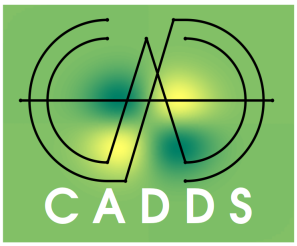
Lecturas de Optimización Numérica:

MÉTODOS DE SUBESPACIOS DE KRYLOV



Prof. Dr. Fredy Vides
Scientific Computing Innovation Center, UNAH &
Centre for Analysis of Data-Driven Systems
E-mail: fredy.vides@unah.edu.hn

ÍNDICE

OBJETIVOS

1. Estudiar algunas propiedades fundamentales de los métodos de subespacios de Krylov.

1. SUBESPACIOS DE KRYLOV

Definición 1.1. Dado un polinomio $p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0$ con coeficientes complejos, y dada una matriz $A \in \mathbb{C}^{m\times m}$, escribiremos $p(A)$ para denotar la matriz

$$p(A) = a_n A^n + a_{n-1} A^{n-1} + \cdots + a_1 A + a_0 I$$

la matriz $p(A)$ se denomina un polinomio en A. Se denotará por $\mathcal{P}_n$ el conjunto de todos los polinomios con coeficiones complejos de grado $\leq n$.

Definición 1.2. Dado un vector $v \in \mathbb{C}^n$ y una matriz $A \in \mathbb{C}^{n\times n}$, se denomina subespacio de Krylov (correspondiente a A, v) de grado m el subespacio $\mathcal{K}_m \subset \mathbb{C}^n$ determinado por la siguiente expresión.

$$\mathcal{K}_m(A, v) = \text{gen}\{v, Av, \ldots, A^{m-1}v\} \tag{1.1}$$

Se denota por $\mathbf{K}(A, b)$ la matriz en $\mathbb{C}^{n\times m}$ determinada por la siguiente expresión.

$$\mathbf{K}_m(A, v) = \begin{bmatrix} v & Av & \cdots & A^{m-1}v \end{bmatrix} \tag{1.2}$$

Observación 1.3. Dado un vector $v \in \mathbb{C}^n$ y una matriz $A \in \mathbb{C}^{n\times n}$, es posible obvervar lo siguiente.

$$\begin{aligned} \mathcal{K}_m(A, v) &= \{p(A)v \ : \ p \in \mathcal{P}_{m-1}\} \\ &= \{\mathbf{K}_m(A, v)a \ : \ a \in \mathbb{C}^m\} \end{aligned}$$

Ejercicio para el lector 1. Verificar la observación 1.3.

Sea J_n la matriz en $\mathcal{C}^n$ determinada por la siguiente expresión.

$$J_n = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \tag{1.3}$$

Dado un vector $v \in \mathbb{C}^n$, una matriz $A \in \mathbb{C}^{n\times n}$ y dado $m \leq n$, es posible obvervar lo siguiente.

$$A\mathbf{K}_m(A,b) = \mathbf{K}_m(A,b)J_m + (A^m v)\hat{e}^*_{m,m} \tag{1.4}$$

donde $\hat{e}_{j,m}$ denota la j-ésima columna de la matriz identidad I_m de $m \times m$.

GMRES: Métodos de resíduo mínimo generalizado. Dado un sistemas de ecuaciones lineales

$$Ax = b, \tag{1.5}$$

para $A \in \mathbb{C}^{n\times n}$ y $b \in \mathbb{C}^n$. Los métodos iterativos de (subespacios de) Krylov producen sucesiones de la forma $x_k = x_0 + q_k$, donde x_0 es una sposición inicial de la solución de (1.5), y donde cada q_k es extraído del subespacio $\mathcal{K}_k(A, r_0)$, para $r_0 = b - Ax_0$. Por la observación 1.3, para cada q_k existe $q \in \mathcal{P}_{k-1}$ tal que $q_k = q(A)r_0$. Si se define el polinomio $p(z) = 1 - zq(z)$, entonces $p \in \mathcal{P}_k$ y es posible medir la convergencia de los métodos utilizando el residuo

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - Ax_0 + Ax_0 - Ax_k \\ &= r_0 - A(x_k - x_0) \\ &= r_0 - Aq_k \\ &= r_0 - Aq(A)r_0 \\ &= (I - Aq(A))r_0 = p(A)r_0 \end{aligned} \tag{1.6}$$

Los diversos métodos iterativos de Krylov difieren en la forma en que calculan los polinomios residuales p correspondientes a cada residuo r_k de la forma (1.6). El objetivo de los métodos GMRES es resolver el problema:

$$p_k = \operatorname{argmin}_{p\in\mathcal{P}_k, p(0)=1} \|p(A)r_0\|_2 \tag{1.7}$$

para cada iteración, encontrando a la vez, un balance entre la factibilidad y computabilidad de las soluciones. Es decir para cada iteración se busca el polinomio $p_k \in \mathcal{P}_k$ tal que $p_k(0) = 1$ y $\|p_k(A)r_0\|_2 \leq \|p(A)r_0\|_2$ para cualquier $p \in \mathcal{P}_k$ tal que $p(0) = 1$. Además, para que sea efectivo, un método iterativo GMRES debe alcanzar valores suficientemente pequeños para $\|r_k\|_2$, para $k \ll n$.

El núcleo del preceso iterativo GMRES es el **proceso de Arnoldi**, un mecanismo que permite calcular construir una base ortonormal $\{u_1, \ldots, u_k\}$ para cada $\mathcal{K}_k(A, r_0)$ siempre que $\operatorname{rk}(\mathbf{K}_k(A,b)) = k > 0$, donde cada u_k puede calcularse aplicando una variación del

teorema de ortogonalización de Gram-Schmidt, utilizando las siguientes ecuaciones de recurrencia.

$$
\begin{aligned}
u_1 &= \frac{1}{\|r_0\|_2} r_0, \\
v_{k+1} &= Au_k - \sum_{j=1}^{k} ((Au_k) \cdot u_j) u_j = Au_k - \sum_{j=1}^{k} (u_j^*(Au_k)) u_j \\
u_k &= \frac{1}{\|u_k\|_2} u_k
\end{aligned} \tag{1.8}
$$

Este preceso de ortonormalización puede volverse altamente costoso computacionalemente a medida que se incrementa k.

Para estudiar el proceso de Arnoldi, es conveniente organizar el proceso de ortogonalización en forma matricial. Sea $H_k = [h_{jk}] \in \mathbb{C}^{k \times k}$ la matriz de Hessenberg, cuyos coeficientes cumplen las condiciones,

$$
h_{jk} = \begin{cases} u_j^*(Au_k), & j > k+1 \\ 0, & j \leq k+1 \end{cases} \tag{1.9}
$$

y sea $\tilde{H}_k$ la matriz en $\mathbb{C}^{(k+1) \times k}$ de la forma

$$
\tilde{H}_k = \begin{bmatrix} H_k \\ (u_{k+1}^*(Au_k)) \hat{e}_{k,k}^* \end{bmatrix} \tag{1.10}
$$

Si se definen las matrices $U_m = \begin{bmatrix} u_1 & \cdots & u_m \end{bmatrix} \in \mathbb{C}^{n \times k}$. Por (1.9) y (1.10) se cumplirá lo siguiente.

$$
\begin{aligned}
AU_k &= U_k H_k + (u_{k+1}^*(Au_k)) u_{k+1} \hat{e}_{k,k}^* \\
&= U_{k+1} \tilde{H}_k
\end{aligned} \tag{1.11}
$$

Por ortonormalidad de los vectores $u_1, \ldots, u_{k+1}$, premultiplicando (1.11) por U_k^* se obtiene la siguiente expresión.

$$
H_k = U_k^* A U_k \tag{1.12}
$$

Observación 1.4. La matriz de Hessenberg H_k es la **restricción/compresión** de la matriz A al subespacio de Krylov $\mathcal{K}_k(A, r_0)$ de grado k.

La versión del algoritmo básico de Arnoldi propuesta por Saad en [2] se presenta a continuación.

Algoritmo 1. **Método de Arnoldi**

Seleccionar $u_1 \in \mathbb{C}^n$ tal que $\|u_1\|_2 = 1$
para $k = 1, \ldots, m$ **hacer**
  Calcular $h_{jk} = u_j^*(Au_k)$, **para** $j = 1, \ldots, k$
  Calcular $w_k = Au_k - \sum_{j=1}^{k} h_{jk} u_j$
  $h_{k+1,k} \leftarrow \|w_k\|_2$
  si $h_{k+1,k} = 0$ **entonces** *Detener*
  $u_{k+1} \leftarrow w_k / h_{k+1,k}$
fin

El algoritmo 1 permita calcular para un vector arbitrario $v \in \mathbb{C}^n$ y una matriz $A \in \mathbb{C}^{n\times n}$, las matrices $U_k, U_{k+1}, H_k, \tilde{H}_k$ que cumplen las condiciones (1.11).

Podemos aplicar el proceso de Arnoldi para calcular cada polinomio p_k que resuelve el problema de optimización (1.7). Sea $\{\nu_k\}_{j=1}^k$ el conjunto de raíces del polinomio $p_k \in \mathcal{P}_k$ a ser determinado. Consideremos en particular le raíz l-ésima de p_k, se cumplirá entonces que p_k puede representarse en la forma:

$$p_k(z) = \left(1 - \frac{z}{\nu_l}\right) q(z)$$

para algún $q \in \mathcal{P}_{k-1}$. Se cumple entonces que $q_k(A) \in \mathcal{K}_k(A, v)$, de manera que existe $y \in \mathbb{C}^k$ tal que $q(A)r_0 = U_k y$. La optimalidad de mínimos cuadrados de (1.7) implica que r_k debe ser ortogonal a cualquier vector en $A\mathcal{K}_k(A, r_0) = \text{gen}\{Ar_0, A^2r_0, \ldots, A^k r_0\}$, por tanto:

$$0 = (AU_k)^* r_k = U_k^* A^* \left(I - \frac{1}{\nu_l} A\right) U_k y.$$

Sustituyendo la identidad $AU_k = U_{k+1}\tilde{H}_k$ en la expresión previa, se obtiene la siguiente ecuación.

$$\nu_l \tilde{H}_k U_k^* U_k y = \tilde{H}_k^* U_{k+1}^* U_{k+1} \tilde{H}_k y,$$

por tanto, como consecuencia de (1.11), ν_l es solución del siguiente problema generalizado de eigenvalores.

$$\tilde{H}_k^* \tilde{H}_k y = \nu_l H_k^* y \tag{1.13}$$

Ejemplo 1. Es posible verificar aproximadamente las propiedades de los subespacios de Krylov previamente estudiadas. Consideremos en particular una matriz $A \in \mathbb{C}^{100\times 100}$ y un vector $v \in \mathbb{C}^{100}$ generados al azar utilizando la siguiente secuencia de comandos de Octave.

```
>> n=100;
>> A=randn(n)+i*randn(n);
>> v=randn(n,1)+i*randn(n,1);
```

Es posible calcular las matrices $U_k, U_{k+1}, H_k, \tilde{H}_k$ correspondientes a $\mathcal{K}_k(A, v)$ y $A\mathcal{K}_k(A, v)$, y que cumplen las condiciones (1.11), utilizando el comando `krylov` de Octave. Para este ejemplo consideraremos $k = 10$.

```
>> k=10;
>> tic,[Vk, hk, Nk] = krylov (A, v, k+1);toc
Elapsed time is 0.010746 seconds.
>> Hk0=hk(1:k,1:k);
>> Hk1=hk(:,1:k);
>> Uk=Vk(:,1:k);
```

Ahora es posible verificar que las condiciones (1.11) se cumplen aproximadamente (debido a los efectos de la aritmética finita, y a los errores de redondeo y truncamiento).

```
>> norm(A*Uk-Vk*Hk1)
ans =    1.7738e-14
```

Es posible calcular aproximadamente el polinomio p_k determinado por el problema (1.7), aplicando el comando `eig` de Octave para resolver el problema (1.13), utilizando la siguiente secuencia de comandos de Octave.

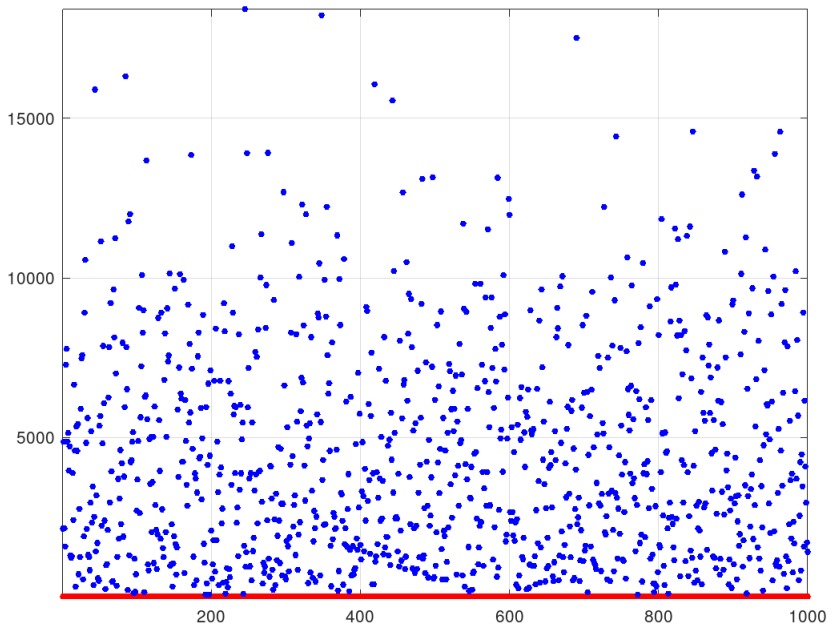


FIGURA 1.1. En esta gráfica se ilustra la verificación aproximada de la condición (1.7) que debe ser cumplida por $p_k \in \mathcal{P}_k^1$.

```
>> tic,l=1./eig(Hk1'*Hk1,Hk0');toc
Elapsed time is 0.0277238 seconds.
>> tic,pk=fliplr(poly(l));toc
Elapsed time is 0.00821304 seconds.
```

Para verificar (aproximadamente) la identidad (1.7) es posible aplicar la siguiente secuencia de comandos.

```
>> ptest=[pk(1:k)+1e-9*randn(10*n,k) ones(10*n,1)];
```

Para generar los coeficientes (en formato Octave) de 1000 elementos en $\mathcal{P}_k^1 = \{p \in \mathcal{P}_k : p(0) = 1\}$ cuyos coeficientes se obtienen perturbando los coeficientes (libres) de p_k. Ahora es posible visualizar de forma aproximada la identidad (1.7) utilizando la siguiente secuencia de comandos.

```
>> tic,for j=1:(10*n),test(j)=norm(polyvalm(ptest(j,:),A)*v);end;toc
Elapsed time is 16.5601 seconds.
>> L=1:(10*n);
>> plot(L,norm(polyvalm(pk,A)*v)*ones(1,10*n),'r.-',...
> 'markersize',12,L,test,'b.','markersize',12);
>> grid on
>> axis tight
```

La salida gráfica producida por la secuencia de comandos anterior se muestra en la figura 1.1.

Ejercicio para el lector 2. Desarrollar un programa Octave llamado Arnoldi.m (**puede utilizar el comando** `krylov` o desarrollar su programa sin utilizar el comando `krylov`), que para cualquier matriz $A \in \mathbb{C}^{n\times n}$, cualquier vector $v \in \mathbb{C}^n$, y cualquier entero positivo $k \leq n$, calcule una base ortonormal de $\{u_1, \ldots, u_k\}$ de $\mathcal{K}_k(A, v)$, y la matriz de Hessenberg H_k correspondiente.

Ejercicio para el lector 3. Dada una matriz $A \in \mathbb{C}^{n\times n}$, un vector $v \in \mathbb{C}^n$, un entero positivo $k \leq n$, y el polinomio p_k determinado por (1.7) para el subespacio $\mathcal{K}_k(A, b)$. Demostrar que existe un proyector $Q \in \mathbb{C}^{n\times n}$ que cumple la siguiente restricción.

$$\|Qv\|_2 = \|p_k(A)v\|_2$$

Ejercicio para el lector 4. Desarrollar un programa Octave llamado `MinPoly.m` que para una matriz $A \in \mathbb{C}^{n\times n}$ y un vector $v \in \mathbb{C}^n$, permita calcular el polinomio p_k determinado por (1.7) para el subespacio $\mathcal{K}_k(A, b)$, resolviendo directamente el problema de mínimos cuadrados (1.7) en lugar del problema generalizado de eigenvalores (1.13). Este nuevo programa puede también utilizar el comando `krylov`, en caso de ser necesario.

Ejercicio para el lector 5. Desarrollar un programa Octave llamado `TestMinPoly.m` que para dos enteros positivos k, n tales que $k \leq n$ (que pueden ingresarse como argumentos de la función `TestMinPoly.m` por el usuario) genere una matriz $A \in \mathbb{C}^{n\times n}$, un vector $v \in \mathbb{C}^n$, y calcule el polinomio p_k determinado por (1.7) para el subespacio $\mathcal{K}_k(A, v)$, aplicando el programa `MinPoly.m`, produciendo una salida gráfica similar a la de la figura 1.1 que permite verificar aproximadamente que el polinomio solución p_k cumple (1.7).

REFERENCIAS

[1] A. S. Householder (1964). The Theory of Matrices in Numerical Analysis. Dover Publications, Inc.
[2] Y. Saad (2003). Iterative Methods for Sparse Linear Systems. 2a Ed. SIAM.
[3] L. N. Trefethen, M. Embree (2005). Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators. Princeton University Press.
[4] R. L. Burden, D. J. Faires, A. M. Burden. (2017). Análisis Numérico. 10a Ed. Cengage Learning Editores.
[5] Golub, G. H., Van Loan C. F (1996). Matrix Computations (3aEd.). The Johns Hopkins University Press.
[6] Quarteroni A., Saleri F., Gervasio P. (2014). Scientific computing with MATLAB and Octave (Textbook).
[7] D. G. Luenberger, Y. Ye. (2016). Linear and Nonlinear Programming. 4a Ed. Springer International Publishing Switzerland.