1 29th of November 2018 — F. Poloni

In this lecture we address the problem of designing methods that use Krylov spaces to solve linear systems.

We need alternative methods to Gaussian elimination because matrices are too large.

1.0.1 Naive idea

Find first a "good" search subspace, then look for best approximation of the solution of $Ax = b$ in that space.

Let us assume that our space is the image of a matrix V, $Im(V)$. $\forall\ x \in Im(V)$ such x may be written as $x = V^1 y_1 + V^2 y_2 + \cdots + V^n y_n$, where V^i are the columns of the matrix V.

The idea is to find a vector y that satisfies $\min_{y\in\mathbb{R}} \|AVy - b\|$, which is a least squares problem.

1.0.2 Improvement

A good search space is $Im(V) = K_n(A, b) = \text{span}(b, Ab, A^2b, \ldots, A^{n-1}b)$, where $K_n(A, b)$ is the Krylov space.

The issue here is that $V = \begin{pmatrix} b & Ab & \cdots & A^{n-1}b \end{pmatrix}$ is a bad basis, because it is still conditioned.

Working with that V is problematic: its columns "tend" to be aligned with the dominant eigenvector and this means that V is close to a rank 1 matrix.

We need a better basis for $K_n(A, b)$, in particular an orthogonal basis.

Do you recall?

An orthogonal basis is a basis in which each couple of vectors are orthogonal.

1.1 Arnoldi algorithm

The idea behind this algorithm is to build an orthogonal basis of $K_n(A, b)$ incrementally.

The algorithm at a generic step, takes an orthogonal basis for $K_n(A, b)$ and adds a vector to produce one of $K_{n+1}(A, b)$.

Let us assume that we start with $\{q_1, q_2, \ldots, q_n\}$, orthogonal basis of $K_n(A, b)$.

We also assume that $q_n = \alpha_0 b + \alpha_1 Ab + \cdots + \alpha_{n-1}A^{n-1}b = p(A)b$, where we impose $\alpha_{n-1} \neq 0$.

$p(A) = \alpha_0 I + \alpha_1 A + \cdots + \alpha_{n-1}A^{n-1}$ has degree exactly 1.

Step 1: $K_1(A, b) = span(b)$, $q_1 = \frac{q_1}{\|q_1\|}$

Generic step:

1. produce a vector $K_{n+1}(A,b) - K_n(A,b) : w = Aq_n$
2. $w \in K_{n+1}(A,b) = 0$, $w = q_1\beta_1 + q_1\beta_2 + \cdots + q_n\beta_n + q_{n+1}\beta_{n+1}$, where $q_1, q_2, \ldots, q_n, q_{n+1}$ is an orthogonal basis of $K_{n+1}(A,b)$. In this context $q_1, q_2, \ldots, q_n$ are know, while q_{n+1} is still to be determined.
   Notice that $\forall i = 1, \ldots, n$ holds the following ${q_i}^T w = {q_i}^T q_1\beta_1 + \cdots + {q_i}^T q_n\beta_n + {q_i}^T q_{n+1}\beta_{n+1} = {q_i}^T q_i\beta_i = \beta_i$, because of the orthonormality of the basis. Now we can compute $q_{n+1}\beta_{n+1} = w - q_1\beta_1 - q_2\beta_2 + \cdots + q_n\beta_n = z$. If we choose $\beta_{n+1} = \|z\|$ we have that $q_{n+1} = \frac{z}{\|z\|}$ and this produces a valid choice of q_{n+1}. We still need to prove that it has norm 1 and it's orthogonal to all the other vectors in the basis.

An implementation of this algorithm is shown in Algorithm 1.1.

ALGORITHM 1.1 Arnoldi algorithm Matlab implementation.

```
function Q = arnoldi(A, b, n)
Q = zeros(length(b), n); % will be filled in
H = zeros(n+1, m);
Q(:, 1) = b / norm(b);
for j = 1 : n
  w = A * Q(:, j);
  for i = 1:j
  % not what we showed earlier here, but stabler
    betai = Q(:, i)' * w;
    w = w - betai * Q(:, i);
    H(i, j) = betai;
  end
  nrm = norm(w);
  H(j+1, j) = nrm;
  Q(:, j+1) = w / nrm;
end
```

Notice that we presented an algorithm where $\beta_i = {q_i}^T w$ for $i = 1, \ldots, n$, then $w \leftarrow w - \beta_1 q_1 - \beta_2 q_2 - \cdots - \beta_n q_n$.

In the implementation we compute $\beta_i = {q_1}^T w$, $w \leftarrow w - \beta_1 q_1$, $\beta_2 = {q_2}^T w$, $w \leftarrow w - \beta_2 q_2$. Why? It is more stable.

At step j:

$$Aq_j = \beta_{1,j}q_1 + \beta_{2,j}q_2 + \cdots + \beta_{j,j}q_j + \beta_{j+1,j}q_{j+1} = Q \begin{bmatrix} \beta_{1,j} \\ \beta_{2,j} \\ \vdots \\ \beta_{j+1,j} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n & q_{n+1} \\ & & & & \end{bmatrix}$$

$$AQ = A \begin{bmatrix} q_1 & q_2 & \cdots & q_n & q_n \\ & & & & \end{bmatrix} = Q \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \cdots & \beta_{1,n} \\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3} & \cdots & \beta_{2,n} \\ 0 & \beta_{3,2} & \beta_{3,3} & \cdots & \beta_{3,n} \\ 0 & 0 & \beta_{4,3} & \cdots & \beta_{4,n} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \beta_{n+1,n} \end{bmatrix}$$

Hence the values q_i, $\beta_{i,j}$ computed by Arnoldi satisfy $AQ_n = Q_{n+1}H$, where $H \in M(n+1, n, \mathbb{R})$ and looks like a triangular matrix plus a diagonal below, namely $H = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \beta_{1,3} & \cdots & \beta_{1,n} \\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3} & \cdots & \beta_{2,n} \\ 0 & \beta_{3,2} & \beta_{3,3} & \cdots & \beta_{3,n} \\ 0 & 0 & \beta_{4,3} & \cdots & \beta_{4,n} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \beta_{n+1,n} \end{bmatrix}$

Notation: $Q_{n+1} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n & q_{n+1} \\ & & & & \end{bmatrix}$, $Q_n = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \\ & & & \end{bmatrix}$

such that $Q_n \in M(m, n, \mathbb{R})$, while $Q_{n+1} \in M(m, n+1, \mathbb{R})$, $A \in M(m, \mathbb{R})$ and $b \in M(m, n)$.

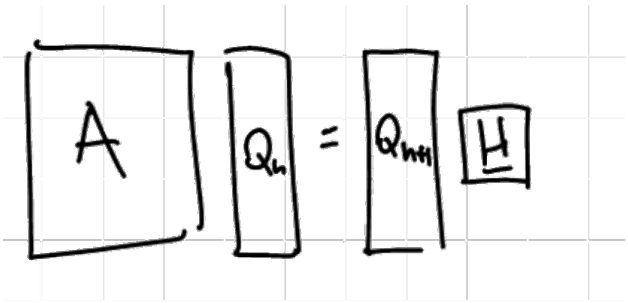

FIGURE 1.1: We started form a matirx A which has many entries and it gets factorized by the product of two smaller matrices.

For every matrix A there exist an Arnoldi factorization.

We don't like that we multiply A by two different matrices on the left and on the right, but we may improve the algorithm by using the same matrix.

$AQ_n = (Q_n|q_{n+1}) \cdot matriToDraw = Q_nH + q_{n+1} \cdot (0, \ldots, 0, *) = Q_nH + q_{n+1} \cdot tre_{n+1}\beta_{j+1,j}$, where $e_i = 0 \ldots 010 \ldots 0$.

Last remark: $AQ_n = Q_{n+1}H$ doesn't allow a factorization of the matrix A, because Q_n isn't invertible, because Q_n is tall, thin and it doesn't have an inverse.

Once we see how Arnoldi works, we would like to see how to use it.

At some point this assumption must become false. For instance, assume we arrive at step

$m = n$: $q_1, q_2, \ldots, q_m$ are a basis of $\mathbb{R}^m$, so:

$$Aq_m = \beta_1 q_1 + \cdots + \beta_m q_m + 0$$

(without an additional term $\beta_{m+1} q_{m+1}$)

Arnoldi factorization for $m = n$ (assuming nothing broke down before):

$AQ_m = Q_m H$ is a factorization into square matrices, formally $Q_m, H, A \in M(m, \mathbb{R})$, so we can write something that we couldn't write before:

$A = A_m H {Q_m}^T$ we recall that H is upper triangular plus another diagonal before.

Definition 1.1. *Let $H \in M(m, \mathbb{R})$ such that $H_{ij} = 0 \forall i > j + 1$. H is called* ***Hessemberg matrix****.*

Fact 1.1. *The QR factorization of an Hessemberg matrix $H \in M(m, \mathbb{R})$ can be computed in $O(m^2)$ operations.*

This approach may be used, but in practice, since A is large and sparse we don't want to go until the end.

Let us analyze what happens if at some point $K_{n+1}(A, b) = K_n(A, b)$?

$$Aq_n = \beta_1 q_1 + \cdots + \beta_n q_n + 0$$

for instance, if b is an eigenvector of A, it happens already at $n = 1$.

In the implementation of the Algorithm 1.1 if we have a breakdown at step $n + 1$, this means that $w = A_q n$ has already enough q_is, so $q_{n+1}\beta_{n+1} = 0$.

Problem: $q_{n+1} = \frac{z}{\|z\|}$ division by 0. We need to change the definition $\beta_{n+1} = \|z\| = 0$. At this point we don't get a basis of the Krylov space, but we can still go on as "nothing happened", as long as these vectors are orthonormal. We go on until the end we get:

$$AQ_m = H_m Q_m$$

$$H_m = \left[\begin{array}{cccc|cccc} * & \cdots & * & * & * & \cdots & \cdots & * \\ * & \cdots & * & * & * & \cdots & \cdots & * \\ 0 & \ddots & * & * & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & * & * & * & \cdots & \cdots & * \\ \hline & & & 0 & * & \cdots & * & * \\ & & & & * & \cdots & * & * \\ & & & & 0 & \ddots & * & * \\ & & & & 0 & 0 & * & * \end{array}\right]$$

The blocks are square.

This is good news, because it allows us to make a lot of manipulations.

$$A = Q_m H_m Q_m^T = \begin{bmatrix} Q_n & \widehat{Q} \end{bmatrix} \begin{bmatrix} H_n & L \\ 0 & M \end{bmatrix} \begin{bmatrix} Q_n & \widehat{Q} \end{bmatrix}^T$$

Do you recall?

We said two things about block triangular matrices:

1. the eigenvalues of the matrix are the eigenvalues of the diagonal blocks, more formally, $eigh(A) = eigh(H) = eigh(H_n) \cup eigh(M)$;
2. we can solve linear systems on block matrices more easily. In this case:

$$\begin{aligned} x = A^{-1}b = Q_m H^{-1} {Q_m}^T b = Q_m H^{-1} \begin{bmatrix} {q_1}^T \\ {q_2}^T \\ \vdots \\ {q_m}^T \end{bmatrix} b \overset{(1)}{=} Q_m H^{-1} \begin{bmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ = \begin{bmatrix} Q_n & \widehat{Q} \end{bmatrix} \begin{bmatrix} {H_n}^{-1} & -{H_n}^{-1} L M^{-1} \\ 0 & M^{-1} \end{bmatrix} \begin{bmatrix} \|b\| \, \mathbf{e}_1 \\ 0 \end{bmatrix} \\ = \begin{bmatrix} Q_n & \widehat{Q} \end{bmatrix} \begin{bmatrix} \|b\| \, H_n^{-1} \mathbf{e}_1 \\ 0 \end{bmatrix} = Q_n \, \|b\| \, H_n^{-1} \mathbf{e}_1 \end{aligned} \tag{1.1}$$

where $\overset{(1)}{=}$ follows from the fact that $b = \|b\| \, q_1$ and q_1 is orthogonal to all the other q_i.

An attentive reader may notice that at step j, when encountering $B_{j+1,j} = 0$, we know the following:

- some eigenvalues of A (those of H_n);
- given an eigencouple v, λ such that $H_n v = \lambda v$ then $Q_n v$ is an eigenvector of A with eigenvalue λ;
- the solution $x = Q_n {H_n}^{-1} \|b\| \, e_1$ of $Ax = b$ and this is called lucky breakdown.

The point is that we have what we need to compute the solution at last step before the breakdown.

Theorem 1.2 ("Lucky breakdown")**.** *If it happens at an early step, we can solve linear systems (or compute some eigenvalues) cheaply: costs n matrix-vector products + $O(mn^2)$.*

What happens when there is no breakdown? After n steps of Arnoldi:

1. if $H_n v = \lambda v$ is an eigenpair of H_n. $Q_n v$, λ is an approximation of an eigenpair of A;
2. $\widetilde{x} = Q_n {H_n}^{-1} \|b\| \, e_1$ is an approximantion of the solution x of $Ax = b$.