This set covers chapters 5–11 of the book *Numerical Optimization* by Nocedal and Wright, 2nd ed.

From the book, the following exercises: 5.1–5.3, 5.6, 5.8, 5.11, 6.1–6.4, 6.6, 7.1–7.3, 7.5–7.6, 8.1, 8.3, 9.2(a), 9.8–9.9, 10.1–10.2, 10.6, 11.1–11.3, 11.5, 11.8, 11.10. In addition, the exercises below. The Matlab programming exercises are 5.1, 5.8, 7.1 and **II.3**–**II.4**.

For exercise 5.8, try 2 different matrices $\mathbf{A}$ generated as follows:

```
% Matrix 1: uniform spectrum
n = 100; l = 1:n; U = gallery('orthog',n); A = U*diag(l)*U';
% Matrix 2: clustered spectrum; try s = 5, 1, 0.001, 0
n = 100; n1 = floor(n/4);
l = [ones(1,n1) n/5*ones(1,n1) n*ones(1,n-2*n1)]; rand('state',314); l = l + s*rand(1,n);
U = gallery('orthog',n); A = U*diag(l)*U';
```

and compare your results with figs. 5.4–5.5.

Hint for exercise 6.1(a): a function is strongly convex on a convex domain if at each point in the domain all the eigenvalues of the Hessian are positive and bounded away from zero.

II.1. Conjugate directions. Let $\mathbf{A}$ be a real, symmetric, positive definite, $n \times n$ matrix with eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_n)$ and associated eigenvalues $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ (in matrix form). Show that: (i) If $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{Q}$, where $\mathbf{Q}$ is any orthogonal matrix, then $\mathbf{A} = \mathbf{R}\mathbf{R}^T$. (ii) If $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are nonzero orthogonal vectors then $\{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ where $\mathbf{p}_i = \mathbf{R}^{-T}\mathbf{v}_i$ are conjugate w.r.t. $\mathbf{A}$ (this shows there is an infinite number of conjugate direction sets). (iii) $\mathbf{p}_i = \mathbf{u}_i$ are conjugate w.r.t. $\mathbf{A}$ (this corresponds to the change of variables $\hat{\mathbf{x}} = \mathbf{S}^{-1}\mathbf{x}$ with $\mathbf{S} = \mathbf{U}$). Hint: use the spectral theorem.

II.2. Quasi-Newton methods. Verify that, in 1D, all 3 quasi-Newton methods BFGS, DFP and SR1 are equivalent to the secant method, where $B_{k+1} = \frac{f'_{k+1} - f'_k}{x_{k+1} - x_k}$ independent of B_k.

II.3. BFGS. Implement algorithm 6.1 (BFGS method) in Matlab with $\mathbf{H}_0 = \mathbf{I}$ and backtracking line search (with initial step length 1). Apply it to the Rosenbrock function (2.23) from two initial points $\mathbf{x}_0 = (1.2, 1.2)$ and $\mathbf{x}_0 = (-1.2, 1)$ (cf. exercise 3.1). Tabulate $\|\mathbf{x}_k - \mathbf{x}^*\|_2$, $\|\nabla f(\mathbf{x}_k)\|_2$ and $\left\|\mathbf{B}_k - \nabla^2 f(\mathbf{x}_k)\right\|_F$ where $\|\cdot\|_2$ is the Euclidean norm (for vectors) and $\|\cdot\|_F$ the Frobenius norm (for matrices). Use your `convseq` function to estimate the convergence rate.

II.4. Newton-CG. Program a pure Newton iteration without line searches, where the search direction is computed by the CG method. Select stopping criteria such that the rate of convergence is linear, superlinear, and quadratic. Try your program on the following quartic function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{x} + \frac{1}{4}\sigma(\mathbf{x}^T\mathbf{A}\mathbf{x})^2 \qquad \mathbf{A} = \begin{pmatrix} 5 & 1 & 0 & \frac{1}{2} \\ 1 & 4 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 3 & 0 \\ \frac{1}{2} & 0 & 0 & 2 \end{pmatrix}$$

where $\mathbf{A}$ is symmetric pd and $\sigma \geq 0$ is a parameter that allows us to control the deviation from a quadratic. The starting point is $\mathbf{x}_0 = (\cos 70°, \sin 70°, \cos 70°, \sin 70°)^T$. Try $\sigma = 1$ or larger values and observe the rate of convergence of the iteration. Use the `convseq` function you wrote in exercise **I.4** to estimate the convergence rate.

Also, considering $\mathbf{A}$ is pd $n \times n$ and $\mathbf{x} \in \mathbb{R}^n$:

1. Prove that f is a strictly convex function (hint: prove that its Hessian is pd everywhere). Find all the stationary points of f and classify them into minima, maxima and saddle points.
2. Compute the cost in multiplications in $\mathcal{O}$–notation of computing $f(\mathbf{x})$, $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$ at a point $\mathbf{x}$.
3. What would be the cost of computing $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$ approximately with finite differences if only function evaluations are allowed?