1 EFFICIENT GMRES

Recall the prototype-GMRES method.

Given \( A, b \) where we can only multiply by \( A \).
for \( i = 1 \) to maxiter
  Update the Arnoldi factorization \( Q_k, H_k \).
  Solve for \( z_k \) by minimizing \( \| H_k z_k - b \| e_1 \| \),
  i.e. \( z_k = \arg \min \| H_k z_k - b \| e_1 \| \).
  Let \( x_k = Q_k z_k \).
  Check \( |Ax_k - b| \).

To implement this, we need to solve a least squares problem at each step. This takes \( O(k^2) \) work because it's a Hessenberg matrix. Then we need to construct the solution and check the residual. These take \( O(nk) \) and another matrix-vector product. We can do all of these steps more efficiently!

Here is the outline for the essential idea to optimize GMRES.

we only need to check the residual at each step, and do not need to compute \( x_k \).

So the method we'll look at optimizing is:

Given \( A, b \) where we can only multiply by \( A \).
for \( i = 1 \) to maxiter
  Update the Arnoldi factorization \( Q_k, H_k \).
  Compute \( r_k \) where
    \( r_k = Ax_k - b \)
    \( x_k = Q_k z_k \)
  \( z_k = \arg \min \| H_k z_k - b \| e_1 \| \)
  and stop once \( |r_k| \) is sufficiently small, i.e.
  Update \( |r_k| \rightarrow |r_{k+1}| \) and stop if it's small enough.
Explicitly compute \( z_k = \arg \min \| H_k z_k - b \| e_1 \| \)
  and return \( x_k = Q_k z_k \) only at the end of the iteration

1.1 THE OPTIMIZATION IDEA

Let's study the quantity we want to compute, let \( |b| = \beta_0 \), then
\[
|r_k| = |b - Ax_k| = |b - AQ_k y_k| = |H_k y_k - \beta_0 e_1|.
\]
After a four steps, this is:
\[
\begin{bmatrix}
  x \\
  x \\
  x \\
  x \\
  0 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\begin{bmatrix}
y_4 - \beta_0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}.
\]
We solve least squares problems via QR, so suppose that
\[
H_k = U_k R_k
\]
is the QR factorization after \( k \)-steps. Then
\[
|r_k| = |U_k R_k y_k - \beta_0 e_1| = |R_k y_k - \beta_0 U_k^T e_1|.
\]
Showing this after a few steps gives us the idea more clearly:
\[
|r_k| = |U_k R_k y_k - \beta_0 e_1| = |R_k y_k - \beta_0 U_k^T e_1| = \beta_0 y_5.
\]
(Remember we solve for \( y_k \) such that this term is zero in the first four components. So we just need to figure out what \( y_5 \) is to get \( |r_k| \).)
1.2 TAKING IT DEEPER

We need to note a two things here to continue our optimization:

1. We only need Givens rotations to get \( H_k \rightarrow R_k \).

2. We only need one rotation to update \( R_k \rightarrow R_{k+1} \). (Woah!)

Let’s review step 1 and see how that will help us with step 2.

\[
\begin{bmatrix}
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

Now, suppose we have \( U_4 \) and \( R_4 \), how do we get \( U_5, R_5 \)?

\[
H_5 = \begin{bmatrix} H_4 & h_{6,5} \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
0 & 0 & 0 & 0 \\
h_{6,5} & & & 
\end{bmatrix}
\]

If we rotate by \( U_4^T \), we get:

\[
\begin{bmatrix} U_4^T & 0 \\
0 & 1 \end{bmatrix} H_5 = \begin{bmatrix} U_4^T & 0 \\
0 & 1 \end{bmatrix} = \begin{bmatrix} R_4 & U_4^T z_4 \\
0 & h_{6,5} \end{bmatrix}
\]

So at this point, we just have the one Givens rotation: \( J \) that we need to do to fix up the element \( h_{6,5} \) and so \( U_5 = J_5 J_4 \ldots J_1 \), which is just one update.

1.3 SEEKING GAMMA.

Note that the elements of gamma are just the first column of \( U_k^T \). Let \( g_k = U_k^T e_1 = \begin{bmatrix} y_1 & y_2 & \ldots & y_k \end{bmatrix}^T \). Then by our previous relationship:

\[
g_{k+1} = U_{k+1}^T e_1 = J_{k+1} U_k^T e_1 = J_{k+1} g_k.
\]

But this is weird, because \( J \) is a \( k + 1 \times k + 1 \) matrix and \( g_k \) is a length \( k \) vector. So what we really mean is

\[
J_{k+1} \begin{bmatrix} g_k \\
0 \end{bmatrix}
\]

where we grew the vector by one element in order to make it work. Note that we don’t need to actually update \( g_k \) even though it should change.

1.4 THE WHOLE ALGORITHM

\[
g = \beta_k e_1;
\]

for \( k = 1 \) to \( \ldots 

Update \( Q_k, H_k \)

Let \( \eta_{k+1} = H_{k+1,k} \).

Let \( z_k = H_{1:k,k} \).

Apply \( J_1 \ldots J_{k-1} \) to \( z_k \), and update \( H \)

Create \( J_k \) to eliminate \( \eta_{k+1} \).

Determine \( g_k \) from \( J_k g_{k-1} \) growing by zeros as needed.

If \( g_k \) (end) is small enough, then stop iterating.

At this point, \( H \) has the factor \( R \), and (if we do keep \( g \) accurate), then \( g \) is the right hand side, so we can just solve \( R_k y_k = g_k \) and then output \( x = Q_k y \).

2 GMRES VS. FOM

See notes.

The major point is that FOM is like CG in that it solves a linear system based on the truncated Arnoldi factorization.

The large scale study by Peter Brown (http://dx.doi.org/10.1137/0912003) concluded: there is little difference, but liked the minimum residual property of GMRES.

This section is incomplete.