Università della Svizzera italiana | Institute of Computing CI

Optimization Methods **Spring 2024**

Student: Birindelli Leonardo **Discussed with:** Morales Jeferson

Midterm Project

Due date: Sunday, 12 May 2024, 11:59 PM

Exercise 1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given by $f(x) = \frac{1}{2}x^T Ax - b^T x$ with A symmetric positive definite. Let x_m be the minimizer of the function f. Le v be an eigenvector of A, and let λ be the associated eigenvalue. Suppose that we use Steepest Descent (SD) method to minimize f and the starting point for the SD algorithm is $x_0 = x_m + v$.

1. Prove that the gradient at x_0 is $\nabla f(x_0) = \lambda v$.

   To begin with the proof, it is required to remark the gradient formula of the quadratic form function which is the following:

$$\nabla f(x) = Ax - b$$

   Then, considering the starting point $x_0 = x_m + v$, it is possible to write the gradient at that position as :

$$\begin{aligned}
\nabla f(x_0) &= Ax_0 - b &(1)\\
&= A(x_m + v) - b &(2)\\
&= Ax_m + Av - b &(3)
\end{aligned}$$

   We know from the hypothesys that x_m is the minimizer of the function f, thus, it means that its gradient must be equal to zero according to the first necessary condition for a minimizer:

$$\begin{aligned}
\nabla f(x_m) &= 0\\
Ax_m - b &= 0\\
Ax_m &= b
\end{aligned}$$

   Substituting Ax_m inside the expression (3), we obtain that :

$$\nabla f(x_0) = b - Av - b = Av$$

In addiction, we know beforehand the definition of eigenvalues and eigenvectors which affirms that :

$$Av = \lambda v \tag{4}$$

Hence it is possible to write that :

$$\nabla f(x_0) = \lambda v$$

Consequently, we have proved that $\nabla f(x_0) = \lambda v$.

2. How many iterations does the SD method take to minimize the function f if we use the optimal step length? Show the computations behind your reasoning.

   First of all, we know that the Steepest Descent method uses the negative value of the gradient of function at the given point as the step direction to reach its minimizer. Therefore, that direction, starting from x_0, is equal to the following value as we proved previously:

$$p = -\nabla f(x_0) = -\lambda v$$

   Now, we can show that the subsequent iteration x_1 is computed as following:

$$\begin{aligned} x_1 &= x_0 - \alpha\lambda v & (5)\\ &= x_m + v - \alpha\lambda v \qquad \text{where } x_0 = x_m + v & (6)\end{aligned}$$

   Secondly, we want to know the number of iterations of the gradient method in the case in which it is used the optimal step lenght, thus, we can write that:

$$\alpha_{opt} = \frac{\nabla f(x_i)^T \nabla f(x_i)}{\nabla f(x_i)^T A \nabla f(x_i)}$$

   Hence, the optimal step size of the starting point x_0 is:

$$\begin{aligned} \alpha_{opt} &= \frac{\nabla f(x_0)^T \nabla f(x_0)}{\nabla f(x_0)^T A \nabla f(x_0)} & (7)\\ &= \frac{\cancel{\lambda^2} v^T v}{\cancel{\lambda^2} v^T A v} & (8)\\ &= \frac{v^T v}{v^T \lambda v} \qquad Av = \lambda v \ (4) & (9)\\ &= \frac{1}{\lambda}\frac{\cancel{v^T v}}{\cancel{v^T v}} & (10)\\ &= \frac{1}{\lambda} & (11)\end{aligned}$$

   Knowing that l'optimal step size is equal to $\frac{1}{\lambda}$ we can semplify the expression 6, thus we achieve that:

$$x_1 = x_m + v - \frac{1}{\cancel{\lambda}}\cancel{\lambda} v$$
$$x_1 = x_m + v - v$$
$$x_1 = x_m$$

Then, knowing that x_m is the function minimizer as initial hypothesis, its gradient is equal to zero according to the first necessary condition for minimizer point and given this information we can write the next equation:

$$\nabla f(x_m) = \nabla f(x_1) = 0$$

In conclusion, it is possible to notice that the Steepest Descent method converges in only 1 iteration.
In particular, the single iteration that brings the iterative method to converge, directly depends on the fact that we are using the optimal step lenght in the process, $\alpha_{opt} = \frac{1}{\lambda}$. This information lets the steepest method to find the best step size in order to get closer towards the minimizer of the function. On the other hand, if $\alpha \neq \frac{1}{\lambda}$, the method could have required at least 1 iteration concluding that:

$$x_1 \neq x_m \wedge \nabla f(x_m) \neq \nabla f(x_1) \Longrightarrow f(x_1) > f(x_m) \wedge \nabla f(x_1) > \nabla f(x_m)$$

Exercise 2

Given a starting point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, p_1, ..., p_{n-1}\}$, we generate the sequence $\{x_k\}$ by setting

$$x_{k+1} = x_k + \alpha_k p_k \tag{12}$$

where

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \tag{13}$$

and r_k is the residual, as defined in class.

Consider the following theorem:
Theorem 1 *For any $x_0 \in \mathbb{R}^n$, the sequence $\{x_k\}$ generated by the conjugate gradient direction algorithm converges to the solution x^* of the linear system $Ax = b$ in at most n steps.*
Prove the theorem and explain carefully every step of your reasoning.

Proof:
To begin with, it can be noticed from the hypothesis that the final solution x^* can be written directly in term of the iterative process (12) where the step size is computed according to the formula (13). Thus, x^* is equal to:

$$x^* = \underbrace{\underbrace{\underbrace{x_0 + \alpha_0 p_0}_{x_1} + \alpha_1 p_1}_{x_2} + \alpha_2 p_2}_{x_3} + \cdots + \alpha_{n-1} p_{n-1}; \tag{14}$$

Since, the directions $\{p_0, p_1, p_2, \ldots, p_{n-1}\}$ are linearly independent and orthogonal due to the conjugacy property between each other :

$$< p_i, Ap_j >= 0 \ \forall i \neq j \text{ where } A \text{ is s.p.d} \tag{15}$$

They must be a span of the whole n real space:

$$\mathbb{R}^n = \{p_0, p_1, p_2, \ldots, p_{n-1}\}$$

This affirmation allows to write a generic vector $x \in \mathbb{R}^n$ merely as a linear combination of these directions:

$$x = \sum_{i=0}^{n-1} \gamma_i p_i$$

Especially, this definition can also be estended to the vectors defined by the difference between the final solution x^* and the initial value x_0 given that also these vectors are part of the same real space $\mathbb{R}^n$:

$$\underbrace{e}_{\text{Initial error}} = x^* - x = \sum_{i=0}^{n-1} \sigma_i p_i$$

for some coefficients σ_i.

Now, I want to proof that the coefficients σ_i of the basis p_i match exactly the the optimal step sizes. To achieve this condition, I need to multiply the initial error with the quantity $p_k^T A$, then:

$$\begin{aligned}
p_k^T A e &= p_k^T A(x^* - x_0) && (16)\\
&= p_k^T A \sum_{i=0}^{n-1} \sigma_i p_i && (17)\\
&= \sum_{i=0}^{n-1} p_k^T A \sigma_i p_i && (18)\\
&= \sum_{i=0}^{n-1} \sigma_i p_k^T A p_i && (19)\\
&= \sigma_k p_k^T A p_k \quad \text{Conjugacy property (15)} && (20)
\end{aligned}$$

Hence, it is possible to write that :

$$p_k^T A(x^* - x_0) = \sigma_k p_k^T A p_k \implies \sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}$$

However, it is still required to prove that $A(x^* - x_0) = -r_k$ (where $-r_k = b - Ax_k$) in order to confirm that the coefficients σ_i correspond to the optimal step sizes.
The generic iteration x_k , if it is generated by the iterative procedure (12), then we have that :

$$x_k = \underbrace{\underbrace{\underbrace{x_0 + \alpha_0 p_0}_{x_1} + \alpha_1 p_1}_{x_2} + \alpha_2 p_2}_{x_3} + \cdots + \alpha_{k-1} p_{k-1};$$

Subsequently, we aim to pre-multiply the expression with $p_k^T A$ and using the conjugacy property we obtain that:

$$p_k^T A p_k = p_k^T A x_0$$

Then, substituing the previous expression to the formula σ_k we obtain that:

$$\begin{aligned}
\sigma_k &= \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} \\
&= \frac{p_k^T A(x^* - x_k)}{p_k^T A p_k} \quad (p_k^T A x_0 = p_k^T A p_k) \\
&= \frac{p_k^T A x^* - p_k^T A x_k}{p_k^T A p_k} \\
&= \frac{p_k^T b - p_k^T A x_k}{p_k^T A p_k} \quad (Ax^* = b) \\
&= \frac{p_k^T (\overbrace{b - Ax_k}^{-r_k})}{p_k^T A p_k} \\
&= -\frac{r_k p_k}{p_k^T A p_k} \\
&= \alpha_k
\end{aligned}$$

In conclusion, we prove that the coefficients $\sigma_k = \alpha_k$ and consequently we verified that the algorithm converges in at most n steps:

$$\begin{aligned}
x^* - x_0 = \sum_{i=0}^{n-1} \sigma_i p_i = \sum_{i=0}^{n-1} \alpha_i p_i \\
x^* = x_0 + \sum_{i=0}^{n-1} \alpha_i p_i
\end{aligned}$$

Exercise 3

Consider the linear system $Ax = b$, where the matrix A is a symmetric positive definitive diagonal matrix constructed in four different ways:

$A = \text{diag}([1 : 10]);$

$A = \text{diag}(ones(1, 10));$

$A = \text{diag}([1, 1, 1, 3, 4, 5, 5, 5, 10, 10]);$

$A = \text{diag}([1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]);$

1. How many distinct eigenvalues has each matrix?
   The first and the fourth matrix have exactly 10 unique eigenvalues, the second one has 1 unique eigenvalue and the third one has 5 distinct eigenvalues.
2. Implement the CG method (CG.m).

   The implementation of the CG method is avaiable in the project folder `code` and it is denominated as `CG.m`.
3. Construct a right-hand side $b = rand(10, 1)$ and apply the Conjugate Gradient method to solve the system for each A.
   The code implementation is available in the matlab file denominated as `code/Ex3.m` in the code section commented as "`3.2 & 3.3`".

4. Compute the logarithm energy norm of the error (i.e. $log((x - x^*)TA(x - x^*)))$ for each matrix and plot it with respect to the number of iteration.

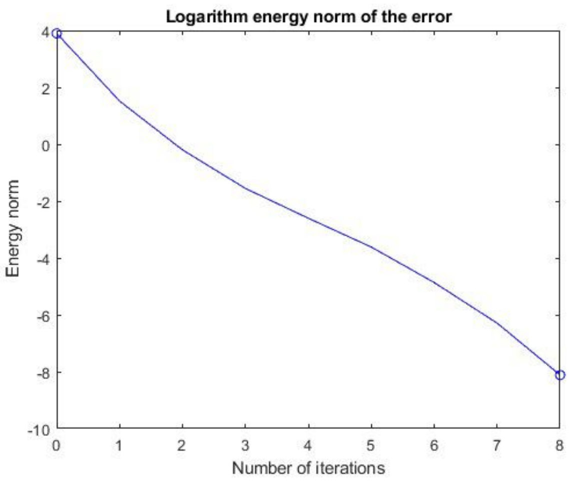

Figure 1: First matrix

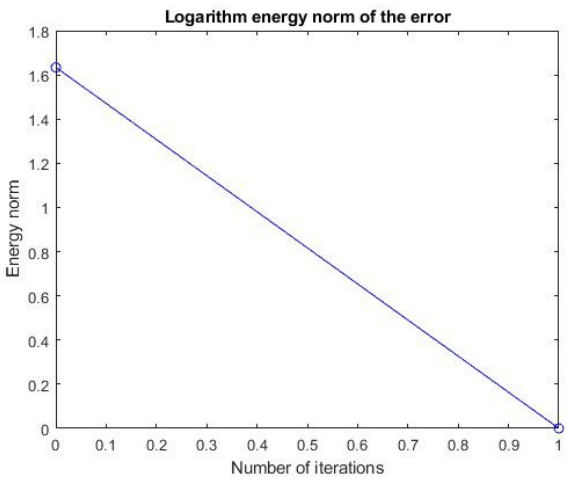

Figure 2: Second matrix

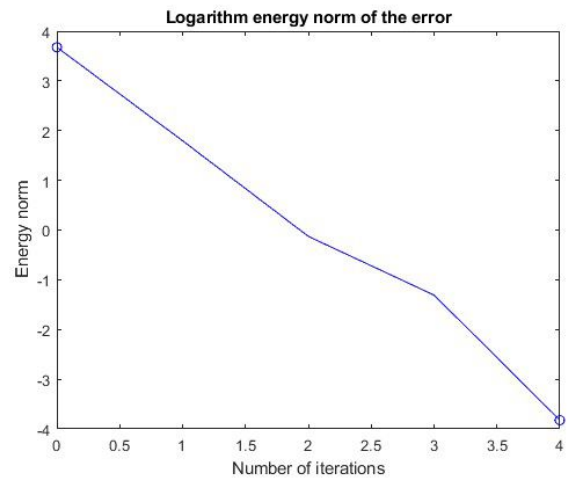

Figure 3: Third matrix

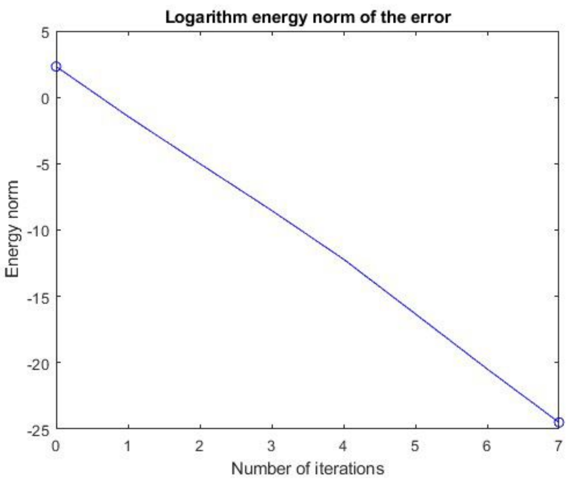

Figure 4: Fourth matrix

Note: the code implementation for generating the plots is available in the matlab file denominated as `code/Ex3.m` in the code section commented as "`3.4`".

5. Comment on the convergence of the method for the different matrices. What can you say observing the number of iterations obtained and the number of clusters of the eigenvalues of the related matrix?

   The A1 and A4 matrixes have equally distributed and distinct eigenvalues and in their domain and this kind of distribution afflicts negatively the performance of the CG method because with a uniform distribution of eigenvalues, the method needs to reduce the error uniformly along all directions, which consequently requires more iterations in the iterative process. On the other hand, the A3 matrix has a skewed distribution of eigenvalues and, moreover, some of them are also repeated in the diagonal of the matrix, those two characteristics allows the method to quickly and firsly reduce the error in the directions corresponding to the large eigenvalues, which can lead to faster convergence.

In conclusion, A2 matrix has only one eigenvalue because has all the elements on the diagonal equal to each other and this situation makes the method converges in only one iteration because the direction of the first residual will already be conjugate to all other directions, and thus the solution can be found immediately.

Exercise 4

Consider the Chapter 4, "Trust-region methods" of the book *Numerical Optimization*, Nocedal and Wright.

1. Explain Cauchy point method and Dogleg method, as well as the connection between them.

   The Cauchy point and Dogleg methods are both optimization techniques that belong to the group of trust region methods: this type of optimization algorithms is based on the idea of defining a region around the iteration of the process in which we are confident that constructing a local quadratic model of the objective function is a good approximation of that function. Subsequently, the method proceeds to compute the step direction and step size to move optimally toward the model minimizer within that region.
   The trust region problem aims to minimize the local model of the function at each iteration of the process and it can be matematically described as such:

   $$\min_{p\in\mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2}p^T B_k p \quad \text{s.t. } \|p\| \leq \Delta_k$$

   where $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, B_k can be the hessian $\nabla^2 f(x_k)$, its approximation or its modified version to guarantee the positive definite property of the matrix, and Δ_k is the radius of the trust region at k-th iteration.
   Taking as a starting reference the iterative procedure described as follows.:

   $$x_{k+1} = x_k + \tau p_k$$

   The Cauchy point method is characterized by the idea of computing the step direction along the direction of the gradient of the function:

   $$p_k^s = -\frac{g_k}{\|g_k\|}\Delta_k \text{ where } g_k = \nabla f(x_k)$$

   Once the step direction is obtained, then the step size is computed depending on the behavior of the model, so we have that :
   If $g_k^T B_k g_k \leq 0$, it means that the model $m_k(\tau p_k^s)$ decreases monotonically with respect to τ and, consequently, the only allowed value that also satisfied the trust region bound is $\tau = 1$. Otherwise, $g_k^T B_k g_k > 0$ the quadratic model is convex and therefore there is a point along the direction of the negative gradient for which the model reaches the minimizer. In fact an unconstrained minimizer of the model is computed and checked that it is not greater than the trust-region boundary. In mathematical terms, it can be written that:

   $$\tau_k = \begin{cases} 1 & g_k^T B_k g_k \leq 0 \\ \min(\frac{\|g_k\|^3}{\Delta_k g_k^T B_k g_k}, 1) & g_k^T B_k g_k > 0 \end{cases} \tag{21}$$

   Note: The quadratic model used to make the local approximation of the objective function in the Cauchy point method is simplified since that method uses only the gradient of the function and does not use the second grade derivative to calculate the step direction, so we have that:

   $$\min_{p\in\mathbb{R}^n} m_k(p) = f_k + g_k^T p \quad \text{s.t. } \|p\| \leq \Delta_k$$

To conclude with the Cauchy point method, its step p_k^c is described as :

$$p_k^c = \tau_k p_k^s$$

Speaking of the Dogleg method, it is another approach used to solve trust region problems. it is a special method because it aims to combine the Cauchy point method and the second grade derivative.
The idea behind the algorithm can be listed as follows :

a) Compute the unconstrained minimizer of the quadratic model and use it if the trust region is large enough.

$$p_B = -B_k^{-1} g_k \text{ where } B_k \text{ is s.p.d}$$

b) Calculate the direction of the steepest descent with the optimal step length for the model of objective function.

$$p_V = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k$$

c) If the Δ_k is too small compared to $\|p_V\|$, the Dogleg step direction is approximated to the Couch point direction (22), otherwise we compute the step direction according to the formula (23) where τ is choosen such that $\|p\|^2 = \Delta^2$:

$$p = \frac{g_k}{\|g_k\|}\Delta_k \tag{22}$$

$$p = \begin{cases} \tau p_V & 0 \le \tau \le 1 \\ p_V + (\tau - 1)(p_B - p_V) & 1 \le \tau \le 2 \end{cases} \tag{23}$$

As it is possible to notice, the Dogleg method allows searching for the minimum of the function by exploiting the information about the curvature of the function contained in the second grade derivative at the cost of being symmetric positive defined, otherwise the expression $p_B = B_k^{-1} g_k$ does not guarantee a descent direction (note that the matrix B_k can be either the exact hessian matrix or its approximation). For example, the second grade information are not used in the Cauchy point method to compute the step direction, but only to compute the step length.
Moreover, it is possible to note its close proximity with the Cauchy point method: in fact, when the radius of the trust region is really small, the curvature information contained in $p^T B_k p$ in the quadratic model become negligible and therefore it approximates the direction to the Cauchy step, while if the Δ_k is very large, larger values of τ will be accepted guaranteeing a greater reduction in the quadratic model. This last expression is in agreement with the lemma of sufficient decreasing in the trust region methods (”**Global convergence 11.5**” *page 33 of ”Notes on the optimization”*(2))where in which it is possible to remark the fact that the reduction of the model produced by the Dogleg method is at least or equal to the reduction of that in the Cauchy point :

$$\underbrace{m_k(0) - m_k(\tilde{p_k})}_{\text{decreasing of Dogleg model}} \ge \underbrace{m_k(0) - m_k(p_k^c)}_{\text{decreasing of Cauchy model}} \overset{(1)}{\ge} \underbrace{c_1 \, \|g_k\| \min(\Delta_k, \frac{\|g_k\|}{\|B_k\|})}_{\text{estimate of the reduction}}$$

Note: the inequity (1) in the previous formula is satisfied by the Cauchy point method if and only if the coefficient $c_1 = \frac{1}{2}$ according to the **Lemma 11.2** *page 34 of ”Notes on the optimization”*(2).

2. Write down the Trust-Region algorithm, along with Dogleg and Cauchy-point computations.

   The trust region algorithm computed with Dogleg method and the algorithm version that uses the Cauchy point method are available in the matlab file named "`code/Trust_Reg_Dogleg`" and "`code/Trust_Reg_Cauchy`" .

3. Consider the following lemma (Lemma 4.2, page 75 *Numerical Optimization*, Nocedal and Wright):

   Lemma 4.2:

   a) Let B be positive definite. Then:

      i. $\|\tilde{p}(\tau)\|$ is an increasing function of τ, and

      ii. $m(\tilde{p}(\tau))$ is a decreasing function of τ.

   Read carefully the proof and explain in detail how each step is obtained.

To begin with the proof of lemma, I would start proving the condition (i.) for the case $0 \leq \tau \leq 1$: Firstly, we can define $h(\tau) = \|\tilde{p}(\tau)\|$ and then computing:

$$\begin{aligned} h(\tau) &= \|\tilde{p}(\tau)\| \\ &= \|\tau p_V\| \\ &= |\tau| \cdot \|p_V\| && \text{Norm homogenity property} \end{aligned}$$

Subsequently, we aim to compute the derivative of the previous function in terms of τ such that :

$$\frac{\partial h}{\partial \tau} = \|p_V\| \tag{24}$$

Since the non-negativity property of norms of vectors, the norm of p_V is always positive : $\|p_V\| \geq 0$. Consequently, the derivative is positive and it is possible to affirm that the function is monotonically increasing.

Now, I want to prove the condition (i.) for the case $1 \leq \tau \leq 2$:
Thus, we can define $h(\tau) = \frac{1}{2}\|\tilde{p}(\tau)\|^2$ in order to avoid computing the squared root and simplify the matematical expression, given these premises we calculate that:

$$\begin{aligned} h(\tau) &= \frac{1}{2}\|\tilde{p}(\tau)\|^2 \\ &= \frac{1}{2}\|p_V + (\tau - 1)(p_B - p_V)\|^2 \end{aligned}$$

Then, we substitute $\alpha = \tau - 1$ in the $\tilde{p}(\tau)$ in order to work in term of that variable, obtaining that:

$$\tilde{p}(\alpha) = p_V + \alpha(p_B - p_V)$$

Therefore, we can calcolate $h(\alpha)$:

$$\begin{aligned} h(\alpha) &= \frac{1}{2}\|p_V + \alpha(p_B - p_V)\|^2 \\ &= \frac{1}{2}[p_V + \alpha(p_B - p_V)]^T \cdot [p_V + \alpha(p_B - p_V)] \\ &= \frac{1}{2}\|p_V\|^2 + \alpha p_V^T(p_B - p_V) + \frac{1}{2}\alpha^2\|p_B - p_V\|^2 \end{aligned}$$

Afterwards, we aim to calculate the derivative of $h(\alpha)$ such that :

$$\begin{aligned}\frac{\partial h}{\partial \alpha} &= (p_V)^T (p_B - p_V) + \underbrace{\alpha \|p_B - p_V\|^2}_{\geq 0} \\ &\geq (p_V)^T (p_B - p_V)\end{aligned}$$

We need to prove that $p_V^T(p_B - p_V) \geq 0$ where $p_V = \frac{g^T g}{g^T B g} g$ and $p_B = -B^{-1}g$ in which $g = \nabla f(x)$ and B is a symmetric positive matrix, it follows that :

$$\begin{aligned}(p_V)^T (p_B - p_V) &= -(p_V)^T (p_V - p_B) \\ &= \frac{g^T g}{g^T B g} g^T \cdot \left[\left(-\frac{g^T g}{g^T B g} g\right) - \left(-B^{-1} g\right)\right] \\ &= \frac{g^T g}{g^T B g} g^T \cdot \left(B^{-1} g - \frac{g^T g}{g^T B g} g\right) \\ &= \frac{g^T g}{g^T B g} g^T B^{-1} g \cdot \left(1 - \frac{g^T g}{(g^T B g)(B^{-1} g)} g\right) \\ &= \underbrace{\frac{g^T g}{g^T B g} g^T B^{-1} g}_{\geq 0} \cdot \left(1 - \frac{(g^T g)^2}{(g^T B g)\,(g^T B^{-1} g)}\right)\end{aligned}$$

We also need to prove that $1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g) \geq 0}$ in order to conclude the proof and, for doing so, we use the Cauchy-Schwarz inequity in which we have $u = B^{-\frac{1}{2}} g, v = B^{\frac{1}{2}} g$:

$$\begin{aligned}&\left|u^T v\right|^2 \leq \|u\|^2 \cdot \|v\|^2 \\ &\left|g^T B^{-\frac{1}{2}} B^{\frac{1}{2}} g\right|^2 \leq \left\|B^{-\frac{1}{2}} g\right\|^2 \cdot \left\|B^{\frac{1}{2}} g\right\|^2 \\ &\left|g^T g\right|^2 \leq \left(g^T \overbrace{B^{-\frac{1}{2}} B^{-\frac{1}{2}}}^{B^{-1}} g\right) \left(g^T \overbrace{B^{\frac{1}{2}} B^{\frac{1}{2}}}^{B} g\right) \\ &(g^T g)^2 \leq \left(g^T B^{-1} g\right) \left(g^T B g\right) \\ &\frac{(g^T g)^2}{(g^T B g)\,(g^T B^{-1} g)} \leq 1 \\ &1 - \frac{(g^T g)^2}{(g^T B g)\,(g^T B^{-1} g)} \geq 0\end{aligned}$$

This final proof allows to finally show that $h(\tau)$ is also monotonically increasing for the case $1 \leq \tau \leq 2$.

Once I have compleately proved the condition (i.) for the cases $0 \leq \tau \leq 1$ and $1 \leq \tau \leq 2$, I start to prove the condition (ii.) for the case $0 \leq \tau \leq 1$:
Firstly, I aim to substitute the model $m(\tilde{p}(\tau))$ in term of the value assumed by $\tilde{p}(\tau)$ in the interval:

$$\begin{aligned} m(\tilde{p}(\tau)) &= f + g^T\tilde{p}(\tau) + \frac{1}{2}\tilde{p}(\tau)^T B\tilde{p}(\tau) \\ &= f + \tau g^T p_V + \frac{1}{2}\tau^2 p_V^T B p_V \end{aligned}$$

Then, I compute the derivative of the model in term of τ :

$$\begin{aligned} \frac{\partial m}{\partial \tau} &= g^T p_V + \tau (p_V)^T B p_V \\ &= g^T \cdot \left(-\frac{g^T g}{g^T B g} g\right) + \tau \left(-\frac{g^T g}{g^T B g} g\right)^T \cdot B \cdot \left(-\frac{g^T g}{g^T B g} g\right) \qquad \text{where } p_V = \frac{g^T g}{g^T B g} g \\ &= g^T \cdot \left(-\frac{g^T g}{g^T B g} g\right) + \tau \left(-\frac{g^T g}{g^T B g} g\right)^T \cdot B \cdot \left(-\frac{g^T g}{g^T B g} g\right) \\ &= -\frac{(g^T g)^2}{g^T B g} + \tau \frac{(g^T g)^2}{(g^T B g)^2} g^T B g \\ &= -\frac{(g^T g)^2}{g^T B g} + \tau \frac{(g^T g)^2}{g^T B g} \\ &= \underbrace{(\tau - 1)}_{\leq 0} \underbrace{\frac{(g^T g)^2}{g^T B g}}_{\geq 0} \qquad \text{where } B \text{ is s.p.d} \\ &\leq \frac{(g^T g)^2}{g^T B g} \\ &\leq 0 \end{aligned}$$

In conclusion, we proved that $m(\tilde{p}(\tau))$ is a decreasing function of τ.
The last proof I want to do it is related to prove the condition (ii.) for the case $1 \leq \tau \leq 2$:
Firstly, I want to write the model $m(\tilde{p}(\tau))$ in term of the value assumed by $\tilde{p}(\tau)$ in the specified interval:

$$\begin{aligned} m(\tilde{p}(\tau)) &= f + g^T\tilde{p}(\tau) + \frac{1}{2}\tilde{p}(\tau)^T B\tilde{p}(\tau) \\ &= f + g^T(p_V + (\tau-1)(p_B - p_V)) + \frac{1}{2}[p_V + (\tau-1)(p_B - p_V)]^T \cdot B \cdot [p_V + (\tau-1)(p_B - p_V)] \end{aligned}$$

Then, we substitute $\alpha = \tau - 1$ in the $\tilde{p}(\tau)$ in order to work in term of the new variable, achieving that:

$$m(\tilde{p}(\alpha)) = f + g^T(p_V + \alpha(p_B - p_V)) + \frac{1}{2}[p_V + \alpha(p_B - p_V)]^T B [p_V + \alpha(p_B - p_V)]$$

After this change, we can calculate $m(\tilde{p}(\alpha))$ such that :

$$\begin{aligned}
m(\tilde{p}(\alpha)) &= f + g^T\left[p_V + \alpha\left(p_B - p_V\right)\right] + \frac{1}{2}\left[p_V + \alpha\left(p_B - p_V\right)\right]^T B\left[p_V + \alpha\left(p_B - p_V\right)\right] \\
&= f + g^T p_V + \alpha g^T\left(p_B - p_V\right) + \frac{1}{2}\left(p_V + \alpha p_B - \alpha p_V\right) B\left(p_V + \alpha p_B - \alpha p_V\right) \\
&= f + g^T p_V + \alpha g^T\left(p_B - p_V\right) + \frac{1}{2}[(p_V)^T B p_V + \alpha (p_V)^T B p_B - \alpha (p_V)^T B p_V + \alpha (p_B)^T B p_V \\
&+ \alpha^2 (p_B)^T B p_B - \alpha^2 (p_B)^T B p_V - \alpha (p_V)^T B p_V - \alpha^2 (p_V)^T p_B + \alpha^2 (p_V)^T B p_V] \\
&= f + g^T p_V + \alpha g^T\left(p_B - p_V\right) + \frac{1}{2}(\alpha^2\left[(p_B)^T B p_B - (p_B)^T B p_V - (p_V)^T B p_B + (p_V)^T B p_V\right] \\
&+ \alpha\left[(p_V)^T B p_B - (p_V)^T B p_V + (p_B)^T B p_V - (p_V)^T B p_V\right] + (p_V)^T B p_V) \\
&= f + g^T p_V + \alpha g^T\left(p_B - p_V\right) + \frac{\alpha^2}{2}\left[(p_B)^T B p_B + (p_V)^T B p_V - 2(p_B)^T B p_V\right] \\
&+ \alpha\left[(p_V)^T B p_B - (p_V)^T B p_V\right] + (p_V)^T B p_V
\end{aligned}$$

Afterwards, I aim to calculate the derivative of $m(\tilde{p}(\alpha))$ so as to conclude that :

$$\begin{aligned}
\frac{\partial m}{\partial \alpha} &= g^T\left(p_B - p_V\right) + \alpha\left[(p_B)^T B p_B + (p_V)^T B p_V - 2(p_B)^T B p_V\right] + \left[(p_V)^T B p_B - (p_V)^T B p_V\right] \\
&= g^T p_B - g^T p_V + \left[(p_B)^T B p_V - (p_V)^T B p_V\right] + \alpha\left[(p_B)^T B p_B - (p_V)^T B p_B + (p_V)^T B p_V - (p_B)^T B p_V\right] \\
&= (p_B)^T g - (p_V)^T g + (p_B)^T B p_V - (p_V)^T B p_V + \alpha\left[(p_B)^T B p_B - (p_V)^T B p_B + (p_V)^T B p_V - (p_B)^T B p_V\right] \\
&= \left(p_B - p_V\right)^T\left(g + B p_V\right) + \alpha\left(p_B - p_V\right) B\left(p_B - p_V\right) \\
&\leq \left(p_B - p_V\right)^T\left(g + B p_V + B\left(p_B - p_V\right)\right) \\
&= \left(p_B - p_V\right)^T\left(g + B p_B\right) \\
&= \left(p_B - p_V\right)^T\left[g + B\left(-B^{-1} g\right)\right] \qquad \text{where } p_B = -B^{-1} g \\
&= \left(p_B - p_V\right)^T \cdot (g - g) \\
&= 0
\end{aligned}$$

In conclusion, obtaining that the derivative of the model $m(\tilde{p}(\alpha))$ is equal to zero, I can finish affirming that $m(\tilde{p}(\tau))$ is a decreasing function of τ.

References

1. Jorge Nocedal and Stephen J Wright. Numerical optimization. Springer, 1999 avaiable on iCorsi platform

2. "Notes on the optimization" avaiable on iCorsi platform