Jacob Leonard

Math 467

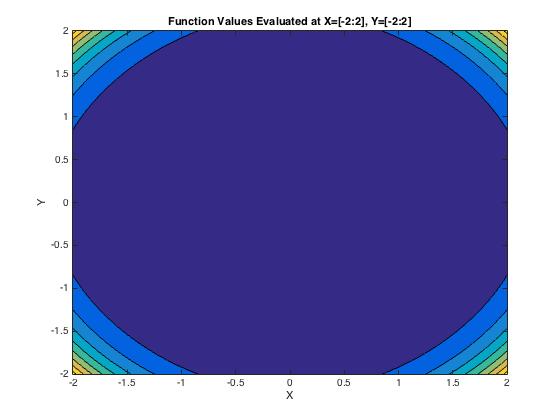
Project #1

Imaginary Analysis

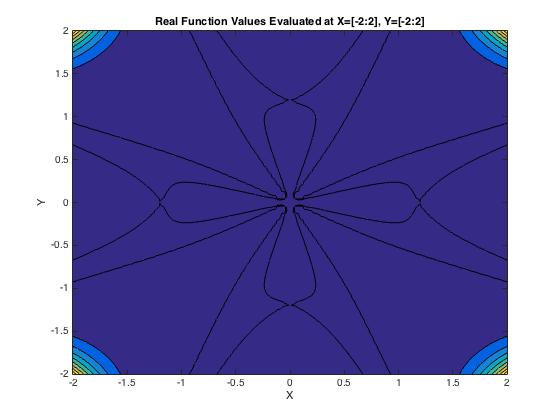
1. Analytically Compute the Gradient and Hessian of the Function:

Evaluating the function at all the points on the intervals:

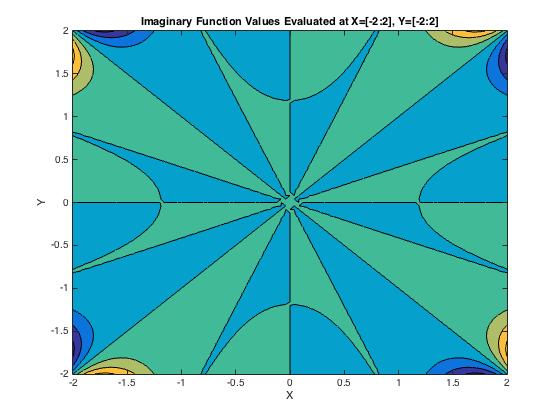
1. Graph of the function , z = (x+iy)



1. Graph of the real values



1. Graph of the imaginary values



1. Gradient:
2. Hessian:

1. Roots:
2. Gradient Along Root Values;
   1. = [0,0]

x = [-2,2], y = ix+1

x = [-2,2], y = ix-1

* 1. = [0,0]

x = [-2,2], y = ix+i

* 1. = [0,0]

x = [-2,2], y = ix-i

1. Gradient Near Root Values: D = +.04
   1. = [-1.4698, - 1.4698i]

x = [-2,2}, y = ix+1.04

* Increasing the y value along this root by .04 produces a negative real X gradient and a negative imaginary Y gradient
  1. = [1.1107, 1.1107i]

x = [-2,2}, y = ix-.96

* Increasing the y value along this root by .04 produces a real positive X gradient, and positive imaginary Y gradient
  1. = [-0.1789 - 1.2698i, 1.2698 - 0.1789i]

x = [-2,2}, y = ix+i+.04

* Increasing the y value along this root by .04 produces a negative X gradient, positive real valued Y gradient and negative imaginary Y gradient
  1. = [-0.1789 + 1.2698i, - 1.2698 - 0.1789i]

x = [-2,2}, y = ix-i+.04

* Increasing the y value along this root by .04 produces a negative Y gradient, negative real valued X gradient and positive imaginary X gradient

1. Gradient Near Root Values: D = -.04
   1. = [1.1107, 1.1107i]

x = [-2,2}, y = ix+.96

* Decreasing the y value along this root by .04 produces a positive gradient that is isometric to its parallel y= , meaning the function is symmetric about y = ix
  1. = [-1.4698 , - 1.4698i]

x = [-2,2}, y = ix-1.04

* Decreasing the y value along this root by .04 produces a positive gradient that is isometric to its parallel y= , meaning the function is symmetric about y = ix
  1. = [-0.1789 + 1.2698i , -1.2698 - 0.1789i]

x = [-2,2}, y = ix+i-.04

* Decreasing the y value along this root by .04 produces a gradient that is isometric to its parallel y=
  1. = [-0.1789 - 1.2698i , +1.2698 - 0.1789i]

x = [-2,2}, y = ix-i-.04

* Decreasing the y value along this root by .04 produces a gradient that is isometric to its parallel y=

1. Hessian Along Root Values:

x = [-2,2], y = ix+1

x = [-2,2], y = ix-1

* The hessian of the function along the parallel lines y = ix+1, y = ix+1 produces isometric matrices, meaning the function denatures at the same quadratic rate

x = [-2,2], y = ix+i

x = [-2,2], y = ix-i

* The hessian of the function along the parallel roots y=ix+1, y=ix-1 produces isometric matrices, meaning the function denatures at the same quadratic rate

1. Hessian Near Root Values: D = .04
   1. = [-44.8995, -44.8995i, -44.8995i , 44.8995]

x = [-2,2}, y = ix+1.04

* Increasing the y value along this root by .04 produces a highly negative hessian, except a positive quadratic Y term
  1. = [-21.7160, -21.7160i, -21.7160i ,21.7160 ]

x = [-2,2}, y = ix-.96

* Increasing the y value along this root by .04 produces a highly negative hessian, except a positive quadratic Y term
  1. = [30.6966 -11.4484i, 11.4484 +30.6966i,  
      11.4484 +30.6966i, -30.6966 +11.4484i]

x = [-2,2}, y = ix+i+.04

* Increasing the y value along this root by .04 produces a symmetric hessian with inverse quadratic terms
  1. = [30.6966 +11.4484i, -11.4484 +30.6966i,  
      -11.4484 +30.6966i, -30.6966 -11.4484i]

x = [-2,2}, y = ix-i+.04

* Increasing the y value along this root by .04 produces a symmetric hessian with inverse quadratic terms, large negative Y and large positive X

1. Hessian Near Root Values: D = -.04
   1. = [-21.7160 , -21.7160i, -21.7160i, 21.7160 ]

x = [-2,2}, y = ix+.96

* Decreasing the y value along this root by .04 produces a hessian equal to the line y = ix-.96, meaning the quadratic decomposition is symmetric about y = ix
  1. = [-44.8995, -44.8995i, -44.8995i , 44.8995]

x = [-2,2}, y = ix-1.04

* Decreasing the y value along this root by .04 produces a hessian equal to the line y = ix+1.04, meaning the quadratic decomposition is symmetric about y = ix
  1. = [30.6966 +11.4484i, -11.4484 +30.6966i,  
      -11.4484 +30.6966i, -30.6966 -11.4484i]

x = [-2,2}, y = ix+i-.04

* Decreasing the y value along this root by .04 produces a hessian equal to the line y = ix-i+.04, meaning the quadratic decomposition is symmetric about y = ix
  1. = [30.6966 -11.4484i, 11.4484 +30.6966i,  
      11.4484 +30.6966i, -30.6966 +11.4484i]

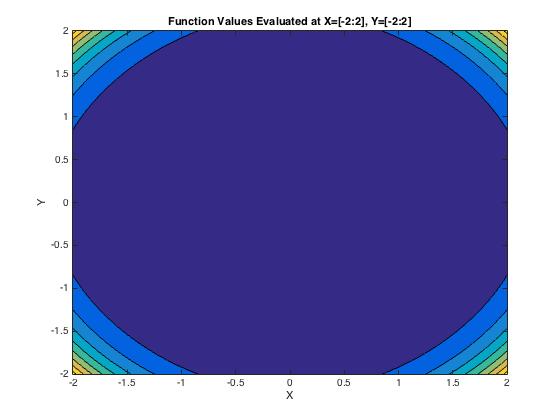
x = [-2,2}, y = ix-i-.04

* Decreasing the y value along this root by .04 produces a hessian equal to the line y = ix+i+.04, meaning the quadratic decomposition is symmetric about y = ix

Real Analysis

The above analysis looked at the function over the complex plane, but for the use of algorithms within Matlab, the following analysis will look at the function:

Evaluating the function at all points on the intervals:



1. Gradient:
2. Hessian:
3. Roots: z = [1,0], [-1,0], [0,1], [0,-1]
4. Gradient Along Root Values:
   1. = [0,0]

x = 1, y = 0

x = -1, y = 0

* 1. = [0,0]

x = 0, y = 1

* 1. = [0,0]

x = 0, y = -1

1. Gradient Near Root Values: D = .04
   1. = [0.0769, 1.2810]
   2. = [-0.0769, 1.2810]

* The gradient at these two points show the function is symmetric about y and opposite about x
  1. = [0, 1.5285]
  2. = [0, 1.0663]
* The gradient at these points shows that the function has an x gradient of 0 about x=0 and positive in the y direction

1. Gradient Near Root Values: D= -.04
   1. = [0.0769, -1.2810]

* The gradient at this point has the same X gradient and negative y gradient, meaning the function is negative symmetric about y=0
  1. = [-0.0769, -1.2810]
* The gradient at this point has the same X gradient and negative y gradient, meaning the function is negative symmetric about y=0
  1. = [0, -1.0663]
* The gradient at this point is the negative gradient at x = 0, y = -.96, meaning the function is negative symmetric about x = 0
  1. = [0, -1.5285]
* The gradient at this point is the negative gradient at x = 0, y = -.96, meaning the function is negative symmetric about x = 0

1. Hessian At Root Values:

* All of the hessians are equal at the roots of the function, showing that the minimum of the function is dictated by the quadratic terms, and denatures in the same way

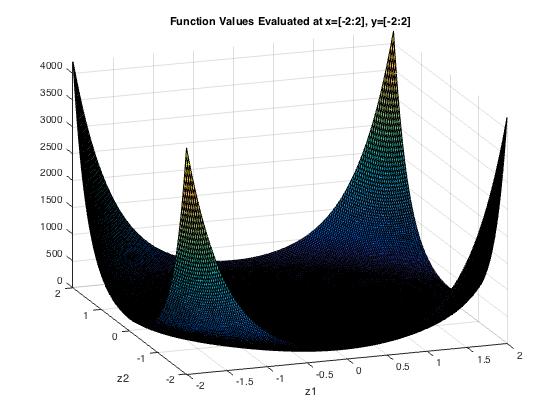
1. Hessian Near Root Values: D = .04

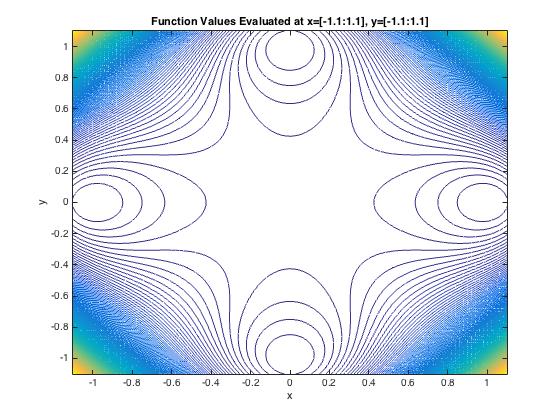
* These two hessians have the same quadratic growth, but denature linearly inversely
* These two hessians both have zero linear terms and positive quadratic terms, and so both increase about x = 0, y 1

1. Hessian Near Root Values: D = -.04

* The hessian at this point is equal to the hessian at x = -1, y=.04, meaning the function is quadratically symmetric about y=-.04x
* The hessian at this point is equal to the hessian at x = 1, y=.04, meaning the function is quadratically symmetric about y=.04x
* The hessian at this point is equal to x=0, y=-.96, meaning the function is symmetric about x = 0
* The hessian at this point is equal to x=0, y=-.96, meaning the function is symmetric about x = 0

Analysis of code and Matlab scripts:



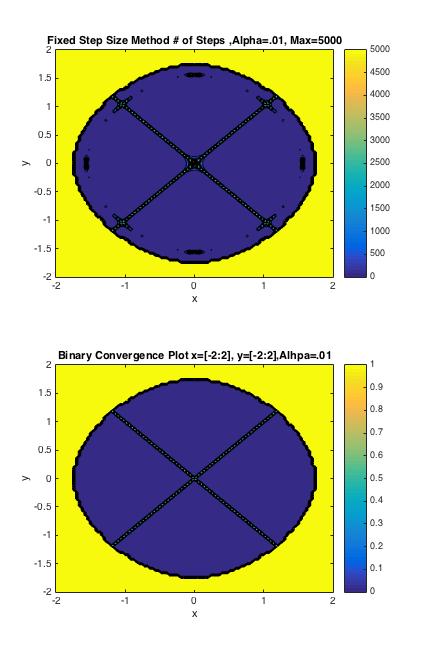


This graph shows a closer look at the origin, and the roots surrounding the origin

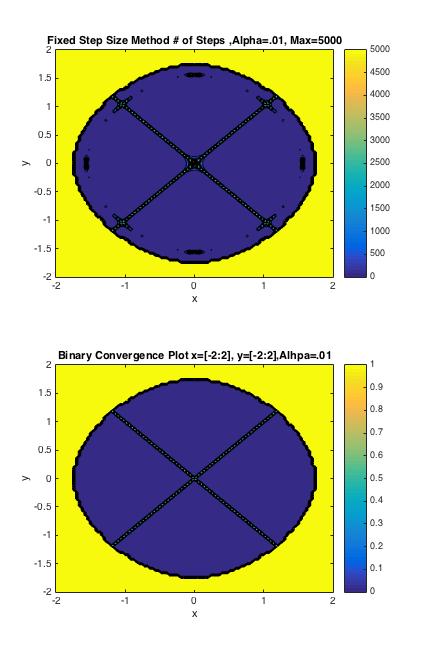
Roots: (-1,0), (1,0), (0,-1), (0,1)

1. Fixed Step Size
   1. This code employs the use of a fixed step size gradient descent method to converge to the minimal solution
   2. The Fixed Step Size Gradient Method requires the step size to be larger than 0, but smaller than 2/lambda(max) if it is to converge. Larger step sizes require fewer steps to converge, but have less accuracy. Smaller lambda take more steps to converge, but increase the accuracy of the estimates
   3. For convergence, the algorithm was considered to have converged if:
      1. The gradient is 0
      2. The iteration value is NaN or Inf
      3. The function value falls within the tolerance
   4. For the following plots, I choose alpha = .05, .01, .005, .001, .0005 (yellow=high, blue=low)

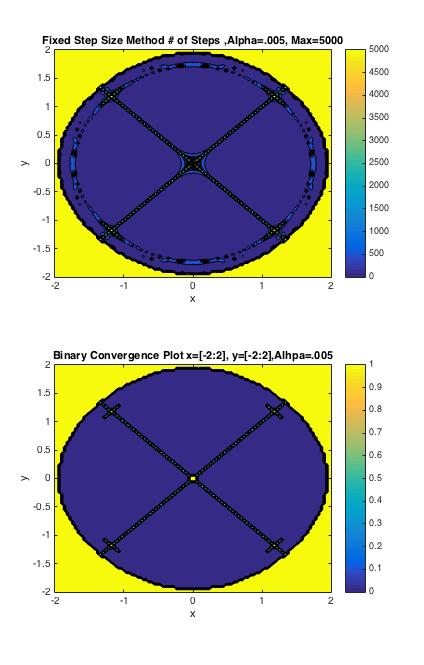
Alpha = .05



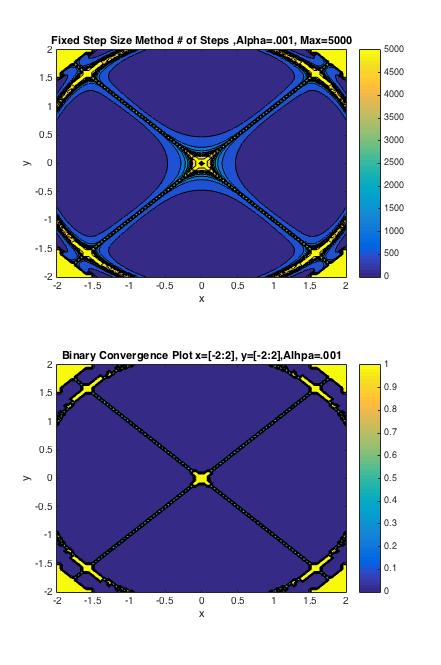
Alpha = .01



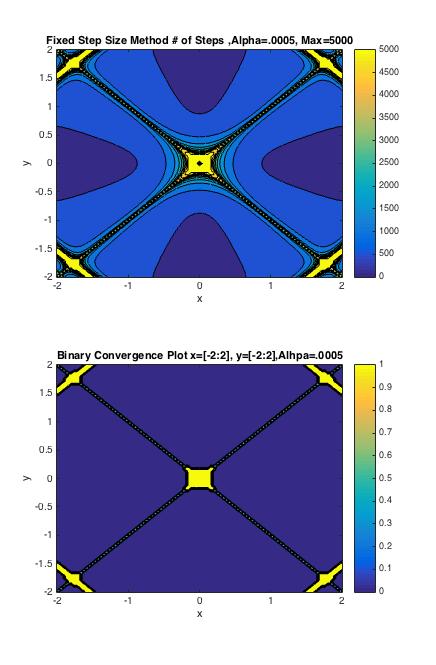
Alpha = .005



Alpha = .001

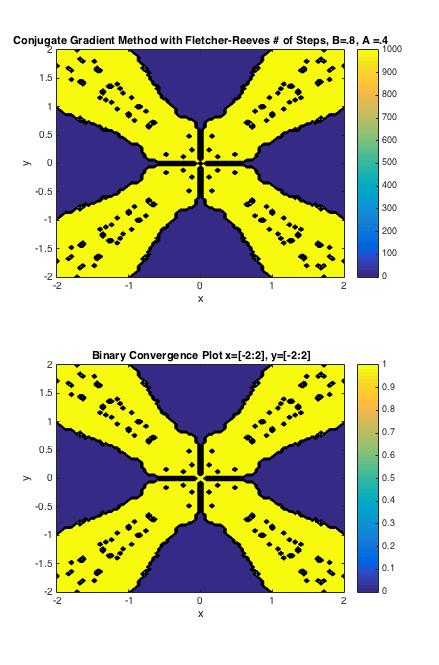


Alpha = .0005



* 1. The first plot shows the number of steps it took the algorithm to converge to a value, with blue being low, and yellow being high. The maximum number of steps is 5000. The second plot shows a binary convergence plot for convergence. If the algorithm converged to the minimum, then it was assigned to 0, and if it diverged or converged to the local maximum at (0,0) it was assigned a value of 1.
  2. You can see from the two plots above that the size of the step involved in the algorithm determines how the various points converge. With a larger alpha, it takes fewer steps for a point to “converge”, but the room for error and over jumping is much greater. With a smaller alpha, the number of steps it takes to get to a minimal value is much larger, but the minimal value is much closer to a true minimum.
  3. Considering the way the algorithm is structured, and the way in which the rules for convergence or divergence were crafted, the algorithm has trouble avoiding the local maxima at (0,0). With smaller step sizes, this error is minimized, but not avoided due to the symmetric nature of the surface. Along the valleys dissecting the roots, the gradient has the highest value, and so if the step size is too large, the algorithm has the possibility of following this path directly to the origin.

1. Conjugate gradient method with Fletcher-Reeves formula and Backtracking
   1. This code employs the use of conjugate gradient directions to dictate the descent of the algorithm to minimum solutions, with the fletcher reeves formula to determine the step sizes of the subsequent directions, and the backtracking algorithm to determine the step size of iterations.
   2. For convergence, the algorithm was considered to have converged if:
      1. The gradient is 0
      2. The iteration value is NaN or Inf
      3. The function value falls within the tolerance
   3. The values for the backtracking algorithm were placed at B = .8 A = .4. Note: smaller values for B and A cause the algorithm to converge faster, but with less accuracy. This is due to the fact that the backtracking algorithm descends to a larger step size fast with smaller B and A, but reduces the accuracy of the most efficient step size. Because of this, the tolerance was used to determine when the backtracking algorithm found an optimal step size
   4. Therefore, further exploration into the optimality of backtracking values should be explored

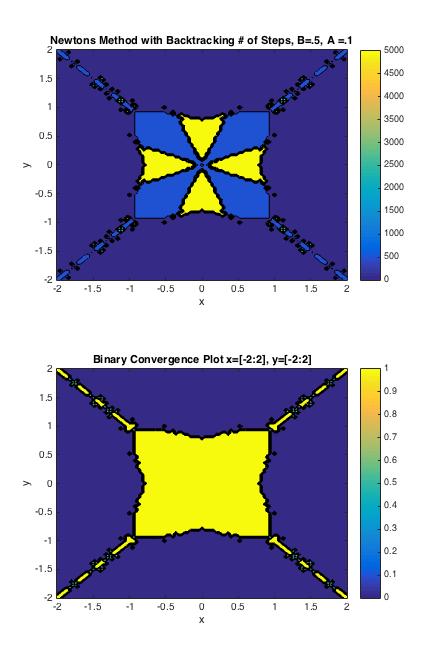


* 1. The first plot shows the number of steps it took the algorithm to converge to a value, with blue being low, and yellow being high. The maximum number of steps is 100 due to processing time. The second plot shows a binary convergence plot for convergence. If the algorithm converged to the minimum, then it was assigned to 0, and if it diverged or converged to the local maximum at (0,0) it was assigned a value of 1.
  2. It can be seen in the two plots that every point that converged reached the minimum, indicating a higher success rate of the algorithm to find the global minimum among its candidates that converge. However, the algorithm has a much wider field of divergence, under the specified conditions, for values closest to the area of global steepest descent - lines y = x, and y = -x. The algorithm could be improved if the line search method was more exact, and not determined by a variable lower threshold.

1. Newton’s Method with Backtracking
   1. This code employs the use of Newton’s method with 1st and 2nd derivatives to determine the direction of the iterations, as well as backtracking to determine the optimal size of alpha at each iteration
   2. For convergence, the algorithm was considered to have converged if:
      1. The gradient is 0
      2. The iteration value is NaN or Inf
      3. The function value falls within the tolerance
   3. Due to computational issues with singular or near-singular matrices within Matlab, the inverse of the 2nd derivative was found using I = pinv(h)
   4. The values for the backtracking algorithm were researched and placed at B = .5, A = .1. Note: smaller values for B and A cause the algorithm to converge faster, but with less accuracy. This is due to the fact that the backtracking algorithm descends to a larger step size fast with smaller B and A, but reduces the accuracy of the most efficient step size.

Because of this, the tolerance was used to determine when the backtracking algorithm found an optimal step size

* 1. Therefore, further exploration into the optimality of backtracking values should be explored



* 1. The first plot shows the number of steps it took the algorithm to converge to a value, with blue being low, and yellow being high. The maximum number of steps is 5000. The second plot shows a binary convergence plot for convergence. If the algorithm converged to the minimum, then it was assigned to 0, and if it diverged or converged to the local maximum at (0,0) it was assigned a value of 1.
  2. It can be seen in the second plot that the starting values surrounding the region containing the minimums, we’re unable to converge due to the local maximum at the origin. In the first plot, it can be seen that the values in light blue, completed the algorithm with a given number of iterations, but eventually terminated due to the gradient equaling zero at the origin. This algorithm could be improved if the backtracking step sizes were smaller, and the gradient had more steps to diverge from the origin.