MATH 612
Computational methods for equation solving and function minimization – Week # 4

Instructor: Francisco-Javier 'Pancho' Sayas

Spring 2014 – University of Delaware

Plan for this week

- Discuss any problems you couldn't solve previous lectures
- Read Lectures 8, 10, and 11
- The first coding assignment is due Friday
- The other two coding assignments will be cut into smaller pieces

Remember that...

... I'll keep on updating, correcting, and modifying the slides until the end of each week.

Important for next week

The next collection of chapters of the book (Lectures 12 to 15) are better read than explained. You'll have a lot of reading next week.

MATLAB TIPS

Vectorizing what's not vectorized

Imagine you have two row lists of numbers

$$[t_1, \ldots, t_m], \qquad [\tau_1, \ldots, \tau_n]$$

and we want to compute the $m \times n$ matrix with values

$$t_i - \tau_j.$$

Here's how...

```
>> t=[1 2 3]; tau=[0 2 4 6];
>> bsxfun(@minus,t',tau)
ans =
     1    -1    -3    -5
     2     0    -2    -4
     3     1    -1    -3
```

Reading backwards?

If you want to read the columns of a matrix from end to beginning, you can do this...

```
>> A=[1 2 3 4;5 6 7 8;9 10 11 12]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> A(:,end:-1:1)
ans =
     4     3     2     1
     8     7     6     5
    12    11    10     9
```

GRAM-SCHMIDT

Review of the classical Gram-Schmidt method

For $j = 1$ to the number of columns of A (assumed to be linearly independent), compute

$$
\begin{aligned}
v_j &= a_j - \sum_{i=1}^{j-1} \underbrace{(q_i^* a_j)}_{r_{ij}} q_i = a_j - \sum_{i=1}^{j-1} q_i q_i^* a_j \\
&= (I - \sum_{i=1}^{j-1} q_i q_i^*) a_j = P_j a_j
\end{aligned}
$$

and then

$$
r_{jj} = \|v_j\|, \qquad q_j = \frac{1}{r_{jj}} v_j .
$$

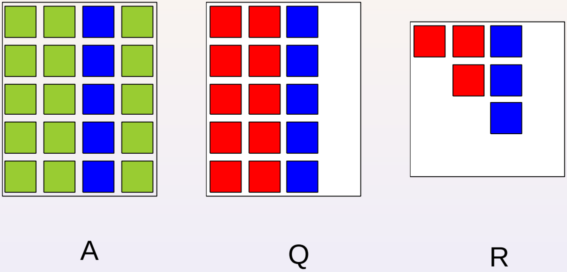
Recycled pseudocode

Remember that the goal is the reduced QR decomposition

$$A = \widehat{Q}\widehat{R}$$

for $j = 1 : n$ % this loop runs on columns of Q and A
    $v_j = a_j$
    for $i = 1 : j - 1$ % this loop is the summation sign
        $r_{ij} = q_i^* a_j$ % the j-th column of R is computed
        $v_j = v_j - r_{ij} q_i$
    end
    $r_{jj} = \|v_j\|_2$
    $q_j = \frac{1}{r_{jj}} v_j$
end

A pictorial representation of classical GS



In **blue** the column of A we are using and the elements of Q and R we are computing. We are in the third go through the loop. The **red** elements of Q and R have been computed already. The **green** elements of A are not active in this step.

An alternative version of the algorithm (not final yet)

Observation

$$P_j = I - \sum_{i=1}^{j-1} q_i q_i^* = (I - q_{j-1} q_{j-1}^*) \ldots (I - q_2 q_2^*)(I - q_1 q_1^*)$$

for $j = 1 : n$ % this loop runs on columns of Q and A
    $v_j = a_j$
    for $i = 1 : j - 1$ % loop for progressive projections
        $r_{ij} = q_i^* v_j$ % we use v_j instead of a_j
        $v_j = v_j - r_{ij} q_i$
    end
    $r_{jj} = \|v_j\|_2$
    $q_j = \frac{1}{r_{jj}} v_j$
end

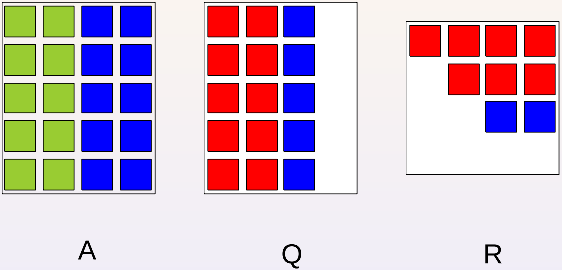
The final version

```
for j = 1 : n
    v_j = a_j
end
for j = 1 : n
    for i = 1 : j − 1
        r_ij = q_i^* v_j
        v_j = v_j − r_ij q_i
    end
    r_jj = ‖v_j‖_2
    q_j = (1/r_jj) v_j
end
```

$$
\begin{array}{l}
\text{for } j = 1 : n \\
\quad v_j = a_j \\
\text{end} \\
\text{for } i = 1 : n \\
\quad r_{ii} = \|v_i\|_2 \\
\quad q_i = \frac{1}{r_{ii}} v_i \\
\quad \text{for } j = i+1 : n \\
\quad\quad r_{ij} = q_i^* v_j \\
\quad\quad v_j = v_j - r_{ij} q_i \\
\quad \text{end} \\
\text{end}
\end{array}
$$

Once the vector q_i is computed, the projection of all columns onto $\langle q_i \rangle$ is subtracted. The matrix R is computed row-wise now.

A pictorial representation of modified GS



In **blue** the columns of A that are being used are using and the elements of Q and R we are computing. (A was copied in V and is modified in each step.) We are in the third go through the loop. The **red** elements of Q and R have been computed already. The green elements of A are not active in this step and won't be any longer.

Operation count

We count flops (sums, substractions, multiplications, and division). A norm and a dot-product need $2m-1$ flops (m is the number of elements of the vectors).
Each run of the internal loop needs $2m-1+2m \sim 4m$ flops.
Each run of the external loop then needs

$$\sim 2m-1+m+(n-i)4m \sim 3m+4m(n-i)$$

and then the total count is

$$\sim 3mn+4m\sum_{i=1}^{n}(n-i) = 3mn+4m\sum_{i=1}^{n} i \sim 2mn^2$$

There's an amazing geometric interpretation of the operation count in the book that you should really understand. It's much simpler than this kind of bean counting.

HOUSEHOLDER

A new goal

Given a matrix A with full column rank, compute a full QR decomposition

$$A = QR, \qquad Q \text{ unitary}, \qquad R \text{ upper triangular}$$

The basic idea

Given $x \in \mathbb{C}^m$, we construct

$$u = \frac{1}{\|x + \sigma\|x\|_2 e_1\|_2}(x + \sigma\|x\|_2 e_1), \qquad \sigma := \operatorname{sign}(x_1)$$

Then, the Householder reflector $H_u = I - 2uu^*$ satisfies

$$\begin{aligned} H_u x &= -\sigma e_1 \\ H_u y &= y \quad \text{if } y \perp u, \\ H_u y &= y - 2u(u^* y) \qquad \text{for a general vector} \end{aligned}$$

Householder's method (rough pseudo-code)

Start with $A^{(1)} = A$. For increasing j, follow this process

$$\begin{aligned}
x_j &:= j\text{-th column of } A^{(j)}, \\
c_j &:= \text{elements } j \text{ to } m \text{ of } x_j \\
\sigma_j &:= \text{sign of the first element of } c_j \\
v_j &:= \frac{1}{\|c_j + \sigma_j \|c_j\|_2 e_1\|_2}(c_j + \sigma_j \|c_j\|_2 e_1) \\
u_j &:= \text{add } j-1 \text{ zeros on top of } v_j \\
A^{(j+1)} &:= (I - 2u_j u_j^*)A^{(j)}
\end{aligned}$$

The matrix

$$R = A^{(n+1)} = (I - 2u_n u_n^*) \dots (I - 2u_1 u_1^*)A$$

is upper triangular.

Householder's method: why it works

- In the first step, the first column of $A^{(2)}$ has its last $m-1$ elements equal to zero
- In the second step, u_2 starts with a zero component, so $H_{u_2} = I - 2u_2u_2^*$ does not modify the first column of $A^{(2)}$. The vector u_2 is chosen so that the last $m-2$ elements of the second column of $A^{(3)}$ vanish.
- In the third step, u_3 starts with two zero elements, so H_{u_3} does not modify the first two columns of $A^{(3)}$. The vector u_3 is chose so that the last $m-3$ elements of the third column of $A^{(4)}$ vanish.
- Et cetera.

Householder delivers QR

The construction is

$$R = (I - 2u_n u_n^*) \dots (I - 2u_1 u_1^*) A$$

and therefore

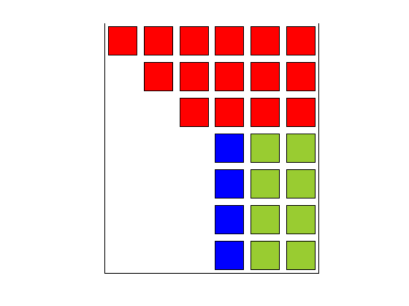
$$A = QR,$$

where

$$Q = (I - 2u_1 u_1^*) \dots (I - 2u_n u_n^*) = (I - 2u_1 u_1^*) \dots (I - 2u_n u_n^*) I$$

is a unitary matrix.

A picture



We are about to begin the fourth step. The elements in **red** will not be modified any longer. The column in **blue** is activated to create a short (4 components) reflection vector. All non-red elements will be modified in this step. Zeros (blanks) are untouched.

A key point in the algorithm

To compute

$$A^{(j+1)} := (I - 2u_j u_j^*)A^{(j)} = A^{(j)} - 2u_j(u_j^* A^{(j)}),$$

note that:

- The first $j-1$ columns will not be modified, so we do not need to operate with them.
- The first $j-1$ rows will not be modified (think of each column vector as the sum of two vectors: one will remain the same, the other one will be modified). Instead of working with u_j we can work with v_j

A similar point can be raised in the computation of Q, if this matrix is wanted at all. We can just store the vectors u_j and an algorithm to multiply by Q, or, even better, the vectors v_j...

LEAST SQUARES

An optimization problem

Let A be an $m \times n$ matrix and $b \in \mathbb{C}^m$. Find $x \in \mathbb{C}^n$ minimizing

$$\|b - Ax\|_2.$$

- By x minimizing $\|b - Ax\|_2$ we mean

$$\|b - Ax\|_2 \leq \|b - Az\|_2 \qquad \forall z \in \mathbb{C}^n.$$

- The vector $r = b - Ax$ is called the residual.
- We will be able to solve this problem because the norm is the $2-$norm. With other norms this problem is actually quite complicated.
- The problem might have more than one of them. In principle, we care about having one solution. We also care about the vector Ax, where x is the solution of the minimization problem.

An optimization problem (2)

The problem of minimizing

$$\|b - Ax\|_2$$

is equivalent to minimizing

$$\|Ax - b\|_2^2 = (Ax - b)^*(Ax - b)$$

It is called the **least squares minimization problem**. A solution of this problem is called a **least squares solution** of the system $Ax = b$.

Remark

A solution of $Ax = b$ is automatically a least squares solution. A least squares solution might not be a solution though. (Think of the case when $Ax = b$ is not solvable.)

An argument leading to a theorem

The cast. A matrix $A \in \mathbb{C}^{m\times n}$, a vector $b \in \mathbb{C}^m$, the orthogonal projection $y = Pb \in \mathbb{C}^m$ of b onto the range of A. (Here P is the orthogonal projector onto range(A).)
The plot. A bystander $z \in$ range(A) enters the scene. Then

$$b - z = \underbrace{b - y}_{\in \text{range}(A)^\perp} + \underbrace{y - z}_{\in \text{range}(A)}$$

and therefore (Pythagoras anyone?)

$$\|b - z\|_2^2 = \|b - y\|_2^2 + \|y - z\|_2^2 \geq \|b - y\|_2^2.$$

Denouement. We recognize $y = Pb = Ax$ for some $x \in \mathbb{C}^n$ (by definition of range of A) and have shown that

$$\|b - Ax\|_2^2 \text{ is minimized} \qquad \Longleftrightarrow \qquad Ax = y = Pb.$$

A chain of conclusions, à la Sherlock

x is a least square solution (x minimizes $\|b - Ax\|_2$)

iff

Ax is the projection of b onto range(A)

iff

$b - Ax$ is orthogonal to range(A)

iff

$b - Ax$ is orthogonal to the columns of A

iff

$$A^*(Ax - b) = 0$$

iff

$$A^*Ax = A^*b$$

Drumroll... and voilà, le théorème

Theorem

The vector x is a least squares solution of $Ax = b$ (i.e., x minimizes $\|b - Ax\|_2$) if and only if

$$A^*Ax = A^*b.$$

The latter equations are called the **normal equations.**

Implicit to the argument is the **existence of at least one** least squares solution. Start with b, find Pb, its orthogonal projection onto range(A). Since $Pb \in$ range(A) there must be at least one x such that $Ax = Pb$. This x (and any other x with the same property) is a least squares solution.

Attaboy,... a fictitious dialogue

- Professor, professor, ... is the solution unique?
- Good question! It might not be. Look again at the argument. Any x such that $Ax = Pb$ works and only these x. These equations are solvable. If A has full rank by columns, the solution is unique. Otherwise, the solution is determined up to elements of null(A).
- Can I get a second opinion?
- You are even entitled to it. We want to solve $A^*Ax = A^*b$. We know these equations are solvable. And we know

$$\text{null}(A^*A) = \text{null}(A).$$

We know it, but we might not remember it, but we should!

The full rank case

Minimizing $\|b - Ax\|_2$ is equivalent to solving the normal equations

$$A^*Ax = A^*b.$$

If (and only if) $\operatorname{rank}(A) = n$ (the number of columns), A^*A is invertible and then

$$x = (A^*A)^{-1}A^*b.$$

Now recall that Ax is the orthogonal projection of b onto $\operatorname{range}(A)$ and note that

$$Ax = \underbrace{A(A^*A)^{-1}A^*}_{P}\, b,$$

which we kind of knew already.

The pseudoinverse revisited

When A has full rank by columns

$$x = (A^*A)^{-1}A^*b$$

is the least squares solution. The matrix

$$A^+ = (A^*A)^{-1}A^*$$

is called the pseudoinverse of A. *Is this the same one we got with the SVD? Yes.* Why? Because we proved that with the other definition, we always got a solution of the normal equations, and in this case the solution of the normal equations is unique.

For a matrix A with full column rank, the pseudoinverse is the operator that for given b outputs the least square solution of $Ax = b$.

Let A have full rank by columns. Its reduced SVD

$$A = \widehat{Q}\widehat{\Sigma}V^*$$

uses

- An $m \times n$ matrix $\widehat{Q}$ with orthonormal columns.
- A square diagonal positive $n \times n$ matrix $\widehat{\Sigma}$ with elements given in non-increasing order.
- A unitary matrix V. (The missing hat is not a typo. In this case the rank is the number of columns and V is the same as in a full SVD.) Again, $V^{-1} = V^*$.

With the new definition...

$$\begin{aligned} A^+ &= (A^*A)^{-1}A^* = (V\widehat{\Sigma}\underbrace{\widehat{Q}^*\widehat{Q}}_{=I}\widehat{\Sigma}V^*)^{-1}V\widehat{\Sigma}\widehat{Q}^* \\ &= (V\widehat{\Sigma}^2V^*)^{-1}V\widehat{\Sigma}\widehat{Q}^* = V\widehat{\Sigma}^{-2}\underbrace{V^*V}_{=I}\widehat{\Sigma}\widehat{Q}^* = V\widehat{\Sigma}^{-1}\widehat{Q}^*. \end{aligned}$$

Least squares and QR

If $A = \widehat{Q}\widehat{R}$ is a reduced QR decomposition of a matrix with full column rank (therefore $\widehat{R}$ is a square invertible upper triangular matris), then

$$\begin{aligned} A^+ &= (\widehat{R}^*\widehat{Q}^*\widehat{Q}\widehat{R})^{-1}\widehat{R}^*\widehat{Q}^* \\ &= (\widehat{R}^*\widehat{R})^{-1}\widehat{R}^*\widehat{Q}^* \\ &= \widehat{R}^{-1}\widehat{Q}^*. \end{aligned}$$

(Please, be sure you can follow all the steps in this computation.) Then x is the least square solution if and only if

$$\widehat{R}x = \widehat{Q}^*b.$$

In summary, given a reduced QR decomposition, finding the LS solution involves: multiplying the r.h.s. by the adjoint of $\widehat{Q}$, solving and upper triangular linear system. Both steps are really easy. Nice!

AN APPLICATION

Polynomial fitting

Given points

$$(x_i, y_i) \qquad i = 1, \ldots m,$$

find a polynomial of degree $n-1$ or less

$$p(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

such that

$$\sum_{i=1}^{m} |y_i - p(x_i)|^2 \qquad \text{is minimum}$$

(where is minimum is to be read as *among all possible choices of the polynomial* p.)

Jargon. The values x_i are the data locations. The values y_i are the data. The polynomial $p(x)$ is the model.

Recasting the problem in our LS format

We change the notation so that the problem fits in the LS format:

$$r_i = y_i - p(x_i) = y_i - \sum_{j=0}^{n-1} a_j x_i^j \qquad b_i = y_i$$

$$A_{ij} = x_i^j, \qquad \begin{array}{ll} i = 1, \ldots, m & \text{(number of data)} \\ j = 0, \ldots, n-1 & \text{(polynomial degree)} \end{array}$$

The polynomial fitting problem is equivalent to minizing

$$\|b - Ax\|_2^2 = \sum_{i=1}^{m} |r_i|^2$$

where x is the vector of coefficients of the best polynomial fit. (Therefore, this problem has always a solution.)

How about uniqueness?

The matrix

$$A_{ij} = x_i^j, \qquad \begin{array}{ll} i = 1, \ldots, m & \text{(number of data)} \\ j = 0, \ldots, n-1 & \text{(polynomial degree)} \end{array}$$

has full rank by columns if and only if:

there are (at least) n distinct points x_i
(which implies that $m \geq n$)

You might want to Google about Vandermonde matrices to understand this statement.