To determine the variable step size (γ) for faster convergence in the gradient descent method for the function

****

**i) Constant Step Size:** In the constant step size approach, you use the same step size (γ) for each iteration of gradient descent. The update rule is:



The constant step size (*γ*) is typically chosen empirically. A common choice is to use a small value such as 0.01, 0.001, or other values that depend on the specific problem.

**ii) Decaying Step Size:** Decaying step size means that the step size (*γ*) decreases over time, which can help achieve faster convergence in the beginning and stability towards the end. One common choice for a decaying step size is:



where *α*0​ is an initial step size, and *k* is the current iteration.

The idea here is that the step size starts relatively large and gradually reduces, which can speed up convergence initially and prevent overshooting in later iterations.

**iii) Bold Driver Algorithm:** The Bold Driver algorithm is a heuristic approach for dynamically adjusting the step size. It uses two parameters: *α*1​ and *α*2​, where *α*1​>1 and 0<*α*2​<1. The update rule for the step size is as follows:

* If the new solution is better than the previous one (i.e. ,



* increase the step size:



* If the new solution is worse than the previous one, decrease the step size:



The Bold Driver algorithm dynamically adapts the step size based on the performance of the updates. This helps in achieving faster convergence by increasing the step size when the gradient descent is making good progress and decreasing it when needed.

The choice of *α*1​ and *α*2​ in the Bold Driver algorithm depends on the specific problem and may require experimentation to find suitable values for optimal convergence.

To compare these three approaches (constant step size, decaying step size, and Bold Driver algorithm), you can run gradient descent with each method on your function *f*(*x*). Monitor the convergence behavior, the number of iterations required to reach a certain accuracy, and the final solution obtained in each case. This comparison will provide insights into which method works best for your particular optimization problem.