1 26th of October 2018 — F. Poloni

1.1 How to construct a QR factorization

In the previous lecture we introduced the QR factorization and we defined what an Householder reflector is.

At the end of the lecture we gave a first MatLab implementation of `householder_vector`:

Algorithm 1.1 Householder vector Matlab implementation.

```
1 function[v,s] = householdervector(x)
2 s = norm(x);
3 v = x;
4 v(1) = v(1) - s;
5 v = v / norm(v);
```

What's the problem of this algorithm? That the subtraction may create a problem with machine numbers, if s and $\|x\|$ are very close. If we take $\|x\|= -s$ the subtraction becomes and addition, and everything works well.

In the end, we would like to obtain this behaviour for every possible value for x and s, so line 2 may be modified as `s = - sign(x(1)) * norm(x)`.

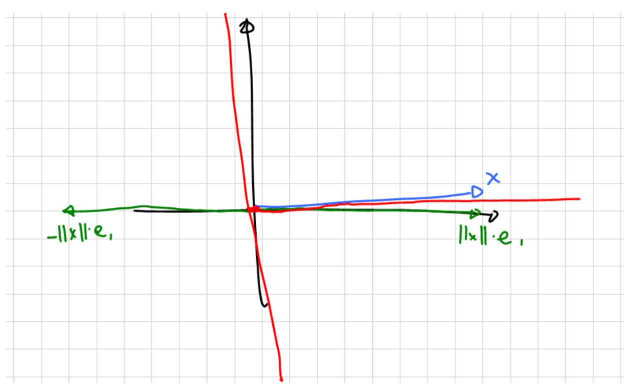

Figure 1.1: If x is oriented as in the plot it's better if we choose $-\|x\|e_1$ verse, since it's opposite to x.

Example 1.1. *Given* $A = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \in \mathcal{M}(m, m, \mathbb{R})$, *where* $m = 5$, *we would like to calculate the* QR *factorization of* A.

Step 1 : *costruct a Householder matrix that sends $A(:,1)$ (first column of A) to a multiple of e_1. Then we have* $H_1A = \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix}$

Step 2 : *take $H_2 \in \mathcal{M}(m-1, m-1, \mathbb{R})$ such that $H_2A(2:end, 2) = \begin{pmatrix} \times \\ 0 \\ 0 \\ 0 \end{pmatrix}$ and compute:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & & & & \\ 0 & & & H_2 & \\ 0 & & & & \\ 0 & & & & \end{pmatrix} \cdot H_1A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & & & & \\ 0 & & & H_2 & \\ 0 & & & & \\ 0 & & & & \end{pmatrix} \cdot \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix}$$
$$= \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{pmatrix} \tag{1}$$

And we denote $Q_2 = \begin{pmatrix} I_{1\times1} & 0 \\ 0 & H_2 \end{pmatrix}$, $Q_1 = H_1$;

Step 3 : *take $H_3 \in \mathcal{M}(m-2, m-2, \mathbb{R})$ such that $H_3A(3:end, 3) = \begin{pmatrix} \times \\ 0 \\ 0 \end{pmatrix}$ and we compute:*

$$Q_3 \cdot (Q_2Q_1A) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & & & \\ 0 & 0 & & H_3 & \\ 0 & 0 & & & \end{pmatrix} \cdot \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{pmatrix}$$
$$= \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{pmatrix} \tag{2}$$

So, $Q_3 = \begin{pmatrix} I_{2\times2} & 0 \\ 0 & H_3 \end{pmatrix}$;

Step 4 : *take $H_4 \in \mathcal{M}(m-3, m-3, \mathbb{R})$ such that $H_4 A(4 : end, 4) = \begin{pmatrix} \times \\ 0 \end{pmatrix}$ and we compute:*

$$\begin{aligned} Q_4 \cdot (Q_3 Q_2 Q_1 A) &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & & \\ 0 & 0 & & H_4 & \\ 0 & 0 & & & \end{pmatrix} \cdot \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{pmatrix} \\ &= \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{pmatrix} \end{aligned} \tag{3}$$

Where, $Q_4 = \begin{pmatrix} I_{3\times 3} & 0 \\ 0 & H_4 \end{pmatrix}$.

In the end, since Q_i is an orthogonal matrix and the product of orthogonal matrices is orthogonal, $Q_1 \cdot Q_2 \cdot Q_3 \cdot Q_4 A = T$, which is an upper triangular matrix.

Theorem 1.1 (Product of block matrices)**.** *Let $I \in \mathcal{M}(k, k, \mathbb{R})$, let $H_i \in \mathcal{M}(m-k, m-k, \mathbb{R})$ and let $B_i \in \mathcal{M}(k, k, \mathbb{R})$, $C_i \in \mathcal{M}(k, m-k, \mathbb{R})$ and $A_i \in \mathcal{M}(m-k, m-k, \mathbb{R})$, then the product between the two following block matrices is exactly the one showed below.*

$$\begin{pmatrix} I & 0 \\ 0 & H_i \end{pmatrix} \cdot \begin{pmatrix} B_i & C_i \\ 0 & A_i \end{pmatrix} = \begin{pmatrix} B_i I & C_i \\ 0 & H_i \cdot A_i \end{pmatrix}$$

Proof. It's trivial computation, using the definition of matrix product. □

1.1.1 Matlab implementation

Algorithm 1.2 First implementation of QR factorization.

```
function [Q, R] = myqr(A)
[m, n] = size(A);
Q = eye(m);
for j = 1:n
    v = householder_vector(A(j:end, j));
    H = eye(length(v)) - 2*v*v';
    A(j:end,j:end) = H * A(j:end,j:end);
    Q(:, j:end) = Q(:, j:end) * H;
end
R = A;
```

Fact 1.2. *The cost of this implementation when A is a square matrix is $O(n^3 + (n-1)^3 + \cdots + 1^3)$. If A is a rectangular matrix, then the computational complexity is $O(m \cdot n^2 + (m-1) \cdot (n-1)^2 + \cdots + (m-n+1)^3)$.*

Proof. Line 7 does a matrix product between matrices of size n, $n-1$, ..., 1, so the resulting cost is $O(m \cdot n^2 + (m-1) \cdot (n-1)^2 + \cdots + (m-n+1)^3)$. □

We may design a faster algorithm, since $HA_j = A_j - 2v(v^T A_j)$.

Algorithm 1.3 More efficient implementation of QR factorization.

```
1  function [Q, A] = myqr(A)
2  [m, n] = size(A);
3  Q = eye(m);
4  for j = 1:n-1
5      [v, s] = householder_vector(A(j:end, j));
6      A(j,j) = s; A(j+1:end,j) = 0;
7      A(j:end,j+1:end) = A(j:end,j+1:end) - ...
8          2*v*(v'*A(j:end,j+1:end));
9      Q(:, j:end) = Q(:, j:end) - Q(:,j:end)*v*2*v';
10 end
```

Let's suppose that A is square matrix, partitioned as: $A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \cdot \begin{pmatrix} R_{1,1} & R_{1,2} \\ 0 & R_{2,2} \end{pmatrix}$ Then we can recover the factorization of A_1 from the factorization of A, since $A_1 = Q \cdot R_{11}$.

Fact 1.3 (Thin QR factorization)**.** *we may replace $Q \in \mathcal{M}(m, m, \mathbb{R})$ and $R \in \mathcal{M}(m, m, \mathbb{R})$ with $Q_1 \in \mathcal{M}(m, n, \mathbb{R})$ and $R_1 \in \mathcal{M}(n, n, \mathbb{R})$ and the same factorization holds: $A = QR = Q_1 R_1$. This is called* ***thin QR factorization****.*

Proof. $A_1 \in \mathcal{M}(m, n, \mathbb{R})$, $A = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q \cdot R = \begin{pmatrix} Q_1 \end{pmatrix} \cdot \begin{pmatrix} R_1 \end{pmatrix} + \begin{pmatrix} Q_2 \end{pmatrix} \begin{pmatrix} 0 \end{pmatrix} = Q_1 \cdot R_1$ □

In order to save space we may work in the following way:

$$Q \cdot B = \begin{pmatrix} 1 & \times & \cdots & \times \\ 0 & & & \\ \vdots & & \text{I-2V}_1\text{V}_1{}^T & \\ 0 & & & \\ 0 & & & \end{pmatrix} \cdot \begin{pmatrix} 1 & \times & \times & \cdots & \times \\ 0 & 1 & \times & \cdots & \times \\ \vdots & 0 & & \text{I-2V}_2\text{V}_2{}^T & \\ 0 & \vdots & & & \\ 0 & 0 & & & \end{pmatrix} \cdot$$

$$\cdot \ldots \cdot \begin{pmatrix} 1 & \times & \times & \cdots & \times \\ 0 & 1 & \times & \cdots & \times \\ \vdots & 0 & 1 & & \\ 0 & \vdots & 0 & \text{I-2V}_n\text{V}_n{}^T & \\ 0 & 0 & 0 & & \end{pmatrix} \cdot B.$$

Fun fact

There are some libraries that store the v_i vectors in the lower part of matrix R which is upper triangular and has only zeros below the main diagonal.

1.2 How to use the thin QR factorization to solve a least squares problem

We would like to solve $\|Ax - b\|$ $\forall A \in \mathcal{M}(m, n, \mathbb{R})$ and $\forall B \in \mathbb{R}^n$ where $m > n$ (a.k.a A is a tall, thin matrix),through the QR factorization. We would like to solve $\min \|Ax - b\|$ through the QR factorization.

We may write first the QR factorization of A, so $\forall A \in \mathcal{M}(m, n, \mathbb{R})$, $\exists Q \in \mathcal{M}(m, m, \mathbb{R})$, $\exists R \in \mathcal{M}(m, n, \mathbb{R})$ such that $A = QR$, where $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ and $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$.

Then,

$$\begin{aligned} \|A\mathbf{x} - \mathbf{b}\| &= \left\|Q^T(A\mathbf{x} - \mathbf{b})\right\| = \left\|Q^T QR\mathbf{x} - Q^T\mathbf{b}\right\| \\ &= \left\|R\mathbf{x} - Q^T\mathbf{b}\right\| = \left\|\begin{pmatrix} R_1 \\ 0 \end{pmatrix}\mathbf{x} - \begin{pmatrix} Q_1{}^T \\ Q_2{}^T \end{pmatrix}\mathbf{b}\right\| \\ &= \left\|\begin{pmatrix} R_1\mathbf{x} - Q_1{}^T\mathbf{b} \\ Q_2{}^T\mathbf{b} \end{pmatrix}\right\| \end{aligned} \tag{4}$$

How can we pick $\mathbf{x}$ to minimize the norm of $A\mathbf{x} - \mathbf{b}$?

Can we choose x such that $R_1\mathbf{x} - Q_1{}^T\mathbf{b} = 0$? Yes, we can, since this is a linear square system, so $\mathbf{x} = R_1{}^{-1}Q_1{}^T\mathbf{b}$

$\|A\mathbf{x} - \mathbf{b}\| = \left\|Q_2{}^T\mathbf{b}\right\|$

We used the fact that R_1 is invertible, but is it always true that R_1 is invertible?

Lemma 1.4. *R_1 is invertible $\Leftrightarrow$ A has full column rank.*

Proof. A has full column rank $\Leftrightarrow$ A^TA is positive definite $\Leftrightarrow$ A^TA is positive semidefinite and invertible, but A^TA is positive semidefinite, so we only need to prove its invertibility.

Let's compute $QR^TQR = R^TQ^TQR = R^TR = \begin{pmatrix} {R_1}^T & 0 \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = {R_1}^T \cdot R_1$.

So, A^TA is invertible $\Leftrightarrow$ R_1 is inverbile. $\square$

Note

${R_1}^T R_1$ is the Cholesky factorization of A^TA.

The computational complexity is asymptotically equal to the one of computing the QR factorization, since the other operations are cheaper (the product $Q_1^T\mathbf{b}$ costs $O(mn)$ and solving the triangular linear system by back-substitution costs $O(n^2)$).