Student Name: Xinxi Zhang
NetID: XZ657
RUID: 219004759

MATH FOUND DS (16:198:501)
Homework 2: Optimization

Problem 1: Momentum Methods and Descent Directions

Consider a function:

$$F(\underline{x}) = \frac{1}{2}\underline{x}^T Q\underline{x} - \underline{x}^T\underline{c} \tag{1}$$

where $\underline{x}$ and $\underline{c}$ are real vectors of dimension D, and Q is a real symmetric $D \times D$ matrix.

Questions:

1. Show that $\nabla_{\underline{x}} F(\underline{x}) = Q\underline{x} - \underline{c}$
   From HW1, we have that:

$$\nabla_{\underline{x}}[\underline{x}^T\underline{c}] = \underline{c}, \quad \nabla_{\underline{x}}[\underline{x}^T Q\underline{x}] = 2Q\underline{x}, \quad (Q \text{ is a real symmetric matrix})$$

$$\begin{aligned} \Rightarrow \quad \nabla_{\underline{x}} F(\underline{x}) &= \frac{1}{2}\nabla_{\underline{x}}[\underline{x}^T Q\underline{x}] + \nabla_{\underline{x}}[\underline{x}^T\underline{c}] \\ &= Q\underline{x} - \underline{c} \end{aligned}$$

2. Show that if Q is positive definite, then F has a unique minimizer given by $\underline{x}^* = Q^{-1}\underline{c}$
   Let:

$$Q = Q^T = \begin{bmatrix} - & \underline{Q}_1^T & - \\ - & \underline{Q}_2^T & - \\ - & \cdots & - \\ - & \underline{Q}_d^T & - \end{bmatrix}$$

$$\Rightarrow \quad \nabla_{\underline{x}} F(\underline{x}) = \begin{bmatrix} \underline{Q}_1^T \cdot \underline{x} \\ \underline{Q}_2^T \cdot \underline{x} \\ \cdots \\ \underline{Q}_D^T \cdot \underline{x} \end{bmatrix} = \begin{bmatrix} \underline{x}^T\underline{Q}_1 \\ \underline{x}^T\underline{Q}_2 \\ \cdots \\ \underline{x}^T\underline{Q}_D \end{bmatrix}$$

$$\begin{aligned} \Rightarrow \quad H(F(\underline{x})) &= \nabla_{\underline{x}}[\nabla_{\underline{x}} F(\underline{x})] \\ &= \begin{bmatrix} | & | & | & | \\ \nabla_{\underline{x}}[\underline{x}^T\underline{Q}_1] & \nabla_{\underline{x}}[\underline{x}^T\underline{Q}_2] & \cdots & \nabla_{\underline{x}}[\underline{x}^T\underline{Q}_D] \\ | & | & | & | \end{bmatrix} \\ &= \begin{bmatrix} | & | & | & | \\ \underline{Q}_1 & \underline{Q}_2 & \cdots & \underline{Q}_D \\ | & | & | & | \end{bmatrix} = Q \end{aligned}$$

So, we know that $H(F(\underline{x}))$ is positive definite meaning that $F(\underline{x})$ is convex through every directions of every dimension.
Therefore, we can conclude that F has a unique minimizer when $\nabla_{\underline{x}} F(\underline{x}) = 0$, which is $\underline{x}^* = Q^{-1}\underline{c}$.

Traditionally, gradient descent updates take the form:

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla F(\underline{x}_k) \tag{2}$$

We occasionally modify this by modifying the descent direction. **Momentum Methods** include an additional descent direction term:

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla F(\underline{x}_k) + \beta_k(\underline{x}_k - \underline{x}_{k-1}) \tag{3}$$

At step k, this moves the iterate from k to $k+1$ slightly in the direction it moved from $k-1$ to k, as though the iterate had some momentum pulling it in this direction. We want to analyze this in a little more detail in this problem.
Consider a general update of the form:

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \underline{p}_k + \beta_k \underline{q}_k \tag{4}$$

where $p_k = \nabla_{\underline{x}} F(\underline{x}_k)$, and q_k represents the additional modification to the descent direction we are going to make.

Question:

- Show that:

$$F(\underline{x}_{k+1}) = F(\underline{x}_k) + \frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k\beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{q}_k \tag{5}$$

  From Taylor, we have:

$$F(\underline{x} + \underline{\delta}) = F(\underline{x}) + [\nabla_{\underline{x}} F(\underline{x})]^T \underline{\delta} + \frac{1}{2}\underline{\delta}^T H(\underline{x})\underline{\delta} + O(||\underline{\delta}||^3)$$

  And we have $H(\underline{x}) = Q$, $p_k = \nabla_{\underline{x}} F(\underline{x}_k)$, so:

$$\begin{aligned} F(\underline{x}_{k+1}) &= F(\underline{x}_k) + p_k^T(-\alpha_k \underline{p}_k + \beta_k \underline{q}_k) + \frac{1}{2}(-\alpha_k \underline{p}_k + \beta_k \underline{q}_k)^T Q(-\alpha_k \underline{p}_k + \beta_k \underline{q}_k) + O(||\underline{\delta}||^3) \\ &= F(\underline{x}_k) + \frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k\beta_k p_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k \end{aligned}$$

Questions:

- For standard Gradient Descent, taking $\beta_k = 0$, find an expression for the optimal step size α_k in terms of $\underline{p}_k$.

  We have already proved equation (5), so when we taking $\beta_k = 0$:

$$F(\underline{x}_{k+1}) = F(\underline{x}_k) + \frac{1}{2}\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - \alpha_k \underline{p}_k^T \underline{p}_k$$

  In order to get the optimal step size $\tilde{\alpha}_k$, we want:

$$\tilde{\alpha}_k = \min_{\alpha_k}(\frac{1}{2}\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - \alpha_k \underline{p}_k^T \underline{p}_k)$$

So we let:

$$\tilde{\alpha}_k = \alpha_k, \text{ when } \frac{\mathrm{d}(\frac{1}{2}\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - \alpha_k \underline{p}_k^T \underline{p}_k)}{\mathrm{d}\alpha_k} = 0$$

$$\Rightarrow \quad \tilde{\alpha}_k \underline{p}_k^T Q \underline{p}_k = \underline{p}_k^T \underline{p}_k$$
$$\Rightarrow \quad \tilde{\alpha}_k = \frac{\underline{p}_k^T \underline{p}_k}{\underline{p}_k^T Q \underline{p}_k}$$

- For the general case, find an expression for the optimal stepsize α_k and generalized momentum factor β_k in terms of p_k and q_k.

  From equation (5), we have:

$$F(\underline{x}_{k+1}) = F(\underline{x}_k) + \frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k \beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k$$

  In order to get the optimal step size $\tilde{\alpha}_k$ and $\tilde{\beta}_k$, we want:

$$\tilde{\alpha}_k = \min_{\alpha_k}(\frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k \beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k)$$
$$\tilde{\beta}_k = \min_{\beta_k}(\frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k \beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k)$$

  So we let:

$$\tilde{\alpha}_k = \alpha_k, \text{ when } \frac{\mathrm{d}(\frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k \beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k)}{\mathrm{d}\alpha_k} = 0$$
$$\tilde{\beta}_k = \beta_k, \text{ when } \frac{\mathrm{d}(\frac{1}{2}(\alpha_k^2 \underline{p}_k^T Q \underline{p}_k - 2\alpha_k \beta_k \underline{p}_k^T Q \underline{q}_k + \beta_k^2 \underline{q}_k^T Q \underline{q}_k) - \alpha_k \underline{p}_k^T \underline{p}_k + \beta_k \underline{q}_k^T \underline{p}_k)}{\mathrm{d}\beta_k} = 0$$

$$\Rightarrow \quad \begin{cases} \tilde{\alpha}_k \underline{p}_k^T Q \underline{p}_k - \tilde{\beta}_k \underline{p}_k^T Q \underline{q}_k - \underline{p}_k^T \underline{p}_k &= 0 \\ \tilde{\beta}_k \underline{q}_k^T Q \underline{q}_k - \tilde{\alpha}_k \underline{p}_k^T Q \underline{q}_k + \underline{q}_k^T \underline{p}_k &= 0 \end{cases}$$

$$\Rightarrow \quad \tilde{\alpha}_k = \frac{\underline{p}_k^T \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{q}_k^T \underline{p}_k \underline{p}_k^T Q \underline{q}_k}{\underline{p}_k^T Q \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{p}_k^T Q \underline{q}_k \underline{p}_k^T Q \underline{q}_k}$$
$$\tilde{\beta}_k = \frac{\underline{p}_k^T \underline{p}_k \underline{p}_k^T Q \underline{q}_k - \underline{q}_k^T \underline{p}_k \underline{p}_k^T Q \underline{p}_k}{\underline{p}_k^T Q \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{p}_k^T Q \underline{q}_k \underline{p}_k^T Q \underline{q}_k}$$

In the remainder of this problem, we want to experiment with the behavior of gradient descent and the modified momentum methods. To do so, we need a Q and a $\underline{c}$.

- Take $D = 10$
- Write a function to generate a D-dimensional vector where each component is drawn from a standard normal distribution.
- Use this function to generate $\underline{c}$.

- Generate a $D \times D$ matrix A, where every column is a vector generated in this way.
- Take $Q = A^T A$.

Questions:

- Explain why Q is almost certainly positive definite when generated this way.

  For any $\underline{x} \neq 0$:

$$\begin{aligned} \underline{x}^T Q \underline{x} &= \underline{x}^T A^T A \underline{x} \\ &= (\underline{x}^T A^T)(A\underline{x}) \\ &= (A\underline{x})^T (A\underline{x}) \\ &= ||A\underline{x}||_2^2 \geq 0 \end{aligned}$$

  So we can state that Q is positive definite except A is not invertible. And A being invertible means that A's collumn space is linearly dependent. The most possible situation is that one column of A can be expressed by the other columns, of which the probability is 0. We can think of it as the probability of randomly throwing a dart so that it hit on a lower dimensional span in a higher dimensional space. It's like the probability to randomly throw a dart exactly on a point of the dartborad, which is also 0.

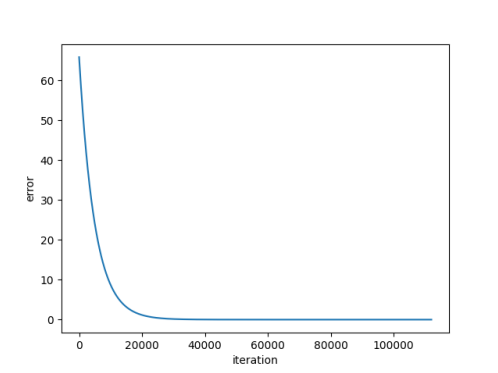
  So we can claim that Q is almost certainly positive definite when generated this way.

Gradient Descent

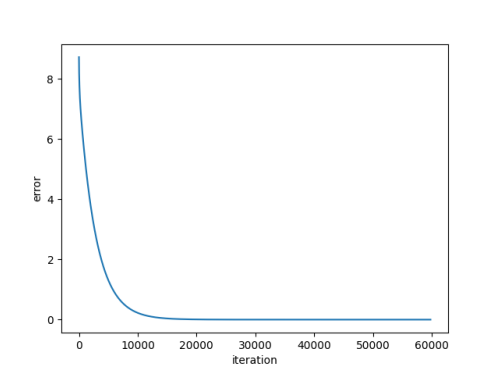
Questions:

- For an $\alpha > 0$ small enough to guarantee convergence, implement gradient descent for this problem. Plot the error of $||\underline{x}_k - \underline{x}^*||$, and show that it agrees with the exponential convergence we expect from the results in class. *How can you verify this?*

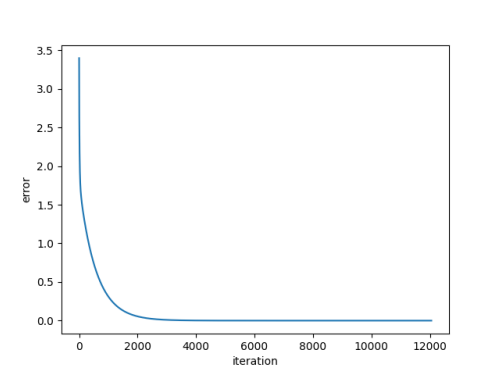
  In order to show generality, the experiments have been run 4 times. And The plots of error $||\underline{x}_k - \underline{x}^*||$ are shown below:



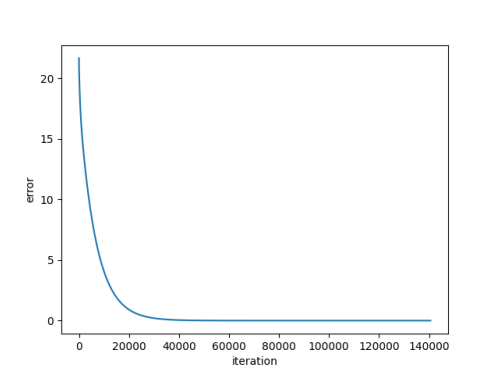
(a.1)



(b.1)

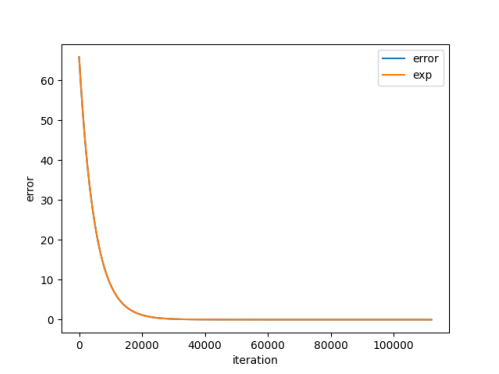


(c.1)

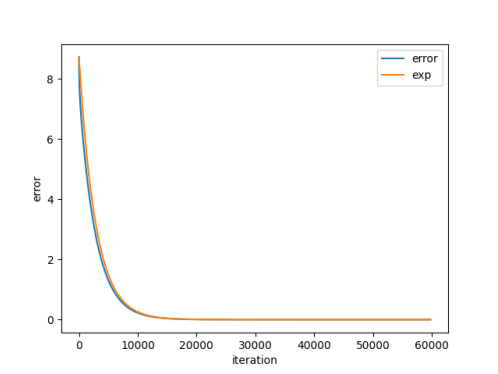


(d.1)

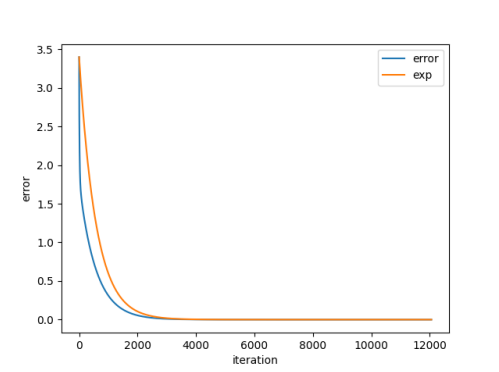
And from class, we expect the error to converge to 0 with exponential speed, which means that $||\underline{x}_k - \underline{x}^*||$ behaves like an exponential function. So what we can do to verify this is to construct a exponential function $exp(x) = c * u^x$ by sampling two points from $error(k) = ||\underline{x}_k - \underline{x}^*||$. Then we can plot both of them to see whether these two function behave the same.



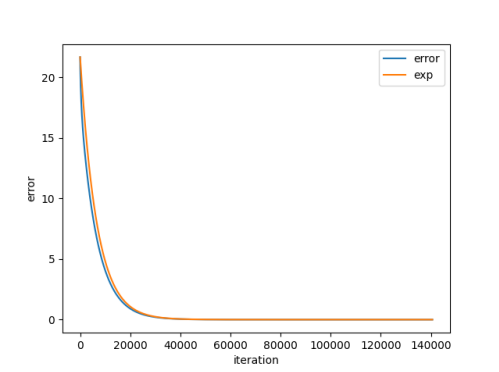
(a.2)



(b.2)



(c.2)



(d.2)

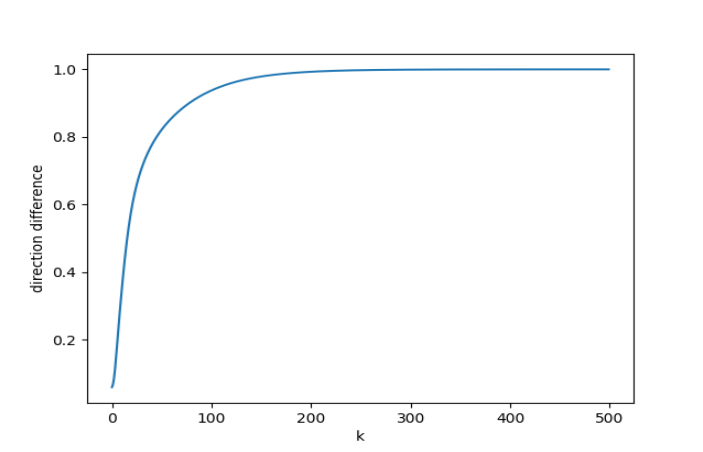
We can conclude from plots above that the error $||\underline{x}_k - \underline{x}^*||$ agrees with the exponential convergence.

- Additionally, we'd like to know in what manner the iterates converge to the minimum. In going from $\underline{x}_k$ to $\underline{x}_{k+1}$, are we aimed directly at the minimizer $\underline{x}^*$, or are we off slightly? We can understand this by looking at the angle between $\underline{x}_{k+1} - \underline{x}_k$ and $x^* - x_k$. To get at this angle, we can plot

$$\frac{[\underline{x}_{k+1} - \underline{x}_k]^T[\underline{x}_* - \underline{x}_k]}{||\underline{x}_{k+1} - \underline{x}_k||||\underline{x}_* - \underline{x}_k]||} \tag{6}$$

as a function of k. What does the plot suggest about how the iterates approach the minimizer?

The plot of the angle between $\underline{x}_{k+1} - \underline{x}_k$ and $x^* - x_k$ is showned below:



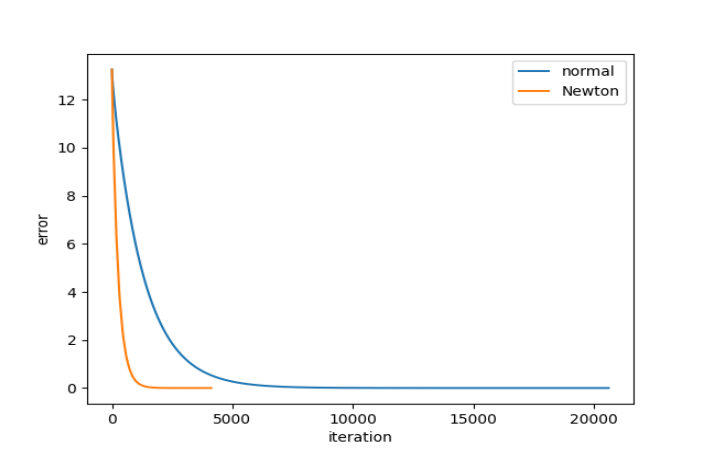
What we can conclude from this is that: if we are far from the minimizer, the information of the first derivative won't directly guide the optimization step to the minimizer. However, when we keep approaching the minimum, the differences between the directions of optimization step and the directions aimed directly at the minimizer become smaller and smaller.

- Are the rates of convergence of the iterates and the behavior of the approach consistent across different starting points, stepsizes, and $Q, \underline{c}$ choices?

  If the starting points are very close to the minimizer, the optimization step may point directly to the minimizer. However, the rates of convergence of the iterates will not change across different starting points, stepsizes, and $Q, \underline{c}$ choices. This is because if the convergence is promised by a sufficiently small α, the error will converge exponentially fast.

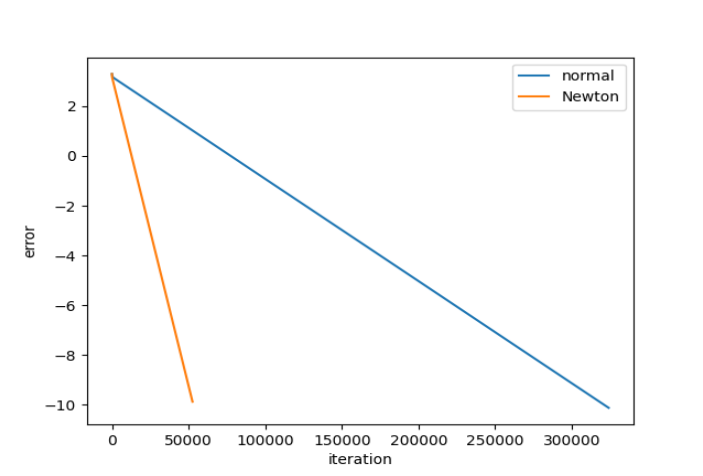
Questions:

- Instead of taking α as a constant, take α_k to be the optimal stepsize for gradient descent as found previously.
- How does this change the rate of convergence? Be as specific as you can.
  Firstly what we can do is to plot two error histories using fixed and optimal α to see the difference between each rate of convergence:



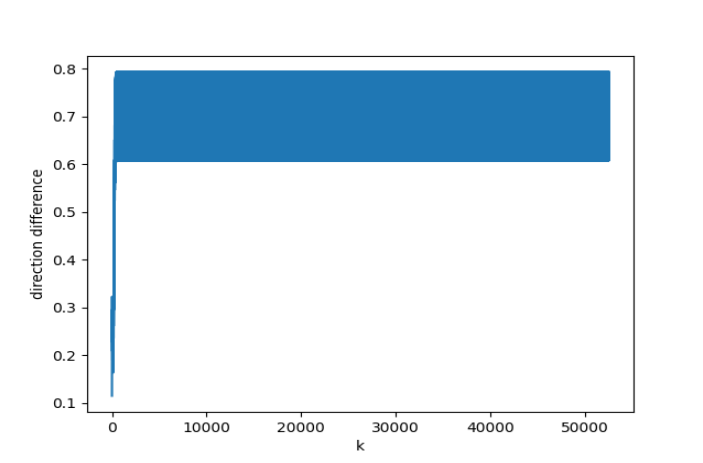
As shown from the plot above, we can see that taking optimal stepsize does accerlate the converge dramatically.

And more interestingly, if we try to plot the log of the errors, we get this:



From this graph, we can see that the convergence of both methods converges exponentially fast in a more intuitive way. And by choosing the optimal stepsizes, we can get a smaller u for the corresponding exponential function.

- How does this change the angle of approach as the iterates converge to the minimum? Similarly, let's look at the angle between $\underline{x}_{k+1} - \underline{x}_k$ and $x^* - x_k$:



What we can get from this is that initially, the angle between $\underline{x}_{k+1} - \underline{x}_k$ and $x^* - x_k$ will shrink to a certain level and then begin oscillating within a range of which the angle is small. It's interesting that instead of directly aiming to the minimizer, the optimization step keep having a small oscillating angle with the direction that points to the minimum. It's like we are walking downhill by taking a rugged path.

- Are these behaviors consistent?

  From the plot above and many other experiments, I believe these oscillating behaviors are consistent. I guess we can interpret this by associating the strategy of taking optimal stepsize with greedy strategy:

  We try to take the biggest descent step with the information of first and second derivative. However, we don't know the true direction between the current point to the minimizer, and we are ignoring the rest terms of the Taylor Expansion. So what happens here is that we overshoot because we are greedy and try to reach the maximum outcome with limited information. However, we can fix some of this overshoot in the next optimization step. And this is the explanation of the behavior of oscillating.

Momentum

In this section, we include a momentum term, given by $\underline{q}_k = \underline{x}_k - \underline{x}_{k-1}$.

- For a constant $\alpha > 0, \beta > 0$, plot the error $||\underline{x}_k$ - $\underline{x}^*||$ as a function of k to show convergence. How can you find β, α to guarantee convergence? Are these the best constants you can find?

  In order to find the best constants, we need to understand how α and β affects the error during the optimization:

  Firstly, we have:

  $$\underline{x}_{k+1} = \underline{x}_k - \alpha(Q\underline{x}_k - c) + \beta(\underline{x}_k - \underline{x}_{k-1})$$

  if we think of $\underline{x}_k = \underline{x}^* + \underline{e}_k$, and $x^* = Q^{-1}\underline{c}$, we can organize above as:

  $$\begin{aligned}\underline{e}_{k+1} &= \underline{e}_k - \alpha Q\underline{e}_k + \beta(\underline{e}_k - \underline{e}_{k-1}) \\ &= (I - \alpha Q + \beta I)\underline{e}_k - \beta\underline{e}_{k-1}\end{aligned}$$

  rather than thinking of this as one vector equation, think of it as two:

  $$\begin{aligned}\underline{e}_{k+1} &= (I - \alpha Q + \beta I)\underline{e}_k - \beta\underline{e}_{k-1} \\ \underline{e}_k &= 1\underline{e}_k + 0\underline{e}_{k-1}\end{aligned}$$

  And think about v_k as a vector:

  $$\underline{v}_k = \begin{bmatrix}\underline{e}_k \\ \underline{e}_{k-1}\end{bmatrix}$$

  Then the above can be described as:

  $$\underline{v}[k+1] = \begin{bmatrix}(I - \alpha Q + \beta I) & -\beta I \\ I & 0\end{bmatrix}\underline{v}_k$$

  where $\underline{v}_k$ is a vector with twice as many dimensions as $\underline{e}_k$ and the above is a matrix twice as big as Q in each direction.

  And we can let:

  $$A = \begin{bmatrix}(I - \alpha Q + \beta I) & -\beta I \\ I & 0\end{bmatrix}$$

  In this case if we can find a basis constructed by A's Eigenvectors, we can represent v_0 as:

  $$\underline{v}_0 = a_1\underline{g}_1 + a_2\underline{g}_2 + \cdots + a_d\underline{g}_d$$

  g_i represents the Eigenvectors of A. And by doing this, we can represent v_k as:

  $$\begin{aligned}\underline{v}_k &= A^{k-1}\underline{v}_0 \\ &= a_1\lambda_1^{k-1}\underline{g}_1 + a_2\lambda_2^{k-1}\underline{g}_2 + \cdots + a_d\lambda_d^{k-1}\underline{g}_d\end{aligned}$$

  So if we can find α and β that secure $\forall\lambda_i : -1 < \lambda_i < 1$, the convergence is promised.

So here I want to take a bold guess that the Eigenvectors of A are in term of:

$$\underline{g}_i = \begin{bmatrix} \underline{z}_i \\ \delta \underline{z}_i \end{bmatrix}$$

where δ is a unknown constant, and $\underline{z}_i$ represent the Eigenvector of Q. So we want:

$$\begin{aligned} A\underline{g}_i &= \begin{bmatrix} (I - \alpha Q + \beta I)\underline{z}_i - \delta\beta I \underline{z}_i \\ \underline{z}_i \end{bmatrix} \\ &= \begin{bmatrix} (1 - \alpha\lambda_i' + \beta - \delta\beta)\underline{z}_i \\ \underline{z}_i \end{bmatrix} \\ &= \lambda_i \begin{bmatrix} \underline{z}_i \\ \delta \underline{z}_i \end{bmatrix} \\ &= \lambda_i \underline{g}_i \end{aligned}$$

where the λ_i' represent the Eigenvalue of Q corresponding to $\underline{z}_i$, and in order to established the equation above, we need:

$$\begin{cases} 1 - \alpha\lambda_i' + \beta - \delta\beta = \lambda_i \\ 1 = \lambda_i \delta \end{cases}$$

so we have:

$$\beta\delta^2 - (1 - \alpha\lambda_i' + \beta)\delta + 1 = 0$$

And for δ to have two roots so that we can have a basis with $2D$ dimension, we need:

$$(1 - \alpha\lambda_i' + \beta)^2 - 4\beta > 0$$

Things get pretty messy if we want to continue our computation above, which is similar to what's in the Professors' note of "Convergence for Momentum Methods.pdf". And according to that, which limits the β between $(0, 1)$ and derives a range of α and β that promises the convergence, all we have is a range or a relationship between α and β. They are not that helpful to formulate the final forms of the optimal constant α and β.

However, if we take a step back and look at the expression of the eigenvalues of A:

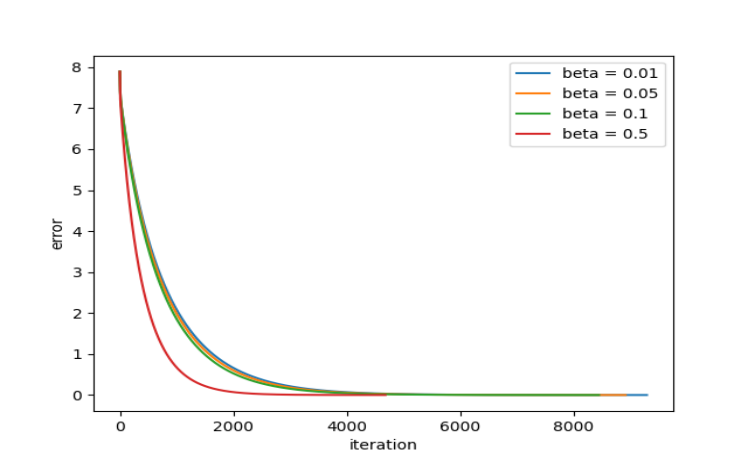
$$\lambda_i = 1 - \alpha\lambda_i' + \beta - \delta\beta$$

We can think of this as searching the constrant minimum of:

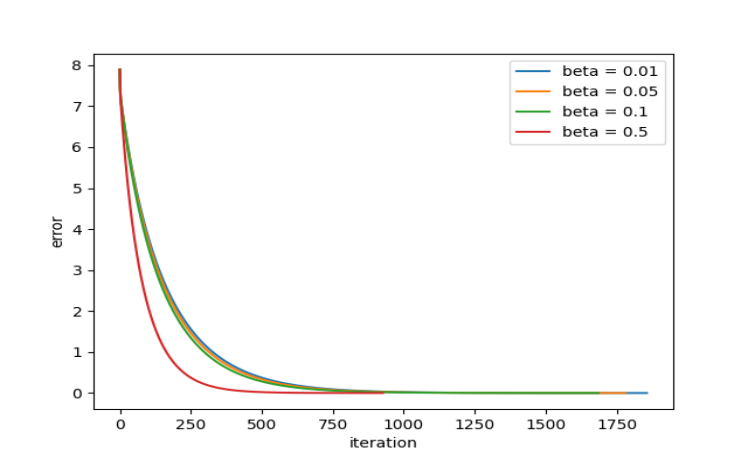
$$\begin{gathered} \min(|\max_i \lambda_i = 1 - \alpha\lambda_i' + \beta - \delta\beta|) \\ \text{s.t. } \forall \lambda_i \in (0, 1) \end{gathered}$$

So yes I think we can get the optimal constant α and β, but I don't think we should do that because it's like we are solving another much more complex problem to get the parameter for the original problem. It's more practical to just try different β during optimizing.

And the plots of the error when choosing different constant α and β are shown in the next page.



(a) fix $\alpha = 0.01$

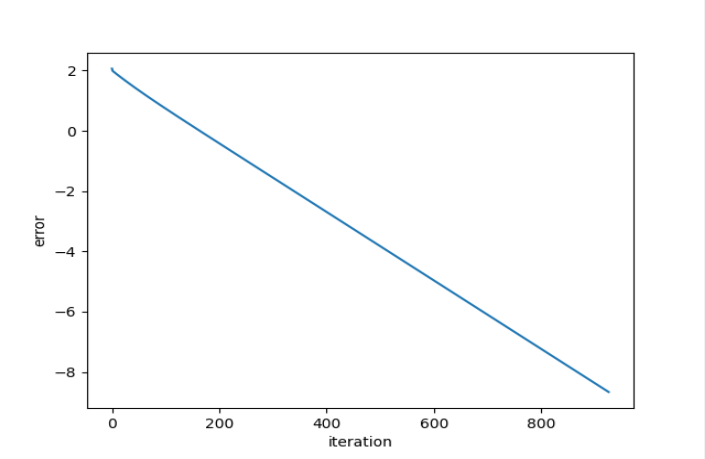


(a) fix $\alpha = 0.05$

So the best pair of α and β in my experiment is $\alpha = 0.05$ and $\beta = 0.5$. And of course we can find better α and β with the cost of much more complex calculation.

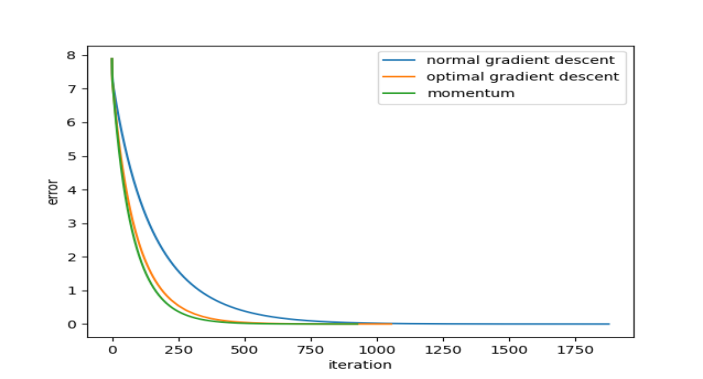
- For the best α, β you can find in the above question, what can you say about the rate of convergence, and how does it compare to gradient descent? Can you find α and β to make the convergence rate better than vanilla gradient descent? How does it compare to optimized gradient descent?

  If we take the log of the error and plot it, we get:

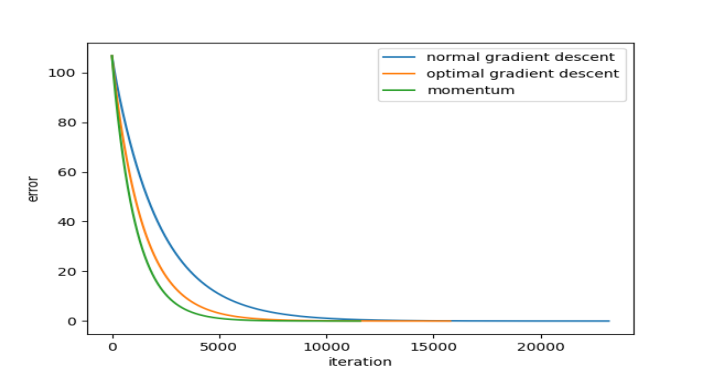


As shown as the plot, we can say that the momentum method follow the same exponential convergence as normal gradient descent.

And if we compare it with gradient descent we get:



(a)



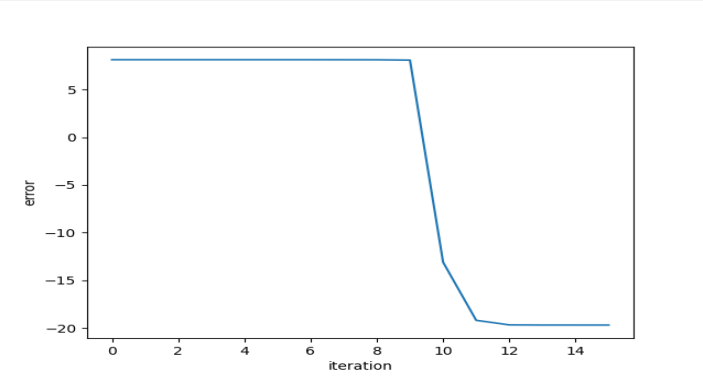
(b)

It surprised me so much that the momentum method has the fastest convergence. This is not intuitive because compared to momentum method trying to extract some second derivative information, optimal gradient descent do have the full access to the information to the second derivative.

And more interestingly, if we use the optimal α and β calculated every single step:

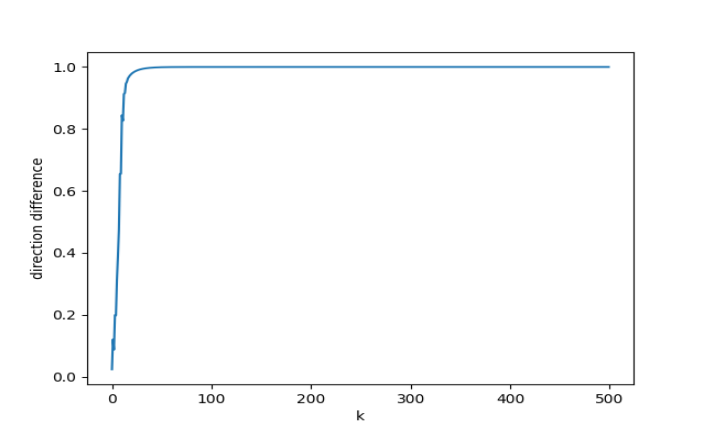
$$\tilde{\alpha}_k = \frac{\underline{p}_k^T \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{q}_k^T \underline{p}_k \underline{p}_k^T Q \underline{q}_k}{\underline{p}_k^T Q \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{p}_k^T Q \underline{q}_k \underline{p}_k^T Q \underline{q}_k}, \quad \tilde{\beta}_k = \frac{\underline{p}_k^T \underline{p}_k \underline{p}_k^T Q \underline{q}_k - \underline{q}_k^T \underline{p}_k \underline{p}_k^T Q \underline{p}_k}{\underline{p}_k^T Q \underline{p}_k \underline{q}_k^T Q \underline{q}_k - \underline{p}_k^T Q \underline{q}_k \underline{p}_k^T Q \underline{q}_k}$$

the error will converge to a very small value within several steps:



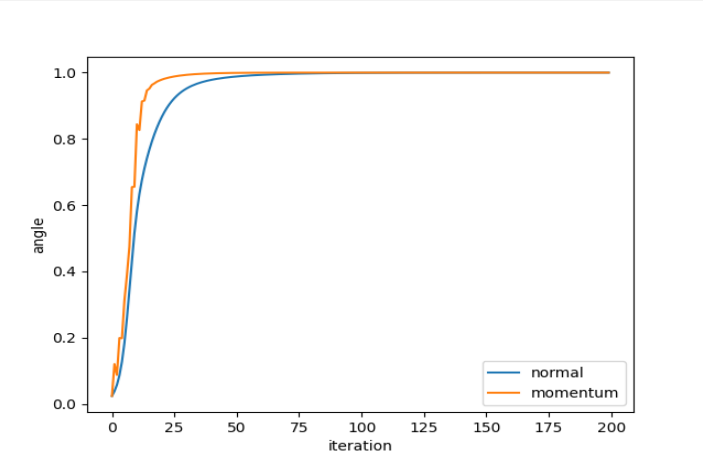
- Again, plot the angle of approach to the minimizer for these momentum iterates. What can you say about the approach to the minimizer, and how does it compare to the previous results?

  Firstly, let's plot the angle of approach to the minimizer for these momentum iterates:



  We can see that the momentum method behaves as the vanilla gradient descend: the angle between the optimization step and the direction of the minimizer decreases as the time of iteration increases.

  However, if we compare the vanilla momentum with the vanilla gradient descent:



  We can see that the angle between the optimization step and the direction of the minimizer shrink faster, which means the momentum do help improve the direction choosing during optimization.

- Do the trends you observe above generalize, with α, β, Q, c? How does vanilla momentum compare with vanilla gradient descent? With optimized gradient descent?

  Through a certain times of experiment and convergence analyse above, I think the behavior we observe generalize, with α, β, Q, c. And from above, we have a surprising discovery that momentum method stands out among all the optimization methods we use.

- Repeat the above, but for optimized momentum, using the optimal stepsizes α_k, β_k from before.

  I think I've already include this part above, please refer to the discussion above.

Is there a better direction?

We can consider the effect of the momentum term of adding a little bit of movement in a direction other than just the gradient (or rather, the negative of the gradient). This widens the space of what the iterates can explore, and in that way it makes sense that it may discover better routes to the optimum. But is this the best approach? Ideally, we'd like to move as directly towards the minimizer as possible.

- What would moving directly towards the minimizer as possible 'look like', in terms of the iterates? How does this compare to the behavior of gradient descent and momentum methods as above?

  I think if we can move directly towards the minimizer, the optimization speed will be boosted dramatically. And if we can do that, we can think of momentum methods as the same as the gradient descent but with a larger stepsize.

An alternative approach we might take is to choose a direction $\underline{q}_k$ that is orthogonal to $\underline{p}_k$, and choose the stepsize and momentum factors to optimize motion in this orthogonal direction.

- Given a vector $\underline{p}_k$, how can we generate an $\underline{q}_k$ that is orthogonal to $\underline{p}_k$?

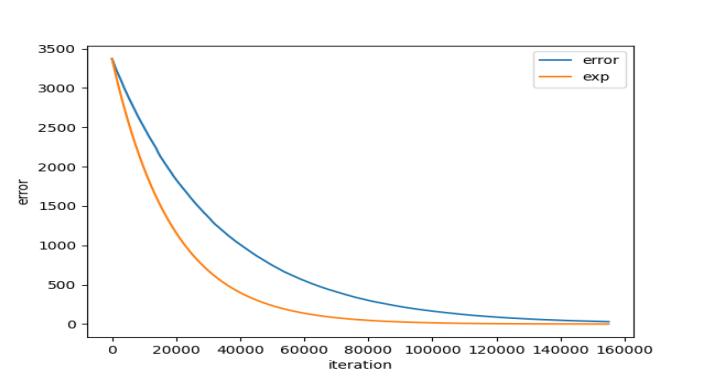
  All we know about orthogonal is that: $\underline{p}_k \cdot \underline{q}_k = 0$. So we can easily build a loop function to generate $\underline{q}_k$ easily:

  1: $sum \leftarrow 0$
  2: **for** i from 1 to $D-1$ **do**
  3: $\quad \underline{q}_k[i] \leftarrow Random()$
  4: $\quad sum \leftarrow sum + \underline{q}_k[i] * \underline{p}_k[i]$
  5: **end for**
  6: $\underline{q}_k[D] \leftarrow \frac{-sum}{\underline{p}_k[D]}$
  7: **Return** $\underline{q}_k$

Constructing orthogonal $\underline{q}_k$ as above, we can implement this modified momentum descent.

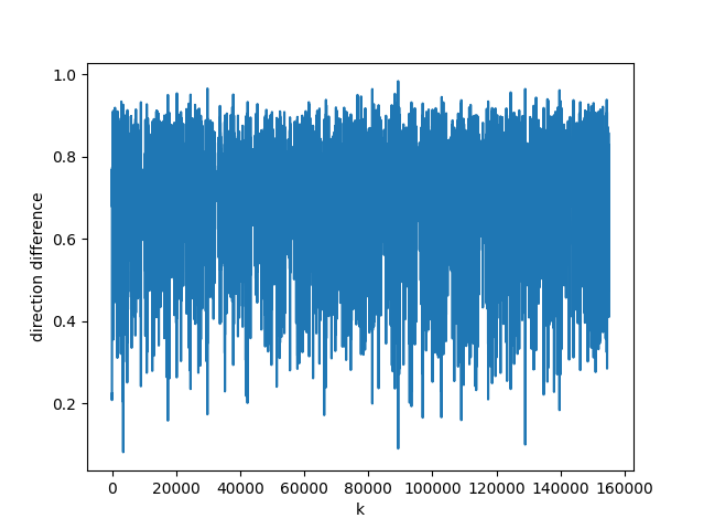
- Implementing this modified momentum descent with optimal α_k, β_k, how does this method of choosing $\underline{q}_k$ influence the convergence rate and the directions of approach? How does it compare with momentum methods generally?

  In order to learn about how choosing this orthogonal $\underline{q}_k$ influence the convergence, let's plot the error of it comparing to the other methods:



As shown by the error plot, the first intuitive observation is that the orthogonal descent is faster than the optimal gradient descent, which is quite reasonable because if we set $\beta = 0$ in orthogonal descent it will become optimal gradient descent. And then we can see that orthogonal descent can perform as well as momentum descent when we fix α and β. However, the astonishing efficiency of the momentum descent with optimal α_k and β_k is still untouchable. So yeah, maybe adding a orthogonal direction during the optimization may produce something useful, but we can conclude that the information catch by the original momentum direction is more useful.

Then let's have a look of how this orthogonal direction influence the angle between the optimization step and the minimizer:



We can see that there is a highly chaotic oscillation with the angle between the optimization step and the minimizer. And this oscillation is much more intense compare to the oscillation we observe in optimal vanilla gradient descent. I think this is reasonable to because we are adding a completely random orthogonal direction to the optimization step. The only thing we can control is the size of this random orthogonal direction. Without containing any information with the minimizer, it's reasonable that the angle between the optimization step and the minimizer will have a very intensive oscillation.

- Try to come up with a better way of generating an additional direction to move in. Note that you cannot use x^* or Q^{-1}, since if we knew either of these, none of this would be necessary.

Problem 2: Branch and Bound

Consider the following problem: find $x_1, x_2, ..., x_10 \in \mathbb{Z}$ to maximize

$$104x_1 + 128x_2 + 135x_3 + 139x_4 + 150x_5 + 153x_6 + 162x_7 + 168x_8 + 195x_9 + 198x_{10} \tag{7}$$

such that

$$9x_1^2 + 8x_2^2 + 7x_3^2 + 7x_4^2 + 6x_5^2 + 6x_6^2 + 5x_7^2 + 2x_8^2 + x_9^2 + x_{10}^2 \leq 68644 \tag{8}$$

Note that since we want to maximize over the integers, most of our usual approaches are not going to work - no derivatives, no continuity.

However, our usual approach can still provide useful information. Because the real numbers contain the integers, the solution to the above problem over the reals must necessarily be greater than or equal to the solution over the integers. This provides a convenient upper bound on possible solutions to the problem we're actually interested in.

- What is the solution to the above problem, when the x_i are taken to be real values instead of integers? Show your work.

  There is only one inequality constraint of the problem, so let's try to solve the question using the duality method. We have:

  $$F(\underline{x}) = -(104x_1 + 128x_2 + 135x_3 + 139x_4 + 150x_5 + 153x_6 + 162x_7 + 168x_8 + 195x_9 + 198x_{10})$$
  $$g(\underline{x}) = 9x_1^2 + 8x_2^2 + 7x_3^2 + 7x_4^2 + 6x_5^2 + 6x_6^2 + 5x_7^2 + 2x_8^2 + x_9^2 + x_{10}^2 - 68644$$

  Then we can construct $L(\underline{x}, \mu)$ with $(\mu > 0)$ that:

  $$\begin{aligned} L(\underline{x}, \mu) = &-(104x_1 + 128x_2 + 135x_3 + 139x_4 + 150x_5 + 153x_6 + 162x_7 + 168x_8 + 195x_9 \\ &+ 198x_{10}) + \mu(9x_1^2 + 8x_2^2 + 7x_3^2 + 7x_4^2 + 6x_5^2 + 6x_6^2 + 5x_7^2 + 2x_8^2 + x_9^2 + x_{10}^2 \\ &- 68644) \end{aligned}$$

  It's easy to see that the Hessian of $L(\underline{x}, \mu)$ is positive definite, which promise that $L(\underline{x}, \mu)$ has a global minimum when $\nabla_{\underline{x}} L(\underline{x}, \mu) = 0$. So, we can try to convert the problem of searching the minimum of $F(\underline{x})$ to searching the maximum of $L(\underline{x}, \mu)$ when $\nabla_{\underline{x}} L(\underline{x}, \mu) = 0$.

  Let $Q(\mu) = \min_{\underline{x}} L(\underline{x}, \mu)$, then we have:

  $$x_1 = \frac{104}{18\mu}, x_2 = \frac{128}{16\mu}, x_3 = \frac{135}{14\mu}, x_4 = \frac{139}{14\mu}, x_5 = \frac{150}{12\mu},$$
  $$x_6 = \frac{153}{12\mu}, x_7 = \frac{162}{10\mu}, x_8 = \frac{168}{4\mu}, x_9 = \frac{195}{2\mu}, x_{10} = \frac{198}{2\mu}$$

  $$\begin{aligned} \Rightarrow \quad Q(\mu) = &-(104 * \frac{104}{18\mu} + 128 * \frac{128}{16\mu} + 135 * \frac{135}{14\mu} + 139 * \frac{139}{14\mu} + 150 * \frac{150}{12\mu} + 153 * \frac{153}{12\mu} \\ &+ 162 * \frac{162}{10\mu} + 168 * \frac{168}{4\mu} + 195 * \frac{195}{2\mu} + 198 * \frac{198}{2\mu}) + \mu(9 * (\frac{104}{18\mu})^2 + 8 * (\frac{128}{16\mu})^2 \\ &+ 7 * (\frac{135}{14\mu})^2 + 7 * (\frac{139}{14\mu})^2 + 6 * (\frac{150}{12\mu})^2 + 6 * (\frac{153}{12\mu})^2 + 5 * (\frac{162}{10\mu})^2 + 2 * (\frac{168}{4\mu})^2 \\ &+ 1 * (\frac{195}{2\mu})^2 + 1 * (\frac{198}{2\mu})^2 - 68644) \end{aligned}$$

So we have:

$$Q(\mu) = -28213.698015873015 * \frac{1}{\mu} - 68644\mu$$

And for functions in terms of $f(x) = -(\frac{a}{x} + bx)$, we know they have a global maximum when $f'(x) = 0$ if $x > 0$. So, if we take $Q'(\mu) = 0$, we get:

$$\frac{28213.698015873015}{\mu^{*2}} - 68644 = 0$$
$$\Rightarrow \quad \mu^* = 0.6411043379183128$$

So we have:

$$\max_{\mu} Q(\mu) = Q(\mu^*) = -88015.93234412931$$

And we can verify whether there is duality gap by checking whether this result satisfies $g(\underline{x}^*) \leq 0$. By representing x^* with μ^*, we get:

$$\begin{aligned} g(\underline{x}^*) &= \mu^*(9 * (\frac{104}{18\mu^*})^2 + 8 * (\frac{128}{16\mu^*})^2 + 7 * (\frac{135}{14\mu^*})^2 + 7 * (\frac{139}{14\mu^*})^2 + 6 * (\frac{150}{12\mu^*})^2 + 6 * (\frac{153}{12\mu^*})^2 \\ &\quad + 5 * (\frac{162}{10\mu^*})^2 + 2 * (\frac{168}{4\mu^*})^2 + 1 * (\frac{195}{2\mu^*})^2 + 1 * (\frac{198}{2\mu^*})^2 - 68644) \\ &= -9.329295977925544 * e^{-12} \end{aligned}$$

So there is no duality gap. The maximum solution of the original problem is 88015.93234412931

Additionally, note that any choice of integer values for $x_1, x_2, \ldots, x_{10}$ necessarily provides a *lower bound* on the above problem - the maximum value must be better than the result of any arbitrary choice of integers we make. If choosing integer values is done in an efficient way, such as greedily, this can generate useful lower bounds on the problem we're actually interested in. Obviously, the larger the lower bound, the better.

- Generate a greedy variable assignment to try to get a good lower bound on the value of the maximum for this problem. The larger this bound, the better.

  To get a good lower bound, all we need to do is to assign x_i with the floor integer of their real solutions:

  $$x_i = \lfloor x_i^* \rfloor$$

Now to solve the original problem, we could potentially brute force it, looping over every possible feasible value for each variable. But the above results suggest a potentially better approach.

A Branch and Bound Algorithm attempts to solve this kind of optimization problem by traversing the tree of possible variable assignments, using the following observations:

- At any time, suppose we have a partial assignment of variables, and an upper and lower bound on the value the maximum attains.
- If we relax the remaining variables to be real valued, and solve for the maximum, if the result falls

below the lower bound *there is no way to complete the variable assignment with integers that will surpass the lower bound.*

- In this case, we know immediately that the partial assignment must be incorrect, and we can backtrack.
- If we complete a variable assignment, and the result gives a value above the lower bound, then this variable assignment provides a *better* lower bound for the full problem, and we can utilize it moving forward.

Applying these ideas recursively, we can test assignments for some variables, determine whether they are feasible, and either explore deeper, or backtrack and test other values for variable assignment. This lets us prune the space of possible variable assignments down, and efficiently identify a maximizing assignment.

- Giving a partial assignment of variables, the above problem will reduce to something of the form: maximize

$$\sum_{i=1}^{n} \alpha_i z_i \tag{9}$$

subject to

$$\sum_{i=1}^{n} \beta_i z_i^2 \leq R^2 \tag{10}$$

where $z_1, z_2, \ldots, z_n \in \mathbb{Z}$.
Consider the relaxation of this to $z_i \in R$, and solve for the constrained maximum in terms of α_i, β_i, R.

Firstly, let $\{x_1, x_2, \ldots, x_j\} \in X$ that x_i has already assigned with integers, and let $\{y_1, y_2, \ldots, y_k\} \in Y$ that y_i is the variable that we relax at this stage. Noted that $X \cap Y = \emptyset, X \cup Y = Z$. And if we let:

$$R' = R^2 - \sum_{i=1}^{j} \beta_i x_i^2$$

Then the original problem is converted to minimizing:

$$-\sum_{i=1}^{k} \alpha_i y_i$$

subject to

$$\sum_{i=1}^{k} \beta_i y_i^2 \leq R'$$

And if we try to solve this using duality method, we have:

$$L(\underline{y}, \mu) = -\sum_{i=1}^{k} \alpha_i y_i + \mu \sum_{i=1}^{k} \beta_i y_i^2 - \mu * R'$$

And if we let:

$$\underline{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_k \end{bmatrix}, \quad B = \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_k \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_k \end{bmatrix}$$

we have:

$$\begin{aligned} & L(\underline{y}, \mu) = -\underline{\alpha}^T \underline{y} + \mu \underline{y}^T B \underline{y} - \mu * R' \\ \Rightarrow \quad & \nabla_{\underline{y}} L(\underline{y}, \mu) = -\underline{\alpha} + 2\mu B \underline{y} \end{aligned}$$

Then if we set $\nabla_{\underline{y}} L(\underline{y}, \mu) = 0$, we have:

$$\underline{y}^* = \frac{B^{-1}\alpha}{2\mu}$$

then we let:

$$\begin{aligned} Q(\mu) &= \min_{\underline{y}} L(\underline{y}, \mu) \\ &= L(\underline{y}^*, \mu) \\ &= -\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{2\mu} + \frac{\underline{\alpha}^T B^{-1T} B B^{-1} \underline{\alpha}}{4\mu} - \mu * R' \\ &= -\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{2\mu} + \frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4\mu} - \mu * R' \\ &= -\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4\mu} - \mu * R' \end{aligned}$$

Note that here we assume all $\beta_k \neq 0$ so that B is invertible. Then we have:

$$Q'(\mu) = \frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4\mu^2} - R'$$

If we set $Q'(\mu) = 0 \quad (\mu \geq 0)$, we get:

$$\mu^* = \sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}}$$

Then if we plug this μ^* back to $Q(\mu)$, we get a probable solution S for the problem:

$$\begin{aligned} S &= -Q(\mu^*) \\ &= \frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4\sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}}} + \sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}} * R' \\ &= \sqrt{R' \underline{\alpha}^T B^{-1} \underline{\alpha}} \end{aligned}$$

The last thing we need to do is to evaluate whether this solution satisfies the constraint.

By replacing $\underline{y}$ with μ^*, we can represent the primal constraint as:

$$\sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}} * (\frac{B^{-1}\alpha}{2\sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}}})^T B (\frac{B^{-1}\alpha}{2\sqrt{\frac{\underline{\alpha}^T B^{-1} \underline{\alpha}}{4R'}}}) \leq R'$$

After simplifying the gross thing above, we can simplify this as evaluating whether:

$$\sqrt{\alpha^T B^{-1} \alpha} \leq 2\sqrt{R'}$$

- Describe a way to efficiently generate an assignment of values to the variables to give a lower bound on the value of this maximum. The larger the bound the better, but not at the cost of efficiency.

  From the last question, we get a very beautiful representation of the maximum value S while fixing X, and relaxing Y:

  $$S = \sqrt{R' \underline{\alpha}^T B^{-1} \underline{\alpha}}$$

  So assuming at this stage we are at the tree node of fixing x_i is some constant integer, what we can do here is to compare S to the lower bound O we obtained before and generate an assignment of x_i as:

  $$\begin{cases} O > S, & \text{Prun all the children of } x_i\text{, and back track to it's parent node to continue searching} \\ O < S, & \text{Sign } O = S \text{ and keep searching by go to the next level of the searching tree} \end{cases}$$

- Using the above solutions, implement a branch and bound algorithm to solve the original problem. What is the maximum value? How many complete variable assignments did you have to visit in order to discover the optimum?

  The implementation of the algorithm is integrated in the **"Branch and Bound.py"**. The maximum value is 88011, and the optimal $\underline{x}^* = [9, 13, 15, 16, 20, 20, 25, 65, 152, 154]^T$. In order to discover the optimum, I have to make 23007 complete variable assignments. The output of the algorithm is shown below:

```
============================================================
New Maximum: 88011
new X: [[  9  13  15  16  20  20  25  65 152 154]]
 Counts = 23007
============================================================
```