Modified Gram-Schmidt Process
29 March 2013

- Page 437, from line -10 to bottom of page, should read:

Schmidt process can be *stabilized*. Instead of computing the vectors $\mathbf{z}_k$using

$$\mathbf{z}_k \leftarrow \mathbf{v}_k - \text{proj}_{\mathbf{z}_1}\mathbf{v}_k - \text{proj}_{\mathbf{z}_2}\mathbf{v}_k - \cdots - \text{proj}_{\mathbf{z}_{k-2}}\mathbf{v}_k - \text{proj}_{\mathbf{z}_{k-1}}\mathbf{v}_k, \qquad \mathbf{q}_k \leftarrow (1/||\mathbf{z}_k||)\mathbf{z_k}$$

we compute $\mathbf{z}_k^{(k)}$ (improved $\mathbf{z}_k$) using

$$\begin{aligned}
\mathbf{z}_k^{(0)} &\leftarrow \mathbf{v}_k - \text{proj}_{\mathbf{z}_1}\mathbf{v}_k, \\
\mathbf{z}_k^{(1)} &\leftarrow \mathbf{z}_k^{(0)} - \text{proj}_{\mathbf{z}_2}\mathbf{z}_k^{(0)}, \\
&\vdots \\
\mathbf{z}_k^{(k-1)} &\leftarrow \mathbf{z}_k^{(k-2)} - \text{proj}_{\mathbf{z}_{k-2}}\mathbf{z}_k^{(k-2)}, \\
\mathbf{z}_k^{(k)} &\leftarrow \mathbf{z}_k^{(k-1)} - \text{proj}_{\mathbf{z}_{k-1}}\mathbf{z}_k^{(k-1)}, \qquad \mathbf{q}_k \leftarrow (1/||\mathbf{z}_k^{(k)}||)\mathbf{z}_\mathbf{k}^{(\mathbf{k})}
\end{aligned}$$

Here each step finds a vector $\mathbf{z}_k^{(i)}$ orthogonal to $\mathbf{z}_k^{(i-1)}$. Moreover, $\mathbf{z}_k^{(i)}$ is orthogonalized against any error introduced in the computation of $\mathbf{z}_k^{(i-1)}$. ‘

- Page 438, top of page to beginning of **EXAMPLE 3**, should read:

An algorithm for the **modified (stabilized) Gram-Schmidt process** is:

for $i = 1$ **to** k
  $\mathbf{v}_i \leftarrow (1/||\mathbf{v}_i||)\,\mathbf{v}_i$ (normalized)
  for $j = 1$ **to** $i - 1$
    $\mathbf{v}_j \leftarrow \mathbf{v}_j - \text{proj}_{\mathbf{v}_i}\mathbf{v}_j$ (remove component in direction $\mathbf{v}_i$)
  end for
end for

Here the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are replaces by an orthonormal set of vectors that spans the same subspace (**orthonormalization**). In exact arithmetic, this computation gives the same result as before but introduces smaller errors in finite-precision computer arithmetic. If each vector has n components, the asymptotical cost of the algorithm is $2nk^2$ floating-point operations.