Abstract

Introduction

* Concepts involved
* Literature review
* Objective I think???
  + Maybe pose questions similar to systemic review
  + Add the desing goals and deliverables
  + Mention briefly how this will be done to seguay
    - Good code
    - End result of some tool if possible

Methods

* Simulation was used
  + Attempted to find links between different variables to explore how the algorithims were performing
* Design perspective Made robust python coding
  + End goal to serve as a tool, coding very generalkly to make it more applicatble
* Python implementation
  + Talk about libraries
  + Talk about sympy
  + Talk about numerics and calculations when using a gradient descent
    - It also mention small rounding errors in python, that may become prevalent in iterations
* Talk about limitations taken for the project
  + AS PER THE POWERPOINT FAM
* Psudocode
  + Have images of all the code and labelled
  + Variables used
    - Define the variables as per their original algorithm
      * Say how in typical case you would want e to be zero, representing the middle part has converged
    - BR
      * Define what the best response it, make reference to intro
      * mention how it is calculated in python
        + probably need a pseudocode
    - grad
      * mention it is symbolic use, so grad is a defined function that is calculate once
  + How variables were chosen
    - Ie alpha and e
      * Define error, give example
  + Talk about runtime in coding
    - Attempted to make as efficient as possible
    - In house python function <<<<< MAYBE NUMPY CITATION THAT ITS BETTER
* Testing procedure
  + Talk about what examples are used
    - Haver some sort of table that summarizes everything together, shows what each example is and gives the iformation relating to it
  + Say what the decided initial conditions, alpha, etc are for each of the fuinctions
    - ALSO MENTION THE STOP CONDITION
      * Didn’t want code to crash / run indefinitely, if over 300 000 iteratiions it is ignored
  + Talk about for ranges what was choses and why: for example, what range do we pick for alpha and the other stuff, etc
  + For the examples what you will do
    - Ie, one in depth to show how the parameters affect different things
    - The 3 other that are linked to show how they relate to one another

Results

* Results for one in depth
  + Show for one of the example as taken from (decided on the first one since the eq is closer so its not as annoying to use
* Results for multiple
  + Talk about the different exponents of all the other things
    - How does this affect the different parameters, pick the important graphs
    - <<<<<<maybe make a new one with everything together
* Talk about 3d for each of the cases
  + Show examples of the 3d plots
* Results for the ui thing maybe
* Also think about 3d 🡨

Discussion

* Limitations of each algorithm
  + Similar to what was said in the presentation
    - BR has some issues
      * Convergence accuracy
      * Talk about how complexity and timing is bad for this one
    - GP has some other issues
      * Talk about how for some alphas it might just not work
* Complexity analysis
  + - Talk about limitations with runtime for both
      * BR has some complexity added since it is finding the min at a point and needs function applied to entire set
      * Gradient descent does not have any
* Talk about the first example
  + How cerain parameters work
* Talk about the different ns
  + How do the compare to each other
  + >>>maybe new graph
* Talk about 3d graphs
  + See if there is some source for the ridges…like why do those occur
    - If you cant find source, find something that talks specifically about no knowing what initial conditions there are

Conclusion

* Next steps as well
  + Looking at more types of functions
  + Something that might be more useful is to look at a specific parameter,
    - This paper looked at breath, it was an introduction and study of the algorithms
    - Next paper can focus specifically on one variable
      * Ie the initial conditions, why they appear in that pattern, is there some way to define a better initial condition to allow a more efficient algorithm to be run
* Modified algorithms (ie when they are made to converge easier
  + Ie perturbed and the other one
    - They were looked at, but not in detail
    - Often conveged very fast and somewhat inaccurately in preliminary experiments
    - Can revisit to see whats useful
* Modifying gradient descent / br
  + Making br more efficient + accurate
  + Making gradient descent not as problematic with alpha
* Biggest issue is error and alpha, maybe running some sort of regression to figure out optimal values
* LOOK IN LITERATURE FOR FINDING OPTIMAL ALPHA<<<<< / optimal ics / optimal error