NE 255, Fa16
Operator Form and Iteration Methods
October 20, 2016

We have now fully discretized the transport equation:

$$\hat{\Omega}_a \cdot \nabla \psi_a^g(\vec{r}) + \Sigma_t^g(\vec{r})\psi_a^g(\vec{r}) = \tag{1}$$
$$\sum_{g'=0}^{G}\sum_{l=0}^{N}\Sigma_{sl}^{gg'}(\vec{r})\left[Y_{l0}^e(\hat{\Omega}_a)\phi_{l0}^{g'}(\vec{r}) + \sum_{m=1}^{l}\left(Y_{lm}^e(\hat{\Omega}_a)\phi_{lm}^{g'}(\vec{r}) + Y_{lm}^o(\hat{\Omega}_a)\vartheta_{lm}^{g'}(\vec{r})\right)\right] + q_{a,e}^g(\vec{r}) \,,$$

$$\phi_{lm}^g = \int_{4\pi} Y_{lm}^e(\hat{\Omega})\psi^g(\hat{\Omega})\, d\hat{\Omega} \,, \quad m \geq 0 \,, \tag{2}$$

$$\vartheta_{lm}^g = \int_{4\pi} Y_{lm}^o(\hat{\Omega})\psi^g(\hat{\Omega})\, d\hat{\Omega} \,, \quad m > 0 \,. \tag{3}$$

Now that we have these equations (+ spatial discretization) we're going to look at how we actually solve this with a computer.

The next set of material comes from

R. N. Slaybaugh. *Acceleration Methods for Massively Parallel Deterministic Transport*, a PhD Dissertation. University of Wisconsin, Madison, WI (2011).

T. M. Evans, A. S. Stafford, R. N. Slaybaugh, and K. T. Clarno. "Denovo – New Three-Dimensional Parallel Discrete Ordinates Code in SCALE." *Nuclear Technology*, **volume 171**(2), pp. 171200 (2010).

Operator Form

We start by expressing the TE in operator notation; which makes it easier to talk about solution techniques. In general, uppercase bolded letters will indicate matrices and lowercase italicized letters will indicate vectors and scalars. The following operators are used to express the transport equation:

$\mathbf{L} = \hat{\Omega} \cdot \nabla + \Sigma_t$ is the transport operator,

$\mathbf{M}$ is the operator that converts harmonic moments into discrete angles,

$\mathbf{S}$ is the scattering matrix,

q_e contains the external source,
f contains the fission source, $\nu\Sigma_f$; $\mathbf{F} = \chi f^T$,
$\mathbf{D} = \mathbf{M^T W} = \sum_{a=1}^{n} Y_{lm}^{e/o} w_a$ is the discrete-to-moment operator.

With this notation, Equation 1 can be written as Equation 4; it can be formulated as an eigenvalue problem by replacing the fixed soruce term with the fission term, $\frac{1}{k}\mathbf{MF}\phi$. This has two unknowns, the angular flux and the moments, which are related by the discrete-to-moment operator as seen in Equation (5) [NOTE: ϕ are the flux moments, **not** the scalar flux].

$$\mathbf{L}\psi = \mathbf{MS}\phi + \mathbf{M}q_e \tag{4}$$

$$\phi = \mathbf{D}\psi \tag{5}$$

The size of the operators can be defined in terms of the granularity of discretization:
G = number of energy groups,
t = number of moments,
n = number of angular unknowns,
c = number of cells,
u = number of unknowns per cell, which is determined by spatial discretization.
These can be combined to define
$a = G \times n \times c \times u$ and
$f = G \times t \times c \times u$.
Using a and f, Equation (4) can be presented in terms of operator size:

$$(a \times a)(a \times 1) = (a \times f)(f \times f)(f \times 1) + (a \times f)(f \times 1)\,.$$

The index variables, their meaning, and their ranges are shown in Table 1.

Table 1: Meaning and Range of Indices Used in Transport Discretization

Variable	Symbol	First	Last
Energy	g	1	G
Solid Angle	a	1	n
Space	suppressed	n/a	n/a
Legendre moment (P_N)	l	0	N
Spherical harmonic moment (Y)	m	0	l

The structures of the vectors and matrices are shown in the next few equations as this can make the whole thing easier to understand and visualize. The angular flux vector is explicitly written first, where for each discrete angle, a, and energy group, g, the set of angular fluxes, ψ_a^g, includes all

spatial unknowns.

$$\psi = \begin{pmatrix} [\psi]_1 & [\psi]_2 & \cdots & [\psi]_g & \cdots [\psi]_G \end{pmatrix}^T, \quad \text{and} \tag{6}$$

$$[\psi]_g = \begin{pmatrix} \psi_1^g & \psi_2^g & \cdots & \psi_a^g & \cdots \psi_n^g \end{pmatrix}^T. \tag{7}$$

$$\mathbf{M} = \begin{pmatrix} [\mathbf{M}]_{11} & 0 & 0 & \cdots & 0 \\ 0 & [\mathbf{M}]_{22} & 0 & \cdots & 0 \\ 0 & 0 & [\mathbf{M}]_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & [\mathbf{M}]_{GG} \end{pmatrix}, \qquad \mathbf{S} = \begin{pmatrix} [\mathbf{S}]_{11} & [\mathbf{S}]_{12} & [\mathbf{S}]_{13} & \cdots & [\mathbf{S}]_{1G} \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & [\mathbf{S}]_{23} & \cdots & [\mathbf{S}]_{2G} \\ [\mathbf{S}]_{31} & [\mathbf{S}]_{32} & [\mathbf{S}]_{33} & \cdots & [\mathbf{S}]_{3G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [\mathbf{S}]_{G1} & [\mathbf{S}]_{G2} & [\mathbf{S}]_{G3} & \cdots & [\mathbf{S}]_{GG} \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} \chi^1[\nu\Sigma_f]^1 & \chi^1[\nu\Sigma_f]^2 & \cdots & \chi^1[\nu\Sigma_f]^G \\ \chi^2[\nu\Sigma_f]^1 & \chi^2[\nu\Sigma_f]^2 & \cdots & \chi^2[\nu\Sigma_f]^G \\ \vdots & \vdots & \ddots & \vdots \\ \chi^G[\nu\Sigma_f]^1 & \chi^G[\nu\Sigma_f]^2 & \cdots & \chi^G[\nu\Sigma_f]^G \end{pmatrix}, \qquad [\mathbf{S}]_{gg'} = \begin{pmatrix} \Sigma_{s\,0}^{gg'} & 0 & \cdots & 0 \\ 0 & \Sigma_{s\,1}^{gg'} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{s\,N}^{gg'} \end{pmatrix},$$

$$[\mathbf{M}]_{gg} = \begin{pmatrix} Y_{00}^e(\hat{\Omega}_1) & Y_{10}^e(\hat{\Omega}_1) & Y_{11}^o(\hat{\Omega}_1) & Y_{11}^e(\hat{\Omega}_1) & Y_{20}^e(\hat{\Omega}_1) & \cdots & Y_{NN}^o(\hat{\Omega}_1) & Y_{NN}^e(\hat{\Omega}_1) \\ Y_{00}^e(\hat{\Omega}_2) & Y_{10}^e(\hat{\Omega}_2) & Y_{11}^o(\hat{\Omega}_2) & Y_{11}^e(\hat{\Omega}_2) & Y_{20}^e(\hat{\Omega}_2) & \cdots & Y_{NN}^o(\hat{\Omega}_2) & Y_{NN}^e(\hat{\Omega}_2) \\ Y_{00}^e(\hat{\Omega}_3) & Y_{10}^e(\hat{\Omega}_3) & Y_{11}^o(\hat{\Omega}_3) & Y_{11}^e(\hat{\Omega}_3) & Y_{20}^e(\hat{\Omega}_3) & \cdots & Y_{NN}^o(\hat{\Omega}_3) & Y_{NN}^e(\hat{\Omega}_3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ Y_{00}^e(\hat{\Omega}_n) & Y_{10}^e(\hat{\Omega}_n) & Y_{11}^o(\hat{\Omega}_n) & Y_{11}^e(\hat{\Omega}_n) & Y_{20}^e(\hat{\Omega}_n) & \cdots & Y_{NN}^o(\hat{\Omega}_n) & Y_{NN}^e(\hat{\Omega}_n) \end{pmatrix}.$$

Note that $[\mathbf{M}]_{11} = [\mathbf{M}]_{22} = \ldots = [\mathbf{M}]_{GG} = [\mathbf{M}]$. These are written with subscripts to simplify the visualization of which blocks correspond to which equations and multiply which other blocks.

The lower triangular part of $\mathbf{S}$ represents downscattering, the diagonal represents in-group scattering, and the upper diagonal is upscattering.

Finally, the vector of flux moments for group g has t entries per spatial unknown, and

$$[\phi]_g = \begin{pmatrix} \phi_{00}^g & \phi_{10}^g & \vartheta_{11}^g & \phi_{11}^g & \phi_{20}^g & \cdots & \vartheta_{NN}^g & \phi_{NN}^g \end{pmatrix}^T, \tag{8}$$

for each group.

Note also that the total size of $\mathbf{D}$ over all groups and spatial unknowns is $(f \times a)$ (where $\mathbf{W}$ is an $(n \times n)$ diagonal matrix of quadrature weights). Also, even though $\mathbf{M}$ maps angular flux moments onto discrete angular fluxes, in general $\psi \neq \mathbf{M}\phi$. This constraint can be removed by using Galerkin quadrature in which $\mathbf{D} = \mathbf{M}^{-1}$.

Solution Procedure

Once the matrices are multiplied together, a series of single-group equations that are each only a function of space and angle result:

$$\begin{aligned}
\mathbf{L}[\psi]_1 &= [\mathbf{M}]\big([\mathbf{S}]_{11}[\phi]_1 + [\mathbf{S}]_{12}[\phi]_2 + \ldots + [\mathbf{S}]_{1G}[\phi]_G\big) + [\mathbf{M}][q_e]_1 \,, \\
\mathbf{L}[\psi]_2 &= [\mathbf{M}]\big([\mathbf{S}]_{21}[\phi]_1 + [\mathbf{S}]_{22}[\phi]_2 + \ldots + [\mathbf{S}]_{2G}[\phi]_G\big) + [\mathbf{M}][q_e]_2 \,, \\
&\vdots \\
\mathbf{L}[\psi]_G &= [\mathbf{M}]\big([\mathbf{S}]_{G1}[\phi]_1 + [\mathbf{S}]_{G2}[\phi]_2 + \ldots + [\mathbf{S}]_{GG}[\phi]_G\big) + [\mathbf{M}][q_e]_G \,,
\end{aligned} \tag{9}$$

Each **within-group** equation is solved for that group's flux.

If the groups are coupled together through upscattering, as they often are, then multiple **multigroup** solves over the coupled portion of the energy range may be required.

If there is fission and the eigenvalue is desired, an additional **eigenvalue** solve is needed as well.

Each of these levels of iteration typically uses a completely different solver.

Note that the typical strategy for solving Equation (4) is to combine it with (5) and form one equation involving only the moments, where $Q = \mathbf{D}\mathbf{L}^{-1}\mathbf{M}q_e$ is comprised of the fixed (or fission) source:

$$(\mathbf{I} - \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\mathbf{S})\phi = Q \,. \tag{10}$$

This is solved for ϕ from which ψ can be determined at the end of the calculation. This format looks like what we're used to seeing in numerical linear algebra: $\mathbf{A}x = b$.

Solver Basics

There are two categories of methods for solving $\mathbf{A}x = b$, **direct** and **iterative**.

Direct methods solve the problem by inverting $\mathbf{A}$ and setting $x = \mathbf{A}^{-1}b$. If $\mathbf{A}$ is invertible this

can be done explicitly. $\mathbf{A}$ is often not invertible, so most direct methods are based on factoring the coefficient matrix $\mathbf{A}$ into matrices that are easy to invert. The problem is then solved in pieces where each factored matrix is inverted to get the final solution. An example where this is done is LU factorization. These methods are often robust and require a predictable amount of time and storage resources. However, direct methods scale poorly with problem size, becoming increasingly expensive as problems grow large.

Iterative methods compute a sequence of increasingly accurate approximations to the solution. They generally require less storage and take fewer operations than direct methods, though they may not be as reliable. Iterative methods are highly advantageous for large problems because direct methods become intractable for systems of the size of those of interest here. For this reason the nuclear energy industry tends to use iterative methods for transport calculations.

Within the category of iterative methods, two further subdivisions can be made that will be useful in this work. Some methods only place data from previous iterations on the right hand side of the equation. The order in which those equations are solved is then irrelevant because only the previous iterate is needed. The other class of methods place data from both the previous and current iteration on the right hand side. These methods are fundamentally sequential and must be solved in order. These two categories will be referred to as *order independent* and *order dependent*, respectively.

Within Group Iterative Methods

The methods that functionally perform a mesh sweep

$$[\psi]_g = \mathbf{L}^{-1}\big([\mathbf{M}]\big([\mathbf{S}]_{11}[\phi]_1 + [\mathbf{S}]_{12}[\phi]_2 + \ldots + [\mathbf{S}]_{1G}[\phi]_G\big) + [\mathbf{M}][q_e]_1\big) \,.$$

Richardson iteration

The simplest iteration scheme used by the nuclear community is source iteration (SI), also known as **Richardson iteration**. SI is applied to the within-group space-angle iterations. Richardson iteration can be thought of as a two-part process for the neutron transport equation, where $\bar{Q}$ includes all sources and k is the inner iteration index:

$$\mathbf{L}[\psi]_g^{k+1} = \mathbf{MS}[\phi]_g^k + [\bar{Q}]_g \,, \tag{11}$$

$$[\phi]_g^{k+1} = \mathbf{D}[\psi]_g^{k+1} \,. \tag{12}$$

The spectral radius, $c = \Sigma_s/\Sigma$, determines the speed of convergence. For problems dominated by scattering, SI will converge very slowly (good explanation of how to think of this physically on p. 2, 3 of https://github.com/rachelslaybaugh/NE250/blob/master/23-iterations/Nov-30Class.tex, which you can build using pdflatex).

Krylov methods

Krylov methods are a powerful class of subspace methods that can be ideal for solving various types of linear and eigenvalue problems. A Krylov method solves $\mathbf{A}x = b$ by building a solution from a Krylov subspace generated by an iteration vector v_1. At iteration k, the subspace is:

$$\mathcal{K}_k(\mathbf{A}, v_1) \equiv span\{v_1, \mathbf{A}v_1, \mathbf{A}^2v_1, ..., \mathbf{A}^{k-1}v_1\} \, . \tag{13}$$

The choice of v_1 varies, but $v_1 = b$ is common.

The dimension of a Krylov space is bounded by n because Krylov methods will give the exact solution after n iterations in the absence of roundoff error. Interestingly, this technically makes Krylov methods a hybrid of direct and iterative methods because an exact answer can be obtained in a predetermined number of steps. Krylov subspace methods are nevertheless generally classified as iterative methods.

Krylov methods are particularly useful in a few pertinent cases. One is when $\mathbf{A}$ is very large because fewer operations are required than traditional inversion methods like Gaussian elimination. Another is when $\mathbf{A}$ is not explicitly formed because Krylov methods only need the action of $\mathbf{A}$. Finally, Krylov methods are ideal when $\mathbf{A}$ is sparse because the number of operations are low for each matrix-vector multiplication. For deterministic transport codes, $\mathbf{A}$ is typically quite large, fairly sparse, and only its action is needed. The action of $\mathbf{A}$ is implemented through the transport sweeps.

In the last few decades Krylov methods have been used widely to solve problems with appropriate properties for several reasons. Krylov methods are robust; the existence and uniqueness of the solution can be established; typically far fewer than n iterations are needed when they are used as iterative solvers; they can be preconditioned to significantly reduce time to solution; only matrix-vector products are required; explicit construction of intermediate residuals is not needed; and they have been found to be highly efficient in practice.

There are, however, a few drawbacks. In some cases Krylov methods can be very slow to converge, causing large subspaces to be generated and thus becoming prohibitively expensive in terms of

storage size and cost of computation. Some methods can be restarted after m steps to alleviate this problem, keeping the maximum subspace size below $\mathcal{K}_{m+1}$. The relatively inexpensive restart techniques can reduce the storage requirements and computational costs associated with a slow-converging problem such that they are tractable. Preconditioners can also help by reducing the number of iterations needed.

Krylov Methods

An aside, because they're useful and I think they're super cool.

Arnoldi Method

A general issue with Krylov subspaces is that the columns of $\mathcal{K}_k(\mathbf{A}, v_1)$ become increasingly linearly dependent with increasing k. To deal with this, there are two factorization methods upon which many Krylov methods are based. Some use the Arnoldi method, which generates an orthonormal basis for the Krylov subspace for non-normal matrices, and others use the Lanczos method, which creates non-orthogonal bases for normal matrices. We'll focus on the Arnoldi method.

Galerkin Method and Weighted Residuals

Fundamentally, Krylov methods are Galerkin or Galerkin-Petrov methods on a Krylov subspace. *Galerkin's method* uses a few fundamental concepts:

- an inner product of two functions is zero when the functions are orthogonal:

$$< f(x), g(x) >= 0 \text{ if } f(x) \text{ and } g(x) \text{ are orthogonal.}$$

- any function $f(x)$ in a subspace $\mathcal{V}$ can be written as a linear combination of the vectors that make a basis for that function space. Let $\mathbf{V} = \{\phi_i(x)\}_{i=0}^{\infty}$ be the basis for $\mathcal{V}$; if $f(x) \in \mathcal{V}$ then

$$f(x) = \sum_{j=0}^{\infty} c_j \phi_j(x)$$

  for some scalar coefficients c_j.

- A *weighted residual method* is a solution technique for solving some linear problem $\mathbf{A}u = b$ where

$$u = u_0 + \sum_{i=1}^{n} c_i \phi_i(x)$$

  is the approximate solution and u_0 is an initial guess. The solution is found by taking the inner product of some arbitrary weight function, $w(x)$, and the residual, $r(x) = b - \mathbf{A}u$, $r(x) \in \mathcal{V}$, such that $< w(x), r(x) >= 0$. The solution is the u satisfying this requirement.

Galerkin's method is a weighted residual method where the weight function is chosen from the basis functions: $w(x)$ is selected from $\mathbf{V}$. In the Galerkin-Petrov method, the weight functions come from a subspace other than $\mathcal{V}$, that is $w(x) \in \mathcal{W}$.

The weighted residual method can also be thought of as a process to minimize the residual. There are a few ways to express this idea.

- If $\hat{u}$ is the exact minimizer of $r(x) = b - \mathbf{A}\hat{u}$, then let $u' = \hat{u} + w(x)$ be a close approximation to $\hat{u}$ with $w(x) \in \mathcal{V}$. The residual is minimized if and only if $< w(x), r(x) >= 0$ for all $w(x) \in \mathcal{V}$, meeting the Galerkin condition just described.
- This can also be written as:

$$\text{find} \qquad u \in u_0 + \mathcal{V} \qquad \text{such that} \qquad r(x) \perp \mathcal{V}\,. \tag{14}$$

  Further, if y is the solution to

$$\mathbf{V}^T\mathbf{A}\mathbf{V}y = \mathbf{V}^T r_0\,, \qquad \text{then} \qquad \hat{u} = u_0 + \mathbf{V}y\,. \tag{15}$$

  This $\hat{u}$ minimizes the residual in the some measure of interest, where the measure is determined by the selection of y.

We can use the Galerkin Method in combination with *Ritz pairs* to understand the Arnoldi method. For us, the subspace from which solutions are derived will be the Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$, and the equation will be switched to $\mathbf{A}x = b$.

Ritz Pairs

A vector $z \in \mathcal{K}_k(\mathbf{A}, v_1)$ is defined as a *Ritz vector* with corresponding *Ritz value*, θ, if it satisfies the Galerkin condition

$$< w, \mathbf{A}z - \theta z >= 0 \, \forall \, w \in \mathcal{K}_k(\mathbf{A}, v_1) \, .$$

To see why Ritz pairs can be important, define $\hat{p}(\mathbf{A})$ to be the minimum characteristic polynomial of $\mathbf{A}$ such that $||\hat{p}(\mathbf{A})v_1|| \leq ||q(\mathbf{A})v_1||$ for all monic polynomials $q \neq \hat{p}$ of degree k. If (θ, z) is a Ritz pair for $\mathbf{A}$, then $\mathbf{A}z - z\theta = \gamma\hat{p}(\mathbf{A})v_1 = g$ for some scalar γ. When $g = 0$, then the Ritz pair is an eigenpair. When g is small, the Ritz pair is likely a close approximation to an eigenpair of $\mathbf{A}$.

Method

These ideas can be assembled to understand why the Arnoldi method works. Let $\mathbf{V}$ be a basis for a Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$, and $\mathcal{X} \in \mathcal{K}_k(\mathbf{A}, v_1)$ be an eigenspace of $\mathbf{A}$. When solving $\mathbf{A}x = b$, the subspace $\mathcal{K}_k(\mathbf{A}, v_1)$ and the vector x must satisfy

1. $x \in \mathcal{K}_k(\mathbf{A}, v_1)$ and
2. $r = \mathbf{A}x - b \perp \mathcal{K}_k(\mathbf{A}, v_1)$.

Let $\mathbf{H} = \mathbf{V}^T\mathbf{A}\mathbf{V}$. There is an eigenpair (θ, x) of $\mathbf{H}$ such that $(\theta, \mathbf{V}x)$ is an eigenpair of $\mathbf{A}$. To reduce notational clutter let $z = \mathbf{V}x$, giving $\mathbf{A}z = \theta z$.

If $\mathbf{V}$ is an orthonormal basis for $\mathcal{K}_k(A, v_1)$, then (θ, z) is a Ritz pair if and only if $x = \mathbf{V}y$ with $\mathbf{H}y = \theta y$ for some y. Noting the definition of $\mathbf{H}$, comparing to Equation (15), and doing some basic matrix manipulation, it can be seen that this y minimizes the residual. As the residual tends toward zero, the Ritz pair converges to an approximate eigenpair of $\mathbf{A}$. Another way to state this is to revisit the polynomial identity expressing the minimum residual. It can be shown that $\mathbf{A}\mathbf{V}y - \mathbf{V}\mathbf{H}y = \gamma\hat{p}(\mathbf{A})v_1 = g$. As $g \to 0$, the Ritz pair approaches the eigenpair of $\mathbf{A}$.

In summary, the eigenpairs of $\mathbf{A}$ are approximated by the eigenpairs of $\mathbf{H}$. These eigenvalues and/or eigenvectors are subsequently used in different ways by different Krylov methods to formulate the solution to $\mathbf{A}x = b$ to achieve specific goals, like minimizing the residual in a certain norm. The Galerkin condition is used to ensure the eigenpairs of $\mathbf{H}$ become increasingly good approximations to those of $\mathbf{A}$ as the size of the Krylov subspace increases.

The Arnoldi method is a process of establishing the $\mathbf{V}$ and $\mathbf{H}$ discussed above. $\mathbf{H}$ is an orthogonal

projection of $\mathbf{A}$ onto the basis $\mathbf{V}$ and is upper Hessenberg in form. The Gram-Schmidt method (or the modified Gram-Schmidt method) computes $\mathbf{V}$. The Arnoldi method generates a Ritz estimate for the Ritz pair at each iteration. The Arnoldi Algorithm is:

$r_0 = b - \mathbf{A}x_0$

$v_1 = \frac{r_0}{||r_0||}$

For $j = 1$ to k:

$h_{i,j} = v_i \mathbf{A} v_j,\ i = 1, \ldots, j$

$\hat{v}_{j+1} = \mathbf{A}v_j - \sum_{i=1}^{j} v_j h_{i,j}$

$h_{j+1,j} = ||\hat{v}_{j+1}||$

$v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$

Form the solution $x_k = x_0 + \mathbf{V}_k y_k$, where $y_k = \mathbf{H}_k^{-1}||r_0||e_1$

The kth step of an Arnoldi factorization can be written as:

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k \mathbf{H}_k + g_k e_k^T\ , \tag{16}$$

where e_k is the k^{th} column of the identity matrix and g is the residual. An alternative way to derive the Arnoldi method is as a truncation of the reduction of $\mathbf{A}$ to Hessenberg form using shifted QR-iteration.

GMRES

One of the more popular Krylov methods, which uses the Arnoldi process, is the GMRES algorithm developed by Saad and Schultz. Recall that $\mathbf{V}_k$ is an orthonormal basis for the Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$ and that $\mathbf{H}_k$ is the representation of the part of $\mathbf{A}$ that is in this Krylov subspace formed from the basis $\mathbf{V}_k$. The notation $\bar{\mathbf{H}}^k$ is the $(k+1) \times k$ upper Hessenberg matrix that includes the newest $h_{k+1,k}$ element generated in the Arnolodi process. This satisfies $\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\bar{\mathbf{H}}_k$.

The distinguishing factor for different Krylov methods is the way in which they select the y_k that makes the solution $x_k = x_0 + \mathbf{V}_k y_k$. GMRES uses the least squares procedure to find the y_k that minimizes the norm of the residual over z in $\mathcal{K}_k(\mathbf{A}, v_1)$. To accomplish this, the least squares

problem

$$\min_{z \in \mathcal{K}_k} ||r_0 - \mathbf{A}z|| \tag{17}$$

is solved. Here $z = \mathbf{V}_k y_k$. The y_k that is selected minimizes $||\beta e_1 - \bar{\mathbf{H}}^k y||$, where $\beta = ||r_0||$.

GMRES picks the best solution within the Krylov subspace with respect to minimizing the residual.