The Conjugate Gradient Method is simply a Galerkin Method

We wish to solve $Au - f = 0$ for u with $u, f \in H$ and $A : H \to H$ with $\dim(H) = n$

Ignoring the words "conjugate" and "gradient" for now, let's think about the iterative CG Method as finding

$$(1) \qquad u^{(k)} = \alpha_{k-1} p^{(k-1)} + \alpha_{k-2} p^{(k-2)} + ... + \alpha_0 p^{(0)}$$

where our only job is to find α's and p's .

Naturally this leads us to think of the Galerkin Method;
Find $u^{(k)} \in H_k \subseteq H$ where $\dim(H_k) = k \le n$ such that

$$(2) \qquad u^{(k)} = \sum_{i=0}^{k-1} \alpha_i p^{(i)} \quad \text{and}$$

$$(3) \qquad \left(Au^{(k)}, p^{(k)}\right)_H = \left(f, p^{(k)}\right)_H \text{ for every } p^{(k)} \in H_k$$

where $H_k = span\{p^{(0)}, p^{(1)}, ..., p^{(k-1)}\}$
Let's impose the requirement that the p's satisfy $\left(p^{(i)}, p^{(j)}\right)_A = 0$ for $i \neq j$ (the reason will become apparent shortly)
Substituting (2) into (3) yields

$$(4) \qquad \left(A\sum_{i=0}^{k-1} \alpha_i p^{(i)}, p^{(k)}\right)_H = \left(f, p^{(k)}\right)_H \text{ for every } p^{(k)} \in H_k$$

and since the p's are A-Conjugate, (4) becomes

$$(5) \qquad \alpha_i \left(Ap^{(i)}, p^{(i)}\right)_H = \left(f, p^{(i)}\right)_H \text{ for } 0 \le i \le k-1 \text{ and}$$

$$(6) \qquad \alpha_i = \frac{\left(f, p^{(i)}\right)_H}{\left(p^{(i)}, p^{(i)}\right)_A}$$

We found our $\alpha's$. Let's compare these α's to the α's we derived in class. In class, we used the fact $e^{(k+1)} \perp_A span\{p^{(k)}, p^{(k-1)}, ..., p^{(0)}\}$ and obtained the following:

Our algorithm from class is:

Set $u^{(0)}, r^{(0)} = f - Au^{(0)}, p^{(0)} = r^{(0)}$
Do $k = 0, 1, 2, ...$ until done

$$\alpha_k = \frac{\left(r^{(k)}, p^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A}$$
$$u^{(k+1)} = u^{(k)} + \alpha_k Ap^{(k)}$$
$$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}$$

$$p^{(k+1)} = Ap^{(k)} - \frac{\left(p^{(k)}, Ap^{(k)}\right)_A}{\left(p^{(k)}, p^{(k)}\right)_A} p^{(k)} - \frac{\left(p^{(k-1)}, Ap^{(k)}\right)_A}{\left(p^{(k-1)}, p^{(k-1)}\right)_A} p^{(k-1)}$$

End.

Both sets of α's are the same because $\frac{\left(r^{(k)}, p^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A} = \frac{\left(f, p^{(k)}\right) - \left(Au^{(k)}, p^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A} = \frac{\left(f, p^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A}$

The last equality is true because $u^{(k)} = span\{p^{(k-1)}, ..., p^{(0)}\}$ and $\left(p^{(i)}, p^{(j)}\right)_A = 0$ for $i \neq j$

The only requirement we imposed on the *p*'s so far is that they are *A*-Conjugate. So right now we have a Conjugate-Gradient Method without the Gradient. From above, we see that using any random *A*-Conjugate orthogonal basis for *H* ,we are guaranteed to find *u* using a linear combination of *n p*'s .

But the idea of the Conjugate-*Gradient* Method is that if we pick our *p*'s meaningfully, we can arrive at a good approximation of *u* by using less than *n* of our *p*'s. In class, we chose our *p*'s to be the vectors indicated by the Cayley-Hamilton Theorem. We know that $u = A^{-1}f$ and $A^{-1} = Q_m(A)$,a degree *m* polynomial in *A*, with $m \leq n-1$. Hence we chose our *p*'s such that $span\{p^{(0)}, p^{(1)}, ..., p^{(i)}\} = span\{f, A^1 f, A^2 f, ..., A^i f\}$ for $0 \leq i \leq m$ (note: if $u^{(0)} = 0$ then $f = r^{(0)}$) This gave us convergence by using $m+1 \leq n$ of our *p*'s and a good approximation with less than that.

I will show how we can find a good set of *p*'s by using gradient theory. And then I will show that it is the same set of *p*'s from class.

From Calculus 10C, we learned we could find the zeros of $f'(x) = ax - b$ if $f''(x) = a > 0$ by minimizing $f(x) = \frac{1}{2}ax^2 - bx$ Likewise, we can find the zeros of $\nabla F(x) = Au - f$ if $\nabla^2 F(x) = A > 0$ (*A* is s.p.d) by minimizing $F(x) = \frac{1}{2}u^T Au - f^T u$ Therefore each new *p* should be the direction that would minimize $F(x)$ the most if we move that way. That direction will be the negative gradient or $-\nabla F(u^{(k+1)}) = f - Au^{(k+1)} = r^{(k+1)}$. Therefore set each new *p* equal to $p^{(k+1)} = r^{(k+1)}$. However, we also need the *p*'s to be *A*-Conjugate therefore let

(7) $$p^{(k+1)} = r^{(k+1)} - \text{projection of } r^{(k+1)} \text{ onto } span\{p^{(0)}, p^{(1)}, ..., p^{(k)}\}$$
$$= r^{(k+1)} - \frac{\left(p^{(k)}, r^{(k+1)}\right)_A}{\left(p^{(k)}, p^{(k)}\right)_A} p^{(k)}$$

Let's compare these *p*'s to the ones we found in class. In class, we used the Cayley Hamilton Theorem and arrived at the *p*'s shown in our class algorithm above. Notice that each $p^{(k+1)}$ gets its new *A*-orthogonal direction from $Ap^{(k)}$. In our gradient derivation here, the new *A*-orthogonal direction for our $p^{(k+1)}$ is from $r^{(k+1)}$ and

(8) $$r^{(k+1)} = \left(-Au^{(k+1)}\right) + f = \left(-\alpha_k Ap^{(k)} - \alpha_{k-1} Ap^{(k-1)} - ... - \alpha_0 Ap^{(0)}\right) + p^{(0)} \qquad (f = p^{(0)} \text{ when } u^{(0)} = 0)$$

By using induction, it is easy to see that the only new *A*-orthogonal direction we get from the terms above is from $Ap^{(k)}$. Therefore, aside from their lengths, the *p*'s derived here are the same as the ones from class.

Here is the CG algorithm derived in this paper:

Set $u^{(0)}, r^{(0)} = f - Au^{(0)}, p^{(0)} = r^{(0)}$
Do $k = 0, 1, 2, ...$ until done

$$\alpha_k = \frac{\left(f, p^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A}$$
$$u^{(k+1)} = u^{(k)} + \alpha_k A p^{(k)}$$
$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$
$$p^{(k+1)} = r^{(k+1)} - \frac{\left(p^{(k)}, r^{(k+1)}\right)_A}{\left(p^{(k)}, p^{(k)}\right)_A} p^{(k)}$$

End.

Observe that we compute the residual recursively to avoid a matrix-vector multiplication.

(9) $$r^{(k+1)} = f - Au^{(k+1)} = f - A\left(u^{(k)} + \alpha_k p^{(k)}\right) = r^{(k)} - \alpha_k p^{(k)}$$

The above algorithm requires 1 matrix-vector multiplication, 3 vector-vector multiplications, and 3 scalar-vector multiplications per iteration making its order of computation per iteration $O(n^2 + 6n)$. You may normalize the p's if you wish without destroying the algorithm. However it seems that the algorithm keeps the p's near each other in size already.

The algorithm from class is more or less the same complexity having $O(2n^2 + 9n)$ per iteration with an extra $O(3n)$ if you wish to A-normalize the p's as you go along. They will naturally get large, so it is wise to normalize them.

For your additional information, let me present you with the most common form of the CG algorithm. Most textbooks and other references present the above algorithm with the following two identities substituted in:

(10) $$r^{(k)^T} r^{(k)} = \alpha_k p^{(k)^T} A p^{(k)}$$

(11) $$r^{(k+1)^T} r^{(k+1)} = -\alpha_k r^{(k+1)^T} A p^{(k)}$$

Therefore the common algorithm is: (which is essentially the one in the original text by Hestenes and Stiefel (1952)):

Set $u^{(0)}, r^{(0)} = f - Au^{(0)}, p^{(0)} = r^{(0)}$

Do $k = 0,1,2,...$ until done

$$\alpha_k = \frac{\left(r^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, p^{(k)}\right)_A}$$
$$u^{(k+1)} = u^{(k)} + \alpha_k A p^{(k)}$$
$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$
$$p^{(k+1)} = r^{(k+1)} - \frac{\left(r^{(k+1)}, r^{(k+1)}\right)}{\left(r^{(k)}, r^{(k)}\right)} p^{(k)}$$

End.

The reasons sited for the substitutions are increased efficiently. But the complexity of this common algorithm is the same as the one presented in this paper, $O(n^2 + 6n)$, so go figure. Also be aware that if you choose to use the common algorithm, with its substitutions, you can no longer change the magnitudes of the p's as you go along without destroying the algorithm.

Below is an example. Both algorithms were programmed into Matlab with the additional step of A-normalization of the p vectors added. I compared the $u^{(k)}$'s and $p^{(k)}$'s generated by each algorithm and as predicted they were the same. Here are the common results:

```
EDU>> A

A =

   2.7345   1.8859   2.0785   1.9442   1.9567
   1.8859   2.2340   2.0461   2.3164   2.0875
   2.0785   2.0461   2.7591   2.4606   1.9473
   1.9442   2.3164   2.4606   2.5848   2.2768
   1.9567   2.0875   1.9473   2.2768   2.4853

EDU>> f'

ans =

   0.7577   0.7431   0.3922   0.6555   0.1712

EDU>> [p u]=CG3(A,f);    %Solve Au=f  and output all the u's and p's.

U_1 is off by:341.206219 in 2-norm and 6.820957 in A-norm and Energy=-0.093197
U_2 is off by:340.772463 in 2-norm and 6.761206 in A-norm and Energy=-0.498973
U_3 is off by:340.376891 in 2-norm and 6.740362 in A-norm and Energy=-0.639683
U_4 is off by:332.147010 in 2-norm and 6.656442 in A-norm and Energy=-1.201816
U_5 is off by:    0.000000 in 2-norm and 0.000000 in A-norm and Energy=-23.355926

EDU>> p   %Display the p's that were used.

p =

   0.1881  -0.7746  -0.4836  -1.3886   7.4659
   0.1845  -0.7653   0.8385   3.1254 -21.8740
   0.0974   0.6002  -0.6800   3.6653 -17.5074
   0.1627  -0.2979   1.0143  -5.9801  39.3848
   0.0425   1.2539  -0.7266   0.8848  -9.8986

EDU>> p'*A*p   %Show that the p's are A-conjugate

ans =

   1.0000  -0.0000  -0.0000  -0.0000  -0.0000
  -0.0000   1.0000   0.0000  -0.0000  -0.0000
  -0.0000   0.0000   1.0000  -0.0000   0.0000
  -0.0000  -0.0000  -0.0000   1.0000   0.0000
  -0.0000  -0.0000   0.0000  -0.0000   1.0000

EDU>> u   %Display all the u_k's

u =

   0.0812   0.7791   0.5225   1.9949   51.6913
   0.0796   0.7690   1.2139  -2.1001 -147.7031
   0.0420  -0.4986  -0.8594  -4.7458 -121.2827
   0.0702   0.3386   0.8767   7.2175  269.3803
   0.0183  -1.1113  -1.4968  -2.4349  -68.3246

EDU>> (A*u(:,5))'   %Test the final u by comparing Au with f above.

ans =

   0.7577   0.7431   0.3922   0.6555   0.1712

EDU>>
```