EN.553.761: Nonlinear Optimization I

Homework Assignment #3

Starred exercises require the use of MATLAB.

Exercise 3.1**:** Solve the trust-region subproblem

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \quad s^T g + \tfrac{1}{2} s^T B s \quad \text{subject to} \quad \|s\|_2 \le \delta \tag{1}$$

in the following cases:

(a)

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \delta = 2,$$

(b)

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \delta = 5/12$$

Hint: $\lambda = 2$ is a root of the nonlinear equation

$$\frac{1}{(1+\lambda)^2} + \frac{1}{(2+\lambda)^2} = \frac{25}{144}.$$

(c)

$$B = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad g = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \delta = 5/12,$$

(d)

$$B = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad g = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \delta = 1/2, \quad \text{and}$$

(e)

$$B = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \quad g = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \delta = \sqrt{2}.$$

Exercise 3.2**:** Consider the solution of problem (1) with data

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

as a function of the trust-region radius δ. In which direction does the solution point as δ shrinks to zero, i.e., what can you say about $\lim_{\delta\to 0} \frac{s(\delta)}{\|s(\delta)\|}$?

Exercise 3.3***:** Write a MATLAB m-function called steihaug_CG.m that implements the truncated linear conjugate gradient method by Steihaug based on Algorithm 4 on slide 81 in the "Trust Region Methods" slides. The function call should have the form

```
[ p, iters, flag ] = steihaug_CG( B, g, radius, tol )
```

where the input B is required to be a symmetric (possibly indefinite) matrix, g is a vector, radius $>$ 0 is the trust-region radius, and tol $\in (0,1)$ is the stopping tolerance. On exit, the resulting approximate Steihaug truncated-CG solution should be stored in the vector p, the number of iterations performed stored in iters, and the parameter flag should be set to one of the following values: -1 if the algorithm terminated because of negative curvature, 0 if the stopping tolerance was met, and 1 if the algorithm returned a boundary solution simply because the CG iterations grew larger than the trust-region radius.

Exercise 3.4*: Consider the problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x)$$

where f is a twice continuously differentiable function.

(a) Write a MATLAB m-function called unc_TR.m that implements a trust-region method with trial steps computed from the Steihaug truncated linear CG method as given in Problem 4.4 above. The function call should have the form

```
[x,F,G,H,iter,status] = unc_TR(fun,x0,maxit,printlevel,tol)
```

where fun is of type *string* and represents the name of a Matlab m-function that computes $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$ for some desired function f; it should be of the form

```
[F,G,H] = fun(x)
```

where for a given value x it returns the values of the function, gradient, and Hessian, respectively. The parameter x0 is an initial guess at a minimizer of f, maxit is the maximum number of iterations allowed, printlevel determines the amount of printout required, and tol is the final stopping tolerance. In the code, if the parameter printlevel has the value zero, then no printing should occur; otherwise, a single line of output is printed (in column format) per iteration. On output, the parameters x, F, G, and H should contain the final iterate, function value, gradient vector, and Hessian matrix computed by the algorithm. The parameter iter should contain the total number of iterations performed. Finally, status should have the value 0 if the final stopping tolerance was obtained and the value 1 otherwise.

(b) Write a separate MATLAB m-file with function declaration [F,G,H] = fun(x) that returns the value F, gradient G, and Hessian H at the point $x \in \mathbb{R}^2$ of the function

$$f(x) = 10(x_2 - x_1^2)^2 + (x_1 - 1)^2.$$

Use your m-function unc_TR.m from part (a) to minimize f with starting point $x_0 = (0,0)$.

Exercise 3.5*: Recall the Levenberg-Marquardt method for solving the nonlinear least squares problem. The trust region subproblem at the iterate x_k is (see slides 29–30 in the "Least Squares" slides)

$$\min_{s \in \mathbb{R}^n} \;\; \frac{1}{2}\|F(x_k)\|_2^2 + s^T J(x_k)^T F(x_k) + \frac{1}{2} s^T J^T(x_k) J(x_k) s \qquad \text{subject to} \qquad \|s\|_2 \le \delta_k$$

From the characterization of global optima for the above problem, we need to find $\lambda^* \ge 0$ and $s^* \in \mathbb{R}^n$ such that

1. $\lambda^* \geq 0$

2. $(J^T(x_k)J(x_k) + \lambda^* I)s^* = -J(x_k)^T F(x_k)$

3. $\lambda^*(\|s^*\| - \delta_k) = 0$

If solve this by applying Newton's method to the *secular equation* $\phi(\lambda) := \frac{1}{\|s(\lambda)\|_2} - \frac{1}{\delta_k} = 0$, then recall that each step requires the computation of the Newton step $-\phi(\lambda)/\phi'(\lambda)$ (for some fixed $\lambda > 0$). From the computations on slide 72 in the "Trust Region Methods" slides, we see that $\phi'(\lambda) = -\frac{s(\lambda)^T \nabla s(\lambda)}{\|s(\lambda)\|_2^3}$, where $s(\lambda)$ is the solution to

$$(J^T(x_k)J(x_k) + \lambda I)s(\lambda) = -J(x_k)^T F(x_k), \tag{2}$$

and $\nabla s(\lambda)$ is the solution to

$$(J^T(x_k)J(x_k) + \lambda I)\nabla s(\lambda) = -s(\lambda). \tag{3}$$

1. Show that the linear system (2) are the normal equations for the linear least squares problem

$$\min_{s \in \mathbb{R}^n} \ \frac{1}{2} \left\| \begin{pmatrix} J(x_k) \\ \sqrt{\lambda} I \end{pmatrix} s + \begin{pmatrix} F(x_k) \\ 0 \end{pmatrix} \right\|_2^2. \tag{4}$$

2. Show that if we use the QR decomposition to solve the linear least squares problem (4) to compute $s(\lambda)$, then one can compute $\nabla s(\lambda)$ from (3) by just solving two triangular linear systems involving the R matrix in the QR decomposition.

3. Show that if we use the SVD decomposition to solve the linear least squares problem (4) to compute $s(\lambda)$, then one can compute $\nabla s(\lambda)$ from (3) by using only matrix-matrix and matrix-vector products, with no need for any linear system solves.