HW3

Friday, February 7, 2025 12:00 PM

1.1 The function value where your current point is right now is 10. You have picked a search direction and find the slope in that direction is -1 at your current point. You then find out that the function value after moving 2 units in that same search direction is 11. What do you know and what do you do next?

Phi = 10
Directional_derivative = -1
Alpha = 2
New_phi = 11

Answer: I essentially just bracketed and can now enter the pinpoint function/algorithm. I know that there is a minimum somewhere between the two points because the slope was negative, but after my step the phi value increased from its previous value showing that at some point in between those two points there was a minimum. I would call pinpoint to find that minimum. Pinpoint would set alpha_low as 0, ie at that initial point, and alpha_high as 2, what gave me the higher point. I could then do bisection, quadratic, or cubic fits to predict a new alpha in between the two, then evaluate for phi at the new point, if it is lower than the old one, evaluate the directional derivative at that point and proceed accordingly. Once pinpoint has finished I would continue my line search by evaluating my stop conditions and choosing a new search direction if they are not yet met.

1.2 Calculate by hand the *first* and *second* search directions for the following function if you were implementing conjugate gradients in your optimization (i.e. results should be two different vectors). Use the Fletcher-Reeves formula for β. Since a line search would be required for the first direction (and would be too much work by hand), assume $\alpha^* = 0.05$ for the first direction. You must show your work for credit (scan or include an image in your report).

$$x_0 = [1, 2, -1]^T, f(x_1, x_2, x_3) = x_1 x_2^2 x_3^2 - x_2 x_3^3$$

This is referring to the conjugate gradient method:

Algorithm 4.6 Nonlinear conjugate gradient

Inputs:
x_0: Starting point
τ: Convergence tolerance
Outputs:
x^*: Optimal point
$f(x^*)$: Minimum function value

```
k = 0                                                  Initialize iteration counter
while ||∇f_k||_∞ > τ do                                Optimality condition
    if k = 0 or reset = true then                      first direction, and at resets
        p_k = -∇f_k / ||∇f_k||
    else
        β_k = (∇f_k^T ∇f_k) / (∇f_{k-1}^T ∇f_{k-1})
        p_k = -∇f_k / ||∇f_k|| + β_k p_{k-1}           Conjugate gradient direction update
    end if
    α_k = linesearch(p_k, α_init)                      Perform a line search
    x_{k+1} = x_k + α_k p_k                            Update design variables
    k = k + 1                                          Increment iteration index
end while
```

Where Beta =

respect to A. One option to compute a β that achieves conjugacy is given by the *Fletcher–Reeves formula*,

$$\beta_k = \frac{\nabla f_k^T \nabla f_k}{\nabla f_{k-1}^T \nabla f_{k-1}}. \quad (4.57)$$

First Direction:

$P_k = \frac{-\nabla f_k}{\|\nabla f_k\|}$

$f(x_1, x_2, x_3) = x_1 x_2^2 x_3^2 - x_2 x_3^3$

$df = \left[\frac{df}{dx_1}, \frac{df}{dx_2}, \frac{df}{dx_3}\right]$

$df = \left[x_2^2 x_3^2,\ 2x_1 x_2 x_3^2 - x_3^3,\ 2x_1 x_2^2 x_3 - 3x_2 x_3^2\right]$

@ $x_0 = \begin{bmatrix}1\\2\\-1\end{bmatrix}$ $df = \left[2^2(-1)^2,\ 2(1)(2)(-1)^2 + 1,\ 2(1)(2)^2(-1) - 3(2)(-1)^2\right]$

$= [4,\ 4+1,\ -8-6]$

$df_0 = [4, 5, -14]$ → horizontal for convenience

$P_0 = \frac{[-4, -5, 14]}{\sqrt{4^2+5^2+14^2}}$

$P_0 = [-0.2598, -0.3248, 0.9094]$ → horizontal for convenience

$\alpha_0 = 0.05 \rightarrow x_1 = x_0 + \alpha_0 P_0 = [0.987, 1.9838, -0.9545]$ → horizontal for convenience

$\beta_1 = \left[x_2^2 x_3^2,\ 2x_1 x_2 x_3^2 - x_3^3,\ 2x_1 x_2^2 x_3 - 3x_2 x_3^2\right] \cdot \begin{bmatrix} x_2^2 x_3^2 \\ 2x_1 x_2 x_3^2 - x_3^3 \\ 2x_1 x_2^2 x_3 - 3x_2 x_3^2 \end{bmatrix}$

top @ k=1 $\beta_1 = \frac{x_2^4 x_3^4 + 4x_1^2 x_2^2 x_3^4 - 4x_1 x_2 x_3^5 + x_3^6}{} = 0.8227$

$\left[2x_1x_2^2x_3 - 3x_2x_3^2\right]$

top @ k=1

Btm @ k=0

$$\beta_1 = \frac{x_2^4x_3^4 + 4x_1^2x_2^2x_3^4 - 4x_1x_2x_3^5 + x_3^6}{x_2^4x^4 + 4x_1^2x_2^2x_3^4 - 4x_1x_2x_3^5 + x^6} = 0.8327$$

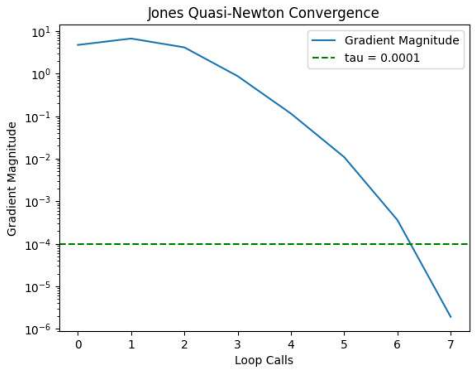
$$P_k = \frac{-\nabla f_k}{\|\nabla f_k\|} + \beta_k P_{k-1} \rightarrow P_1 = \frac{-\nabla f_1}{\|\nabla f_1\|} + \beta_1 P_0 \rightarrow \boxed{P_1 = \begin{bmatrix} -0.4716 \\ -0.5863 \\ 1.6711 \end{bmatrix}}$$

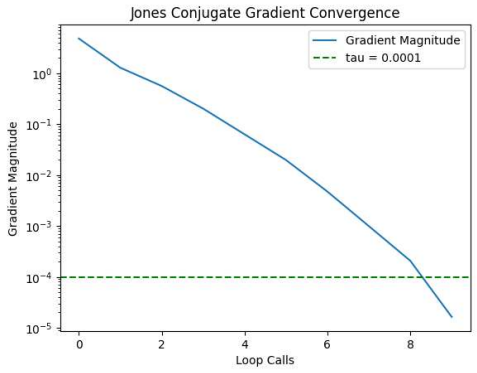
1.4 Create a table reporting the number of function calls

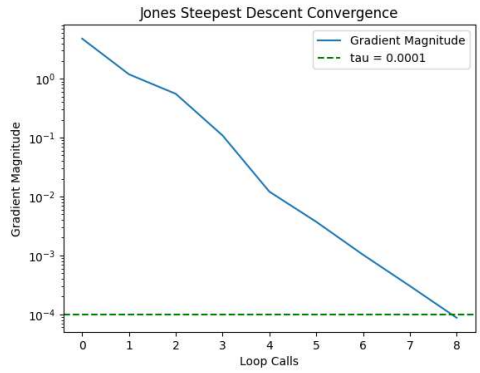
IterationTable

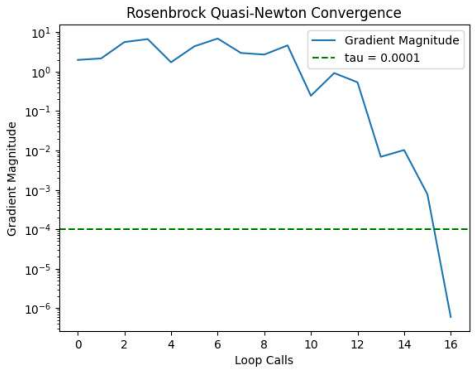
	Steepest Descent	Conjugate Gradient	Quasi-Newton	Scipy Minimize
Slanted Quad	400	231	79	25
Rosenbrock	16057	6302	235	91
Jones	318	201	134	43

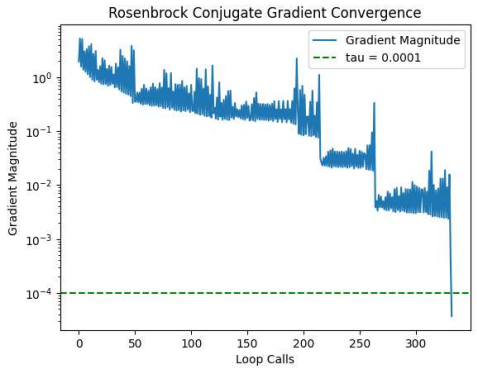
1.5 Convergence Plots- I chose the magnitude of grad f as my criteria with a tau of 1e-4

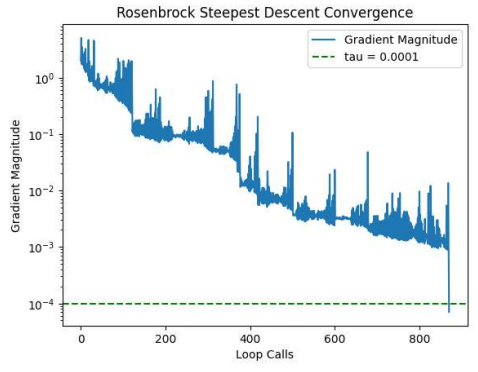


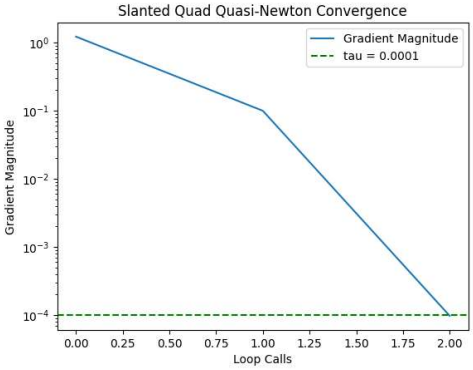


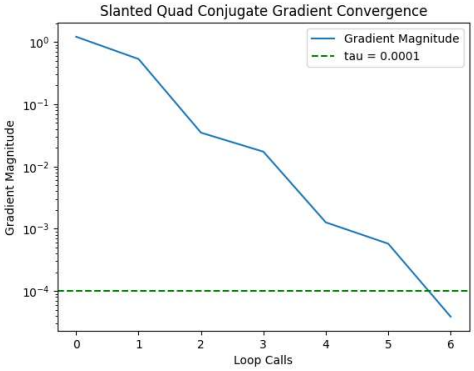


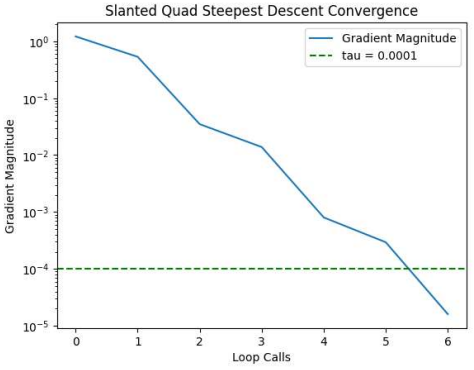












What I learned:
The mu 2 condition (or sufficient curvature) is really important. I had mine much too low, and was getting into weird issues with the pinpointing function as it was trying to find a slope that was a very small fraction of the initial slope and wouldn't be able to. I also learned the importance of matrix multiplication and shapes. Changing some of my * to @ in python for Quasi-Newton helped considerably and things actually behaved as expected. I should have put more thought into what I expected the shapes to be. Figure 4.51 in the textbook would have been very helpful, but I ignored it. Looking at it now I see we get the right sizes with the matrix multiplication AI recommended (when I was trying to debug). The Rosenbrock function is very difficult to find the minimum, I think due to a very narrow canyon and very high walls