CS 726 Assignment 2

Ruochen Lin

March 2, 2018

1

Given $0 < c_2 < c_1 < 1$, we simply need to find a function that satisfy the two inequalities in distinct regions. For example: This function satisfies

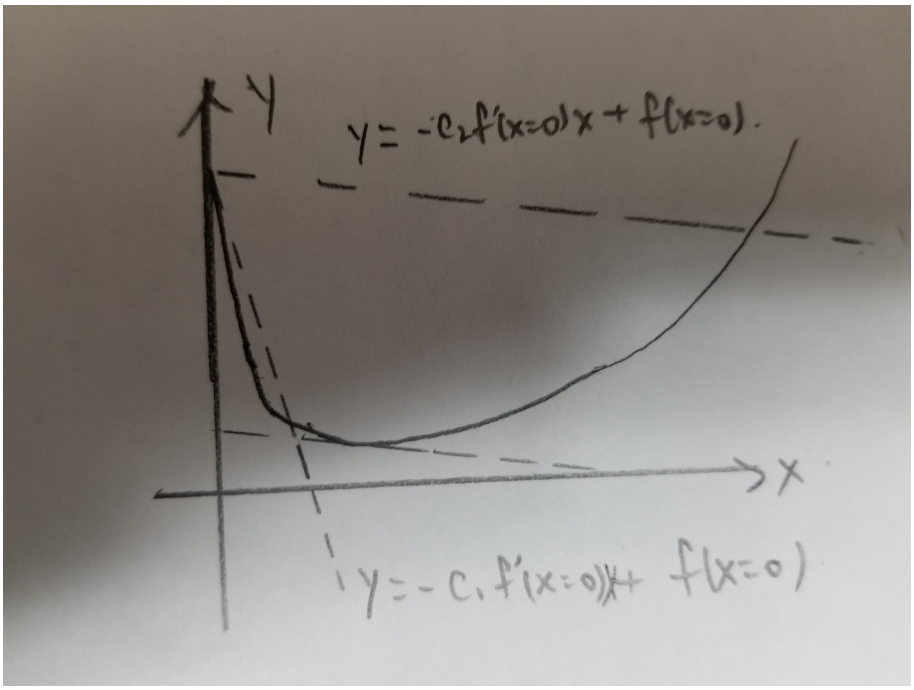

Figure 1: A function that violates weak Wolfe condition at $x_k = 0$

the first inequality in the region left to the intersection between $f(x)$ and $y = -c_1 f'(0)x + f(0)$, but only satisfies the second inequality in the region between the tangent shown in Figure 1 all the way to the right. Since the two

region has no overlap, the given function violates the weak Wolfe condition at $x_k = 0$.

2

Given $f(x) = \frac{1}{2}x^T Q x - b^T x$, if $x_0 - x^*$ is parallel to an eigenvector of Q, namely

$$x_0 - x^* = cv,\ Qv = \lambda v,$$

then $x_0 - x^*$ is also an eigenvector of Q with eigenvalue λ:

$$Q(x_0 - x^*) = \lambda(x_0 - x^*),$$

with the assumption that Q is nonsingular and thus $\lambda \neq 0$. In addition,

$$\begin{aligned} \nabla f(x^*) &= Qx^* - b = 0 \\ \Rightarrow\ \ b &= Qx^* \end{aligned}$$

and

$$\begin{aligned} \nabla f(x_0) &= Qx_0 - b = Q(x_0 - x^*) \\ &= \lambda(x_0 - x^*). \end{aligned}$$

If we apply steepest descent with exact line search to f, then

$$x_1 = x_0 - \alpha_0 \nabla f(x_0) = x_0 - \alpha_0 \lambda(x_0 - x_*),$$

with α_0 given by

$$\begin{aligned} \alpha_0 &= \frac{\nabla f(x_0)^T \nabla f(x_0)}{\nabla f(x_0)^T Q \nabla f(x_0)} \\ &= \frac{[\lambda(x_0 - x^*)]^T [\lambda(x_0 - x^*)]}{[\lambda(x_0 - x^*)]^T Q [\lambda(x_0 - x^*)]} \\ &= \frac{\lambda^2 \|x_0 - x^*\|^2}{\lambda^3 \|x_0 - x^*\|} \\ &= \frac{1}{\lambda}. \end{aligned}$$

Plug α_0 into the expression for x_1, we have:

$$\begin{aligned} x_1 &= x_0 - \frac{1}{\lambda}\lambda(x_0 - x^*) \\ &= x_0 - x_0 + x^* \\ &= x^*. \end{aligned}$$

So, if $x_0 - x^*$ is an eigenvector of Q, we can get to the solution with but one step of steepest descent with exact line search.

3

Running `descentLS.m` with $\eta = 0$ yielded the following output: To bet-

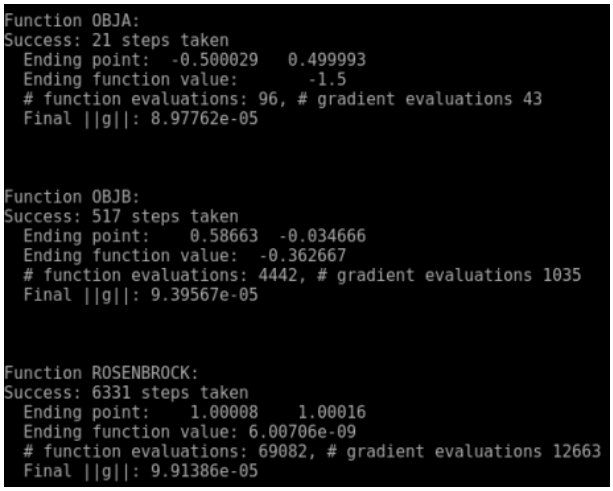

Figure 2: Output of `descentLS.m` with $\eta = 0$

ter show the change of number of iterations (`nIter`), of function evaluations (`numf`), and gradient evaluations (`numg`) needed to reach the convergence criterion with the three functions (textttobja, textttobjb, and rosenbrock), we tabulated the output in files `ls_obja.dat`, `ls_objb.dat`, and `ls_rosenbrock.dat`, which can be found in Supplementary materials. Plotting the natural logarithm of the three values, with respect to the corresponding η, gets us the following figures: We see that textttobja needed the least computation to converge, with `objb` being the next, and `rosenbrock` needed the largest number of iterations and function/gradient evaluations to converge.

By taking a closer look at the figures, we note that $\eta = 0$ seems to have similar performance in terms of iterations and gradient evaluations as $\eta \in (1, 20]$ for functions `obja` and `objb`; but for `rosenbrock` $\eta = 0$ seems to give better performance than most values in $\eta \in (1, 20]$. Also, we notice that, for all three functions, the change in η along $(1, 20]$ does not seem to impact the number of iterations and gradient evaluations too much; but there is a discernible increase in `numf` as we increase η, and $\eta = 0$ seems to require more function evaluations than, say, $\eta \in (1, 5]$. We also note that at $\eta = 15.8$, the algorithm failed to converge within $10,000$ steps into $\|\nabla x_k\| \leqslant 10^{-4}$.

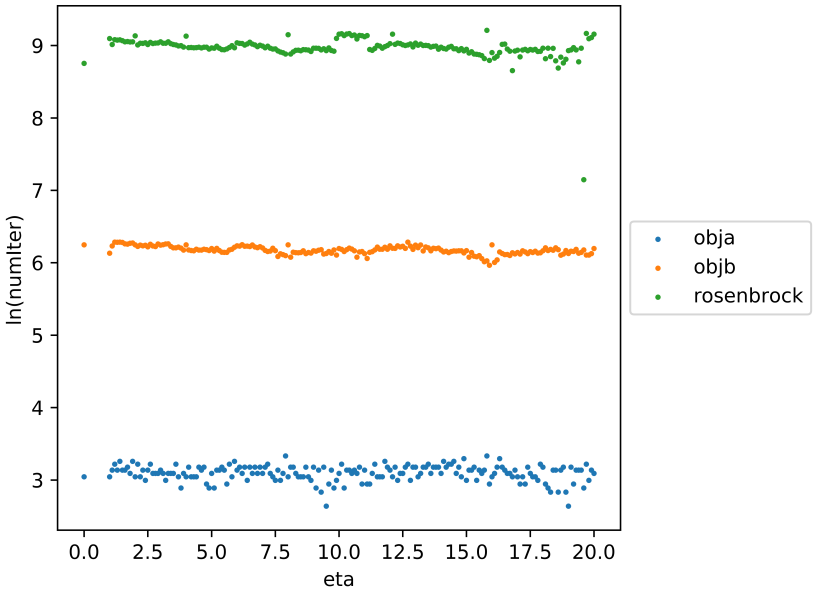

Figure 3: Number of iterations needed for steepest descent with exact line search to converge

Lastly, there is an outlier at $\eta = 19.6$ for `rosenbrock`, at which point the result converged within 1269 iterations, while it usually takes $6000 - 9000$ iterations to converge.

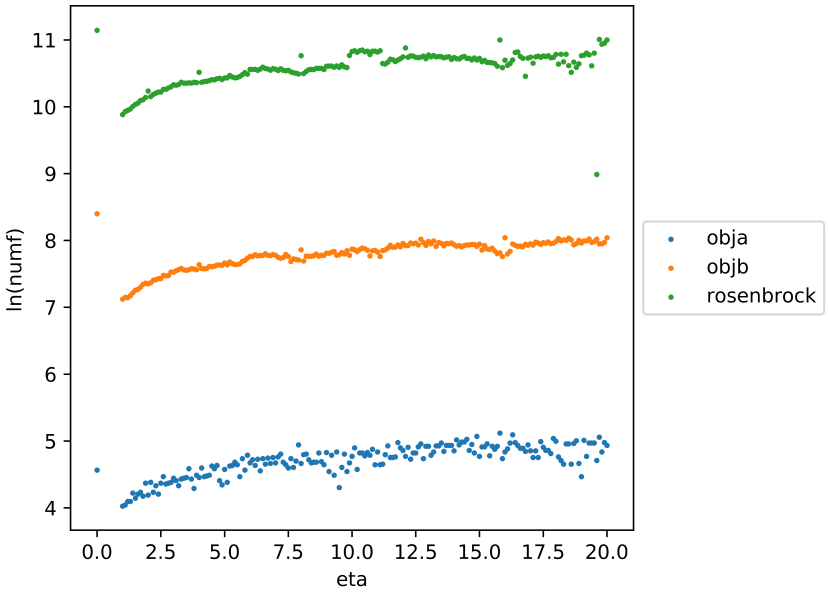

Figure 4: Number of function evaluations needed for steepest descent with exact line search to converge

4

Similar to the previous part, we run `descentBacktrack.m` with $\eta = 0$ and got the following output: We noticed that for all three functions, `descentBacktrack.m` converged with the same number of iterations and function evaluations as `descentLS.m`; but the number of gradient evaluations were almost cut in half. This corroborates the claim in the textbook that backtracking line search steepest descent will be more economical when gradient evaluation is expensive.

By comparing the performance of algorithm to that of the preceeding section, we found that the two algorithms required similar numbers of iterations and gradient evaluations to converge; but `descentBacktrack.m` requires about $\frac{1}{2}$ to $\frac{1}{3}$ less of function evaluations than `descentLS.m`; for example, for `obja`, `descentBacktrack.m` needed e^3 function calls, but `descentLS.m` requires about e^4. Also, the $\eta = 19.6$ outlier in `descentLS` is not present here. Apart from this, the qualitative observations we had for steepest descent

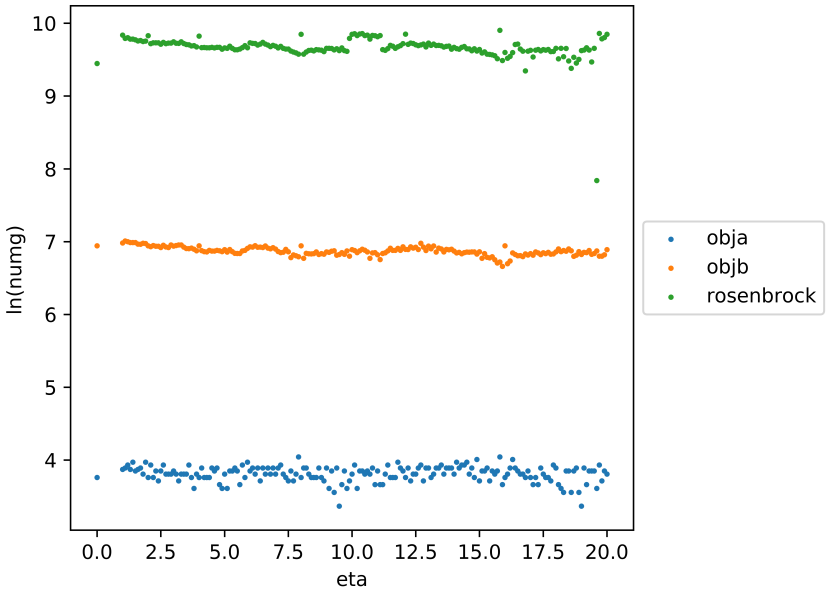

Figure 5: Number of gradient evaluations needed for steepest descent with exact line search to converge

with exact line search are mostly still valid in steepest descent with backtracking line search: in terms of iterations and gradient evaluations, $\eta = 0$ still performs comparably with $\eta > 1$; but it requires significantly more function evaluations than $\eta \in (0, 5]$. Increasing η along $(1, 20]$ also increases the number of function evaluations needed. Finally, we note that the program failed to converge at ηs 1.0, 8.0, and 15.8; this is a curious case because at $\eta = 15.8$ the exact line search algorithm also failed the converge and at the point of $\eta = 15.8$.

```
Function OBJA:
Success: 21 steps taken
  Ending point:  -0.500029   0.499993
  Ending function value:       -1.5
  # function evaluations: 96, # gradient evaluations 22
  Final ||g||: 8.97762e-05


Function OBJB:
Success: 517 steps taken
  Ending point:    0.58663  -0.034666
  Ending function value:  -0.362667
  # function evaluations: 4442, # gradient evaluations 518
  Final ||g||: 9.39567e-05


Function ROSENBROCK:
Success: 6331 steps taken
  Ending point:    1.00008    1.00016
  Ending function value: 6.00706e-09
  # function evaluations: 69082, # gradient evaluations 6332
  Final ||g||: 9.91386e-05
```

Figure 6: Output of `descentBacktrack.m` with $\eta = 0$

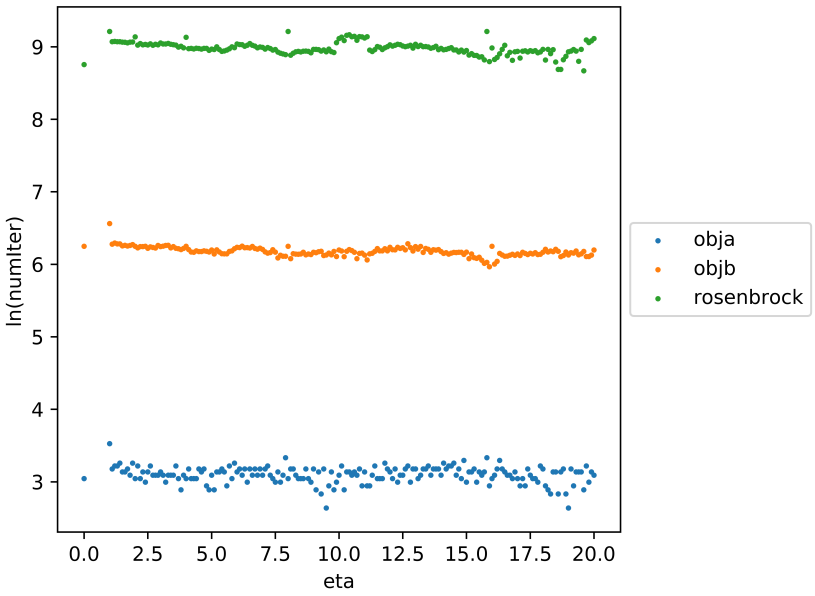

Figure 7: Number of iterations needed for steepest descent with backtrack line search to converge

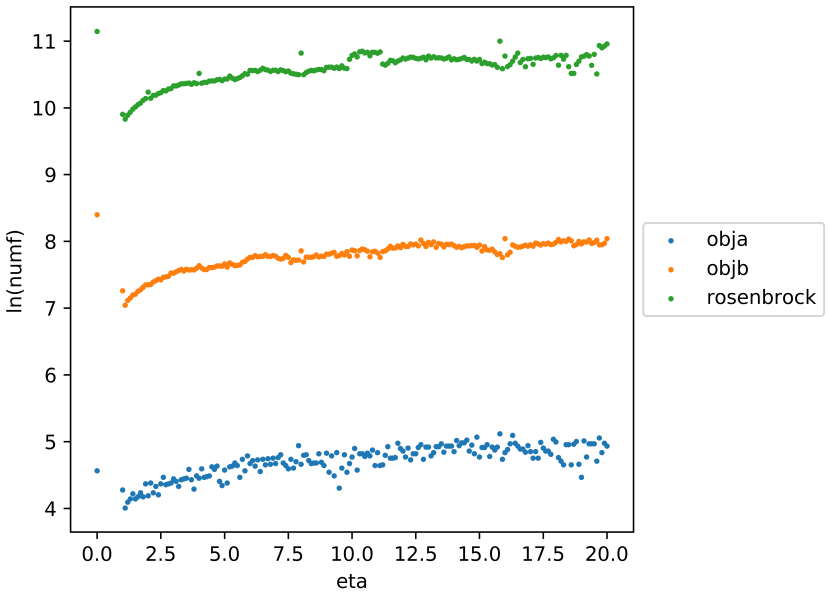

Figure 8: Number of function evaluations needed for steepest descent with backtrack line search to converge

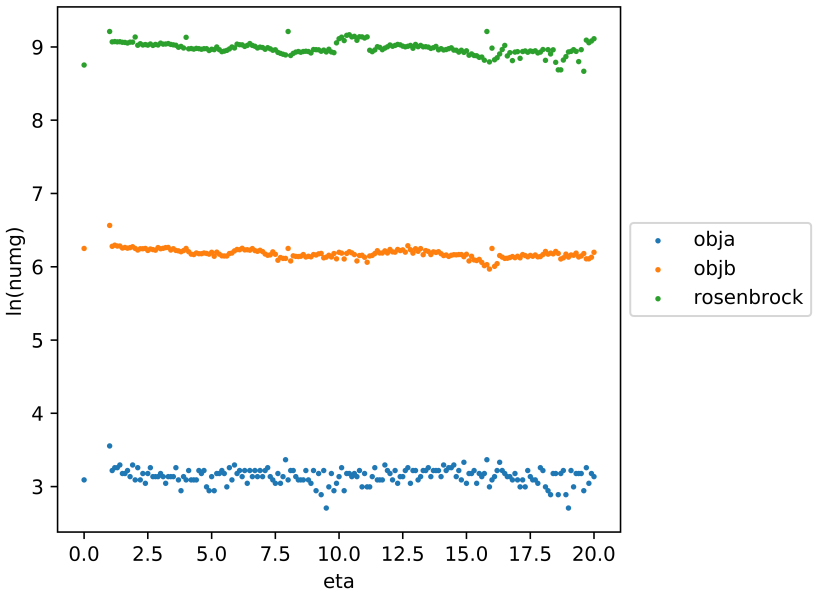

Figure 9: Number of gradient evaluations needed for steepest descent with backtrack line search to converge