MATH 612
Computational methods for equation solving and function minimization – Week # 10

F.J.S.

Spring 2014 – University of Delaware

Plan for this week

- Discuss any problems you couldn't solve from previous lectures
- We will cover Lectures 35, 36, and 38, and be done with Linear Algebra
- Coding assignment #3 is due next Monday
- Once again, start thinking about your final coding assignment

Remember that...

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

THE ROAD TO GMRES

Review (1): Krylov and Arnoldi

Let $b \in \mathbb{C}^m$ and $A \in \mathbb{C}^{m\times m}$. We consider the subspaces

$$\mathcal{K}_n := \langle b, Ab, A^2 b, \ldots, A^{n-1} b\rangle \qquad n \geq 1.$$

We care about these spaces as long as their dimensions are equal to n. After that all of them are the same.

Arnoldi's method is the modified Gram-Schmidt method applied to find an orthonormal basis of the Krylov space $\mathcal{K}_n$.

$$\langle b, Ab, A^2 b, \ldots, A^{j-1} b\rangle = \langle q_1, \ldots, q_j\rangle$$

We do not compute the QR decomposition of the Krylov matrix

$$K_n = \left[\begin{array}{c|c|c|c} b & Ab & \ldots & A^{n-1}b \end{array}\right].$$

Still we keep score of all the computations in the GS process (the entries of R, except for the first one $\|b\|$, which will magically reappear later).

Review (2): Krylov and Arnoldi

```
$q_1 = \frac{1}{\|b\|_2} b$        % first step apart
for $j \geq 1$        % count on the next
    $v = Aq_j$        % the newcomer
    for $i = 1 : j$
        $h_{ij} = q_i^* v$        % not R anymore
        $v = v - h_{ij} q_i$
    end
    $h_{j+1,j} = \|v\|_2$        % stop if zero
    $q_{j+1} = \frac{1}{h_{j+1,j}} v$
end
```

$$Aq_j = \underbrace{h_{1j}q_1 + h_{2j}q_2 + \ldots + h_{jj}q_j}_{\text{orth. proj. onto } \mathcal{K}_j} + h_{j+1,j}q_{j+1}$$

Review (3): Krylov and Arnoldi

$$Aq_j = h_{1j}q_1 + h_{2j}q_2 + \ldots + h_{jj}q_j + h_{j+1,j}q_{j+1}, \qquad j = 1, \ldots, n$$

$$A \underbrace{\left[\begin{array}{c|c|c|c} q_1 & q_2 & \ldots & q_n \end{array}\right]}_{Q_n}$$

$$= \underbrace{\left[\begin{array}{c|c|c|c|c} q_1 & q_2 & \ldots & q_n & q_{n+1} \end{array}\right]}_{Q_{n+1}} \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \ldots & h_{2n} \\ & h_{32} & h_{33} & \ddots & h_{3n} \\ & & \ddots & \ddots & \vdots \\ & & & h_{n,n-1} & h_{nn} \\ & & & & h_{n+1,n} \end{bmatrix}}_{\widetilde{H}^n}$$

Review (4): Krylov and Arnoldi

$$AQ_n = Q_{n+1}\widetilde{H}_n$$

- A is A
- Q_n contains an orthonormal basis of $\mathcal{K}_n$
- Q_{n+1} contains one more vector, making for an orthonormal basis of $\mathcal{K}_{n+1}$
- $\widetilde{H}$ is $(n+1) \times n$ Hessenberg

And now a computation and a definition:

$$Q_n^* A Q_n = Q_n^* Q_{n+1} \widetilde{H}_n = H_n$$

where H_n is $n \times n$ Hessenberg, obtained by removing the last row of $\widetilde{H}_n$. Its eigenvalues are called Ritz values of A. They are not eigenvalues of A, but they approximate them.

An optimization problem

Assume that we have x_0, a first guess for the solution of $Ax = b$. We can think of a modified problem

$$Ae = r, \qquad r = b - Ax_0, \qquad e = x - x_0.$$

We are going to try and find e. For the same price, forget about x_0 (the book does –on purpose–) and think that $x_0 = 0$.
A sort of frozen step of GMRES consists of minimizing

$$\|b - Ax\|_2 \qquad \text{with } x \in \mathcal{K}_n$$

A sequence of equivalent problems

- Minimize $\|b - Ax\|_2$ with $x \in \mathcal{K}_n$
- Minimize $\|AK_n c - b\|_2$ with $c \in \mathbb{C}_n$
- Minimize $\|AQ_n y - b\|_2$ with $y \in \mathbb{C}_n$
- Minimize $\|Q_{n+1}\widetilde{H}_n y - b\|_2$ with $y \in \mathbb{C}_n$
- Minimize $\|\widetilde{H}_n y - Q^*_{n+1} b\|_2$ with $y \in \mathbb{C}_n$
- Minimize $\|\widetilde{H}_n y - \|b\|_2 e_1\|_2$ with $y \in \mathbb{C}_n$.

If we can find a QR decomposition of the rectangular $(n+1) \times n$ Hessenberg matrix $\widetilde{H}_n$, we are done. Why?

A skeleton of GMRES

```
$x_0 = 0$
Start with $b$ and compute $q_1 = (1/\|b\|_2)b$
for $j \geq 1$
    Compute the vector $q_{j+1}$ in Arnoldi's method
    Compute and store the $j$-th column of $\widetilde{H}_j$
    Compute a QR decomposition of $\widetilde{H}_j$
    Minimize $\|\widetilde{H}_j y_j - \|b\|_2 e_1\|_2$ with $y_j \in \mathbb{C}_j$
    Compute $x_j = Q_j y_j \approx x$
    Decide if $x_j$ is close enough to $x_{j-1}$. Stop if it is.
end
```

Small variation. Start with $x_0 \neq 0$ and substitute b by $r_0 = b - Ax_0$. In this case $x_j \in x_0 + \mathcal{K}_j$, so what's in the Krylov space is the error correction $x_j - x_0$ and not x_j itself. And the Krylov space is triggered by r_0, not b.

GIVENS AND THE MISSING LINK

Givens rotations: facts

- Givens' method is similar to the Householder method but it performs rotations instead of reflections to make zeros. The resulting orthogonal matrices (for the simple steps) are not symmetric.
- A Givens rotation $G_{ij}(\theta)$ locates the block

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

  in the $(i,j) \times (i,j)$ submatrix. Other than that it is an identity. $G_{ij}(-\theta) = G_{ij}(\theta)^{-1}$.
- Multiplying $G_{ij}(\theta)A$ involves only the i and j rows of A. It's quite cheap therefore.
- Givens rotations can be applied to triangularize a matrix: $(n-1)+(n-2)+\ldots+1 = \frac{n(n-1)}{2}$ rotations are needed.

Givens rotations for Hessenberg matrices

If H is an $m \times n$ Hessenberg matrix (many zero rows at the end have been added to customize the rotation matrices size), then

$$G_{n,n-1}(\theta_{n-1}) \dots G_{32}(\theta_2) G_{21}(\theta_1) H = R$$

is upper triangular.
To minimize $\|Hx - b\|$, solve

$$Rx = G_{n,n-1}(\theta_{n-1}) \dots G_{32}(\theta_2) G_{21}(\theta_1) b = b_n.$$

Progressive Hessenberg matrices

We have worked with a Hessenberg $m \times n$ matrix $\widetilde{H}_n$, and a fixed right hand side $k\, e_1$. (Note that we have artificially added zeros to the end of the matrix and of the RHS. In GMRES they were shorter. The minimization problem is the same. Why?) We want to compute the next one, that is, how much of what we used can we re-use? Before...

$$G_{n,n-1}(\theta_{n-1}) \dots G_{32}(\theta_2) G_{21}(\theta_1) \widetilde{H}_n = R_n$$

... and after...

$$G_{n,n}(\theta_n) G_{n,n-1}(\theta_{n-1}) \dots G_{32}(\theta_2) G_{21}(\theta_1) \widetilde{H}_{n+1} = R_{n+1}$$

so modify the RHS in the same way

$$G_{n,n}(\theta_n) G_{n,n-1}(\theta_{n-1}) \dots G_{32}(\theta_2) G_{21}(\theta_1) b = G_{n,n}(\theta_n) b_n.$$

Final words and we are done

In GMRES/Arnoldi/Givens, in one step...

- We compute q_{n+1} and the $(n+1)$-th column of $\widetilde{H}_{n+1}$
- We apply all past rotations to the new column (we couldn't do it before, because we didn't know it)
- We find the new rotation to make $\widetilde{H}_{n+1}$ triangular. This adds one column to the right of upper triangular matrix. Nothing else is modified. Why?
- We apply the new rotation to the RHS
- We solve the new upper triangular system

TWO OBSERVATIONS

A frozen step of GMRES

$$\|Ax - b\|_2 = \text{minimum}, \qquad x = K_n c, \qquad c \in \mathbb{C}_n.$$

Once again, the Krylov matrix is implicit to the Arnoldi process even if never explicitly produced

$$K_n = \left[\begin{array}{c|c|c|c} b & Ab & \dots & A^{n-1}b \end{array} \right]$$

$$\begin{aligned}
K_n c &= c_1 b + c_2 Ab + \ldots + c_n A^{n-1} b \\
AK_n c &= c_1 Ab + c_2 A^2 b + \ldots + c_n A^n b \\
b - AK_n c &= b - c_1 Ab - c_2 A^2 b - \ldots - c_n A^n b \\
&= (I - c_1 A - c_2 A^2 - \ldots - c_n A^n) b = p_n(A) b
\end{aligned}$$

where p_n is a polynomial of degree n with $p_n(0) = 1$.

An equivalent optimization problem

Find a polynomial

$$p_n(x) = 1 - c_1 x - \ldots - c_n x^n$$

making

$$\|p_n(A)b\|_2 \qquad \text{minimum.}$$

With those coefficients we compute $x_n = K_n c \approx x = A^{-1} b$
Believe it or not, this is how GMRES convergence is analyzed.

Warning. With exact arithmetic GMRES delivers the exact solution after n steps. Why? If we need to take n steps, GMRES requires storing the large Q_j matrices (basically nothing else, and A is only used as an operator). If we restart GMRES after k iterations (remember x_0?), then the method is called restarted GMRES or GMRES(k).

When to solve and stop

We had phrased the stopping criterion as

```
Minimize $\|\widetilde{H}_j y_j - \|b\|_2 e_1\|_2$ with $y_j \in \mathbb{C}_j$
Compute $x_j = Q_j y_j \approx x$
Decide if $x_j$ is close enough to $x_{j-1}$. Stop if it is.
```

What we do is

```
Compute a full QR decomposition $\widetilde{H}_j = \widehat{Q}_j \widetilde{R}_j$
Define $\xi = \widehat{Q}_j^*(\|b\|_2 e_1) \in \mathbb{C}^{j+1}$
Solve $\widetilde{R}_j y_j = \xi_j$ ignoring the last row
Then res:=$\|\widetilde{H}_j y_j - \|b\|_2 e_1\|_2 =$ abs of last comp. of $\xi_j$
If res < tol
    Compute $x_j = Q_j y_j$
    Stop
end
```

LANCZOS

Arnoldi becomes Lanczos (1): Arnoldi's iteration

```
...
for $j \geq 1$        % count on the next
    $v = Aq_j$        % the newcomer
    for $i = 1 : j$
        $h_{ij} = q_i^* v$        % not R anymore
        $v = v - h_{ij} q_i$
    end
    ...
end
```

$$Aq_j = h_{1j}q_1 + h_{2j}q_2 + \ldots + h_{jj}q_j + h_{j+1,j}q_{j+1}$$

We are going to focus now on the case of Hermitian matrices.

Arnoldi becomes Lanczos (2): the key property

Assume that $j \geq 3$ (we are trying to compute q_{j+1}). We create $v = Aq_j$. We then compute

$$h_{1j} = q_1^* A q_j = \underbrace{(Aq_1)^*}_{\in \mathcal{K}_2} \underbrace{q_j}_{\perp \mathcal{K}_{j-1}} = 0,$$

and we correct

$$v = v - h_{1j} q_1 = v = Aq_j.$$

Proceeding by induction

$$h_{ij} = 0 \qquad i < j - 1 \qquad \Longleftrightarrow \qquad i + 1 < j.$$

This means that the j column of the Hessenberg matrix $\widetilde{H}_n$ contains only three entries: $h_{j-1,j}, h_{j,j}, h_{j+1,j}$.

Arnoldi becomes Lanczos (3): first ideas

Let me repeat this. This means that the j column of the Hessenberg matrix $\widetilde{H}_n$ contains only three entries: $h_{j-1,j}, h_{j,j}, h_{j+1,j}$.

- Instead of a Hessenberg matrix we have a tridiagonal matrix.
- The loop $i = 1 : j$ can be reduced to $i = j - 1 : j$ or actually $i = \max\{1, j-1\} : j$. This fact reduces the complexity of the Arnoldi iteration in one entire loop.

But there's more... Remember how

$$Q_n^* A Q_n = Q_n^* Q_{n+1} \widetilde{H}_n = H_n$$

where H_n is like $\widetilde{H}_n$ without the last row? Can you see why H_n is Hermitian? This means that $h_{j-1,j} = \overline{h_{j,j-1}}$, and we do not even need to compute the first of the dot products in the internal loop.

Arnoldi becomes Lanczos (4): almost there

```
$q_1 = (1/\|b\|)b$
$h_{1,0} = 0$          % fictitious element
for $j \ge 1$          % count on the next
    $v = Aq_j$          %
    $v = v - \overline{h_{j,j-1}}q_{j-1}$
    $h_{jj} = q_j^* v$
    $v = v - h_{jj}q_j$
    $h_{j+1,j} = \|v\|$
    $q_{j+1} = (1/h_{j+1,j})v$
end
```

The **red lines** correspond to the old loop. Only two substractions/projections are needed. For one of them we already know the coefficient.

Arnoldi becomes Lanczos (5): we got it

We rename $\alpha_j = h_{jj}$, $\beta_j = h_{j+1,j}$

```
$q_1 = (1/\|b\|)b$
$\beta_0 = 0$
for $j \geq 1$
    $v = Aq_j$  %
    $v = v - \overline{\beta_{j-1}} q_{j-1}$
    $\alpha_j = q_j^* v$
    $v = v - \alpha_j q_j$
    $\beta_j = \|v\|$
    $q_{j+1} = (1/\beta_j)v$
end
```

```
$q_1 = (1/\|b\|)b$
$\beta_0 = 0$
for $j \geq 1$
    $v = Aq_j$
    $\alpha_j = q_j^* v$
    $v = v - \overline{\beta_{j-1}} q_{j-1} - \alpha_j q_j$
    $\beta_j = \|v\|$
    $q_{j+1} = (1/\beta_j)v$
end
```

$$Aq_j = \beta_j q_{j+1} + \alpha_j q_j + \overline{\beta_{j-1}} q_{j-1}$$

The associated decomposition(s)

$$AQ_n = Q_{n+1} \begin{bmatrix} \alpha_1 & \overline{\beta_1} & & & \\ \beta_1 & \alpha_2 & \overline{\beta_2} & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \overline{\beta_{n-1}} \\ & & & \beta_{n-1} & \alpha_n \\ & & & \ddots & \beta_n \end{bmatrix} = Q_{n+1} \widetilde{T}_n$$

$$Q_n^* A Q_n = \begin{bmatrix} \alpha_1 & \overline{\beta_1} & & & \\ \beta_1 & \alpha_2 & \overline{\beta_2} & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \overline{\beta_{n-1}} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix} = T_n$$

A first possible use of Lanczos

Assume that b is such that $\mathcal{K}_n$ grow as far as $\mathcal{K}_m = \mathbb{C}^m$. Then we have a unitary matrix Q_m such that

$$Q_m^* A Q_m = T_m$$

where T_m is tridiagonal. We then use the QR method (or other eigenvalue methods) to diagonalize T_m, which is a much cheaper matrix to work with that A.

A FAST VIEW OF CG

GMRES vs CG

GMRES (Generalized Minimal RESidual) minizes the norm of the residual in the Krylov space

$$\|b - Ax_n\|_2 = \text{minimum}, \qquad x_n \in \mathcal{K}_n.$$

In CG (Conjugate Gradient), we use a different norm

$$\|x\|_A = (x^T A x)^{1/2}$$

(Today we'll only do real symmetric matrices), and we minimize the $A-$norm of the error

$$\|x_* - x_n\|_A = \text{minimum}, \qquad x_n \in \mathcal{K}_n$$

where x_* is the unknown solution of the system $Ax_* = b$.

A simple computation

$$\begin{aligned}\tfrac{1}{2}\|x - x_*\|_A^2 &= \tfrac{1}{2}(x - x_*)^T A(x - x_*)\\ &= \tfrac{1}{2}x^T Ax - x^T Ax_* + \tfrac{1}{2}x_* Ax_*\\ &= \tfrac{1}{2}x^T Ax - x^T b + \text{constant}\end{aligned}$$

This shows that, up to a constant that is not known but it's not relevant either, we are minimizing the functional

$$\tfrac{1}{2}x^T Ax - x^T b$$

in the Krylov space $\mathcal{K}_n$. Next week we'll start with optimization. We'll see how this is a convex optimization problem. The Steepest Descent method will be the result of applying a general optimization method for this quadratic minimization problem.

Four sequences of vectors

- Approximate solutions x_n
- Residuals $r_n = b - Ax_n$
- Errors (not computable, but being minimized) $e_n = x_* - x_n = A^{-1} r_n$
- Descent directions p_n (these ones are new)

In CG, by definition

$$\|e_n\|_A \leq \|e_{n-1}\|_A \qquad \forall n.$$

This is because the *n*-th step of the minimization problem is carried out in a bigger subspace $\mathcal{K}_{n-1} \subset \mathcal{K}_n$.
The **descent directions** p_n constitute an orthogonal (but not orthonormal) basis of $\mathcal{K}_n$ w.r.t. the inner product defined by A.
They will satisfy

$$p_j^T A p_i = 0 \qquad i \neq j.$$

They are called **conjugate** directions.

The CG algorithm (α_n is shifted w.r.t. the book)

CG hides a Lanczos iteration. You can find the derivation in the wikipedia article (that derivation is not easy) or prove that the next algorithm is equivalent to our claims (see Chapter 38 of the book). We'll do a simpler argument at the end. This is one step of the method

$$\begin{aligned}
&\alpha_{n-1} = (r_{n-1}^\top r_{n-1})/(p_{n-1}^\top A p_{n-1}) \\
&x_n = x_{n-1} + \alpha_{n-1} p_{n-1} && \text{\% advance} \\
&r_n = b - A x_n = r_{n-1} - \alpha_{n-1} A p_{n-1} && \text{\% res. update} \\
&\beta_n = (r_n^\top r_n)/(r_{n-1}^\top r_{n-1}) \\
&p_n = r_n + \beta_n p_{n-1} && \text{\% next descent direction}
\end{aligned}$$

There's no need to store α_n and β_n. One matrix-vector multiplication by iteration. Two inner products by iteration.

General error estimate

Theorem

If A is symmetric and positive definite, $Ax_* = b$, then for the CG method

$$\|x_* - x_n\|_A \le 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n \|x_* - x_0\|_A$$

where κ is the condition number of A.

The proof of this result is not specially difficult. We won't do it though.

The bad news is that for badly conditioned matrices the expected convergence rate is not very good.

PRECONDITIONED CG

The idea

If we can find a matrix M that is easy to invert (this means, solving $My = z$ is cheap) and $M \approx A$ in the sense that $M^{-1}A$ is well conditioned, then the system

$$M^{-1}Ax = M^{-1}b$$

might be faster to solve than $Ax = b$. HOWEVER, the matrix $M^{-1}A$ will not be symmetric positive definite. Here are our hypotheses:

- A is symmetric PD
- M is symmetric PD and easy to invert

We will now work with a decomposition $M = PP^\top$ for a matrix P that **we will not need to compute**.

The preconditioned system

Instead of considering the system

$$Ax = b$$

we consider the system

$$P^{-1}AP^{-T}y = P^{-1}b, \qquad x = P^{-T}y,$$

where

$$P^{-T} = (P^{-1})^T = (P^T)^{-1}.$$

Note that

$$B = P^{-1}AP^{-T}$$

is symmetric PD.

Let me insist, we will get rid of P at the end.

A step of CG for the preconditioned system

Version 1:

$$\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1})/(\widehat{p}_{n-1}^T B \widehat{p}_{n-1}) \\
&y_n = y_{n-1} + \alpha_{n-1} \widehat{p}_{n-1} \\
&\widehat{r}_n = \widehat{r}_{n-1} - \alpha_{n-1} B \widehat{p}_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n)/(\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&\widehat{p}_n = \widehat{r}_n + \beta_n \widehat{p}_{n-1}
\end{aligned}$$

Version 2: computing also iterates for original system

$$\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1})/(\widehat{p}_{n-1}^T P^{-1} A P^{-T} \widehat{p}_{n-1}) \\
&y_n = y_{n-1} + \alpha_{n-1} \widehat{p}_{n-1} \\
&x_n = L^{-T} y_n \\
&\widehat{r}_n = \widehat{r}_{n-1} - \alpha_{n-1} P^{-1} A P^{-T} \widehat{p}_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n)/(\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&\widehat{p}_n = \widehat{r}_n + \beta_n \widehat{p}_{n-1}
\end{aligned}$$

A step of CG for the preconditioned system (2)

$$\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1}) / ((P^{-T}\widehat{p}_{n-1})^T A P^{-T}\widehat{p}_{n-1}) \\
&y_n = y_{n-1} + \alpha_{n-1}\widehat{p}_{n-1} \\
&x_n = L^{-T} y_n \\
&\widehat{r}_n = \widehat{r}_{n-1} - \alpha_{n-1} P^{-1} A P^{-T}\widehat{p}_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n) / (\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&\widehat{p}_n = \widehat{r}_n + \beta_n \widehat{p}_{n-1}
\end{aligned}$$

We then define $p_n = P^{-T}\widehat{p}_n$ and substitute

$$\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1}) / (p_{n-1}{}^T A p_{n-1}) \\
&y_n = y_{n-1} + \alpha_{n-1}\widehat{p}_{n-1} \\
&\widehat{r}_n = \widehat{r}_{n-1} - \alpha_{n-1} P^{-1} A p_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n) / (\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&\widehat{p}_n = \widehat{r}_n + \beta_n \widehat{p}_{n-1}
\end{aligned}$$

A step of CG for the preconditioned system (3)

$$
\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1})/(p_{n-1}^T A p_{n-1}) \\
&y_n = y_{n-1} + \alpha_{n-1}\widehat{p}_{n-1} \\
&\widehat{r}_n = \widehat{r}_{n-1} - \alpha_{n-1} P^{-1} A p_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n)/(\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&\widehat{p}_n = \widehat{r}_n + \beta_n \widehat{p}_{n-1}
\end{aligned}
$$

Note that

$$r_n = b - Ax_n = P(P^{-1}b - P^{-1}AP^{-T}y_n) = P\widehat{r}_n.$$

and multiply the equations....

$$
\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T \widehat{r}_{n-1})/(p_{n-1}^T A p_{n-1}) \\
&P^{-T} y_n = P^{-T} y_{n-1} + \alpha_{n-1} P^{-T}\widehat{p}_{n-1} \\
&P\widehat{r}_n = P\widehat{r}_{n-1} - \alpha_{n-1} A p_{n-1} \\
&\beta_n = (\widehat{r}_n^T \widehat{r}_n)/(\widehat{r}_{n-1}^T \widehat{r}_{n-1}) \\
&P^{-T}\widehat{p}_n = P^{-T} P^{-1} P\widehat{r}_n + \beta_n P^{-T}\widehat{p}_{n-1}
\end{aligned}
$$

A step of CG for the preconditioned system (4)

We substitute again $p_n = P^{-T}\widehat{p}_n$, $x_n = P^{-T}y_n$, $P\widehat{r}_n = r_n$

$$
\begin{aligned}
&\alpha_{n-1} = (\widehat{r}_{n-1}^T\widehat{r}_{n-1})/(p_{n-1}^T A p_{n-1})\\
&x_n = x_{n-1} + \alpha_{n-1}p_{n-1}\\
&r_n = r_{n-1} - \alpha_{n-1}Ap_{n-1}\\
&\beta_n = (\widehat{r}_n^T\widehat{r}_n)/(\widehat{r}_{n-1}^T\widehat{r}_{n-1})\\
&\widehat{p}_n = M^{-1}r_n + \beta_n p_{n-1}
\end{aligned}
$$

... and finally note that

$$\widehat{r}_n^T\widehat{r}_n = (P^{-1}r_n)^T(P^{-1}r_n) = r_n^T P^{-T}P^{-1}r_n = r_n^T M^{-1} r_n$$

Final version (P has disappeared)

(In the first step $p_0 = P^{-T}\widehat{p}_0 = P^{-T}\widehat{r}_0 = P^{-T}P^{-1}r_0 = M^{-1}r_0$.)

```
$x_0$ is given
$r_0 = b - Ax_0$
$p_0 = M^{-1}r_0$
for $n \geq 1$
    $\alpha_{n-1} = (r_{n-1}^T M^{-1} r_{n-1})/(p_{n-1}^T A p_{n-1})$
    $x_n = x_{n-1} + \alpha_{n-1} p_{n-1}$
    $r_n = r_{n-1} - \alpha_{n-1} A p_{n-1}$
    $\beta_n = (r_n^T M^{-1} r_n)/(r_{n-1}^T M^{-1} r_{n-1})$
    $\widehat{p}_n = M^{-1} r_n + \beta_n p_{n-1}$
end
```

Coding trick: use an additional sequence of vectors $z_n = M^{-1}r_n$ to minimize preconditioning solves.

Final version (algorithm)

```
$x_0$ is given
$r_0 = b - Ax_0$
$z_0 = M^{-1} r_0$
$p_0 = z_0$
for $n \geq 1$
    $v = Ap_{n-1}$                          % Only matrix-vector multiplication
    $\alpha = (r_{n-1}^T z_{n-1})/(p_{n-1}^T v)$
    $x_n = x_{n-1} + \alpha p_{n-1}$
    $r_n = r_{n-1} - \alpha v$
    $z_n = M^{-1} r_n$                        % Solve a simple system
    $\beta = (r_n^T z_n)/(r_{n-1}^T z_{n-1})$
    $p_n = z_n + \beta p_{n-1}$
end
```

Final version (algorithm – stopping criterion missing)

Additional savings available: store $r_{n-1}^T z_{n-1}$ and do not keep r_{n-1}. Update variables all the time

```
$x_0$ is given
$r = b - Ax_0$
$z = M^{-1} r$ ;        $\xi_{\text{old}} = r^T z$
$p = z$
for $n \geq 1$
    $v = Ap$
    $\alpha = (r^T z)/(p^T v)$
    $x = x + \alpha p$
    $r = r - \alpha v$
    $z = M^{-1} r$;        $\xi_{\text{new}} = r^T z$
    $\beta = \xi_{\text{new}}/\xi_{\text{old}}$;        $\xi_{\text{old}} = \xi_{\text{new}}$
    $p = z + \beta p$
end
```