Nonlinear Optimization Fall 2021
HW 5 Sample Solutions

Q1 (a) This follows from our Taylor Approximation Theorem,

$$\left| f(x_k+s) - \left(f(x_k) + \nabla f(x_k)^T s\right)\right| \leq \frac{L}{2}\|s\|_2^2, \quad \forall s$$

by taking $s = -\alpha B_k^{-1} g_k$. This gives

$$f(x_{k+1}) - \left(f(x_k) + g_k^T(-\alpha B_k^{-1} g_k)\right) \leq \frac{L}{2}\|-\alpha B_k^{-1} g_k\|_2^2.$$

(dropping the absolute value only shrinks the LHS above)

Hence $$f(x_{k+1}) \leq f(x_k) - \alpha g_k^T \overbrace{B_k^{-1} g_k}^{-p_k} + \frac{L\alpha^2}{2}\|\overbrace{B_k^{-1} g_k}^{p_k}\|_2^2. \quad \square$$

Then the claim follows from the following two bounds...

$$g_k^T B_k^{-1} g_k \geq \lambda_{min}(B_k^{-1})\|g_k\|_2^2$$
$$= \lambda_{max}^{-1}(B_k)\|g_k\|_2^2$$

Critically $B_k \succ 0$ and so its eigenval are all positive. So its largest eigenval is 1/its inverses smallest eigenvalue.

and

$$\|B_k^{-1} g_k\|_2 \leq \lambda_{max}(B_k^{-1})\|g_k\|_2$$
$$= \lambda_{min}^{-1}(B_k)\|g_k\|_2$$

And vice versa needed here for 2nd bound

$$\Rightarrow f(x_{k+1}) \leq f(x_k) - \left(\frac{\alpha}{\lambda_{max}} - \frac{L\alpha^2}{2\lambda_{min}^2}\right)\|g_k\|_2^2. \quad \square$$

(b) Note this is a concave quadratic in α.

So it maximizes at $\nabla\left(\frac{\alpha}{\lambda_{max}} - \frac{L\alpha^2}{2\lambda_{min}^2}\right)\|g_k\|_2^2 = 0$

$$\Leftrightarrow \left(\frac{1}{\lambda_{max}} - \frac{L\alpha}{\lambda_{min}^2}\right)\|g_k\|_2^2 = 0$$

$$\Leftrightarrow \quad \alpha = \frac{\lambda_{min}^2}{\lambda_{max}} \cdot \frac{1}{L}.$$

This maximizing α is positive since $\lambda_{min}, \lambda_{max}, L > 0$ (strictly by assumption).

The maximum value is the

$$\|g_k\|_2^2\left(\frac{\lambda_{min}^2}{\lambda_{max}^2 L} - \frac{\lambda_{min}^2}{2\lambda_{max}^2 L}\right) = \frac{\lambda_{min}^2 \|g_k\|_2^2}{2\lambda_{max}^2 L} > 0$$

and so a well choosen stepsize can always give descent.

(c) By (a) and (b), we know the descent condition

$$f(x_{k+1}) \le f(x_k) - \frac{\lambda_{min}^2 \|g_k\|_2^2}{2\lambda_{max}^2 L}.$$

Iteratively applying this, we know for any minimizer x^*,

$$f(x^*) \le f(x_{k+1}) \le f(x_0) - \sum_{i=0}^{k} \frac{\lambda_{min}^2}{2\lambda_{max}^2 L}\|g_k\|_2^2.$$

$$\Rightarrow \min_{i=0..k} \|g_i\|_2^2 \le \frac{\sum_{i=0}^{k}\|g_i\|_2^2}{k+1} \le \frac{2L(f(x_0)-f(x^*))\lambda_{max}^2}{(k+1)\lambda_{min}^2}.$$

Taking a squareroot to measure the gradient norm without a square gives the claimed guarantee

$$\min_{i=0..k} \|\nabla f(x_i)\|_2 \le \frac{\sqrt{2L(f(x_0)-f(x^*))}\,\lambda_{max}}{\sqrt{k+1}\;\lambda_{min}}. \qquad \square$$

(d) Selecting $B_k = LI$, our descent direction $p_k = -B_k^{-1}\nabla f(x_k) = -\frac{\nabla f(x_k)}{L}$

becomes aligned with the gradient direction.

Note $\lambda_{min}(B_k) = \lambda_{max}(B_k) = L$

Our optimal stepsize $\alpha = \frac{\lambda_{min}^2}{\lambda_{max} L} = \frac{L^2}{L^2} = 1$.

So our algorithm iterates $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$.

This is exactly gradient descent with our theoretically best stepsize from lecture.

Moreover, the rate in (c) reduces to be exactly our GD rate from lecture

$$\min_{i=0..k} \|\nabla f(x_i)\| \le \sqrt{\frac{2L\,(f(x_0) - f(x^*))}{k+1}}.$$

(in lecture, we wrote the rate for $\|\nabla f(x_i)\|_2^2$, so the rate may look different by a square but it is equivalent to our previous work).

$\boxed{\text{Q2}}$ (a) Noting B_k is symmetric and the rank one updates are symmetric ($y_{k+1} y_{k+1}^T$ and $B_k s_{k+1} s_{k+1}^T B_k^T$), it follows that there sum is symmetric.

Then we only need to check B_{k+1} has all positive eigenvalues. Equivalently, let's show $z^T B_{k+1} z > 0$ for all $z \neq 0$. (If we only show ≥ 0, then we only have positive semidefiniteness)

$$z^T B_{k+1} z = z^T B_k z + \frac{(y_{k+1}^T z)^2}{y_{k+1}^T s_{k+1}} - \frac{(z^T B_k s_{k+1})^2}{s_{k+1}^T B_k s_{k+1}}$$

$$= \underbrace{\frac{(z^T B_k z)(s_{k+1}^T B_k s_{k+1}) - (z^T B_k s_{k+1})^2}{s_{k+1}^T B_k s_{k+1}}} + \underbrace{\frac{(y_{k+1}^T z)^2}{y_{k+1}^T s_{k+1}}}$$

≥ 0 by Cauchy-Schwartz using the inner product of $\langle a, b \rangle_{B_k} = a^T B_k b$ (which states $\langle a,a \rangle \langle b,b \rangle - \langle a,b \rangle^2 \geq 0$ and $= 0$ only if $a \parallel b$.)

≥ 0 since $y^T s > 0$ by assumption and $= 0$ only if $y_{k+1} \perp z$

Thus $z^T B_{k+1} z \geq 0$ and so B_k is positive semidefinite. To rule out this being $= 0$, the two inequalities above are only tight if $z \parallel s_{k+1}$ and $y_{k+1} \perp z$.

Hence this $= 0$ only when $s_{k+1} \perp y_{k+1}$, but we assume $s_{k+1}^T y_{k+1} > 0$.

$$\cancel{\Leftrightarrow y_{k+1}^T B_k s_{k+1} = 0}$$

$$\cancel{\Leftrightarrow (\nabla f(x_{k+1}) - \nabla f(x_k))^T B_k(-\alpha B_k^{-1} \nabla f(x_k)) = 0}$$

$$\cancel{\Leftrightarrow (\nabla f(x_{k+1}) - \nabla f(x_k))^T \nabla f(x_k) = 0}$$

$\Rightarrow z^T B_{k+1} z > 0$ for all nonzero z. $\square$

[Showing an alternative to the direct Woodbury formula here]

(b) We can do this using the Sherman Morrison formula from class twice, so long as we check the needed nonzero condition at each application (i.e. that our matrices stay invertible).

Recall BFGS has
$$B_{k+1} = \underbrace{B_k + \frac{y_{k+1} y_{k+1}^T}{y_{k+1}^T s_{k+1}}}_{=: M_k} - \frac{B_k s_{k+1} s_{k+1}^T B_k^T}{s_{k+1}^T B s_{k+1}}.$$

First let's invert this M_k, rank one update (Check it is invertible: $B_k \succ 0$ and $y_{k+1} y_{k+1}^T \succeq 0$ $\Rightarrow M_k \succ 0$.)

$$M_k^{-1} = B_k^{-1} - \frac{(B_k^{-1} y_{k+1})(B_k^{-1} y_{k+1})^T}{(y_{k+1}^T s_{k+1})^2 + y_{k+1}^T B_k^{-1} y_{k+1}}$$

Equivalently, need this denominator nonzero, which follows from $B_k \succ 0$ and $y^T s > 0$.

Then we can invert B_{k+1} as a rank one update from M_k (Check invertibility: (a) gives this since $B_{k+1} \succ 0$).

$$B_{k+1}^{-1} = M_k^{-1} - \frac{(M_k^{-1} B_k s_{k+1})(M_k^{-1} B_k s_{k+1})^T}{(s_{k+1}^T B_k s_{k+1})^2 - s_{k+1}^T B_k^T M_k^{-1} B_k s_{k+1}}$$

$$= B_k^{-1} - \frac{(B_k^{-1} y_{k+1})(B_k^{-1} y_{k+1})^T}{(y_{k+1}^T s_{k+1})^2 + y_{k+1}^T B_k^{-1} y_{k+1}}$$

$$- \frac{\left(s_{k+1} - \frac{(B_k^{-1} y_{k+1} y_{k+1}^T s_{k+1})}{(y_{k+1}^T s_{k+1})^2 + y_{k+1}^T B_k^{-1} y_{k+1}}\right)\left(s_{k+1} - \frac{B_k^{-1} y_{k+1} y_{k+1}^T s_{k+1}}{(y_{k+1}^T s_{k+1})^2 + y_{k+1}^T B_k^{-1} y_{k+1}}\right)^T}{(s_{k+1}^T B_k s_{k+1})^2 - s_{k+1}^T B_k^T \underbrace{\left(s_{k+1} - \frac{B_k^{-1} y_{k+1} y_{k+1}^T s_{k+1}}{(y_{k+1}^T s_{k+1})^2 + y_{k+1}^T B_k^{-1} y_{k+1}}\right)}_{= M_k^{-1} B_k s_{k+1}}}$$

$\boxed{\text{Lots of Cancellations}}$

$$= \left(I - \frac{s_{k+1} y_{k+1}^T}{y_{k+1}^T s_{k+1}}\right) B_k^{-1} \left(I - \frac{y_{k+1} s_{k+1}^T}{y_{k+1}^T s_{k+1}}\right) + \frac{s_{k+1} s_{k+1}^T}{y_{k+1}^T s_{k+1}}.$$

This is similar to DFP applied to B_k^{-1} (instead of B_k), swapping the roles of s_{k+1} and y_{k+1}.

(Recall DFP sets $B_{k+1} = \left(I - \frac{y_{k+1} s_{k+1}^T}{y_{k+1}^T s_{k+1}}\right) B_k \left(I - \frac{s_{k+1} y_{k+1}^T}{y_{k+1}^T s_{k+1}}\right) + \frac{y_{k+1} y_{k+1}^T}{y_{k+1}^T s_{k+1}}$.)

[As a result, we can view tracking the inverse of B_k^{-1} during the execution of a quasi-Newton method as repeatedly applying DFP to approximate the inverse Hessian with the inverted Secant Equation $B_k^{-1} y_{k+1} = s_{k+1}$.]

In [1]:

```
import numpy as np
import scipy
from scipy.optimize import minimize_scalar, rosen, rosen_der, rosen_h
#These packages provide definitions for the rosenbrock function and i
#They are just a line or two to implement from scratch, if not using
```

In [2]:

```
def linesearch(x,p): #Given (x,p) return a minimizing f(x+ap)
    f = lambda a: rosen(x+a*p)
    return minimize_scalar(f).x
```

In [3]:
```
#(a) Run a 100 steps of gradient descent, after ~4000, we would get n
x = np.array([0,0])
for i in range(100):
    print(x)
    p = -1*rosen_der(x)
    a = linesearch(x,p)
    x = x+a*p
```

```
[0 0]
[0.16126202 0.        ]
[0.16126202 0.02600544]
[0.21133889 0.02600544]
[0.21133889 0.04466413]
[0.24508248 0.04466413]
[0.24508248 0.06006542]
[0.27112227 0.06006542]
[0.27112227 0.07350729]
[0.29257001 0.07350728]
[0.29257001 0.08559721]
[0.3109308  0.08559721]
[0.3109308  0.09667796]
[0.32705634 0.09667796]
[0.32705634 0.10696585]
[0.34147918 0.10696585]
[0.34147918 0.11660803]
[0.35455629 0.11660803]
[0.35455629 0.12571016]
[0.36653957 0.12571016]
[0.36653957 0.13435126]
[0.37761395 0.13435126]
[0.37761395 0.14259229]
[0.38791953 0.14259229]
[0.38791953 0.15048156]
[0.39756524 0.15048157]
[0.39756524 0.15805812]
[0.40663753 0.15805812]
[0.40663753 0.16535408]
[0.41520632 0.16535408]
[0.41520632 0.17239629]
[0.42332899 0.17239629]
[0.42332899 0.17920743]
[0.43105328 0.17920743]
[0.43105328 0.18580693]
[0.43841936 0.18580693]
[0.43841936 0.19221153]
[0.44546139 0.19221153]
[0.44546139 0.19843585]
[0.45220869 0.19843585]
[0.45220869 0.20449269]
[0.45868659 0.2044927 ]
[0.45868659 0.21039338]
[0.46491719 0.21039338]
[0.46491718 0.21614799]
[0.47091985 0.21614799]
[0.47091985 0.22176551]
[0.47671168 0.22176551]
[0.47671168 0.22725403]
```

```
[0.48230784 0.22725403]
[0.48230784 0.23262085]
[0.48772184 0.23262085]
[0.48772184 0.23787259]
[0.49296581 0.2378726 ]
[0.49296581 0.24301528]
[0.49805063 0.24301529]
[0.49805063 0.24805442]
[0.50298614 0.24805443]
[0.50298614 0.25299506]
[0.50778127 0.25299506]
[0.50778127 0.25784182]
[0.51244412 0.25784182]
[0.51244412 0.26259898]
[0.5169821  0.26259898]
[0.5169821  0.26727049]
[0.52140198 0.26727049]
[0.52140198 0.27186002]
[0.52570997 0.27186002]
[0.52570997 0.27637097]
[0.52991179 0.27637097]
[0.52991179 0.2808065 ]
[0.53401271 0.2808065 ]
[0.53401271 0.28516957]
[0.53801761 0.28516957]
[0.53801761 0.28946295]
[0.54193101 0.28946295]
[0.54193101 0.29368922]
[0.54575709 0.29368922]
[0.54575709 0.2978508 ]
[0.54949976 0.2978508 ]
[0.54949976 0.30194998]
[0.55316265 0.30194999]
[0.55316265 0.30598892]
[0.55674916 0.30598892]
[0.55674916 0.30996962]
[0.56026245 0.30996962]
[0.56026245 0.31389401]
[0.5637055  0.31389401]
[0.5637055  0.31776389]
[0.5670811  0.31776389]
[0.5670811  0.32158097]
[0.57039186 0.32158097]
[0.57039186 0.32534687]
[0.57364025 0.32534687]
[0.57364025 0.32906314]
[0.57682859 0.32906314]
[0.57682859 0.33273122]
[0.57995907 0.33273122]
[0.57995907 0.33635252]
[0.58303375 0.33635252]
```

In [4]:
```
#(b) Run a 100 steps of Newton, after 2, we get optimal
x = np.array([0,0])
for i in range(100):
    print(x)
    p = scipy.linalg.solve(rosen_hess(x), -1*rosen_der(x))
#    a = linesearch(x,p) #If we include a linesearch it takes !10 ste
    x = x+p
```

```
[0 0]
[1. 0.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
```

```
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
```

In [5]:
```
#(c) Run a 100 steps of BFGS (after ~13 steps, we reach optimal, then
x = np.array([0,0])
B = np.array([[1,0],[0,1]])
for i in range(100):
    print(x)
    g = rosen_der(x)
    p = -1*scipy.linalg.solve(B, -1*g)
    a = linesearch(x,p)
    x = x+a*p
    s = a*p
    y = rosen_der(x)-g
    if (np.linalg.norm(s)>=0.00000001):
        B = B + np.outer(y,y)/np.dot(y,s) - np.outer(np.dot(B,s),np.d
```

```
[0 0]
[0.16126202 0.        ]
[0.2928363 0.0505949]
[0.34461573 0.12707842]
[0.46111375 0.19622896]
[0.53849589 0.26517212]
[0.8719792  0.75566096]
[0.86857966 0.75309137]
[0.93942519 0.8775552 ]
[0.96969259 0.94192242]
[0.99046445 0.9803038 ]
[1.0007949  1.00138922]
[0.99992822 0.99985329]
[0.99999959 0.99999927]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
```

```
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]
```

```
[1. 1.]
[1. 1.]
```