Sarah Elaine Hale

Spring 2015

Weekly Journal

1 Week 1: The Preliminaries

With a limited background in HTML as my only web experience, I began my journey learning JavaScript. The final goal is an interactive website in which others can explore and contribute to the research. For anyone interested in continuing this project or starting their own JavaScript iourney I have added a section at the end which contains the resources I found most useful.

1.1 Progress

I have started to learn the following languages:

- JavaScript: Allows interactivity and computation
  - numeric.js: JavaScript library for linear algebra
  - jQuery: JavaScript library for various purposes (not yet implemented, but was useful for understanding d3)
  - d3: JavaScript library for data visualization
- HTML(Hypertext Markup Language): Provides the most basic website layout
- CSS(Cascading Style Sheets): Describes how to render each piece of the HTML
- SVG(Scaling Vector Graphics): D3 uses these to render graphics, CSS can style these as well

Using the languages above and the provided Python code, I have created a website that renders a contour plot of the error for T2G1. It matches the plots generated by Python and is done completely within the browser. I have an interactive version, but it uses JavaScript prompts (pop-up boxes). They seem obtrusive and I think using input boxes on the page will be more user-friendly. Currently the contour plot is mostly hard coded for T2G1, which I plan to change in the future.

1.2 Misadventures

Using the already existing Python code directly on the website will probably not happen. From what I have found, Python seems to be used for server-side purposes. We wanted something that can be used client-side.

Even though I can generate the contour plot, it has no labels or axes. Using the inspector in Google Chrome, I can see that they are generated.

I can achieve simultaneous plotting, but it needs to be refined. Right now, I have separate CSS id's for each simultaneous graph. This works for a set number of graphs. If we want to allow the user to add another plot to compare, something more dynamic will need to be created.

1.3 Items of Interest

SVG uses a coordinate system that starts from the top-left corner instead of the bottom right. I chose to generate the alphas in reverse order for this reason.

I do not know if this is at relevant, but I noticed a trend in standardized errors. You can stack them to create a continuous image. You can also rotate 180 degrees and get a continuous image side-by-side.

1.4 Upcoming Goals

- Create a toy program that uses linear algebra
- Get user input from the page directly instead of using prompts
- Add axes/labels to the plots (successfully)
- Create a re-usable contour plot function
- Create a slide-show of pre-generated graphs

2 Week 2: Completing Step 1 of the Project

This week was focused on improving and expanding on what I have already done, as well as preparing for the presentation.

2.1 Progress

- Created a toy program that uses linear algebra
- Got user input from the page directly instead of using prompts
- Added axes to the plots
- Used new CSS stylesheet
- Fixed simultaneous plotting so that each graph deletes separately
- Updated documentation to be more thorough
- Added descriptions to improve user experience

2.2 Misadventures

There are no major misadventures to note this week. I did struggle with the axes more than I would have liked to. I made sure to comment the code to explain each line for any future developers/when I look back on it.

2.3 Upcoming Goals

- Add labels to the plots
- Create a re-usable contour plot function
- Create a slide-show of pre-generated graphs
- Use a slide-bar to get user input
- Include a short mathematical description

3 Week 3: Focusing on First Case of Interest

I am continuing to work on the first case of interest, this week focusing on a comprehensive page about it.

3.1 Progress

- Added slider bar to control amplitude
- Developing a more generalized contour plot function
- Fixed a memory leak caused by setting innerhtml to blank rather than removing the SVG objects in the code
- Wrote code to generate the values of the Hessian stability

3.2 Misadventures

The slider bar generates all large errors for some amplitudes. This only occurs in a new implementation I have created to try to generalize the contour plot function. Overall, generalizing the contour plot functions is more work than I anticipated. The webpage works well in Chrome using the original contour function. However, in Firefox and IE11 it seems to have a memory leak.

3.3 Upcoming Goals

- Add labels to the plots
- Finish a re-usable contour plot function
- Include a short mathematical description
- Identify and fix memory leak
- Implement the Hessian using the generalized contour plot function

4 Week 4: Continuing With First Case of Interest

I am continuing to work on Step Two of the project.

4.1 Progress

- Completed a re-usable contour plot (color map) function [prototype, maybe]
- Implemented the Hessian using the color map function
- Uploaded first draft of website to my page

4.2 Misadventures

Switching the first angle to be slightly larger than zero did not fix the error that occurs at amplitude -1. I am currently in the process of trying to speed up the code, but I am having trouble treating the axes and data points separately.

4.3 Upcoming Goals

- Expand on mathematical outline
- Speed up code

5 Week 5: More T2G1

I spent this week continuing to work on the T2G1 case webpage.

5.1 Progress

- Improved speed (removed axes in favor of explaining in words and changed so the slide bar will change the fill but not redraw each rectangle)
- Added checkbox to compare graphs
- Added option to generate a high quality plot if not comparing graphs
- Added checkboxes for the error/Hessian plots
- Added $\alpha = \phi$ line
- Added rendering after changing dimension
- Changed size to depend on screen size for better scalability

5.2 Misadventures

Can we discuss the Hessian plot to ensure I have a good description?

5.3 Upcoming Goals

- Add colorbars with scales
- Explain interpretation of the Hessian
- Decide what else to include on this page for Step Two

6 Week 6: Finishing Step Two

This week involved working toward a more complete and polished webpage for the simplest case.

6.1 Progress

- Added colorbars with scales (dynamic for Hessian)
- Explained interpretation of the Hessian
- Completed description on the simple case

6.2 Items of Interest

Is there a meaning to the way the Hessian scale changes?

6.3 Upcoming Goals

- Work toward other cases
- Possibly using some git repository

7 Week 7: Moving Forward

Now that Step Two is finished, we are figuring out what to include in Step Three.

7.1 Progress

There has been no change in code. There has been discussion and thought put into how to leave the code for future contributors.

7.2 Upcoming Goals

- Work toward other cases, most likely ignoring the Hessian due to possible change
- Make code more modular and ready to be used by others
- Update descriptions to be more specific
- Minor changes to code such as removing the pink for errors

8 Week 8: Modularizing the Code

This week I have been working on modularizing the code and implementing the new T2G1 functions in JavaScript.

8.1 Progress

- Coded multiple functions in JavaScript, currently in a new .js file.
- Began to modularize code to generalize for any data

8.2 Misadventures

- In trying to generalize the plot function, I have some interconnectedness between dynamic sizing and the data function. I am trying to decide the best way to split them apart such that they are not static.
- In the Python code, in pipgrad and other functions, it seems that we are taking the dot product of two different size vectors.

```
Hybrid Code:
function pipgrad(alphas)
{
    var d = alphas.length;
    var cosas = alphas.map(cos);
    out = Uint8Array(d);
for(var j=0; j<alphas.length; ++j)
{
        //cosasnot = numpy.concatenate((cosas[:j],cosas[j+1:]))
        //out[j] =- sin(alphas[j])*numpy.product(cosasnot)

}
    return out;
}
```

8.3 Upcoming Goals

- Implement all Python code in JavaScript and test newly implemented code against Python to ensure functionality of numeric.js
- Make code more modular and without reducing functionality (dynamic sizing)

9 Week 9: Separating Code and Moving Forward

9.1 Progress

- Separated plotting code and data generation code
- Translated most of the T2G1 Python to JavaScript

9.2 Misadventures

I had a bug from trying to initialize an array. In JavaScript, it is best to use dynamic allocation in most cases.

9.3 Upcoming Goals

- Finish T2G1 and test code
- Start and finish T2G2 code, noticing which functions are not needed for plotting
- Complete Overview and T2G2 descriptions

10 Week 10: T2G2

I have completed all necessary JavaScript code for T2G2. In this coming week, I will only need to focus on descriptions and documentation.

10.1 Progress

- Completed translating Python to JavaScript
- Completed T2G2 error plots
- Competed T2G2 Hessian plots
- Added regularization to both T2G1 and T2G2

10.2 Upcoming Goals

- Complete Overview and T2G2 descriptions
- Complete commenting the code

11 Week 11: Final Presentation

I have completed all necessary JavaScript code, descriptions, and documentation.

11.1 Progress

- Completed Step Three

11.2 Upcoming Goals

- Touch up anything mentioned during presentation
- Pass the torch

12 Resources

Tutorials:

- http://alignedleft.com/tutorials/d3
- https://www.dashingd3js.com/
- http://www.codecademy.com/