Equal contribution
KU-IDs: qfz440 / bdf506 / gks462
Date: February 25, 2023

Numerical Optimization - Handin 3

This paper seeks to investigate implementations of the algorithms `CG` and `Approximate Newton`. The `Approximate Newton` algorithm is tested for different schedules of η_k to see the convergence rate of the different schedules. At last, the original `Newtons Algorithm` and `steepest descent` are compared to the `Approximate Newton` algorithm.

1 Theory

Towards Cheap Orthogonalization

Inserting $v_k = -\nabla f(x_k)$ into the second to last equation above eq. 20 from the lecture notes yields the following results. Note that this equation is equivalent to eq. 20 and we have shifted $k \leftarrow k+1$ to make the indices match the desired results:

$$v_k = -\nabla f(x_k) \overset{(20)}{=} p_k + \sum_{i=0}^{k-1} \beta_i p_i$$
$$\iff \nabla f(x_k) = -p_k + \sum_{i=0}^{k-1} -\beta_i p_i$$

Now if we define $\gamma_i = \begin{cases} -1 & \text{if } i = k \\ -\beta_i & \text{otherwise} \end{cases}$ we can rewrite the above as follows:

$$\nabla f(x_k) = \gamma_k p_k + \sum_{i=0}^{k-1} \gamma_i p_i = \sum_{i=0}^{k} \gamma_i p_i \tag{1}$$

We furthermore wish to use this result to show that $\nabla f(x_k)^\intercal \nabla f(x_j) = 0$ for all $j < k$:

$$\nabla f(x_k)^\intercal \nabla f(x_j) \overset{(1)}{=} \nabla f(x_k)^\intercal \sum_{i=0}^{j} \gamma_i p_i = \sum_{i=0}^{j} \gamma_i \nabla f(x_k)^\intercal p_i$$

Finally using (19) of theorem 6 of the lecture notes we have $\nabla f(x_k)^\intercal p_i = 0$ for $0 \le i < k$. Since $j < k$ we can apply this result directly to finalize the proof:

$$\nabla f(x_k)^\intercal \nabla f(x_j) = \sum_{i=0}^{j} \gamma_i \nabla f(x_k)^\intercal p_i = 0 \quad \square$$

2 Correctness of our Implementation

To ensure correctness of our implementation we for a variety of different initialization tested whether the algorithm ϵ-converged to the theoretical optima of function f_1, f_4 and f_5. For the bounded parameters e.g. c_1 and ρ we always assert that they are within their constrains. We also did some qualitative comparisons with the results of Scipys `minimize` function with parameter `method=Newton-CG` and our implementation got similar results.

3 Experiments

All plots are based on quadratic η-schedule unless otherwise specified (see figure 1).

3.1 What functions do we experiment with?

We include f_1, `The Ellipsoid Function`. For $\alpha > 0$ it has a positive-definite hessian. This is stated in the case studies and it can be shown quite simply that the hessian is a diagonal matrix with entry i being $2\alpha^{\frac{i-1}{d-1}}$.
We exclude f_2, the `The Rosenbrock Banana Function`. In the case studies it says that for $d > 2$ the function is no longer uni-modal which means it does not have a positive semi-definite Hessian hence it is excluded as we are examining $d \geq 20$.
We exclude f_3, the `The Log-Ellipsoid Function`. While it is a composition of f_1 which is always positive (semi-)definite, it is not convex as mentioned in the case studies.
We include f_4. As the dimensions of the input are summed together, the hessian is diagonal, so verifying that each entry in the diagonal is strictly positive for all inputs shows positive definiteness. Using maple, we can get the second derivative of the term inside the sum to equal the following:

$$\frac{\ln(1+\mathrm{e}^x)\mathrm{e}^x + 100\ln(1+\mathrm{e}^x)\mathrm{e}^x + \mathrm{e}^{2x} + 100}{50000000(1+\mathrm{e}^x)^2}$$

As the denumerator and all terms in the numerator are strictly positive, we can conclude that the hessian is positive definite.
We include f_5, the sum of `Sum of Different Powers Function`. The second derivative of the last dimension is 0, so it is only positive semi-definite.

4 Results/Discussion

Evaluation on different schedules

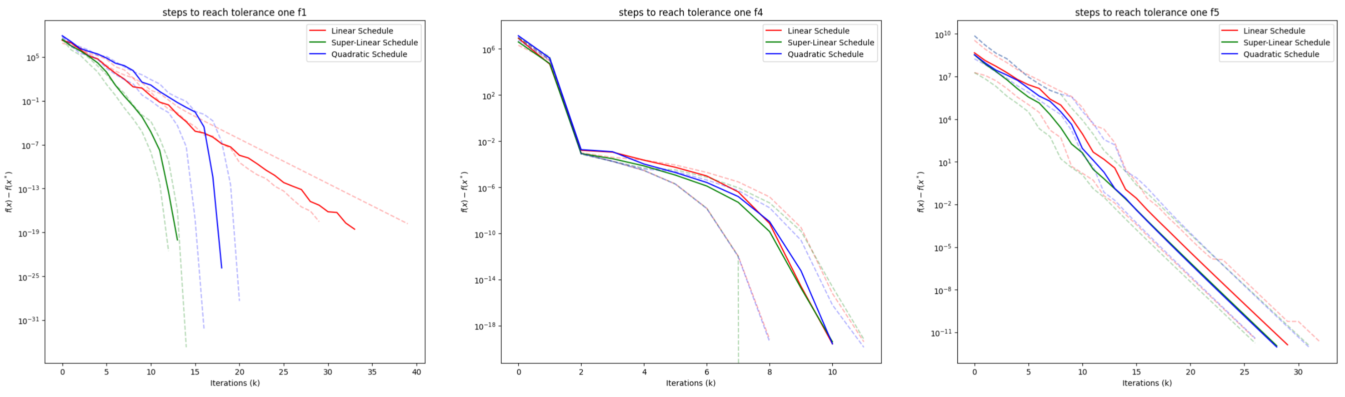

Figure 1: Plot of different schedules of η for `approx newton`. Median number of steps taken is the most significant line, while the transparent lines denote the minimum and maximum number of steps taken for each of the schedules.

By looking at figure 1 we see the `approx newton` under different schedules. To evaluate it, we take note of the median runs, however, the difference in performance can only really be seen in the first plot, where the super-linear schedule performs best, followed by the quadratic schedule and at last the linear schedule. For the other plots, we see that the difference in performance is only different by a small margin. Do note that this evaluation is solely based on steps taken until convergence as performance metric.

Are the Theoretical Convergence Rates met?

On f_1, we see the expected results. The linear schedule looks linear, the quadratic one looks steeper than the super linear one. The latter two schedules are however, as mentioned in the script, hard to differentiate in plots.
On f_4 and f_5 we are indeed surprised by the results. The choice of η does not seem to have a significant impact on the convergence rates. We suspect these results may be due to the specific nature of f_4 and f_5 somehow.

Number of CG steps performed during each step of optimization, observations and explanations

See 2. If η is chosen with the quadratic convergence setting, we see that the amount of steps in the internal model grows very quickly as the gradient norm goes to zero. The CG stopping condition gets harder and harder to fulfill as we reach the optimum, so more work has to put in to reach the required precision. As the dimensionality grows, more work has to be put in, because CG in general has to take a step for each dimension before reaching convergence. In our case we never see more steps than dimensions, but this could be necessary in higher dimensions because of numerical issues. On f_1, which is fully captured by the derivative and hessian, `approx newton` converges as soon as `CG` has optimized all 20 dimensions. Even on f_5 `approx newton` reaches its convergence criteria as soon as `CG` gets to optimize along every single p_i. Something strange happens on f_4, where the internal model reaches the convergence criteria, often after one or two steps. We hypothesize that this is because the attractive sector function gets flat quickly, satisfying the CG convergence criteria.

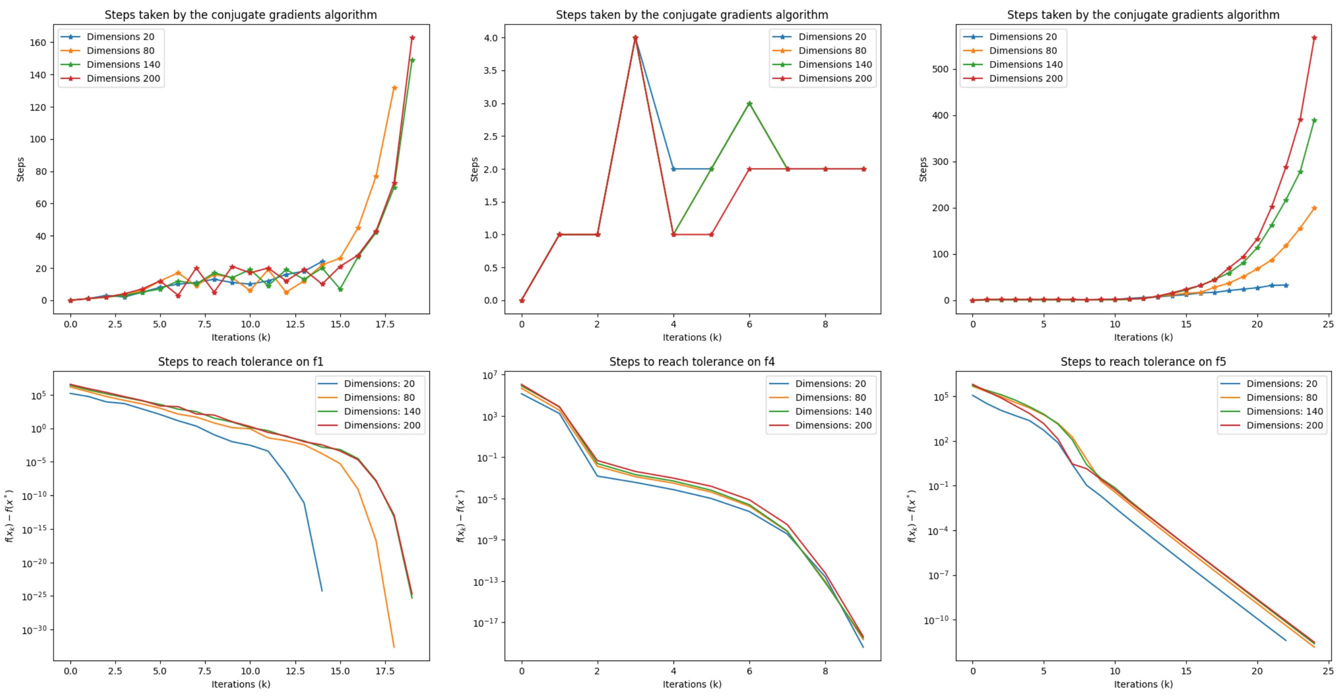

Figure 2: Steps taken by CG per iteration, and difference between the current and lowest function value on the real function. Trajectories are generated by a random initialization.

Comparison of `steepest descent`, `original newton` and `approx newton`

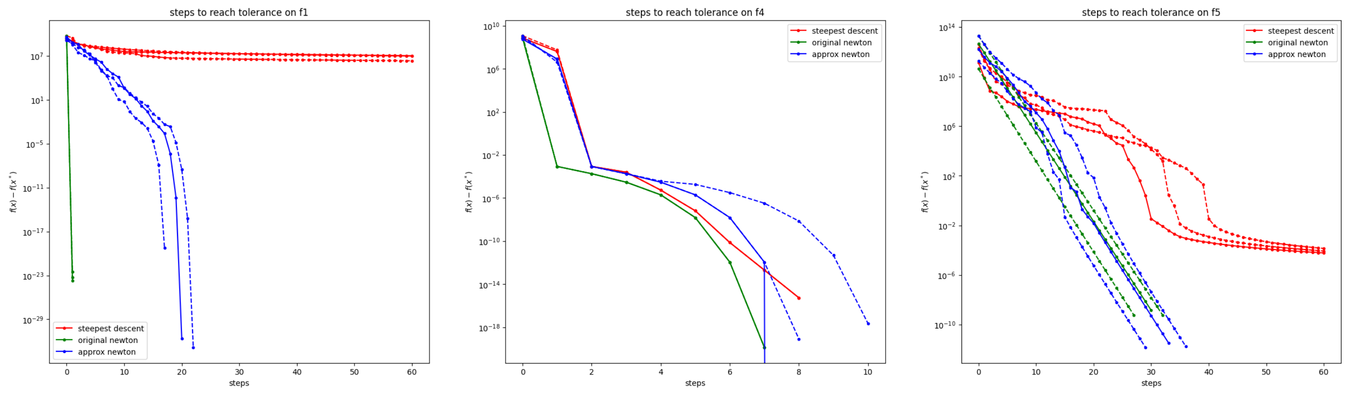

Figure 3: Plot showing comparisons between `Steepest Descent`, `Original Newton` and `Approximate Newton` where $d = 20, \epsilon = 1e-6$, $\rho = 0.5$, $c_1 = 0.5$, $max\_iter = 60$ and the fastest, slowest and median run is plotted.

What we can interpret from figure 3 is that `original newton` is superior in terms of number of steps taken to convergence, even though `approx newton` gives a good fight for the first place. We chose the max_iter variable relatively low, as we noted that convergence was reached for most of the cases beneath this threshold. We saw that `steepest descent` did not seem to come closer to convergence, even with larger values for max_iter, hence the low value should suffice to show a reasonable comparison of the algorithms. It can be seen that `steepest descent` has some serious issues in f_1 and f_5 in terms of convergence and perhaps non-termination, hence this can be deemed the worst one of the three. In the case of steps taken to convergence, we prefer `original newton`, as can be seen is best if an average over all the objective functions is taken.

Time taken until convergence

Refer to figure 4 and 5 for plots showing how much time the different optimizers take until convergence, measured in seconds. Approx newton takes more steps until convergence, but takes them much more rapidly than `original newton`. `steepest descent` takes steps very quickly as it does not have to compute any hessian, however it makes little progress. `original newton` is a lot more step efficient, but the step is expensive to compute.

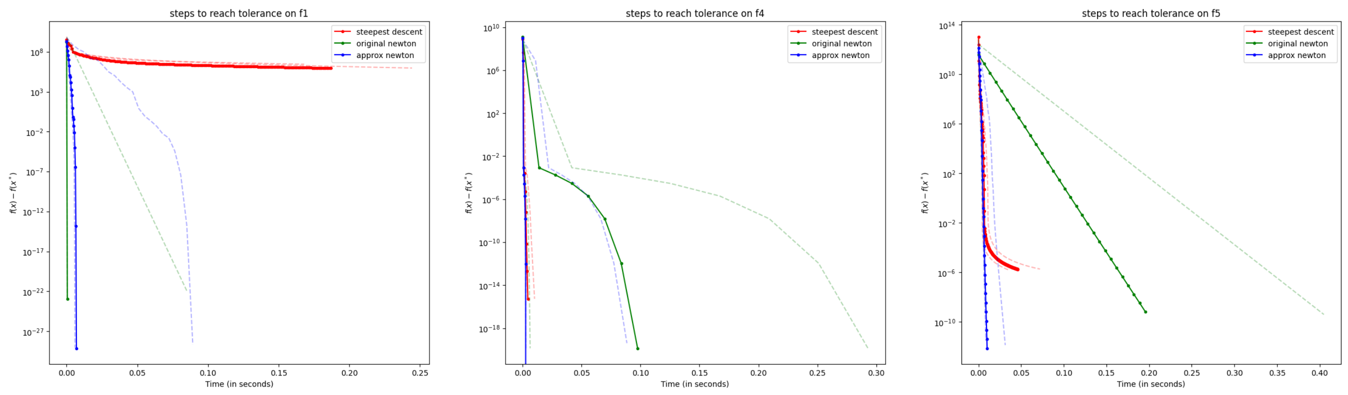

Figure 4: Plot showing comparisons between `Steepest Descent`, `Original Newton` and `Approximate Newton` where $d = 20, \epsilon = 1e-6$, $\rho = 0.5$, $c_1 = 0.5$, $max\_iter = 60$ and the fastest, slowest and median run is plotted (measured in seconds until convergence on the x-axis.)

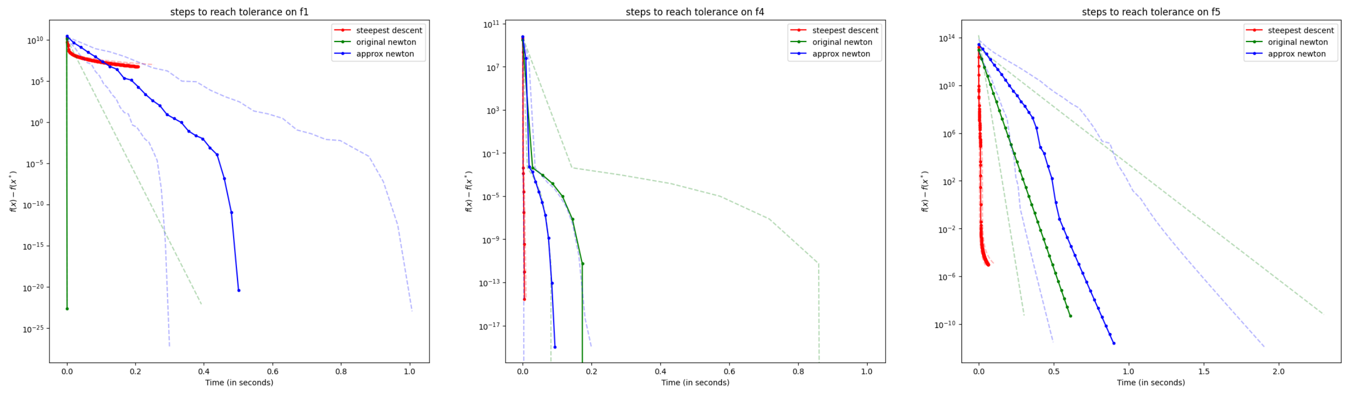

Figure 5: Same as figure 4, but $d = 100$