Optimization methods – Homework 3

Claudio Maggioni

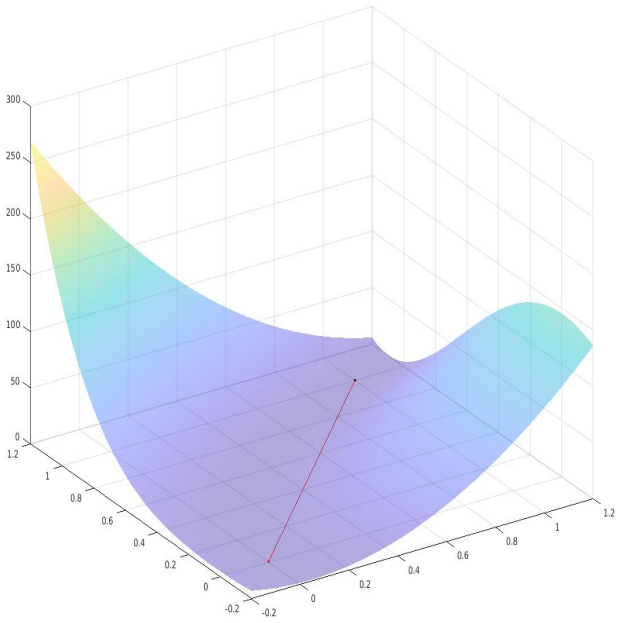
April 23, 2021

1 Exercise 1

1.1 Exercise 1.1

Please consult the MATLAB implementation in the files `Newton.m`, `GD.m`, and `backtracking.m`. Please note that, for this and subsequent exercises, the gradient descent method without backtracking activated uses a fixed $\alpha = 1$ despite the indications on the assignment sheet. This was done in order to comply with the forum post on iCorsi found here: `https://www.icorsi.ch/mod/forum/discuss.php?d=81144`.

Here is a plot of the Rosenbrock function in 3d, with our starting point in red ($(0, 0)$), and the true minimizer in black ($(1, 1)$):

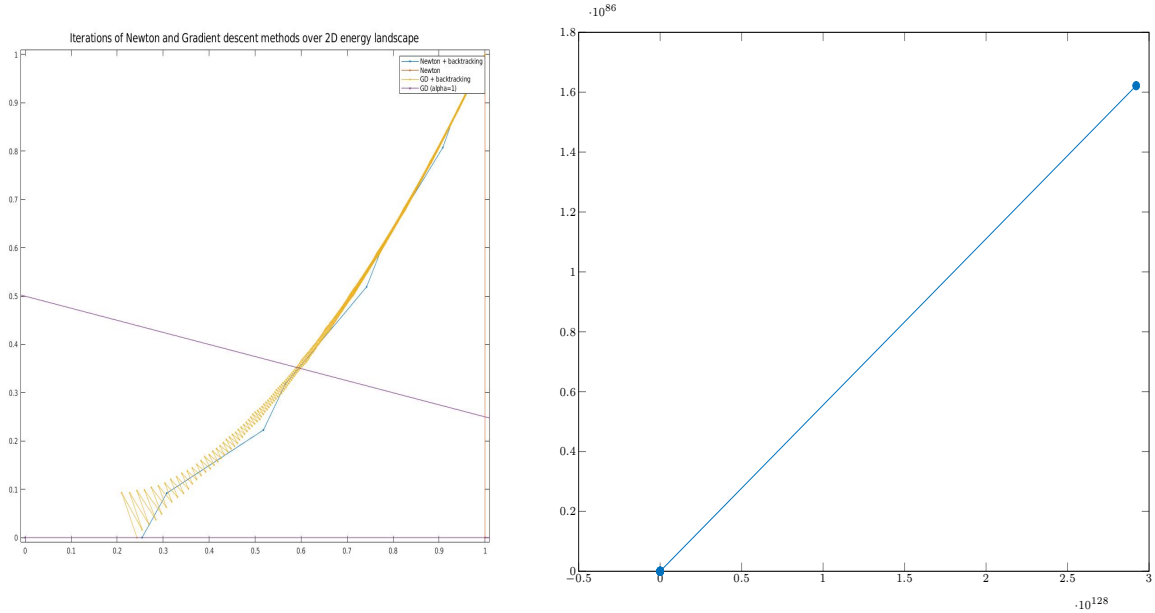


1.2 Exercise 1.2

Please consult the MATLAB implementation in the file `main.m` in section 1.2.

1.3 Exercise 1.3

Please find the requested plots in figure 1. The code used to generate these plots can be found in section 1.3 of `main.m`.



(a) Zoomed plot on $x = (-1, 1)$ and $y = (-1, 1)$ (b) Complete plot (the blue line is GD with $\alpha = 1$)

Figure 1: Steps in the energy landscape for Newton and GD methods

1.4 Exercise 1.4

Please find the requested plots in figure 2. The code used to generate these plots can be found in section 1.4 of `main.m`.

2 Exercise 1.5

The best performing method for this very set of input data is the Newton method without backtracking, since it converges in only 2 iterations. The second best performing one is the Newton method with backtracking, with convergence in 12 iterations. The gradient method achieves convergence with backtracking slowly with 21102 iterations, while with a fixed $\alpha = 1$ the method diverges in a dozen of iterations resulting in catastrophic numerical instability leading to `x_k = [NaN; NaN]`.

Analyzing the movement in the energy landscape (figure 1), the more "coordinated" method in terms of direction of iterations steps appears to be the Netwton method with backtracking. Other than performing a higher number of iterations when compared with the classical variant, the method maintains its iteration directions approximately at a 45 degree angle from the x axis, roughly always pointing at the minimizer x^*. However, this movement strategy is definitely inefficient compared with the 2-step convergence achieved by Newton without backtracking, which follows a conjugate gradient like path finding in each step a component of the true minimizer. GD with backtracking instead follows an inefficient zig-zagging pattern with iterates in the vicinity of the Netwon + backtracking iterates. Finally, GD without backtracking quickly degenerates as it can be seen by the enlarged plot.

From the initial plot in the Rosenbrock's function, we can see why algorithms using backtracking follow roughly the $(0,0)-(1,1)$ diagonal: since this region is effectively a "valley" in the 3D energy landscape, due to the nature of the backtracking methods and their strict adherence of the Wolfe conditions these methods avoid wild "climbs" (unlike the Netwon method without backtracking) and instead finding iterates with either sufficient decrease or a not to high increase.

When looking at gradient norms and objective function values (figure 2) over time, the degeneration of GD without backtracking and the inefficiency of GD with backtracking can clearly be seen. Newton with backtracking offers fairly smooth gradient norm and objective value curves with an exponential decreasing slope in both for the last 5-10 iterations. Netwon without backtracking instead shoots at the first iteration at gradient $\nabla f(x_1) \approx 450$ and objective value $f(x_1) \approx 100$, but quickly has both values decrease to 0 for its second iteration achieving convergence.

What has been observed in this assignment matches with the theory behind the methods. Since the Newton method has quadratic convergence and uses quadratic information (i.e. using the hessian in the step direction calculation) the number of iterations required to find the minimizer when already close to it (as we are with $x_0 = [0;0]$) is significantly less than the ones required for linear methods, like gradient descent. However, it must be said that for an objective with an high number of dimensions a single iteration of a quadratic method is significantly more costly than a single iteration of a linear method due to the quadratically growing number of cells in the hessian matrix, which makes it harder and harder to compute as the number of dimensions increase.

3 Exercise 2

3.1 Exercise 2.1

Please consult the MATLAB implementation in the file `BGFS.m`.

3.2 Exercise 2.2

Please consult the MATLAB implementation in the file `main.m` in section 2.2.

3.3 Exercise 2.3

Please find the requested plots in figure 3. The code used to generate these plots can be found in section 2.3 of `main.m`.

3.4 Exercise 2.4

Please find the requested plots in figure 4. The code used to generate these plots can be found in section 2.4 of `main.m`.

3.5 Exercise 2.5

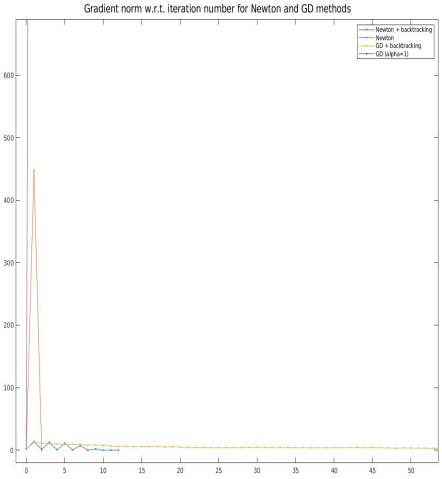
The following table summarizes the number of iterations required by each method to achieve convergence:

Method	**Backtracking**	**# of iterations**
Newton	No	2
Newton	Yes	12
BGFS	Yes	26
Gradient descent	Yes	21102
Gradient descent	No ($\alpha = 1$)	Diverges after 6

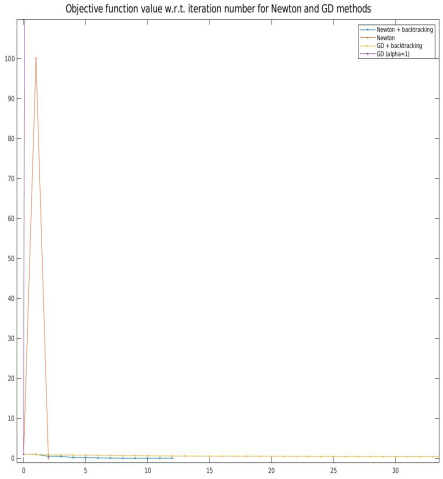
From the table above we can see that the BGFS method is in the same performance order of magnitude as the Newton method, albeit its number of iterations required to converge are more than double (26) of the ones of Newton with backtracking (12), and more than ten times of the ones required by the Newton method without backtracking (2).

From the iterates plot and the gradient norm and objective function values plots (respectively located in figure 3 and 4) we can see that BGFS behaves similarly to the Newton method with backtracking, loosely following its curves. The only noteworthy difference lies in the energy landscape plot, where BGFS occasionally "steps back" performing iterations in the opposite direction of the minimizer. This behaviour can also be observed in the plots in figure 4, where several bumps and spikes are present in the gradient norm plot and small plateaus can be found in the objective function value plot.

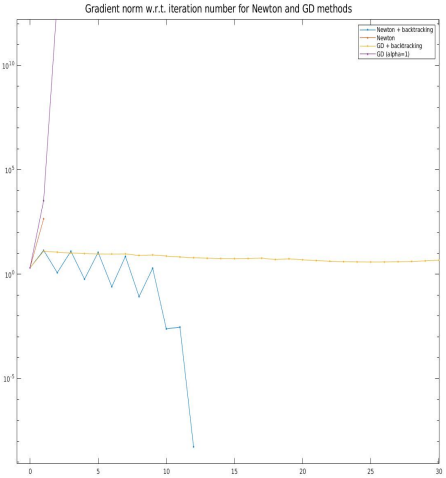
Comparing these results with the theory behind BFGS, we can say the results that have been obtained fall within what we expect from the theory. Since BFGS is a superlinear but not quadratic method of convergence, its "speed" in terms of number of iterations falls within linear methods (like GD) and quadratic methods (like Newton).



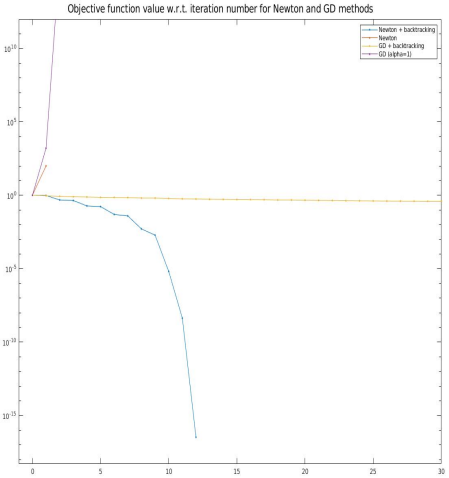
(a) Gradient norms
(zoomed, y axis is linear for this plot)



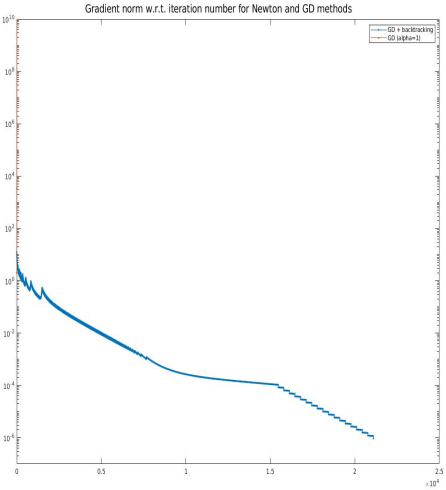
(b) Objective function values
(zoomed, y axis is linear for this plot)



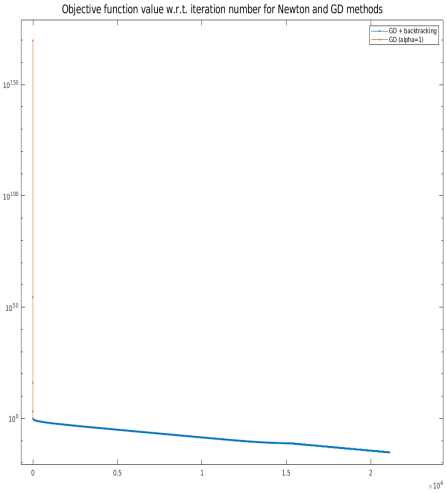
(c) Gradient norms (zoomed)



(d) Objective function values (zoomed)



(e) Gradient norms



(f) Objective function values

Figure 2: Gradient norms and objective function values (y-axes) w.r.t. iteration numbers (x-axis) for Newton and GD methods (y-axis is log scaled, points at $y = 0$ not shown due to log scale)

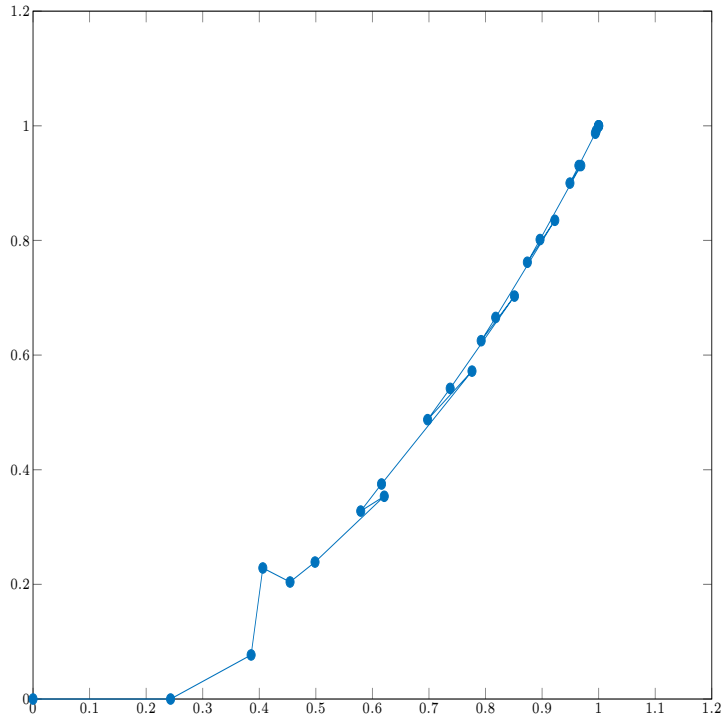
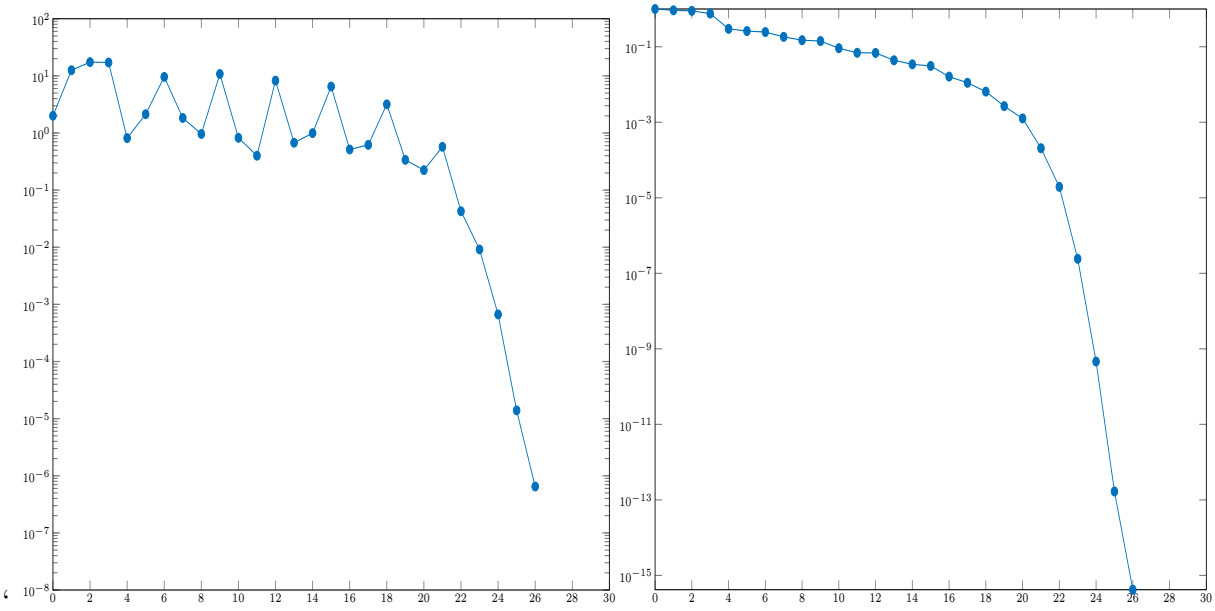


Figure 3: Steps in the energy landscape for BGFS method



(a) Gradient norms

(b) Objective function values

Figure 4: Gradient norms and objective function values (y-axes) w.r.t. iteration numbers (x-axis) for BFGS method (y-axis is log scaled, points at $y = 0$ not shown due to log scale)