Exercises for Week 10

1 Finite-difference approximation of the Hessian (by hand)

In this exercise, we derive the finite-difference approximation formula for the Hessian. According to if the gradient available, we consider two cases. Here, we assume that second derivatives of f exist and are Lipschitz continuous near $\boldsymbol{x}$.

1. **The gradient is available.** In many methods, we do not require knowledge of the full Hessian, but in each iteration we need supply the Hessian-vector product $\nabla^2 f(\boldsymbol{x})\boldsymbol{p}$ for a given $\boldsymbol{p}$.

   (a) By applying Taylor's theorem, we have

   $$\nabla f(\boldsymbol{x} + \epsilon\boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \epsilon\nabla^2 f(\boldsymbol{x})\boldsymbol{p} + O(\epsilon^2).$$

   Derive the forward-difference approximation of $\nabla^2 f(\boldsymbol{x})\boldsymbol{p}$ (The formula (8.20) in the textbook).

   (b) Referring to the derivation of the central-difference formula of the gradient, derive the central-difference approximation of $\nabla^2 f(\boldsymbol{x})\boldsymbol{p}$. Here, you would need to apply Taylor's theorem on $\nabla f(\boldsymbol{x} - \epsilon\boldsymbol{p})$ as well.

2. **The gradient is NOT available.** We use Taylor's theorem again, and have

   $$f(\boldsymbol{x} + \epsilon\boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{p} + O(\|\boldsymbol{p}\|_2^3).$$

   Substitute the vectors $\boldsymbol{p} = \epsilon\boldsymbol{e}_i$, $\boldsymbol{p} = \epsilon\boldsymbol{e}_j$, and $\boldsymbol{p} = \epsilon(\boldsymbol{e}_i + \boldsymbol{e}_j)$ into this formula, and derive an approximation of $\frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x}) = \boldsymbol{e}_i^T\nabla^2 f(\boldsymbol{x})\boldsymbol{e}_j = \boldsymbol{e}_j^T\nabla^2 f(\boldsymbol{x})\boldsymbol{e}_i$ using only function values.

2 Automatic differentiation (by hand)

In this exercise, we practise automatic differentiation technique for finding the derivative of the function

$$f(\boldsymbol{x}) = e^{x_1} + x_1\sin(x_2).$$

1. Express the evaluation of f in arithmetic terms, like (8.27) in the textbook.
2. **Forward mode.** Set $\boldsymbol{p}_1 = [1, 0]^T$ and $\boldsymbol{p}_2 = [0, 1]^T$. Use the forward mode to derive the directional derivatives for $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ associated with each variables, and further obtain $\frac{\partial f}{\partial x_1}(\boldsymbol{x})$ and $\frac{\partial f}{\partial x_2}(\boldsymbol{x})$.
3. **Reverse mode (Optional).** Referring to the slides, apply the reverse mode to derive the gradient $\nabla f(\boldsymbol{x})$.

3 Coordinate search method (in Matlab)

In this exercise, we will implement the coordinate search method and apply it to solve Rosenbrock problem:

$$\min_{\boldsymbol{x}} 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{1}$$

1. Implement the coordinate search method with the search directions $\{\boldsymbol{e}_1, \boldsymbol{e}_2, -\boldsymbol{e}_1, -\boldsymbol{e}_2\}$. Here, we consider the nonopportunistic search, i.e., we check all potential points in the pattern, $\mathcal{D}_k = \{\boldsymbol{x}_k \pm \gamma_k \boldsymbol{e}_i : i = 1, 2\}$, and find

$$\boldsymbol{t} = \arg\min_{\boldsymbol{y} \in \mathcal{D}_k} f(\boldsymbol{y}).$$

   If $f(\boldsymbol{t}) < f(\boldsymbol{x}_k)$, then we set $\boldsymbol{x}_{k+1} = \boldsymbol{t}$; otherwise, we reduce γ_k by half.

   You can use the following Matlab template as the start.

```
function [x,stat]=coordinate(fundfun,x0,gamma0,varargin)
% Solver settings and info
maxit = 10000;
tol   = 1.0e-6;

stat.converged = false;          % converged
stat.nfun = 0;                   % number of function calls
stat.iter = 0;                   % number of iterations

% Initial iteration
gamma = gamma0;
x = x0;
it = 0;
f = feval(fundfun,x,varargin{:});
stat.nfun = 1;
converged = false;

% Store data for plotting
stat.X = x;
stat.F = f;

% Main loop of coordinate search
while ~converged && (it < maxit)
    it = it+1;

    % TODO -- evaluate the function values for all points in D_k,
    %                 then update x or reduce gamma
    % ============================================
```

```
    % ========================================

    converged = (gamma <= tol);

    % Store data for plotting
    stat.X  = [stat.X  x];
    stat.F  = [stat.F f];
end

% Prepare return data
if ~converged
     x = [];
end
stat.converged = converged;
stat.iter = it;
```

2. Use the starting point $\boldsymbol{x}_0 = [-1.2, 1]^T$ and set $\gamma_0 = 1$. Apply the coordinate search method to solve the Rosenbrock problem.

3. Use `semilogy` to plot $e_k = \|\boldsymbol{x}_k - [1, 1]^T\|_2$ and $f(\boldsymbol{x}_k)$.

4. Plot the iterates on the contour plot of the Rosenbrock function, which has been implemented in Week 5.

5. Compare with the results from the steepest descent method with backtracking line search in Week 5. Which method converges faster?

4 Interpolation model (in Matlab)

Find the quadratic function

$$m(\boldsymbol{x}) = m(x_1, x_2) = c + g_1 x_1 + g_2 x_2 + \frac{1}{2} H_{11} x_1^2 + H_{12} x_1 x_2 + \frac{1}{2} H_{22} x_2^2$$

that approximates the function $f(\boldsymbol{x})$ by interpolating the following data: $\boldsymbol{x}_1 = [0, 0]^T$, $\boldsymbol{x}_2 = [1, 0]^T$, $\boldsymbol{x}_3 = [2, 0]^T$, $\boldsymbol{x}_4 = [1, 1]^T$, $\boldsymbol{x}_5 = [0, 2]^T$, $\boldsymbol{x}_6 = [0, 1]^T$, and $f(\boldsymbol{x}_1) = 1$, $f(\boldsymbol{x}_2) = 2.0084$, $f(\boldsymbol{x}_3) = 7.0091$, $f(\boldsymbol{x}_4) = 1.0168$, $f(\boldsymbol{x}_5) = -0.9909$, $f(\boldsymbol{x}_6) = -0.9916$.

5 `fminsearch` in Matlab's Optimization Toolbox

The Matlab function `fminsearch` applies the Nelder-Mead method to find a minimizer of an unconstrained minimization problem.

1. Read about `fminsearch` using `doc fminsearch`.

2. Use `optimset` to set displaying output at each iteration.

3. Call `fminsearch` to solve the Rosenbrock problem in (1) with the starting point $[-1.2, 1]^T$.

4. Now in the command window you should see that at each iteration which point (reflected point or expansion point or outside contraction point or inside contraction point) is used to update the iterate. In addition, you can see how many function evaluations are calculated.

5. You can compare the number of iterations with other starting point, like $[1.2, 1.2]^T$.