NUMERICAL OPTIMISATION
TUTORIAL 31/01/20
ASSIGNMENT 2 (submit by 11pm on Thursday 13/02)

Marta Betcke

EXERCISE 1 [DEMO]

Derive the 2D subspace trust region method for convex functions (with s.p.d. Hessian). Note that

(i) when p is constraint to a subspace $V = \mathrm{span}(g, B^{-1}g)$, it can be expressed as a linear combination of basis vectors $p = Va$. You can use any basis, here orthonormal basis is useful;

(ii) use the result in Theorem 4.1 to obtain optimal p. Observe that complementarity condition (Theorem 4.1 equation (4.8a)) results in two cases;

(iii) use Theorem 4.1 equation (4.8a) to obtain an explicit expression for each coefficient a_i and plug them into the remaining condition; *Hint: After using the eigenvalue decomposition of $B_V = V^T BV$, this can be reduced to finding the roots of a 4th order polynomial.*

[0pt]

EXERCISE 2 [DEMO]

Implement the 2D subspace trust region method for convex functions (with s.p.d. Hessian). This implementation should return the *Cauchy point* whenever the gradient and Newton steps are collinear. **[0pt]**

EXERCISE 3

Implement the Dogleg trust region method for convex functions (with s.p.d. Hessian), which can be found in Nocedal Wright.
Submit your implementation via `MATLAB Grader`. **[20pt]**

EXERCISE 4

Implement a trust region framework function based on Algorithm 4.1 in Nocedal Wright. Let this function take a handle to a solver for the constraint quadratic model problem as an argument. This will allow us to plug in different solvers to obtain different trust region methods.
Submit your implementation via `MATLAB Grader`. **[20pt]**

EXERCISE 5

Apply the trust region method to the Rosenbrock function with a nearby starting point $(1.2, 1.2)$ and a remote point $(-1.2, 1)$. Pay attention to the choice of trust region radius in each case. In your submission please include:

- Which of the two quadratic solvers you are using.
- **All** the parameters of the minimization.
- Convergence plots and their discussion in light of the theory.
- Trust region radii at each iteration.
- Brief explanation of all plots and conclusions from the experiment.

Hint: You can choose between the given 2D subspace implementation or your Dogleg implementation as a quadratic solver.
Submit solution via ***`TurnitIn`***. **[20pt]**

EXERCISE 6 [DEMO]

Consider the linear system $Ax = b$ with $A \in \mathbb{R}^{n \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$.

(a) Implement the linear preconditioned Conjugate Gradient method.
Submit your implementation via ***`MATLAB Grader`***. **[0pt]**

Consider starting point $x_0 = (0, \dots, 0)^T$, tolerance `tol = 1e-12` and a dimension `n = 100`. Let b be the right hand side vector defined by $Ax^* = b$ for the following values of x^*:

```
xtrue = zeros(n,1);
xtrue(floor(n/4):floor(n/3)) = 1;
xtrue(floor(n/3)+1:floor(n/2)) = -2;
xtrue(floor(n/2)+1:floor(3/4*n)) = 0.5;
```

(b) Solve the given linear system with the following A matrices:

- `A1 = diag(1:n);`
- `A2 = diag([ones(n-1, 1) 100]);`
- 1d negative Laplacian:
  `A3 = -diag(ones(n-1, 1), -1) - diag(ones(n-1, 1), 1) + diag(2*ones(n, 1));`

Compare the theoretical and practical convergence rates according to the distribution of eigenvalues of A_i, $i = 1, 2, 3$. Use the true solution[1] to evaluate the convergence rate.
Submit your solution via ***`TurnitIn`***. **[0pt]**

[1] In practice, the true solution is not available, so a common practice is to consider the norm of the residuals.

EXERCISE 7

(a) Implement the Fletcher-Reeves conjugate gradient method.
Submit your implementation via `MATLAB Grader`. **[10pt]**

(b) Implement the Polak-Ribière conjugate gradient method.
Submit your implementation via `MATLAB Grader`. **[10pt]**

(c) Minimise the function

$$f(x, y) = x^2 + 5x^4 + 10y^2$$

from the initial points $x_0 = (10, 10)^T$ and $x_0 = (-5, 7)^T$ to tolerance `tol = 1e-12` using your implementation of **both** non-linear conjugate gradient methods. Explain your results highlighting any potential problems. Propose a way to ensure convergence. *Submit your solution via* `TurnitIn`. **[20pt]**

Remark. The submission to `TurnitIn` should not be longer than 8 pages (this is not a hard limit, no penalty for longer submissions). Avoid submitting more code than needed (if any) and focus on explaining your results.