Unit 8. Descent Methods

Motivation:

We know that to ~~solve~~ find the minimum or maximum of a function $f(x)$, we need to solve $f'(x)=0$. For example, $f(x)=\frac{1}{2}\alpha x^2-\beta x$. Solving $f'(x)=\alpha x-\beta=0$ $\Rightarrow \alpha x=\beta$ will give us the minimum.

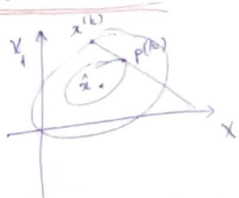
Now for the case with matrices, if we have a function $f(x)=\frac{1}{2}x^TAx-x^Tb$. Solving $\nabla f(x)=Ax-b=0$ will give us the minimum.

Theorem 8.1.1.1

Let A be SPD and assume that $A\hat{x}=b$. Then vector $\hat{x}$ minimizes the function $f(x)=\frac{1}{2}x^TAx-x^Tb$ ~~or $=\frac{1}{2}(x-\hat{x})^TA(x-\hat{x})-\frac{1}{2}\hat{x}^TA\hat{x}$~~ $(A\hat{x}=b)$

Basics of descent methods

Visualize:



$x=x^{(k)}+\alpha_k p^{(k)}$

- Choose direction $p^{(k)}$
- Pick α_k
- Repeat until $x-\hat{x}=0$

Algorithm: Given $A, b, x^{(0)}$. We try to solve $\nabla f(x)=Ax-b=0$

$r^{(0)} := b-Ax^{(0)}$ <residual>

$k:=0$

while $r^{(k)}\neq 0$

  $p^{(k)}:=$ next direction <Choose direction>

  $x^{(k+1)}:=x^{(k)}+\alpha_k p^{(k)}$ <minimize ~~$\nabla f(x)$~~ $f(x)$ by choose appropriate α_k, and step toward that direction>

  $r^{(k+1)}:=b-Ax^{(k+1)}$ <update residual>

  $k:=k+1$

endwhile

Expanding $f(x^{(k+1)})=f(x^{(k)}+\alpha_k p^{(k)})$

$$=\frac{1}{2}p^{(k)T}Ap^{(k)}\alpha_k^2-p^{(k)T}r^{(k)}\alpha_k+f(x^{(k)})$$

Minimizing $f(x^{(k+1)})$ requires solving $\frac{d f(x^{(k)}+\alpha_k p^{(k)})}{d\alpha_k}=0$

$$\Rightarrow \alpha_k=\frac{p^{(k)T}r^{(k)}}{p^{(k)T}Ap^{(k)}} \quad \text{and} \quad x^{(k+1)}=x^{(k)}+\alpha_k p^{(k)}$$

So the now its just the problem of picking the appropriate $p^{(k)}$

Detailed Algorithm:

Given: $A, b, x^{(0)}$

$r^{(0)} := b-Ax^{(0)}$

$k:=0$

while $r^{(k)}\neq 0$

  $p^{(k)}:=$ next iteration

  $\alpha_k^{(k)}:=\frac{p^{(k)T}r^{(k)}}{p^{(k)T}Ap^{(k)}}$

  $x^{(k+1)}:=x^{(k)}+\alpha_k p^{(k)}$

  $r^{(k+1)}:=b-Ax^{(k+1)}$

  $k:=k+1$

endwhile

Hw 8.2.1.1

The algorithm consists of 2 dot products, 2 matrix-vector multiplications and 1 axpy operation (exclude $p^{(k)}:=$ next iteration)

Toward practical descent methods

Major cost of solving $Ax=b$ via descent methods is in the matrix-vector multiplication, so reduce the cost here is essential (how to reduce?)

Hw 8.2.2.1

We can prove that $r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$

So we can reuse $A p^{(k)}$ to reduce the cost. ~~We come up~~ From this insight, a more cost-efficient variant of algorithm is:

Practical Algorithm:

Given $A, b, x^{(0)}$
$r^{(0)} := b - Ax^{(0)}$
$k := 0$
while $r^{(k)} \neq 0$
  $p^{(k)} :=$ next iteration
  $q^{(k)} := Ap^{(k)}$
  $\alpha_k := \dfrac{p^{(k)T} r^{(k)}}{p^{(k)T} q^{(k)}}$
  $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
  $r^{(k+1)} := r^{(k)} - \alpha_k q^{(k)}$
  $k := k+1$
endwhile

Hw 8.2.2.2

The practical algorithm costs: 1 matrix-vector multiplication, 2 dot products, 2 axpy operations

Practical algorithm (without storing history)

Given A, b, x
$r := b - Ax$
while $r \neq 0$
  $p :=$ next iteration
  $q := Ap$
  $\alpha := \dfrac{p^T r}{p^T q}$
  $x := x + \alpha p$
  $r := r - \alpha q$
endwhile

Note that sometimes we do want to store the trajectory of solving $Ax = b$ via descent method.

5

How to pick search direction? Relation to splitting methods

Hw 8.2.3.1 Picking standard basis vectors for search direction

If we pick $p^{(0)} = e_0$. Then:

$$x_0^{(1)} = x_0^{(0)} + \frac{1}{\alpha_{0,0}} \left(\beta_0 - \sum_{j=0}^{n-1} \alpha_{0,j} x_j^{(0)} \right)$$

$$= \frac{1}{\alpha_{0,0}} \left(\beta_0 - \sum_{j=1}^{n-1} \alpha_{0,j} x_j^{(0)} \right)$$

$\Rightarrow$ This looks like Gauss - Seidel

Now that is just e_0 for $p^{(0)}$, if we pick e_k for $p^{(k)}$, we essentially updating $x_0^{(k+1)} \to x_{n-1}^{(k+1)}$, which is vector $x^{(k+1)}$

So n iteration of picking e_0 to e_{n-1} for $p^{(0)}$ to $p^{(n-1)}$ equals 1 iteration of Gauss - Seidel

Method of ~~steepest~~ steepest descent

Given function $f(x)$, its steepest decline is $-\nabla f(x)$.
With $f(x) = \frac{1}{2} x^T A x - x^T b \Rightarrow -\nabla f(x) = -(Ax - b)$
$= b - Ax$
$\hookrightarrow$ which is the residual!

Therefore, our algorithm now become:

Given: $A, b, x^{(0)}$
$r^{(0)} := b - Ax^{(0)}$
$k := 0$
while $r^{(k)} \neq 0$:
  $p^{(k)} := r^{(k)} \leftarrow$ changes
  $q^{(k)} := Ap^{(k)}$
  $\alpha_k := \ldots$
  $x^{(k+1)} := \ldots$
  $r^{(k+1)} := r^{(k)} - \alpha_k q^{(k)}$
  $k := k+1$
endwhile

4

Preconditioning:

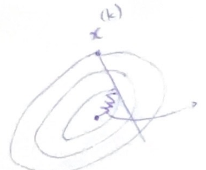
Sometime using steepest descent is to solve $Ax = b$ is not enough since A is often ill-conditioned. We can instead solve

$$\boxed{M^{-1}Ax = M^{-1}b}$$, where $M \approx A$ so that ~~MM⁻¹~~ $M^{-1}M \approx M^{-1}\cancel{A} \approx I$

$\boxed{M^{-1} \text{ is called precondition}}$

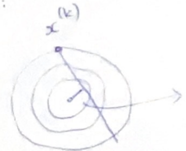
Visualization:

ill-conditioned A:



it goes zic-zac steps to the solution

well-condioned A:



less steps, straight to solution

But:

It is not guaratee that $M^{-1}A$ is SPD, so we instead do

$$\boxed{\underbrace{L_M^{-1} A L_M^{-T}}_{\tilde{A} \text{ (which } \approx I)} \underbrace{L_M^T x}_{\tilde{x}} = \underbrace{L_M^{-1} b}_{\tilde{b}} \quad \text{where} \begin{cases} M = L_M L_M^T \\ A = LL^T \\ M \approx A \text{, both are SPD} \end{cases}}$$

Now the algorithm becomes:

```
Given: A, b, x^(0), M
r^(0) := b - Ax^(0)
k := 0
while r^(k) ≠ 0
    p^(k) := M^-1 r^(k)
    q^(k) := Ap^(k)
    α_k := ...
    x^(k+1) := ...
    r^(k+1) := ...
    k := k+1
endwhile
```


Detailed algorithm:

$$\begin{array}{l}
\text{Given } A, b, x^{(0)}, M = LL^T, \\
\tilde{A} = L^{-1} A L^{-T}, \\
\tilde{b} = L^{-1} b, \\
\tilde{x}^{(0)} = L^T x^{(0)}, \\
\tilde{r}^{(0)} := \tilde{b} - \tilde{A} x^{(0)} \\
k := 0 \\
\text{while } \tilde{r}^{(k)} \neq 0 \\
\quad \tilde{p}^{(k)} := \tilde{r}^{(k)} \\
\quad \tilde{q}^{(k)} := \tilde{A}\tilde{p}^{(k)} \\
\quad \tilde{\alpha}_k := \dfrac{\tilde{p}^{(k)T}\tilde{r}^{(k)}}{\tilde{p}^{(k)T}\tilde{q}^{(k)}} \\
\quad \tilde{x}^{(k+1)} := \tilde{x}^{(k)} + \tilde{\alpha}_k \tilde{p}^{(k)} \\
\quad \tilde{r}^{(k+1)} := \tilde{r}^{(k)} - \tilde{\alpha}_k \tilde{q}^{(k)} \\
\quad x^{(k+1)} := L^{-T}\tilde{x}^{(k+1)} \\
\quad k := k+1 \\
\text{endwhile}
\end{array}$$

Hw 8.2.5.1

Prove that the detailed algorithm is compute the same result as the algorithm in page 5

THE CONJUGATE GRADIENT METHOD

A-conjugate directions

We know so far that the current direction/result can be expressed by previous directions/results, i.e.:

$$x^{(k+1)} = \alpha_0 p^{(0)} + \alpha_1 p^{(1)} + \dots + \alpha_k p^{(k)}$$

Since $x^{(k+1)}$ is linear combination of $p^{(0)} \dots p^{(k)}$; $x^{(k+1)} \in \text{Span}(p^{(0)} \dots p^{(k)})$

Now, if $p^{(0)} \dots p^{(k)}$ are linearly independent, then the result is guaranteed to complete in at most n iterations. Because:

$\text{Span}(p^{(0)}, \dots, p^{(n-1)}) = \mathbb{R}^n$

so that:

$$\boxed{f(x^{(n)}) = \min_{x \in \text{span}(p^{(0)}, \dots, p^{(n-1)})} f(x) = \min_{x \in \mathbb{R}^n} f(x)}$$

Hence $Ax^{(n)} = b$

Hw 8.3.1.1

Given $P^{(k)} = (p^{(0)} | \dots | p^{(k)})$, $y = \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_k \end{pmatrix}$

or $P^{(k)} = (P^{(k-1)} | p^{(k)})$, $y = \begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix}$

Then: $\min_{x \in \text{Span}(p^{(0)}, \dots, p^{(k)})} f(x) = \min_y f(P^{(k)} y)$

$$= \min_y \Big[\tfrac{1}{2} y_0^T P^{(k-1)T} A P^{(k-1)} y_0 - y_0^T P^{(k-1)T} b + \psi_1 y_0^T P^{(k-1)T} A p^{(k)} + \tfrac{1}{2} \psi_1^2 p^{(k)T} A p^{(k)} - \psi_1 p^{(k)T} b \Big]$$

this would disappear if $P^{(k-1)T} A p^{(k)} = 0$ (picking $p^{(k)}$ that is orthogonal to the previous directions)

So, if $P^{(k-1)T} A p^{(k)} = 0$, then:

$$\min_{x \in \text{Span}(p^{(0)}, \dots p^{(k-1)}, p^{(k)})} f(x) = \min_{x \in \text{Span}(p^{(0)}, \dots, p^{(k-1)})} f(x) + \min_{\psi_1} \Big[\tfrac{1}{2} \psi_1^2 p^{(k)T} A p^{(k)} - \psi_1 p^{(k)T} b \Big]$$

Minimizing ψ_1 is given by $\psi_1 = \dfrac{p^{(k)T} b}{p^{(k)T} A p^{(k)}}$

A sequence of vector $p_0 \dots p^{(k)}$ where $P^{(k-1)T} A p^{(k)}$ is A-conjugate

Definition 8.3.1.2 A-conjugate directions

Let A be SPD. A sequence $p^{(0)}, p^{(1)}, \dots p^{(k-1)} \in \mathbb{R}^n$ such that

$$p^{(j)T} A p^{(i)} = 0 \quad \text{if only if} \quad i \neq j$$

is said to be A-conjugate

So in conclusion, the this tells us how to pick direction $p^{(k)}$, A-conjugate.

Hw 8.3.1.2

ALWAYS: $P \in \mathbb{R}^{n \times k}$ are A-conjugate if only if:

$P^T A P = D$, where D is diagonal and has positive values on its diagonal

Algorithm with A-conjugate search direction:

Given: A, b

$x^{(0)} := 0$

$r^{(0)} := b$

$k := 0$

while $r^{(k)} \neq 0$

choose $p^{(k)}$ such that $p^{(k)T} A P^{(k-1)} = 0$ and $p^{(k)T} r^{(k)} \neq 0$

$\alpha_k := \dfrac{p^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$

$x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$

$r^{(k+1)} := r^{(k)} - \alpha_k A p^{(k)}$

$k := k+1$

endwhile

Remark 8.8.1.4

The important observation is that if $p^{(0)}, \dots, p^{(k)}$ are chosen to be A-conjugate, then $x^{(k+1)}$ not only minimize

$$f(x^{(k)} + \alpha_k p^{(k)})$$

but also minimize:

$$\min_{x \in \mathrm{span}(p^{(0)}, \dots p^{(k-1)})} f(x)$$

Existence of A-conjugate search directions

We need to ask this question when picking search direction p:

- $p \perp \mathrm{Span}(Ap_0, Ap_1, \dots, Ap_{(k-1)})$ <A-conjugate>
- p is not $\perp$ to $r^{(k)}$

What happen when $p \perp r^{(k)}$, or $p^T r^{(k)} = 0$?

$$p^T r^{(k)} = p^T(b - Ax^{(k)}) = p^T(b - AP^{(k-1)} a^{(k-1)}) = 0$$

$$\Leftrightarrow p^T b = 0 \quad \text{for all } p \perp \mathrm{span}(Ap^{(0)}, \dots, Ap^{(k-1)})$$

$$\Rightarrow b \in \mathrm{Span}(Ap^{0}, \dots, Ap^{(k-1)})$$

Therefore, $\begin{cases} b = AP^{(k-1)} z & \text{for some } z \in \mathbb{R}^k \\ x = P^{(k-1)} z & \text{solves } Ax = b \end{cases}$

Basically, when $p \perp r^{(k)}$, then we already found the solution since $x^{(k)}$ minimizes $\cancel{f(x) =} \min_{x \in \mathrm{Span}(p^{(0)}, \dots p^{(k-1)})} \underbrace{\| b - Ax \|_2}_{f(x)}$

so $Ax^{(k)} = b$, $r^{(k)} = 0$

Conjugate Gradient Method Basics:

In summary, the idea behind Conjugate Gradient Method is:

- At k iteration, we have an approximation $x^{(k)}$ to the solution to $Ax = b$. $x^{(k)} = \alpha_0 p^{(0)} + \dots + \alpha_{k-1} p^{(k-1)}$

(9)

- The residual $r^{(k)} = b - Ax^{(k)}$

$$= b - \alpha_0 Ap^{(0)} - \dots - \alpha_{k-1} Ap^{(k-1)}$$

$$= r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)}$$

- If $r^{(k)} = 0$, then $x^{(k)}$ solves $Ax = b$, and we are done.

How to construct $p^{(k)}$ that is A-conjugate to previous directions and $p^{(k)T} r^{(k)} \neq 0$, assume that $r^{(k)} \neq 0$?

- We ideally would like it to be the steepest descent, so $r^{(k)} = b - Ax^{(k)}$
- However, it is rarely the case, so we pick $p^{(k)}$ that is A-conjugate and is closest to $r^{(k)}$, so:

$$\| p^{(k)} - r^{(k)} \|_2 = \min_{p \perp \mathrm{Span}(Ap^{(0)}, \dots, Ap^{(k-1)})} \| r^{(k)} - p \|_2$$

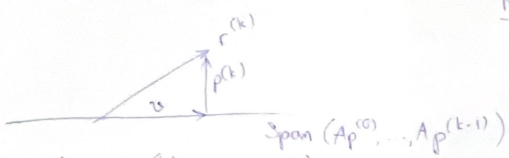
This yield the algorithm: **Conjugate Gradient Method**

```
Given: A, b
x^(0) := 0
r^(0) := b
k := 0
While r^(k) ≠ 0:
    if k = 0:
        p^(k) = r^(0)
    else:
        p^(k) minimizes  min_{p ⊥ Span(Ap^(0), ..., Ap^(k-1))} ||r^(k) - p||_2
    endif
    α_k := p^(k)T r^(k) / p^(k)T A p^(k)
    x^(k+1) := x^(k) + α_k p^(k)
    r^(k+1) := r^(k) - α_k A p^(k)
    k := k+1
endwhile
```

(10)

Technical Details

How do we pick $p^{(k)}$ that actually minimizes $\min\limits_{p \perp Span(Ap^{(0)}, \dots, Ap^{(k-1)})} \|r^{(k)} - p\|_2$?



Notice that: $r^{(k)} = v + p^{(k)}$

We can restate the problem as:

$$\|r^{(k)} - v\|_2 = \min_{w \in Span(Ap^{(0)}, \dots, Ap^{(k-1)})} \|r^{(k)} - w\|_2$$

$$= \min_{z \in \mathbb{R}^k} \|r^{(k)} - AP^{(k-1)} z\|_2 \quad \langle v = AP^{(k-1)} z^{(k)} \rangle$$

↳ This is solving LLS problem!

Conjugate Gradient Method Observations

$$\|r^{(k)} - p^{(k)}\|_2 = \min_{p \perp Span(Ap^{(0)}, \dots, Ap^{(k-1)})} \|r^{(k)} - p\|_2$$

- $p^{(k)} = r^{(k)} - v = r^{(k)} - AP^{(k-1)} z^{(k)}$
  $\Rightarrow \|r^{(k)} - AP^{(k-1)} z^{(k)}\|_2 = \min\limits_{z} \|r^{(k)} - AP^{(k-1)} z\|_2$
- $P^{(k-1)T} r^{(k)} = 0$
- $Span(p^{(0)}, \dots, p^{(k-1)}) = Span(r^{(0)}, \dots, r^{(k-1)})$ < Krylov subspace >
  $= Span(b, Ab, A^2 b, \dots, A^{(k-1)} b)$
- $r^{(i)T} r^{(j)} = 0$ if $i \neq j$

So, we can conclude that to pick $p^{(k)}$ where:

$$p^{(k)} = r^{(k)} - AP^{(k-1)} z^{(k)}$$

where $z^{(k)}$ solves

$$\min_{z \in \mathbb{R}} \|r^{(k)} - AP^{(k-1)} z\|_2$$

↳ This implies that to find $p^{(k)}$, we need "memory" of all

previous searches, as stated by $P^{(k-1)}$.

However, if we push through the theory, we actually only need the "memory" of the last iteration, as stated in theorem 8.3.4.4

Theorem 8.3.4.4

For $k \geq 1$, the search directions generated by the Conjugate Gradient Method satisfy

$$p^{(k)} = r^{(k)} + \gamma_k p^{(k-1)} \quad \text{for some constant } \gamma_k$$

This means we don't have to store all those "memory"!

Practical Conjugate Gradient Method Algorithm

Given: A, b
$x^{(0)} := 0$
$r^{(0)} := b$
$k := 0$
while $r^{(k)} \neq 0$:
  if $k = 0$:
    $p^{(k)} = r^{(0)}$
  else:
    $\gamma_k := \dfrac{-p^{(k-1)T} A r^{(k)}}{p^{(k-1)T} A p^{(k-1)}}$
    $p^{(k)} := r^{(k)} + \gamma_k p^{(k-1)}$
  endif
  $\alpha_k := \dfrac{p^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$
  $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$
  $r^{(k+1)} := r^{(k)} - \alpha_k A p^{(k)}$

  $k := k + 1$
endwhile

Hw 8.3.5.1

$\alpha_k := \dfrac{p^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$ has an alternative $\alpha_k := \dfrac{r^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$

This homework suggest an alternative (refined) Conjugate Gradient Method

Alternative Conjugate Gradient Methods Algorithms:

Alternative 1:

Given: A, b

$x^{(0)} := 0$

$r^{(0)} := b$

$k := 0$

while $r^{(k)} \neq 0$

  if $k = 0$:

    $p^{(k)} = r^{(0)}$

  else:

    $\gamma_k := \dfrac{-p^{(k-1)T} A r^{(k)}}{p^{(k-1)T} A p^{(k-1)}}$

    $p^{(k)} := r^{(k)} + \gamma_k p^{(k-1)}$

  endif

  $\alpha_k := \dfrac{r^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$

  $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$

  $r^{(k+1)} := r^{(k)} - \alpha_k A p^{(k)}$

  $k := k + 1$

endwhile

Alternative 2:

Given. A, b

$x^{(0)} := 0$

$r^{(0)} := b$

$k := 0$

while $r^{(k)} \neq 0$:

  if $k = 0$:

    $p^{(k)} = r^{(0)}$

  else:

    $\gamma_k := \dfrac{r^{(k)T} r^{(k)}}{r^{(k-1)T} r^{(k-1)}}$

    $p^{(k)} := r^{(k)} + \gamma_k p^{(k-1)}$

  endif

  $\alpha_k := \dfrac{r^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$

  $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$

  $r^{(k+1)} := r^{(k)} - \alpha_k A p^{(k)}$

  $k := k + 1$

end while

Hw 8.3.5.2

Can be proven that: $\begin{cases} r^{(k)T} r^{(k)} = -\alpha_k r^{(k)T} A p^{(k-1)} \\ p^{(k-1)T} A p^{(k-1)} = \dfrac{r^{(k-1)T} r^{(k-1)}}{\alpha_{k-1}} \end{cases}$

Therefore, $\gamma_k = \dfrac{-p^{(k-1)T} A r^{(k)}}{p^{(k-1)T} A p^{(k-1)}} = \dfrac{r^{(k)T} r^{(k)}}{r^{(k-1)T} r^{(k-1)}}$

Stopping Criteria:

In theory, the Conjugate Gradient Method requires at most n iterations for $r^{(k)} = 0$ so that $x^{(k)}$ solves $Ax = b$ exactly.

In practice, that rarely happens that $r^{(k)} = 0$ exactetly because
- floating point arithmetic
- catastrophic cancellation since $r^{(k)}$ start b and keep shrinking every iterations.

To mitigate this we either
- set stop when $\|r^{(k)}\|_2 < \varepsilon_{mach} \|b\|_2$
- set stop after some m iterations

Preconditioning revisit

In previous discussion, we know that method of Steepest Descent can be greatly accelerated by using a preconditioner.

Let apply that to our finalized algorithm:

Given: $A, b, M = \tilde{L} \tilde{L}^T$

$x^{(0)} := 0$

$r^{(0)} := b$

$k := 0$

while $r^{(k)} \neq 0$

  $z^{(k)} := M^{-1} r^{(k)}$

  if $k = 0$:

    $p^{(k)} := z^{(0)}$

  else:

    $\tilde{\gamma}_k := \dfrac{r^{(k)T} z^{(k)}}{r^{(k-1)T} z^{(k-1)}}$

    $p^{(k)} := z^{(k)} + \tilde{\gamma}_k p^{(k-1)}$

  endif

continue:

  $q^{(k)} := A p^{(k)}$

  $\tilde{\alpha}_k := \dfrac{r^{(k)T} z^{(k)}}{p^{(k)T} q^{(k)}}$

  $x^{(k+1)} := x^{(k)} + \tilde{\alpha}_k p^{(k)}$

  $r^{(k+1)} := r^{(k)} - \tilde{\alpha}_k q^{(k)}$

  $k := k + 1$

endwhile