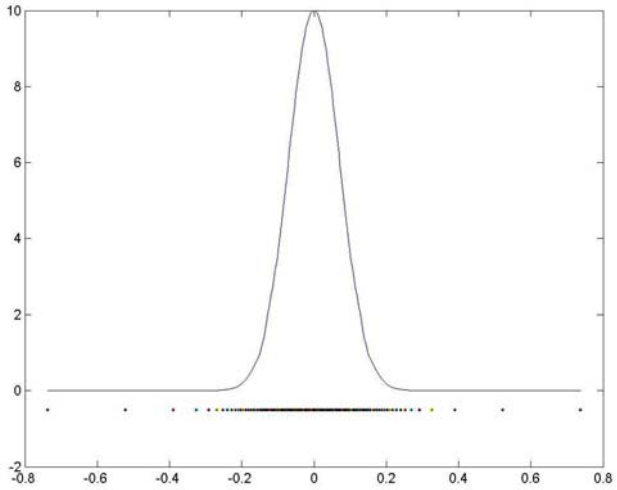

Figure 1: Graph taken from results in Table 1

It is apparent that our idea to equally weight the beta values was allowing for a faster and efficient monitoring function, but more improvement could still possibly be made. Now that we had found target beta values, I tested for a continuously changing beta function to determine new beta values for equal weight at each iteration. The results had no dramatic improvement but we still retain that continuously changing variable values could be important for use later on.

Adaptive mesh in single dimension through heat and advection equations

Discovering the targeted variable values for our monitor function allowed testing to proceed for more complex solutions. A Gaussian solution has served for the basis of our monitor function, because we knew the exact solution, and will continue to be our solution curve we test through the heat and advection equations while running our monitor. Again, we found having equal weight within the monitor during the heat equation tests produced result with minimal error. The advection equation tested very differently than expected. The most efficient results produced during the advection equation was turning off beta variables for amplitude and laplacian and leaving the

gradient variable (β2) at a value of one hundred. Some of these results can be seen below; initial values for beta are given when continuously changing reference is used. Adaptive reference is taken when finding either the max amplitude, gradient, laplacian at every interval, each beta value being adjusted after every iteration.

β1	β2	β3	Adp ref	2-norm Error
10	4.35	0.43	none	0.0051
230	100	10	none	0.0086
1	0.435	0.043	β1	0.0086
2.3	1	0.1	β2	0.0120
23	10	1	β3	0.1291
10	4.35	0.43	β1	0.0992
23	10	1	β2	0.1133
230	100	10	β3	0.1458
0	10	0	none	0.0037

Table 2: Results from Heat Equation in 1-D

β1	β2	β3	Adp ref	2-norm Error
10	5.8	1.04	none	0.6895
17.25	10	1.8	none	0.6231
1	0.58	0.104	β1	0.9927
1.725	1	0.18	β2	0.3233
9.61	5.57	1	β3	0.6938
10	5.8	1.04	β1	0.3291
17.25	10	1.8	β2	1.0917
96.1	55.7	10	β3	1.7461
0	100	0	none	0.0448

Table 3: Results from Advection Equation in 1-D

Runtime comparison for single dimension

Concluding our efficiency testing in one-dimension we followed with runtime comparisons between adaptive mesh code and standard fixed uniform code. The runtime for the heat equation was at best seven times slower than the uniform grid. This had been the case because as the heat equation runs the solution curve becomes flatter and the grid from the monitoring function becomes more like a uniform grid and has no real advantage. The runtime for the advection equation was approximately two hundred times faster than that of the uniform grid. As the spike in the solution curve moved, in order to keep with comparable error we had to increase the number of points to five thousand in the uniform grid as opposed to twenty-five for the monitor.

Monitoring Function in two-dimension

As with the monitor function in one dimension, we would test the adaptive two dimension mesh with equal weights in addition to simply turning the β variables on and off with values of one and zero. The results behaved as expected with equal weights providing far more efficient results with a minimal number of iterations. Again we saw having β1 as our reference value and with a factor of ten the best results were produced. We now felt we had sufficient values to test our monitor through the heat and advection equations in two dimensions. In addition to the Gaussian spike, we also tested the mesh over other typical optical waveguide structures.

Figures 2-6 represent results using our two-dimension adaptive monitor code to create meshes for structures common in optical waveguides.

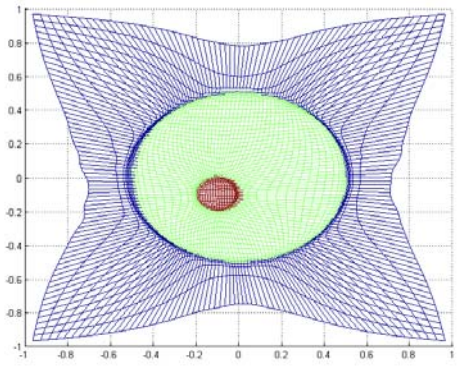

Figure 2

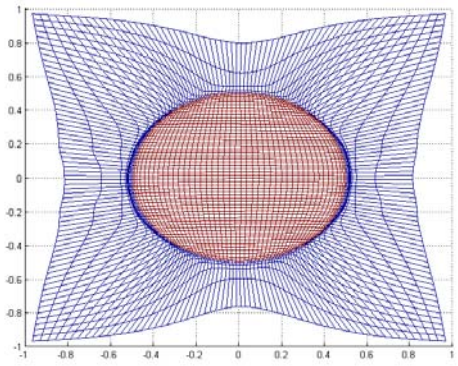

Figure 3

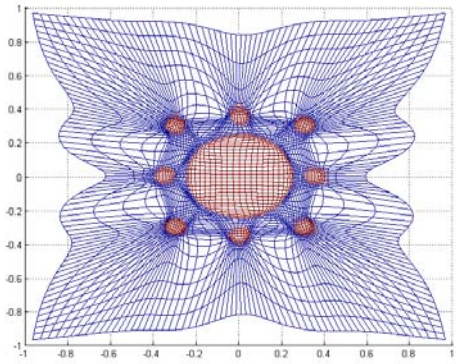

Figure 4

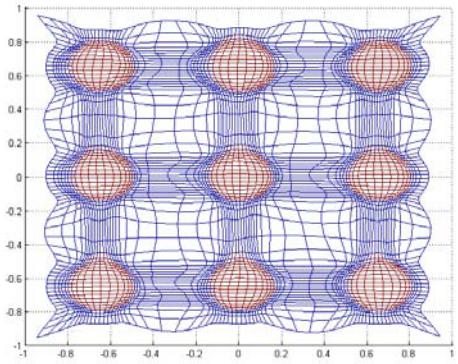

Figure 5

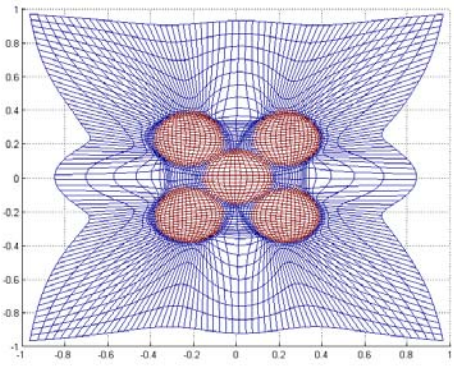

Figure 6

Heat and Advection Equations in two-dimension

Moving closer to our objective the testing procedure has become expanded to include many more situations and different factors. Beginning with simple monitoring function variables turned on and off (one or zero), we also are testing with equal weight with and without adaptive variable values, we are also constantly changing reference points and multiplication factors, adaptive time steps, different amplitude values and

number of total points in the solution, and even changing final time and converging tolerance.

The heat equation has in the past usually given us results with little trouble. However, it is apparent there are bugs in the code for our two dimensional monitor when running through the heat equation. Almost every situation we ran the 2-D monitor through failed as the solutions diverged.

The advection equation seemed not to be affected as the heat equation had in our monitor function. These results were much more ideal. With error values small, we were able to use these results and also find the computer runtime. Again we discovered our weighted values to be more efficient. Our previous reference point of β1 providing the best results were now joined with referencing our equal weights with β2. Both of which provided comparable efficient results with β2 providing a faster runtime. The most significant results are below:

β1	β2	β3	Adp. Ref.	Runtime	Error
1	1	0	none	98.232	2.0766
1	0.037	0.008	none	62.374	7.0845
2.726	1	0.206	none	92.685	2.5507
27.26	10	2.06	none	116.418	2.98230
13.232	4.85	1	none	115.9030	2.9866
132.32	48.5	10	none	108.2000	3.1346
2.726	1	0.206	b2	98.1840	1.4532
27.26	10	2.06	b2	121.0900	3.9314
10	1	0	b1	112.9020	1.0734

Table 4: Advection Equation in 2D

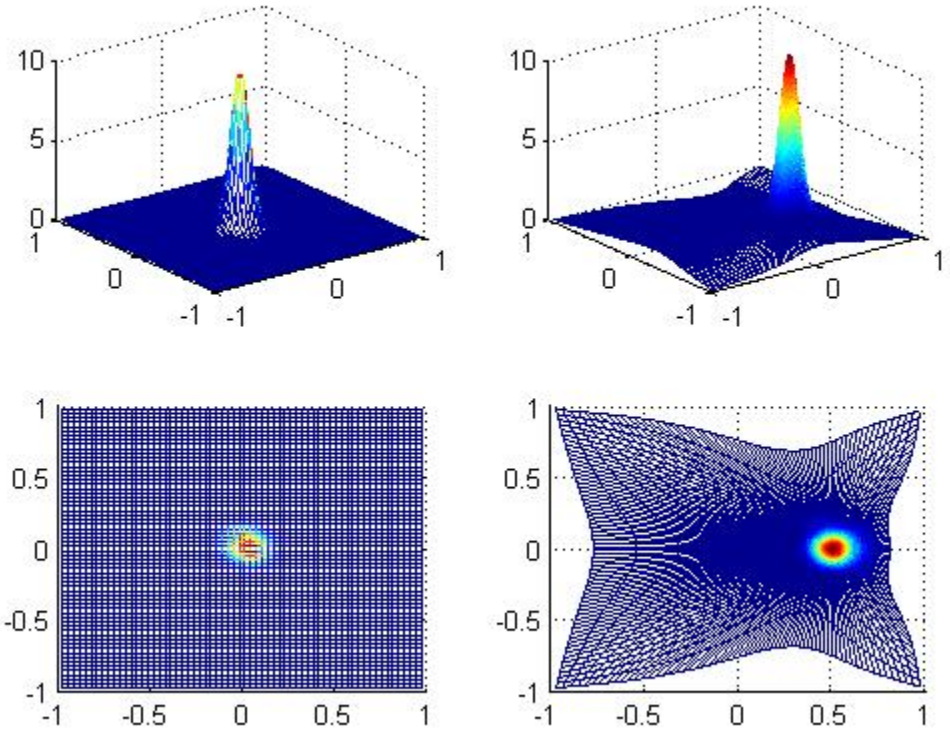

Figure 7: Graph taken from results in Table 4

Future Work/ Continuation

Currently Dustin Ditchen is reworking the current adaptive mesh code to include an orthagonality option for the adaptive mesh. As our mesh adapts and relocates points to areas of interest orthagonality is lost in the grid. Though he has not yet discovered a solution to this, it is the idea this new option would allow the points to move and still provide a grid with orthogonal cross intersections. Dustin is also trying to implement a scalar BPM (beam propagation method) in the code to run for various optical structures. It will be my job to test his code and adjust it code as I see fit.

We are also in the process of developing and debugging our adaptive mesh two-dimension code for the heat equation. We have already made numerous changes in our time stepping and sub class functions for earlier coding, we expect similar changes to be made for the heat equation in two-dimension. It is also very possible these improvements could change our advection equation results which will again have to be tested.

Continuing the project will also mean more runtime comparisons for our two-dimension results with the heat and advection equations. We expect to finally see a large discrepancy in the favor of the monitor. Finishing these results we will begin the task of adapting our monitor to include more solutions other than the current Gaussian we are using now. Becoming more adaptable for multiple solutions is key for real world usage. As our tests are many times long and time consuming, our goal remains to eventually convert the MATLAB code into the C language to speed up the testing process and for a more user friendly design.

References

[1] Ceniceros, Hector D. and Hou, Thomas Y. "An Efficient Dynamically Adaptive Mesh for Potentially Singular Solutions." Journal of Computational Physics **172,** 609–639 (2001).

[2] Knupp, Patrick and Steinberg, Stanley. Fundamentals of Grid Generation. CRC Press, Inc. 1993. p. 25-43