APM 523
Unconstrained Optimization Project

Matt Kinsinger, Hongjun Choi, Adarsh Akkshai

October 12, 2017

This assignment requires us to first code the Extended Rosenbrock and Powell functions and their gradients. These can be found at **extended_rosen.m , extended_rosen_gradient.m , extended_powell.m, extended_powell_gradient.m**

1 Functions

1.1 Extended Rosenbrock Function

The definition of $f(\bar{x}) \in \mathbb{R}^n$ is as follows:

$$f(\bar{x}) = \sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2, \quad \text{where} \quad \bar{x} = [x_1, \dots x_n]^T$$

Then the $\nabla f \in \mathbb{R}^n$ is as follows

$$\nabla f = \begin{vmatrix} -400(x_{2i-1} - x_i^2)x_i + 2(x_{2i-1} - 1) \\ 200(x_{2i} - x_{2i-1}) \end{vmatrix}, \quad \forall \quad i = 1 \dots \frac{n}{2}.$$

1.2 Extended Powell Function

The definition of $f(\bar{x}) \in \mathbb{R}^n$ is as follows:

$$f(\bar{x}) = \sum_{i=1}^{n/4} (x_{4i-3} - 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$$

where $\quad \bar{x} = [x_1 \dots x_n]^T$

$$\nabla f = \begin{vmatrix} 2(x_{4i-3} - 10(x_{4i-2})) + 40(x_{4i-3} - x_{4i})^3 \\ -20(x_{4i-3} - 10x_{4i-2}) + 4(x_{4i-2} - 2x_{4i-1})^3 \\ 10(x_{4i-1} - x_{4i}) - 8(x_{4i-2} - 2x_{4i-1})^3 \\ -10(x_{4i-1} - x_{4i}) - 40(x_{4i-3} - x_{4i})^3 \end{vmatrix}, \quad \forall \quad i = 1 \dots \frac{n}{4}.$$

2 Non-Linear CG

After computation of Jacobians of function we move on to the actual non-linear CG methods. These can be found at **nonlin_CG_FR.m, nonlin_CG_PR.m, nonlin_CG_PRplus.m**. To run these algorithms one needs to execute **run_nonlinear_CG.m**. The code later requires us to manually specify algorithm we desire to execute **"extended_Powell"** or **"extended_rosen"** into the first argument in the function.

xres n̄onlin_CG_FR('extended_rosen','extended_rosen_gradient',x0);

2.1 THE FLETCHER-REEVES METHOD

We would go through some algorithms about nonlinear variants of the conjugate gradient. First, Fletcher and Reeves showed how to extend the conjugate gradient method to nonlinear functions by making two simple changes in general conjugate algorithms (Algorithm 5.2). As we change given formulas, these changes give rise to the following algorithm for nonlinear optimization.

Algorithm 1 Fletcher and Reeves

```
1:  Given $x_0$;
2:  Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;
3:  Set $p_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$;
4:  while $\nabla f_k \neq 0$; do
5:      Compute $\alpha_k$ and set $x_{k+1} = x_k + \alpha_k p_k$;
6:      Evaluate $\nabla f_{k+1}$;
7:
8:      $\beta_{k+1}^{FR} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;
9:      $p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k$;
10:     $k \leftarrow k + 1$;
```

If we choose f to be a strongly convex quadratic and α_k to be the exact minimizer, this algorithm reduces to the linear conjugate gradient method, general algorithm and the Fletcher Reeves method is appealing for large nonlinear optimization problems because each iteration requires only evaluation of the objective function and its gradient.

2.2 THE POLAK-RIBIERE METHOD AND VARIANTS

There are many variants of the Fletcher-Reeves method that differ from each other mainly in the choice of the parameter β_k. An important variant, proposed by Polak and Ribiere, defines this parameter as follows:

$$\beta_{k+1}^{FR} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}. \tag{1}$$

We refer to the algorithm in which equation (1) replaces FR's β_{k+1}^{FR} as **Algorithm PR**.
A surprising fact about Algorithm PR is the strong Wolfe conditions do not guarantee that p_k is always a descent direction. If we define the β parameter as

$$\beta_{k+1}^{+} = max(\beta_{k+1}^{PR}, 0), \tag{2}$$

giving rise to an algorithm we call **Algorithm PR+**, then a simple adaptation of the strong Wolfe conditions ensures that the descent property holds.

3 Complete BFGS for Non-linear problems

For the sake of completeness we restate the BFGS algorithm here. The individual components of the algorithm are present in **LineSearch_wolfe.m, zoom.m, BFGS.m**. To execute the code for non-linear problems, one needs to execute **run_BFGS.m**. In addition one needs to include the arguments to the function as above by specifying the function type.

xres $\bar{\text{B}}$FGS('extended_rosen',x0);

Algorithm 2 BFGS Method

```
    Given starting point $x_0$, and convergence tolerance $\epsilon > 0$
 2: Inverse Hessian Approximation $H_0$
    $k \leftarrow 0$;
 4: while $\|\nabla f_k\|$; do
        Compute search Direction $p_k = -H_k \nabla f_k$
 6:     $\alpha_k \leftarrow$ Procedure (LineSearch)
        Set $x_{k+1} = x_k + \alpha_k p_k$
 8:     $s_k = x_{k+1} - x_k$
        $y_k = \nabla f_{k+1} - \nabla f_k$
10:     $\rho_k = \frac{1}{{y_k}^T s_k}$
        $H_{k+1} = (I - \rho_k s_k {y_k}^T) H_k (I - \rho_k y_k {s_k}^T) + \rho_k s_k {s_k}^T$
12:     k $\leftarrow$ k +1;
```

4 Comparisons with backtracking Method

All the algorithms above have been also made to work with backtracking methods in the code **BFGS_backtrack.m, nonlin_CG_backtrack.m, nonlin_CG_PR_backtrack.m, nonlin_CG_PRPlus_backtrack.m, nonlin_FR_backtrack.m**. To run these files one would need to execute **run_nonlinear_CG_backtrack.m, run_BFGS_backtrack.m**

Algorithm 3 Line Search

```
    Set $\alpha_0 \leftarrow 0$, choose $\alpha_{max} > 0$ and $\alpha_1 \in (0, \alpha_{max})$
    $i \leftarrow 1$;
3:  **repeat**
    **while**
        **do**
6:      Evaluate $\phi(\alpha_i)$
        **if** $\phi(\alpha_i) > \phi(0) + c_1 \alpha_i \phi^{'}(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1]$ **then**
        $\alpha_* \leftarrow$ **zoom**$(\alpha_{i-1}, \alpha_i)$ and **break;**
9:      Evaluate $\phi^{'}(\alpha_i)$
        **if** $|\phi^{'}(\alpha_i)| \leq -c_2 \phi^{'}(0)$ **then**
            **set** $\alpha_* \leftarrow \alpha_i$ and **break;**
12:     **if** **then**$\phi^{'}(\alpha_i) \geq 0$
            **set** $\alpha_* \leftarrow$ **(**zoom) $(\alpha_i, \alpha_{i-1})$ and **break;**
        Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{max})$;
15:     $i \leftarrow i + 1$;
```

If the line search algorithm chooses its candidate step length appropriately, by using a a so-called backtracking algorithm, we can remove the extra condition and use just the sufficient condition to terminate the line search procedure. The basic form of line search proceeds as follows.

In our procedure, the initial step length is chosen to be 0.01. An acceptable step length will be found after a finite number of trials because α_k will eventually become small enough that the sufficient decrease condition holds.

Algorithm 4 zoom

```
    repeat
    Interpolate (using quadratic, cubic, or bisection) to find a trial step length
    $\alpha_j$ between $\alpha_{lo}$ and $\alpha_{hi}$;
    Evaluate $\phi(\alpha_j)$
 4: if $\phi(\alpha_j) > \phi(0) + c_1\alpha_j\phi^{'}(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{lo})$ then
        $\alpha_{hi} \leftarrow \alpha_j$;
    else
        Evaluate $\phi^{'}(\alpha_j)$;
 8:     if $|\phi^{'}(\alpha_j)| \leq -c_2\phi^{'}(0)$ then
            Set $\alpha_* \leftarrow \alpha_j$ and stop;
        if $\phi^{'}(\alpha_j)(\alpha_{hi} - \alpha_{lo}) \geq 0$ then
            $\alpha_{hi} \leftarrow \alpha_{lo}$
12:     $\alpha_{lo} \leftarrow \alpha_j$
    end (repeat)
```

Algorithm 5 Backtracking Line Search

```
    Choose $\overline{\alpha} > 0$, $\rho \in (0,1)$, $c \in (0,1)$; Set $\alpha \leftarrow \overline{\alpha}$;
    repeat    until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha\nabla f_k^T p_k$
    $\alpha \leftarrow \rho\alpha$;
    end (repeat)
 5: Terminate with $\alpha_k = \alpha$
```

5 Convergence analysis

The order of convergence according to theoretical conclusions for:

- Non-Linear CG Methods: PR+ $<$ PR $<$ FR
- BFGS $<$ Backtracking BFGS
- Line Search $<$ Backtracking Search

Overall $n = 500$. The common conditions for BFGS: Tolerance is 10^{-8}. For nonlinear CG the common tolerance is 10^{-5}. Conditions for Line Search are: $c2 = 0.9$ $c1 = 10^{-4}$, $r = 0.001$

Table 1: Extended Rosenbrock

Algorithm	Iterations
FR	293
PR	2549
PR+	5233
BFGS	1099

Table 2: Extended Powell

Algorithm	Iterations
FR	(see below)
PR	
PR+	
BFGS	

Our results seemingly do not follow these convergence orders. However, they match with global solution of the system. In some cases such as the extended Powell, they get stuck at a step length inside zoom, with $\alpha_{hi} == \alpha_{lo}$ and never exit the loop.