1 Problem 4

For this problem I we need to first find the gradient and hessian of the function:

$$\nabla f(x) = \begin{vmatrix} 20x_1^2, & x_2^2 \end{vmatrix}^t$$

$$\nabla^2 f(x) = \begin{matrix} \frac{1}{20} & 0 \\ 0 & \frac{1}{2} \end{matrix}$$

To find the Lipschitz constant we write:

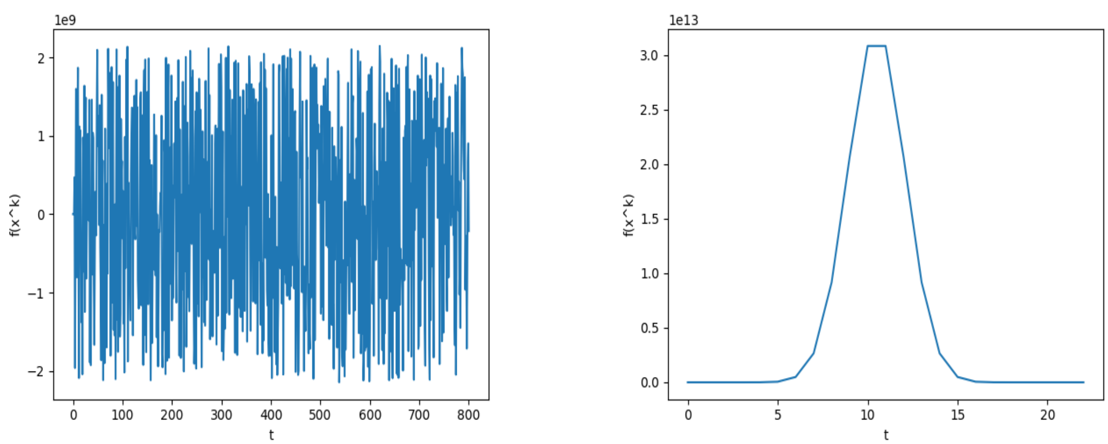
$$\begin{aligned} \|f(x) - f(y)\| &= \sqrt{400(x_1 - y_1)^2 + 4(x_2 - y_2)^2} \\ &= 2 * \sqrt{100(x_1 - y_1)^2 + (x_2 - y_2)^2} \\ &< 20 * \sqrt{100(x_1 - y_1)^2 + (x_2 - y_2)^2} \end{aligned}$$

So we have k = 1, and k2 = 1 as $S^t = \nabla f(x)$, so to ensure convergence for gradient descent we have:

$$0 < \gamma < \frac{2k_2}{k}$$

$$0 < \gamma < \frac{1}{10}$$

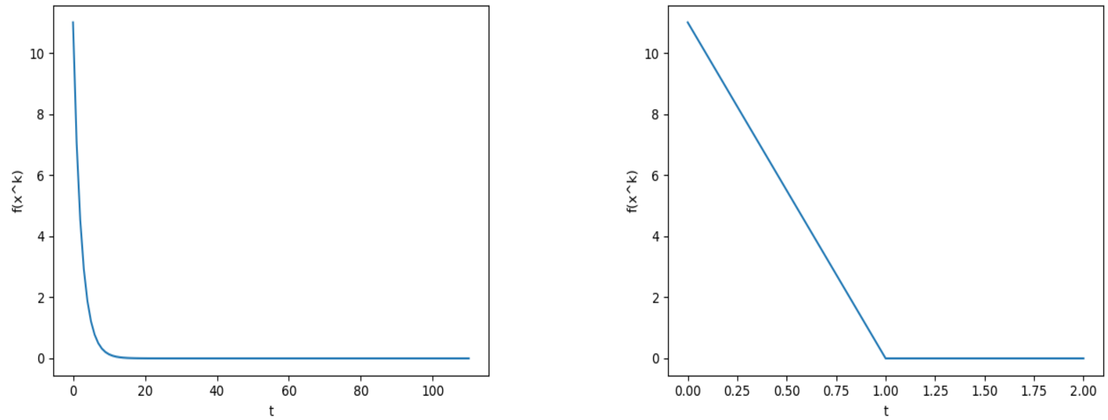
Below are the results for $f(x^t)vst$ after simulating for $\gamma = 1$, $\frac{1}{t}$, and .9, as well as the result for the method using Newton's Method:



(a) (b)

Figure 1: Gradient Descent with a) gamma = 1, b) gamma = 1/t

As we can see the function does not converge for $\gamma = 1$, but will for $\gamma = \frac{1}{t}$ after around 18 iterations.



(a) (b)

Figure 2: a)Gradient Descent with gamma = .9, b) Newton's Method with gamma = 1

gamma = .9 converges fairly quickly, and Newton's method converges in a single step (unsurprisingly). For the distributed case, I chose the consensus matrix P:

$$P = \begin{matrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{matrix}$$

If we represent each local copy of [x1, x2] as v1 and v2, the update equation then becomes:

$$v_1^{t+1} = P_{11} * v_1^t + P_{12} * v_2^t - \gamma^t \frac{1}{20} \Big| 20x_1, 0 \Big|^t$$

$$v_2^{t+1} = P_{21} * v_1^t + P_{22} * v_2^t - \gamma^t \frac{1}{2} \Big| 0, 2x_2 \Big|^t$$

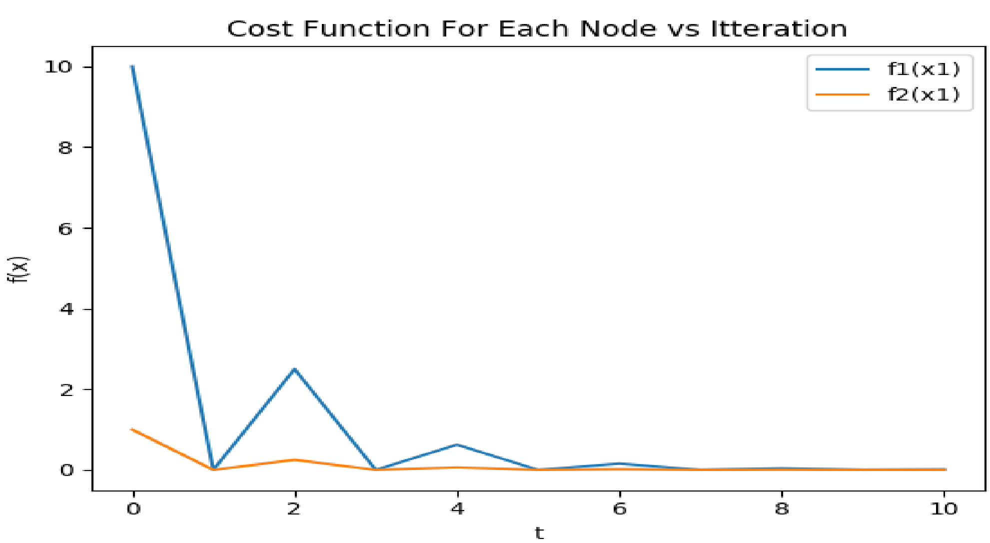


Figure 3: gamma = 1

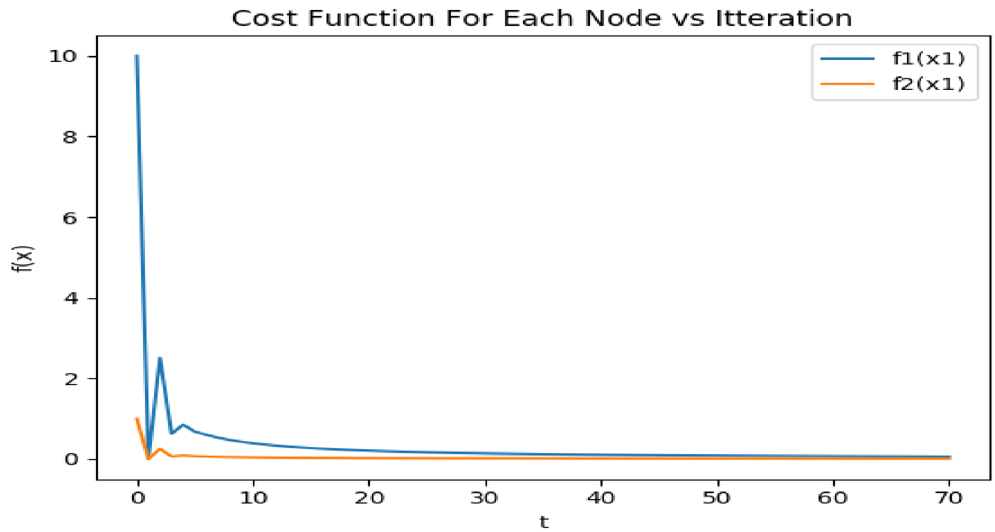


Figure 4: gamma = 1/t

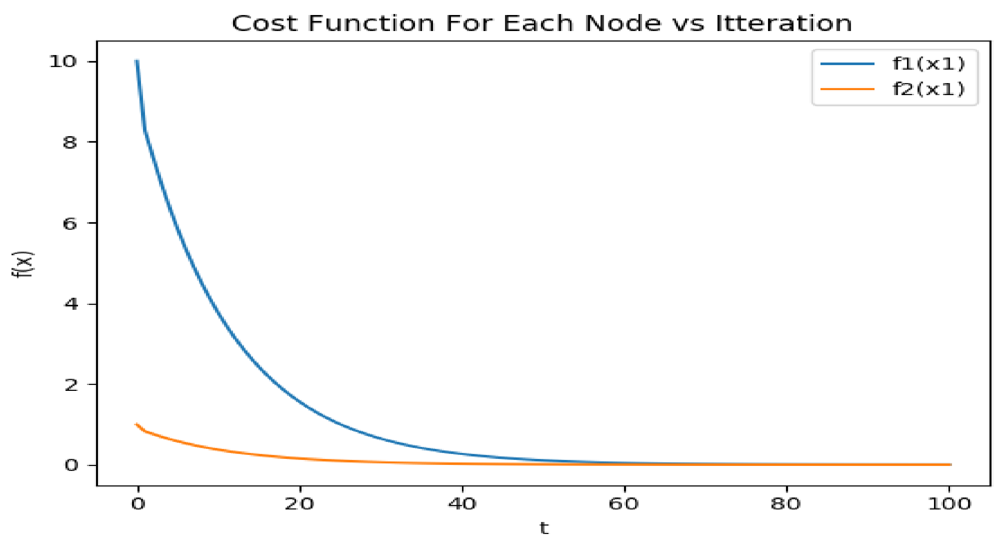


Figure 5: gamma = .09

As we can see, the cost function converges. For this algorithm, each node approaches the optimal solution from opposite ends (i.e. the x1 and x2 values are flipped between node 1 and node 2). The higher step sizes converges faster for the higher gammas, with gamma = 1 performing similarly to the non-distributed case where gamma = .09. None of these values converged in a single step, as f1(x1) tended towards zero more slowly, and thus the distributed Jacobi Gradient Descent doesn't perform as well