

5. Métodos Numéricos en Matlab

De las más de 500 funciones o comandos que posee Matlab actualmente, un gran número están dedicadas a resolver problemas del área de los llamados **Métodos Numéricos**.

Gracias a que Matlab posee una metodología de operación basada en vectores y matrices se ha transformado el concepto de la programación para los métodos del análisis numérico y matemático.

Podemos afirmar, sin lugar a dudas, que Matlab es la herramienta ideal para la enseñanza y aprendizaje en las áreas de programación de computadoras, programación estructurada, métodos numéricos y cualquier otra disciplina donde el ingrediente principal sea el análisis numérico, visualización gráfica y la elaboración de algoritmos de soluciones matemáticas.

Algunas características fundamentales que lo hacen ideal para los métodos numéricos son:

- ✓ Una programación más sencilla y estructurada.
- ✓ Mayor amplitud y exactitud numérica.
- ✓ Manejo transparente entre números enteros, reales y complejos.
- ✓ Una biblioteca de funciones matemáticas muy completa. Opción de crear funciones de usuario.
- ✓ Capacidades de graficación muy eficientes.
- ✓ Vinculación con lenguajes de programación de alto nivel; tales como C / C++, Java y Fortran.
- ✓ Transportabilidad de datos y programas con otras aplicaciones.

Una gran ventaja que tiene Matlab con respecto a los métodos numéricos es que no hay distinción entre variables reales, enteras, complejas, de precisión sencilla o de precisión doble.

Cualquier variable puede contener cualquier tipo de número o valor sin una declaración especial durante su definición. Por lo tanto, las operaciones entre distintos tipos de datos se hacen transparentes al usuario. Esto es algo que resulta muy eficiente con respecto a otros paquetes similares o lenguajes de programación.

A continuación se estudiarán algunos de los comandos más útiles para métodos numéricos de la biblioteca de funciones de Matlab, agrupados por área de conocimiento:

5.1 Álgebra Lineal

En esta sección estudiaremos una serie de comandos de Matlab para comprender y resolver varios de los problemas del álgebra matricial y operaciones con matrices tomadas **como un todo** y no como las operaciones de arreglos de elemento a elemento, estudiadas previamente.

No es nuestra intención repasar los fundamentos del álgebra lineal. Se hace la suposición de que el alumno domina los conceptos, al menos básicos, sobre definiciones y términos principales de matrices, álgebra de matrices y sus operaciones fundamentales. De igual forma, se estudiará la manera de resolver sistemas de ecuaciones lineales usando comandos Matlab.

❖ Suma y resta de matrices

Podemos realizar directamente las operaciones de suma (+) y resta (-) matriciales, siempre y cuando las dos matrices tengan el mismo tamaño. Por ejemplo:

$$\begin{aligned} C &= A + B ; \\ X &= Z - Y ; \end{aligned}$$

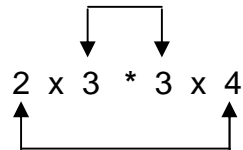
❖ Multiplicación de matrices

Se podrá aplicar el operador de multiplicación (*) para obtener el producto de dos matrices siempre y cuando el **número de columnas de la primera sea igual al número de filas de la segunda**. La matriz resultante tendrá por tamaño el número de filas de la primera con el número de columnas de la segunda. Por ejemplo:

Si A es una matriz de 2 x 3
Y B es una matriz de 3 x 4

Y se tiene que $C = A * B$, entonces:

Si son compatibles para producto, pues el tamaño de columnas y filas son iguales en la parte media:



De donde el tamaño para la matriz producto resultante C, será de 2 x 4 que son las filas y columnas de los extremos.

❖ Trasposición de matrices

Como sabemos, la **traspuesta** de una matriz es otra matriz en la que las filas de la matriz original se convierten en las columnas de la nueva.

En Matlab, la trasposición se realiza con el operador:

[107] ' (Apóstrofo)

Por ejemplo, si $A = \begin{pmatrix} 4 & 6 & 9 \\ 3 & 7 & 1 \end{pmatrix}$

Su traspuesta A' es:

$$A' = \begin{pmatrix} 4 & 3 \\ 6 & 7 \\ 9 & 1 \end{pmatrix}$$

Nótese que para indicar la traspuesta, el operador se coloca a la derecha del nombre de la matriz o vector.

El operador de trasposición (') se utiliza con mucha frecuencia para convertir vectores fila a columna y viceversa:

$$X = [8 \ 2 \ 1 \ 5] \quad Y = X' = \begin{pmatrix} 8 \\ 2 \\ 1 \\ 5 \end{pmatrix}$$

Otra regla a recordar es: $(A * B)' = B' * A'$

❖ Potencias de matrices

Es posible realizar exponenciación de matrices tomándolas como un todo y no elemento a elemento, usando el operador \wedge . Por ejemplo,

$$\begin{array}{ll} A^2 & \text{equivale a: } A * A, \\ B^3 & \text{equivale a: } B * B * B \end{array}$$

La única restricción es que como son productos ligados en cascada las filas y columnas resultantes de cada subproducto deben de mantenerse compatibles; por lo tanto, la exponenciación sólo se puede aplicar a matrices cuadradas.

❖ Matriz inversa y Rango de una matriz

La **inversa** de una matriz A se denota como A^{-1} y es tal que los productos $A * A^{-1}$ y $A^{-1} * A$ son ambos iguales a la matriz identidad I .

La inversa de una matriz se obtiene en Matlab con el comando:

```
[108] >>inv ( A );
```

Si la inversa no existe, se despliega un mensaje de error. Sólo las matrices cuadradas tienen inversa.

Recordemos que no existe la inversa de una matriz que sea **singular** o esté **mal acondicionada**. Esto se relaciona con un sistema de ecuaciones en el que las ecuaciones no son independientes entre sí.

Una matriz singular será aquella en la que al menos una fila (o columna) puede ser expresada restando o sumando otras filas (o columnas). Si esto no ocurre, todas las filas (o columnas) son linealmente independientes entre sí y la matriz es **no singular**.

El **rango** de una matriz, que indica el número de ecuaciones independientes entre sí; representadas por las filas de la matriz, se obtiene con el comando:

```
[109] >>rank ( A );
```

Si el rango que se obtiene para una matriz es exactamente igual a su número de filas, podemos afirmar que es una matriz no singular y su inversa si existe.

Otra regla a recordar es: $(A * B)^{-1} = B^{-1} * A^{-1}$

❖ Determinante de una matriz

El **determinante** de una matriz cuadrada es un valor escalar, calculado a partir de sus elementos, muy importante asociado con la singularidad de dicha matriz.

En Matlab lo obtenemos con:

```
[110] >>det ( A );
```

Si la matriz representa un arreglo de coeficientes de un sistema de ecuaciones lineales, no podremos obtener una solución única si el valor del determinante es igual con cero. De igual forma, si el determinante es cero la matriz es singular y no existe su inversa.

El determinante de una matriz se denota matemáticamente como $\det(A)$ o también como $|A|$.

El cálculo del determinante para matrices triangulares (inferior o superior) es sumamente fácil de obtener al igual que para una matriz diagonal (ceros arriba y debajo de la diagonal).

Cuando se desea calcular el determinante de una matriz dada, es recomendable transformarla en un producto de matrices; a través del uso de su **matriz triangular inferior (L – lower)** y de su **matriz triangular superior (U – upper)** equivalentes, de tal manera que se puede realizar la operación: $\det(A) = \det(L) \cdot \det(U)$.

Regla a recordar: $\det(A * B) = \det(A) * \det(B)$

❖ Factorización triangular LU

Este es un método para hacer **factorizaciones** ó **descomposiciones** con las que podemos lograr soluciones a problemas matriciales con mayor eficiencia.

La factorización triangular LU transforma a una matriz A en el producto de dos matrices triangulares: una matriz triangular inferior *estricta* ó *permutada*, llamada **L**, y una matriz triangular superior, llamada **U**. De tal forma que tenemos que:

$$A = L * U$$

La factorización triangular LU no es única para una matriz dada.

La factorización LU se aplica en casos de querer obtener el determinante de una matriz grande, calcular la inversa de matrices o resolver sistemas de ecuaciones lineales.

Con respecto a los sistemas de ecuaciones lineales del tipo $A * x = b$, podríamos introducir la factorización LU re-escribiendo la ecuación como:

$$L * U * x = b$$

Entonces, si tomamos $U * x = w$,

$$L * w = b$$

De aquí podríamos obtener w muy fácilmente debido a la naturaleza triangular de L.

Teniendo w podemos encontrar x, que es la solución original, también de una forma sencilla debido a que U también es triangular.

Para hacer este tipo de factorizaciones LU, Matlab nos proporciona el comando:

$$[111] \quad \gg [L, U] = \text{lu}(A);$$

Esta descomposición LU resulta muy útil cuando debemos resolver varios conjuntos de ecuaciones lineales con la misma matriz de coeficientes pero diferentes juegos de términos independientes (términos derechos) en lugar de utilizar la eliminación de gauss en cada caso.

❖ Valores y vectores propios de una matriz

Si se tiene una matriz A , cuadrada de $n \times n$. Sea X un vector columna de tamaño n , y sea λ un escalar. Se considera la siguiente ecuación:

$$A * X = \lambda * X$$

Los dos lados de la ecuación dan por resultado un vector de n filas.

A los valores que resulten de λ de tal forma que X sea distinto de cero se les denomina **valores propios** de la matriz A , y los valores correspondientes de X se denominan **vectores propios** de la matriz A .

Los cálculos para obtener los vectores propios y los correspondientes valores propios en matrices pequeñas es un proceso matemático relativamente sencillo; pero al aumentar el tamaño de las matrices los cálculos se pueden complicar considerablemente.

Para evitarnos el problema de los cálculos complicados, Matlab nos proporciona una forma de encontrar los vectores y valores propios de una matriz:

```
[112] >>eig ( A );
ó
    >>[ V, L ] = eig ( A );
```

En su forma básica, el comando nos proporciona un vector columna que contiene los valores propios de la matriz A .

En la segunda forma, `eig` nos regresa una matriz cuadrada V que contiene a los vectores propios de A como columnas y también obtiene otra matriz cuadrada L , que contiene los valores propios (λ) de la matriz A situados a lo largo de la diagonal principal.

Los valores que resultan V y L , son tales que cumplen con las características:

$$V * V' = I \quad (\text{matriz identidad})$$

y

$$A * V = V * L$$

Además, el producto de los valores propios de una matriz es numéricamente igual al valor del determinante de dicha matriz.

Por otro lado, tenemos que el comando:

```
[113] >>trace ( A );
```

Nos permite obtener la suma de los elementos de la diagonal principal de la matriz A ; valor que es numéricamente igual a la suma de los valores propios de dicha matriz.

También se les conoce como Eigenvectores y Eigenvalores. Estos valores característicos de una matriz toman relevancia en ciertas aplicaciones de la ingeniería relacionadas con las áreas de control y simulación de sistemas.

❖ Solución a sistemas de ecuaciones lineales

Matlab facilita en gran medida la solución de sistemas de ecuaciones lineales del tipo:

$$(a) \quad A * x = b$$

Donde A representa la matriz de coeficientes del conjunto de m ecuaciones, x representa un vector columna de las n incógnitas y b es un vector columna que contiene los n valores de los términos independientes del sistema.

La ecuación anterior también podría haberse escrito como:

$$(b) \quad x' * A' = b'$$

La única diferencia es que en este caso A es de dimensiones $n \times m$ y x' y b' son vectores fila.

Para obtener la solución del caso representado con la ecuación en (a), matemáticamente haríamos:

$$x = \frac{b}{A}$$

Que equivale a:

$$x = A^{-1} * b$$

Esto se resuelve muy fácilmente en Matlab con:

$$\gg x = \text{inv} (A) * b ;$$

Otra solución para el mismo caso es ocupar el operador de división izquierda (un poco empolvado por el desuso) en la forma:

$$\gg x = A \backslash b ;$$

El resultado en ambos casos es el mismo pero las operaciones de cálculo que realiza la computadora son muchas menos y se obtiene una ganancia en el tiempo de respuesta (con matrices grandes llega a ser hasta del 50% de tiempo menos con el segundo método).

Para resolver el sistema de acuerdo con la representación de la ecuación en (b) hacemos:

$$\gg x = b' / A' ;$$

Otras formas de escribir expresiones que arrojen el mismo resultado para el caso (a) serían:

$$\gg x = A \wedge (-1) * b ;$$

$$\gg x = b' * \text{inv} (A') ;$$

La primera, genera como respuesta un vector columna y la segunda, un vector fila.

Ejemplo, resuelva el sistema:

$$\begin{pmatrix} 2 & 2 & -4 \\ -3 & 4 & 1 \\ 3 & 2 & -3 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -4 \\ 12 \\ 6 \end{pmatrix}$$

❖ Otras funciones de manipulación matricial

- Para girar una matriz de izquierda a derecha

```
[114] >>fliplr ( A );
```

Voltea las columnas en la dirección de izquierda a derecha.

- Para girar una matriz de arriba hacia abajo

```
[115] >>flipud ( A );
```

Voltea las filas en la dirección de arriba hacia abajo.

- Para rotar una matriz 90 grados

```
[116] >>rot90 ( A );
```

Rota la matriz 90 grados en la dirección contraria a las manecillas del reloj.

- Para extraer la parte triangular inferior

```
[117] >>tril ( A );
```

Deja solo los elementos de la diagonal principal hacia abajo. La parte superior con ceros.

- Para extraer la parte triangular superior

```
[118] >>triu ( A );
```

Deja solo los elementos de la diagonal principal hacia arriba. La parte inferior con ceros.

- Para crear o extraer elementos de la diagonal principal

```
[119] >>diag ( x );  
ó  
>>z = diag ( A );
```

Si x es un vector (fila o columna) de n elementos, la primera opción crea una matriz diagonal cuadrada $n \times n$ en donde los elementos de x forman la diagonal principal y el resto queda en ceros.

Si A es una matriz cuadrada, la segunda opción crea un vector columna z de tamaño n en el cuál los elementos serán los correspondientes a la diagonal principal de A .

- Para crear una **matriz de Pascal**

```
[120] >>pascal ( n );
```

Crea una matriz de Pascal de orden n de números enteros a partir del triángulo de Pascal.

- Para crear un **cuadrado mágico**

```
[121] >>magic ( n );
```

Crea una matriz cuadrada de tamaño n de números enteros en donde tenemos sumas iguales en filas y columnas.