OSM Lab Boot Camp **Math Problem Set 6**

Harrison Beard

30 July 2018

Exercise 9.1.

Prove that an unconstrained linear objective function is either constant or has no minimum.

Solution. Proof by contradiction. If an objective function f is *not* constant, then we know there must be a $\tilde{\mathbf{x}}$ such that $f(\tilde{\mathbf{x}}) \neq f(\mathbf{x}^*)$ for the $\mathbf{x}^*$ that minimizes f, so $f(\tilde{\mathbf{x}})$ cannot be less than $f(\mathbf{x}^*)$ without loss of generality. Then, this implies $f(\tilde{\mathbf{x}}) > f(\mathbf{x}^*)$, so it follows that $f(\mathbf{x}^* - \tilde{\mathbf{x}}) < 0$ by linearity, but then we have

$$\begin{aligned} f(2\mathbf{x}^* - \tilde{\mathbf{x}}) &= f(\mathbf{x}^*) + f(\mathbf{x}^* - \tilde{\mathbf{x}}) \\ &< f(\mathbf{x}^*), \end{aligned}$$

which is a contradiction because $\mathbf{x}^*$ is the minimizer by assumption. ■

Exercise 9.2.

Prove that if $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathrm{M}_{m\times n}(\mathbb{R})$, then the problem of finding an $\mathbf{x}^* \in \mathbb{R}^n$ to minimize $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ is equivalent to minimizing

$$\mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} - 2\mathbf{b}^\mathsf{T}\mathbf{A}\mathbf{x}. \tag{9.21}$$

In Voume 1, Chapter 3 we use projections to prove that this is equivalent to solving the normal equation

$$\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{b}.$$

Use the first- and second-order conditions to give a different proof that minimizing (9.21) is equivalent to solving the normal equation.

Solution. We will use the fact that $\mathbf{A}^\mathsf{T}\mathbf{A}$ is symmetric and *positive definite* as we have shown previously.

Note that if $\mathbf{x}$ is such that the FONC is satisfied, then $\mathbf{x}$ is a global minimizer of the function

$$(\mathbf{A}\mathbf{x} - \mathbf{b})^\mathsf{T}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} - 2\mathbf{b}^\mathsf{T}\mathbf{A}\mathbf{x}.$$

Next, note that our FOC is $2\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} - 2\mathbf{A}^\mathsf{T}\mathbf{b} = 0$, so we have that $\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{b}$, and by the SOC and positive definiteness of $\mathbf{A}^\mathsf{T}\mathbf{A}$, so $2\mathbf{A}^\mathsf{T}\mathbf{A} > 0$, and so we are done. ■

Exercise 9.3.

For each of the multivariable optimization methods we have discussed in this section, list the following:

(*i*). The basic idea of the method, including how it differs from the other methods in the list. Include any geometric description you can give of the method.

(*ii*). What types of optimization problems it can solve and cannot solve.

(*iii*). Relative strengths of the method.

(*iv*). Relative weaknesses of the method.

Solution.

(*i*). The method of **Gradient Descent** is to follow the function f's negative gradient $-\mathbf{D}f^{\mathsf{T}}(\mathbf{x}_i)$ and iterate over the $\mathbf{x}_i$'s, recalculating the gradient, until a minimum is reached. Geometrically, this looks like a zig-zag pattern across the level sets of f. **Newton's Method** is used when the dimensionality is small and if the Hessian $\mathbf{D}^2 f(\mathbf{x}_i)$ is positive definite; its appeal is that it converges quadratically and acts as both a local approximation and a descent method. **Conjugate Gradient** can optimize a quadratic in a single step, and is very quick for small-dimensional problems. It is usually contrasted with gradient descent geometrically, as it appears as a straight line perpendicular to each of the level sets of f by moving along $\mathbf{Q}$-conjugate directions, as seen in figure (9.1) in the textbook. **Gauss-Newton** is an adaptation of Newton's method to efficiently optimize nonlinear least squares (NLS) problems. One of the quasi-Newton innovations, **Broyden-Fletcher-Goldfarb-Shanno (BFGS)**, was developed to improve computational cost involves only computing the Hessian initially, but has a lower convergence rate.

(*ii*). **Conjugate Gradient** is used when f is differentiable. **Newton's Method** is used when $\mathbf{D}^2 f(\mathbf{x}_i) \in \mathrm{PD}_n(\mathbb{R})$. **Gauss-Newton** and its derivatives are used for NLS problems.

(*iii*). **Newton** has quadratic convergence and is best when the dimension of the problem is not large and when $\mathbf{x}_0 \approx \mathbf{x}^*$. **Gauss-Newton** is best for NLS problems. The niche appeal to both Gauss-Newton and **BFGS** is that they both are effective when $\left(\mathbf{D}^2 f(\mathbf{x})\right)^{-1} \mathbf{D}f(\mathbf{x})$ is expensive or error-prone. When $\mathbf{x}_0$ is far from $\mathbf{x}^*$, then **Gradient Descent** is the fastest algorithm and most desirable if the dimensionality is reasonable. **Conjugate Gradient** is best for solving large quadratic linear systems when $f = \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{Q}\mathbf{x} - \mathbf{b}^{\mathsf{T}}\mathbf{x} + c$ and $\mathbf{Q}$ is symmetric, positive definite, and sparse.

(*iv*). **Newton** and its derivatives becomes prohibitively expensive when the dimension of the problem is large or when $\mathbf{x}_0$ starts very far away from $\mathbf{x}^*$. **BFGS** fails under similar criteria. **Conjugate Gradient** loses its appeal when the number of nonzero entries m of $\mathbf{Q}$ becomes very large, since its temporal and spatial complexity $\mathcal{O}(m)$ is often contrasted with Newton's single-iteration complexity $\mathcal{O}\left(n^3\right)$.

■

Exercise 9.4.

Let $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{Q}\mathbf{x} - \mathbf{b}^{\mathsf{T}}\mathbf{x}$, where $\mathbf{Q} \in \mathrm{M}_{m \times n}(\mathbb{R})$ satisfies $\mathbf{Q} > 0$ and $\mathbf{b} \in \mathbb{R}^n$. Show that the Method of Steepest Descent (that is, gradient descent with optimal line search), converges in one step (that is, $\mathbf{x}_1 = \mathbf{Q}^{-1}\mathbf{b}$), if and only if $\mathbf{x}_0$ is chosen such that $\mathbf{D}f(\mathbf{x}_0)^{\mathsf{T}} = \mathbf{Q}\mathbf{x}_0 - \mathbf{b}$ is an eigenvector of $\mathbf{Q}$ (and α_0 satisfies (9.2)).

Solution. For $\mathbf{D}f(\mathbf{x}_0)^\mathsf{T} = \mathbf{Q}\mathbf{x}_0 - \mathbf{b}$ with corresponding eigenvalue $\lambda_\mathbf{Q}$, we have

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \frac{\mathbf{D}f(\mathbf{x})^\mathsf{T}\mathbf{D}f(\mathbf{x})}{\mathbf{D}f(\mathbf{x})^\mathsf{T}\mathbf{Q}\mathbf{D}f(\mathbf{x})}\mathbf{D}f(\mathbf{x}) \\
&= \mathbf{x}_0 - \frac{\mathbf{D}f(\mathbf{x})^\mathsf{T}\mathbf{D}f(\mathbf{x})}{\mathbf{D}f(\mathbf{x})^\mathsf{T}\lambda_\mathbf{Q}\mathbf{D}f(\mathbf{x})}\mathbf{D}f(\mathbf{x}) \\
&= \mathbf{x}_0 - \frac{1}{\lambda}\mathbf{D}f(\mathbf{x}) \\
&= \mathbf{x}_0 - \mathbf{Q}^{-1}\mathbf{D}f(\mathbf{x}) \\
&= \mathbf{x}_0 - \mathbf{Q}^{-1}\left(\mathbf{Q}\mathbf{x}_0 - \mathbf{b}\right) \\
&= \mathbf{Q}^{-1}\mathbf{b}.
\end{aligned}$$

■

Exercise 9.5.

Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^1$. Let $\{\mathbf{x}_k\}_{k=0}^\infty$ be defined by the Method of Steepest Descent. Show that if $\mathbf{x}_{k+1} - \mathbf{x}_k$ is orthogonal to $\mathbf{x}_{k+2} - \mathbf{x}_{k+1}$ for each k.

Solution. We have

$$\begin{aligned}
\frac{\mathrm{d}f\left(\mathbf{x}_{k+1}\right)}{\mathrm{d}\alpha_k} &= \mathbf{D}f\left(\mathbf{x}_{k+1}\right)^\mathsf{T}\mathbf{D}f\left(\mathbf{x}_k\right) \\
&= 0,
\end{aligned}$$

and since $f\left(\mathbf{x}_{k+1}\right) \coloneqq f\left(\mathbf{x}_k + \alpha_k\mathbf{D}f\left(\mathbf{x}_k\right)^\mathsf{T}\right)$, we have that

$$\mathbf{x}_{k+1} - \mathbf{x}_k = -\alpha_k\mathbf{D}f\left(\mathbf{x}_k\right)^\mathsf{T},$$

where $\mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{x}_{k+2} - \mathbf{x}_{k+1}$ are orthogonal by assumption. ■

Exercise 9.6.

Write a Python/NumPy routine for implementing the steepest descent method for quadratic functions (see Example 9.2.3).

Given a small number ε, given Numpy arrays $\mathbf{x}_0, \mathbf{b}$ of length n, and given an $n \times n$ matrix $\mathbf{Q} > 0$, your code should return a close approximation to a local minimizer $\mathbf{x}^*$ of $f = \mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x} + c$.

For the stopping criterion, use the condition $\|\mathbf{D}f\left(\mathbf{x}_k\right)\|$ for some small value of ε.

Solution.

```
import numpy as np
import matplotlib.pyplot as plt

def p9_6(Q,b,x_0,epsilon=1e-8,K=500):
   """
   Steepest Descent.
   """
```

```python
norm,k=1,1
while (k<K) and (norm>epsilon): # stopping criteria

   # calculate Df
   Df = Q @ x_0 - b
   norm = np.linalg.norm(Df)
   alpha = Df Df.T @ Df / (Df.T @ Q @ Df)

   # update x sequence
   x_1 = x_0 - alpha * Df
   x_0 = x_1

if k<K: print("Converged!")

print("\nx_0:\n",x_0)
```

■

Exercise 9.7.

Write a simple Python/NumPy method for computing $\mathbf{D}f$ using forward differences and a step size of $\sqrt{\mathrm{Rerr}_f}$. It should accept a callable function $f : \mathbb{R}^n \to \mathbb{R}$, a point $\mathbf{x} \in \mathbb{R}^n$, and an estimate $\mathrm{Rerr}_f > \varepsilon$ for the maximum relative error of f near $\mathbf{x}$. It should return an estimate for $\mathbf{D}f(\mathbf{x})$.

Solution.

```python
def p9_7(f,x_0,rerr):
   """
   Computing Df with forward differences.
   """

   # set dims
   m,n = f(x_0).shape[0],x_0.shape[0]
   if len(m)==0: m=1

   Df = np.zeros((m,n)) # initialize
   h = 2*np.sqrt(rerr)

   for i in range(n):
      unit_vec = np.zeros(n)
      unit_vec[i] = 1
      Df[:,i] = (f(x_0+h*unit_vec)-f(x_0)) / h

   return Df
```

■

Exercise 9.8.

Use your differentiation method from the previous problem to construct a simple Python/NumPy method for implementing the steepest descent method for arbitrary functions, using the secant method (Exercise 6.15) for the line search.

Your method should accept a callable function f, a starting value $\mathbf{x}_0$, a small number ε, a NumPy array $\mathbf{x}_0$ of length n, and return a close approximation to a local minimizer $\mathbf{x}^*$ of f.

For the stopping criterion, use the condition $\|\mathbf{D}f(\mathbf{x}_k)\|$.

Solution.

```
import matplotlib.pyplot as plt
import numpy as np from scipy
import linalg as la

# Globals
MAX_IT = 10_000
RERR = 1+1e-10
# X = np.linspace(-10,10,1_000)
TOL = 1e-10
STARTING_VALS = [1,.5]

def Df(f,x,rerr):
   Df=np.zeros(len(x))
   for i in range(len(x)):
      Df[i]=
          ((-3*f(x)+4*f(x+2*np.sqrt(rerr)*np.eye(len(x))[:,i])-f(x+4*np.sqrt(rerr)*np.eye(len(x))[:,i]))/
         (4*np.sqrt(rerr)))
   return Df

def secant_method(x_0, x_1, ep, f):        f_pr = p9_7(f,x_0,RERR)
   k = 0
   x_k = x_0
   x_kp1 = x_1
   while k < MAX_IT:
      x_km1 = x_k
      x_k = x_kp1
      x_kp1 = x_k - f_pr(x_k) * (x_k - x_km1)/(f_pr(x_k) - f_pr(x_km1))
      k += 1
#        plt.plot(X, f(X), "b--")
#        plt.plot(x_k, f(x_k), "ko")
#        plt.show()
   if la.norm(x_kp1 - x_k) < ep * la.norm(x_k):
      break
   return x_k

def steepest_descent(f,x_vec,ep):
   x_k,k = x_vec,1
   norm = 1 + ep
   while (k<MAX_IT) and (norm>ep)
      Df_k=Df(f,x_k,ep)
      f_a = lambda x: Df(f,x_k-x_vec*Df_k.T,ep) @ -Df_k.T
      alph = secant_method(STARTING_VALS[0],STARTING_VALS[1],ep,f_a)
      x_kp1 = x_k - alph*Df_k
      norm = la.norm(Df_k)
      x_k = x_kp1
      k+=1
```

```
    return x_k
```

■

Exercise 9.9.

Apply your code from the previous problem to the Rosenbrock function

$$f(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$$

with an initial guess of $(x_0, y_0) = (-2, 2)$.

Solution.

```
def F(x_vec):
    """ rosenbrock function """
    return 100*(x_vec[1]-x_vec[0]**2)**2+(1-x_vec[0])**2
def p9_9():
    x_s = steepest_descent(F,np.array([-2,2]),TOL)
    print("optimum at",x_s,"with value of",F(x_s))
```

The optimum I found through the function I wrote above was at $\mathbf{x}^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ with value 0. ■

Exercise 9.10.

Consider the quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x}$, where $\mathbf{Q} \in \mathrm{M}_n(\mathbb{R})$ is symmetric and positive definite and $\mathbf{b} \in \mathbb{R}^n$. Show that for any initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, one iteration of Newton's method lands at the unique minimizer of f.

Solution. We know that the FOC is that $\mathbf{D}f = 0$, so since $\mathbf{D}f = \mathbf{Q}\mathbf{x} - \mathbf{b}$ and $\mathbf{D}^2 f = \mathbf{Q}$, then we know that for any $\mathbf{x}_0$, we have

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{x}_0 - \mathbf{Q}^{-1}\mathbf{D}f(\mathbf{x}_0) \\ &= \mathbf{x}_0 - \mathbf{Q}^{-1}(\mathbf{Q}\mathbf{x}_0 - \mathbf{b}) \\ &= \mathbf{Q}^{-1}\mathbf{b}, \end{aligned}$$

so $\mathbf{x}_1$ must optimize f. ■

Exercise 9.12.

Prove that if $\mathbf{A} \in \mathrm{M}_n(\mathbb{F})$ has eigenvalues $\lambda_1, \ldots, \lambda_n$ and $\mathbf{B} = \mathbf{A} + \mu\mathbf{I}$, then the eigenvectors of $\mathbf{A}$ and $\mathbf{B}$ are the same, and the eigenvalues of $\mathbf{B}$ are $\mu + \lambda_1, \mu + \lambda_2, \ldots, \mu + \lambda_n$.

Solution. Note that

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x},$$

where $\mathbf{A} = \mathbf{B} - \mu\mathbf{I}$ so we have

$$\mathbf{B}\mathbf{x} = (\lambda + \mu)\mathbf{x}$$

through some algebra. ■

Exercise 9.15.

Prove the Sherman-Morrison-Woodbury formula (9.13).
(9.13). *Let* $\mathbf{A}$ *be a nonsingular* $n \times n$ *matrix,* $\mathbf{B}$ *an* $n \times \ell$ *matrix,* $\mathbf{C}$ *a nonsingular* $\ell \times \ell$ *matrix, and* $\mathbf{D}$ *an* $\ell \times n$ *matrix. We have*

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1}. \tag{9.13}$$

Solution. Left-multiply the RHS by the LHS, and we get the identity:

$$\begin{aligned}
& (\mathbf{A}+\mathbf{BCD})\left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{C}\right)^{-1}\mathbf{DA}^{-1}\right) \\
= \ & \mathbf{I}+\mathbf{BCDA}^{-1} - \left(\mathbf{B}+\mathbf{BCDA}^{-1}\mathbf{B}\right)\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1} \\
= \ & \mathbf{I}+\mathbf{BCDA}^{-1} - \mathbf{BC}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1} \\
= \ & \mathbf{I}+\mathbf{BCDA}^{-1} - \mathbf{BCDA}^{-1} \\
= \ & \mathbf{I}.
\end{aligned}$$

■

Exercise 9.16.

Use (9.13) to derive (9.14).

$$\mathbf{A}_k^{-1} = \mathbf{A}_{k-1}^{-1} + \frac{(\mathbf{s}_{k-1}-\mathbf{A}_{k-1}^{-1}\mathbf{y}_{k-1})\mathbf{s}_{k-1}^{\mathsf{T}}\mathbf{A}_{k-1}^{-1}}{\mathbf{s}_{k-1}^{\mathsf{T}}\mathbf{A}_{k-1}^{-1}\mathbf{y}_{k-1}}. \tag{9.14}$$

Solution. We start with the Broyden's Method updating criteria

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \frac{\mathbf{y}_k - \mathbf{A}_k\mathbf{s}_k}{\|\mathbf{s}_k\|^2}\mathbf{s}_k^{\mathsf{T}}$$

and recognize the following matrices:

$$\underbrace{\mathbf{A}_{k+1} = \underbrace{\mathbf{A}_k}_{n\times n} + \underbrace{\frac{\overbrace{\mathbf{y}_k}^{n\times\ell} - \overbrace{\mathbf{A}_k}^{n\times n}\overbrace{\mathbf{s}_k}^{n\times\ell}}{\|\mathbf{s}_k\|^2}}_{\mathbf{B}} \cdot \underbrace{1}_{\mathbf{C}} \cdot \underbrace{\overbrace{\mathbf{s}_k^{\mathsf{T}}}^{\ell\times n}}_{\mathbf{D}}.}_{(*)}$$

From here, we apply the Sherman-Morrison-Woodbury formula,

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1},$$

to equation $(*)$ to yield

$$\mathbf{A}_k^{-1} = \mathbf{A}_{k-1}^{-1} + \frac{\left(\mathbf{s}_{k-1}-\mathbf{A}_{k-1}^{-1}\mathbf{y}_{k-1}\right)\mathbf{s}_{k-1}^{\mathsf{T}}\mathbf{A}_{k-1}^{-1}}{\mathbf{s}_{k-1}^{\mathsf{T}}\mathbf{A}_{k-1}^{-1}\mathbf{y}_{k-1}},$$

which is of the form (9.14). ■

Exercise 9.18.

Let $\mathbf{Q} \in \mathrm{M}_n(\mathbb{R})$ satisfy $\mathbf{Q} > 0$, and let f be the quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{Q}\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x} + c$. Given a starting point $\mathbf{x}_0$ and $\mathbf{Q}$-conjugate directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{n-1}$ in $\mathbb{R}^n$, show that the optimal line search solution for $\mathbf{x}_{k-1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ (that is, the α which minimizes $\phi_k(\alpha) = f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$) is given by $\alpha_k = \frac{\mathbf{r}_k^\mathsf{T}\mathbf{d}_k}{\mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k}$, where $\mathbf{r}_k = \mathbf{b} - \mathbf{Q}\mathbf{x}_k$.

Solution. For $\mathbf{Q} > 0$, we have that $\mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k > 0$ by definition, so we can find a minimizer α_k of $\phi_k(\alpha_k)$ for $\phi_k(\alpha_k)$ defined as follows.

$$\begin{aligned}
\phi_k(\alpha_k) &= f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \\
&= \frac{1}{2}\mathbf{x}_k^\mathsf{T}\mathbf{Q}\mathbf{x}_k + \alpha_k \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{x}_k \\
&\quad + \frac{1}{2}\alpha_k^2 \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k - \mathbf{x}_k^\mathsf{T}\mathbf{b} - \alpha_k \mathbf{d}_k^\mathsf{T}\mathbf{b}.
\end{aligned}$$

Taking FOCs,

$$\begin{aligned}
0 &= \frac{\partial \phi_k(\alpha_k)}{\partial \alpha_k} \\
&= \alpha_k \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k - \mathbf{d}_k^\mathsf{T}\mathbf{b} + \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{x}_k
\end{aligned}$$

so

$$\begin{aligned}
\alpha_k \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k &= \mathbf{d}_k^\mathsf{T}\mathbf{b} + \mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{x}_k \\
&= \mathbf{d}_k^\mathsf{T}(\mathbf{b} - \mathbf{Q}\mathbf{x}_k) \\
&= \mathbf{d}_k^\mathsf{T}\mathbf{r}_k,
\end{aligned}$$

so

$$\alpha_k = \mathbf{d}_k^\mathsf{T}\mathbf{r}_k \left(\mathbf{d}_k^\mathsf{T}\mathbf{Q}\mathbf{d}_k\right)^{-1}.$$

■

Exercise 9.20.

Prove Lemma 9.5.5.

Lemma 9.5.5. *In the Conjugate Gradient Algorithm, $\mathbf{r}_i^\mathsf{T}\mathbf{r}_k = 0$ for all $i < k$.*

Solution. Let $\mathcal{B}_i = \{\mathbf{b} - \mathbf{Q}\mathbf{x}_k\}_{k=0}^{i-1}$ be our basis. Then, through Gram-Schmidt, we would have

$$\mathbf{r}_k = \mathbf{b} - \mathbf{Q}\mathbf{x}_k - \sum_{i=0}^{k-1} \frac{\langle \mathbf{r}_i, \mathbf{b} - \mathbf{Q}\mathbf{x}_k \rangle}{\|\mathbf{r}_i\|^2}\mathbf{r}_i;$$

here we see that each $\mathbf{r}_i$ is a linear combination of some elements in $\{\mathbf{b} - \mathbf{Q}\mathbf{x}_k\}_{k=0}^{i-1}$, so then it follows that $\mathcal{B}_i = \operatorname{span}\left(\{\mathbf{r}_k\}_{k=0}^{i-1}\right)$. So, our conjugate gradient problem becomes

$$\min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{B}_i} f(\mathbf{x}),$$

where $\mathbf{x}^* = \mathbf{x}_i$, so we have that $h = 0$ minimizes

$$\min_h f\left(\mathbf{x}_i + h\left(\mathbf{b} - \mathbf{Q}\mathbf{x}_j\right)\right)$$

for all $i > j$. So, our FOC is

$$\begin{aligned} 0 &= \mathbf{D}f\left(\mathbf{x}_i\right)\left(\mathbf{b} - \mathbf{Q}\mathbf{x}_j\right) \\ &= \left(\mathbf{Q}\mathbf{x}_i - \mathbf{b}\right)^{\mathsf{T}}\left(\mathbf{b} - \mathbf{Q}\mathbf{x}_j\right) \\ &= -\left(\mathbf{b} - \mathbf{Q}\mathbf{x}_i\right)^{\mathsf{T}}\left(\mathbf{b} - \mathbf{Q}\mathbf{x}_j\right). \end{aligned}$$

■