# NORTHERN ARIZONA UNIVERSITY RESEARCH EXPERIENCE FOR UNDERGRADUATES 2007 PAPER

DAVID J. RIEKSTS

ABSTRACT. This paper is a report on research done for the basins of attraction for a PdE on C3 using Newton's method. It contains a brief review of Newton's method along with some other definitions. The next section will deal with the computer program that was made to do the research. This will be followed by some of the results of the program. And the final section will be a brief discussion of further possibilities.

## 1. INTRODUCTION.

1.1. **Newton's Method.** Newton's method is a way to aproximate the zeros of a function. The equation for Newton's Method:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Basic example:
$f = x^2, x_0 = 1$
first iteration: $x_1 = 1 - \frac{1^2}{2*1} = \frac{1}{2}$
second iteration: $x_2 = \frac{1}{2} - \frac{(\frac{1}{2})^2}{2*(\frac{1}{2})} = \frac{1}{4}$
third iteration: $x_2 = \frac{1}{4} - \frac{(\frac{1}{4})^2}{2*(\frac{1}{4})} = \frac{1}{8}$
etc.
Newton's will converge toward 0 with more and more iterations.

However we will be using the PdE $-Lu + u^3 + su = 0$. So our we will have to use a variety of Newton's method for the PdE.
The equation for Newton's Method on the PdE:

$$u_{k+1} = u_k - Jac(u_k)^{-1}N(u_k)$$

Where $u_{k+1}$ and $N(u_k)$ are vectors and $Jac(u_k)^{-1}$ is a matrix,

1.3. **Other definitions.** There are a few other definitions that will be used in this paper. They are as follows:
The basin of attraction for a point, p, is the set of all points that converge to p using Newton's method
For the PdE, the basin of attraction for a point u is the set of all $u_0$ that converge to u using Newton's method.
The Morse Index of a solution is the number of negative eigenvalues of the negative Jacobian at that solution
The signature is the number of negative eigenvalues of the negative Jacobian at the intial guess

## 2. The Program

2.1. **Program Parameters.** The first thing I would like to look at in this section are the program's parameters. The program will only go over a slice of the four dimensional region given by u1, u2, u3 and s. So the most the parameters are to define what slice is to be taken. The first one is s0 which will determines the value of s. There are also parameters u1i, u1f, u2i, u2f which will define the window of the slice. In the current program configuration, we only look at the plane u1 + u2 + u3 = 0, so it is not necessary to have any more initial parameters to define the area on which the guesses will be made. On a side note, the reason we use this plane is because several solutions lie on it.
Two other parameters are used to define how many guesses are made, precision1 and precision2. As of now, these are set up in such a way that the amount of guesses in the x direction is precision1 multiplied by the magnetude of the range of u1. And it is set up similarly for u2 and precision2.
The final parameter that is used is n, which will determine how many iterations are allowed before Newton's method terminates.

2.2. **Basics of the Program.** This section will give a quick over view of the way the program works. The first thing the program will do is to make numerous initial guesses in a grid pattern on the plane u1 + u2 + u3 = 0. With each initial guess, the main function will call Newton's method and find a solution. If no solution is found in the specified number of iterations, n, the Newton's method function will return a flag to alert the main loop that no solution was found. In this case the flag is u = [Inf; Inf; Inf]. Once the main loop has a solution (or a flag) it will check a matrix, A, of solutions to see if it has already been found. If it has been found before, the matrix will already have a distinct number for that solution. This number will be put into a different matrix, guesses, in order to display where each guess converged. If the matrix A does not have that solution, a number will be assigned to it based on the size of A. Once all the intial guesses have been made, the program will display a picture of the basins of attraction by using the number assigned to each guess as a distinct color. The program will also format the matrices in several different ways and output them to different files.

Pseudocode for the main loop:
$e1 = [\frac{1}{\sqrt{2}}; \frac{-1}{\sqrt{2}}; 0]$;
$e2 = [\frac{1}{\sqrt{6}}; \frac{1}{\sqrt{6}}; \frac{-2}{\sqrt{6}}]$;
precision1 = precision1*abs(u1f - u1i);
precision2 = precision2*abs(u1f - u1i);
for j = 0 to precision1
u1 = u1i + j*abs(u1i - u1f)/precision1;
for k = 0 to precision2
u2 = u2i + k*abs(u2i - u2f)/precision2;
u0 = u1*e1 + u2*e2;
get the signature
call Newton's method. Will return the solution
get MI
put solution and the signature into matrices
end for loop
end for loop
format output and put it into a file

## 3. Results

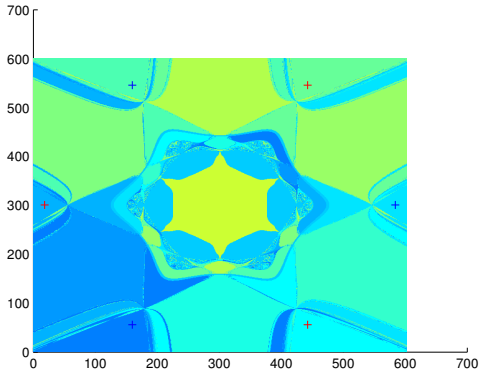


FIGURE 1. s = .001, u1 and u2 range from -3 to 3, precision = 100, colored by the solution

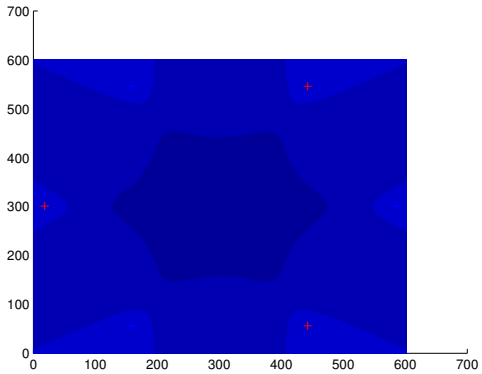


FIGURE 2. s = .001, u1 and u2 range from -3 to 3, precision = 100, colored by signature

3.1. **Types of Output.** The program can format its output in several ways. They are listed below:
The basins of attraction colored by the solution: This is the most basic format. Each solution is assigned a number. When a guess is made and a solution found, that guess will be assigned the number. Then when it is time to display the basin of attraction, the number is converted into a color. There is a lot of infromation in this format. However, it is often hard to abstract this infromation due to the complexity of the picture. This lead me to try a few different formats. See figure 1.
The signatures of the guesses: While the initial guesses are being made, the program will find the signature of each guess. This infromation is put into a matrix, sigs. The picture generated by this matrix is much easier to understand since there are only four possible signatures and hence only four possible colors. However, this does not contain any information as to where Newton's method coverges since the signature is computed using the guess. See figure 2.
The boundary of the solutions: Once we have the matrix guesses, we can make a picture of the boundary of the basins of attraction by formatting the matrix. For each initial guess, the program will look at all of the guesses that are next to it. If all of the neighboring guesses have the same solution, the color of that initial guess will be set to a certain color reserved for areas inside the boundary. If any of the neighboring solutions are different, it will keep its original color. See figure 3.
The areas where Newton's method did not converge in the specified number of iterations: Because the Newton's method function will reuturn a flag if it does not converge in a certain number of iterations, we can display the areas where Newton's method does not converge. The motivation for

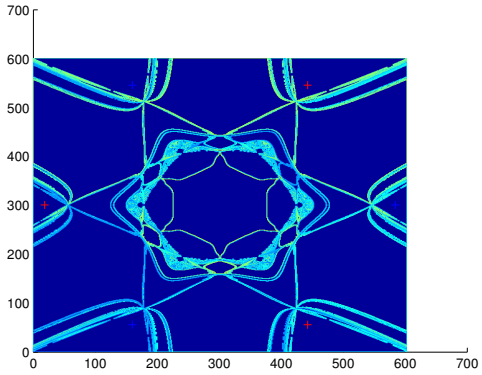


FIGURE 3. s = .001, u1 and u2 range from -3 to 3, precision = 100, The boundary of the basins

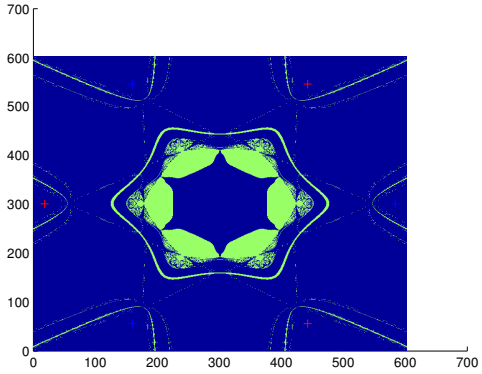


FIGURE 4. s = .001, u1 and u2 range from -3 to 3, precision = 100, Where Newton's does not converge

doing this is that Newton's method will have trouble converging close to the boundary between two basins and it would be very useful if the we could define the boundary. However, this perticular picture does not show any new information. The areas where Newton's method does not converge can be clearly seen in the picture of the Morse Index. See figure 4.

The basins of attraction colored by the Morse Index of the solution: This perticular picture has a lot of information in it. It is really three pictures in one: a picture of the Morse Index, a picture of the areas where Newton's method does not converge and a picture of the norm of the solution. The reason the picture contains the areas where Newton's does not converge is rather simple. The Morse index is calculated using the soltion. If Newton's does not converge, there is no solution to use. But the picture still requires that a number be used for those guesses, so we have to give a specific number for the areas that do not converge. The reason the picture of the norm and the picture of the Morse index are the same is not so clear. We know that if the norm of two solutions is the same, there Morse index will the same. But it is not clear if the converse is true. However, in all the experiments I have run thus far, the two pictures are the same with the exception of subtle shading, the pictures are the same. See figure 5.

The basins of attraction colored by the norm of the solution: This picture, along with the Morse index picture, were attempts to color the picture in such a way that the different symmetries of a solution would get the same color as that solution. But as mentioned before, this piture gives no additional infromation to the picture of the Morse index. See figure 6.

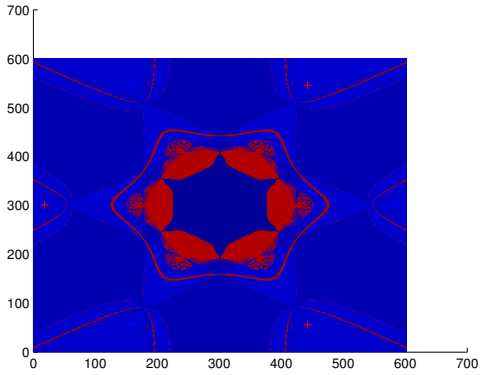


FIGURE 5. s = .001, u1 and u2 range from -3 to 3, precision = 100, colored by Morse index Note: The red area in the figure shows the convergance

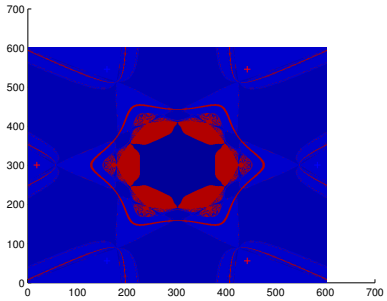


FIGURE 6. s = .001, u1 and u2 range from -3 to 3, precision = 100, colored by the norms of the solutions

The areas where the signature of the guess is different than the Morse Index of the solution: Because the signature is keep in the matrix sigs and the Morse index is calculated using guesses, we can look at both simultaniously to come up with a matrix that has the areas where the Morse index is the same as the signature as one color and the areas where they are different as another. This is again an attempt to find another way of relating to the boundary. If the signature and the Morse index are different it is likely to be due to the chaos of a fractal boundary. It is also suspected that using damped Newton's method will shrink this area, possibily giving a better picture of the boundary in the process. See figure 7.

There were a several other experiments that I ran. I will only be mentioning a few. The first was at an s value just after the last bifurcation point, -1.5. This was to get a picture that would have all the solutions that would still be near a point of interest. As you can see from the picture there are many more color in the picture and thus more solutions. See figure 8.
Another picture I thought would be appropirate to generate would be one with a larger window, so I ran an experiment with the range of x and y four times what it usually is. However, I also cut the precision parameters by four, so the pictures have the same number of initial guesses. As you can see from figure 9, it appears that the interesting fractal boundaries occur in the original window of -3 to 3.
One of the better results I was when I tested a conjecture by Dr. Neuberger of Northern Arizona University. He suspected that part of the boundary was the areas where the negative Jacobian was not invertible. I ran an experiment to these areas and see if it looked like the boundary. However, because it was so hard to hit the points where the determinate of the negative Jacobian was zero,

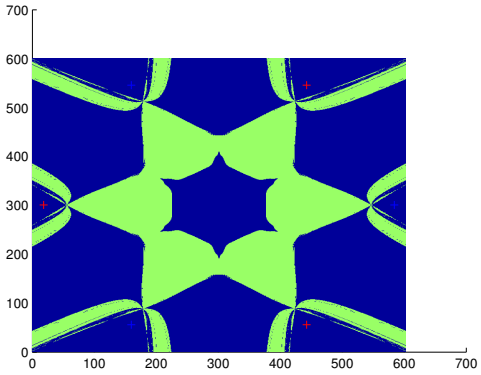


FIGURE 7. s = .001, u1 and u2 range from -3 to 3, precision = 100, The area where the signature is different than the Morse index

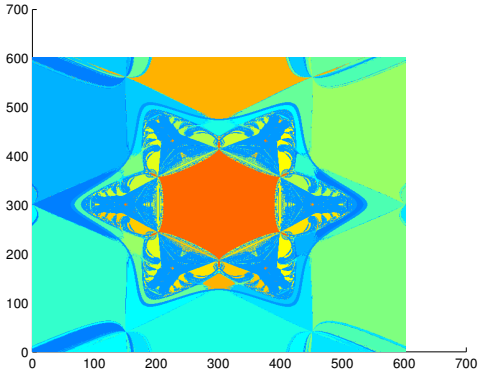


FIGURE 8. s = -1.5001, u1 and u2 range from -3 to 3, precision = 100

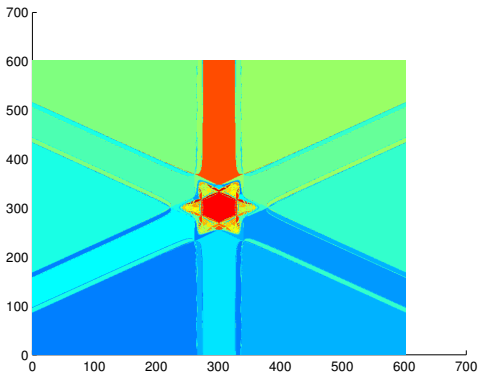


FIGURE 9. s = -1.9999, u1 and u2 range from -12 to 12, precision = 25

the first picture doesn't show the whole area. (See figure 10.) But even here, the similarities to the boundary picture can be seen. The biggest difference is in the middle where there is no boundary but the negative Jacobian is not invertible. However, this could be due to how close the s value is to a bifurcation point.

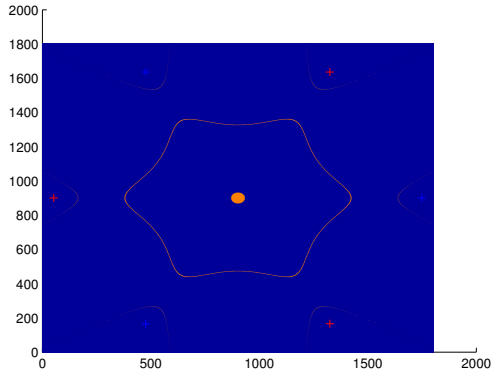


FIGURE 10. s = -.001, u1 and u2 range from -3 to 3, precision = 300

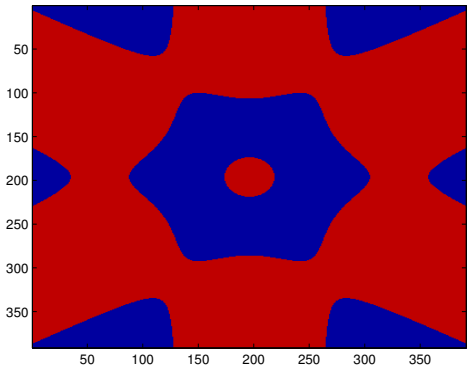


FIGURE 11. s = -.001, u1 and u2 range from -3 to 3, precision = 65

A much better picture to see this boundary is figure 11, which is a picture of where the determinate is negative and where it is positive. The border between the two colors should be where the determinate is zero, and thus where the negative Jacobian is not invertible. It should be mentioned that neighter of these where tested numerically.

## 4. FURTHER POSSIBILITIES

There are still many more possibilities for further research. It is most likely that this project was just getting to the point where a lot of research could be done when it ended.
Graph specific solutions : A very easy addition that could be made to the program would be the ability to graph the basin of attraction for a single solution. Slightly harder would be to graph all the solutions of a single symmetry.
Try damped Newton's method: Another easy possibility would be to try the damped Newton's method and see if the area where the signature and the Morse index are different shrinks. In the process, it would be very easy to check if damped Newton's changed any other pictures.
Try other potential boundaries: From the pictures there is a clear relationship to where the negative Jacobian is not invertible and at least part of the boundary. However, it is also clear that it does not define the whole boundary. Given another potential boundary, this could be a relatively easy experiment to run. Although exactly how easy would depend on the potential boundary somewhat.
Another experiment relating to the boundary that would be require little additional code would be

to modify the boundary format to make everything on the boundary one color and everything off the boundary a different color. After this is would be a simple matter to compare the boundary with a potential boundary. Graphing the areas where the potential boundary is different from the boundary would reveal a lot about there relationship to each other.
try other planes: Currently the program only takes a slice from the plane u1 + u2 + u3 = 0. By adding a constant to the equation for the plane, it would be possible to make animations of a single s value moving through different planes. It would also currently be possible to make an animation of the current plane moving through different s values.
try different graphs: The program is set up in such a way that part of it can be immediately used for graphs other than C3 and part of it would need to be modified. It would be somewhat involved, but should be quite possible to automate the code to work for any graph it is given. If this were done, it would probably be best to link the code to the graph programs that Dr. Neuberger, Dr. Sieben, and Dr. Swift have developed at Northern Arizona University.
Make smarter initial guesses: This one is another of the more involved possibilities. But making smarter guesses should substantially speed up the code. One possibility for making smarter initial guesses would be to run the code with the precision parameters set to a low number. After this, the program could get the boundary and try more initial guesses closer to it. The output would be a little more tricky to format because the guesses would not be evenly spaced. But perhaps the area inside the boundary could just be filled with the color found inside the boundary at a lower precision. There are many other possibilities for research in this area that I have not even given thought. It is indeed unfortunate that the program ended with so many possible avenues of investigation left open.

*E-mail address*: driek801@kutztown.edu

19 Sycamore Lane Oley, PA 19547, USA