UNIVERSIDAD DE IBAGUÉ

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA MECÁNICA

Introducción al Método de Elementos Finitos
Revisión del Álgebra Matricial con Matlab

Ing. Alfonso Cubillos V.

Notas del curso dictado en el segundo semestre del 2004

Revisión de Álgebra Matricial

Para una mejor comprensión de los procedimientos matriciales utilizados en la implementación computacional del método de elementos finitos se presentará un breve resumen de álgebra matricial y la aplicación de algunas funciones utilizadas en Matlab (definidas por los cuadros).

¿Qué es MatLab?
MatLab es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros de Control en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa.
MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos.
MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.
El nombre de MATLAB proviene de la contracción de los términos MATrix LABoratory y fue inicialmente concebido para proporcionar fácil acceso a las librerías LINPACK y EISPACK, las cuales representan hoy en dia dos de las librerías más importantes en computación y cálculo matricial.
MATLAB es un sistema de trabajo interactivo cuyo elemento básico de trabajo son las matrices. El programa permite realizar de un modo rápido la resolución numérica de problemas en un tiempo mucho menor que si se quisiesen resolver estos mismos problemas con lenguajes de programación tradicionales como pueden ser los lenguajes Fortran, Basic o C.
MATLAB goza en la actualidad de un alto nivel de implantación en escuelas y centros universitarios, así como en departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales. En entornos universitarios, por ejemplo, MATLAB se ha convertido en una herramienta básica, tanto para los profesionales e investigadores de centros docentes, como una importante herramienta para la impartición de cursos universitarios, tales como sistemas e ingeniería de control, álgebra lineal, proceso digital de imagen, señal, etc. En el mundo industrial, MATLAB está siendo utilizado como herramienta de investigación para la resolución de complejos problemas planteados en la

realización y aplicación de modelos matemáticos en ingeniería. Los usos más característicos de la herramienta los encontramos en áreas de computación y cálculo numérico tradicional, prototipaje algorítmico, teoría de control automático, estadística, análisis de series temporales para el proceso digital de señal y análisis de vibraciones.

MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyos especializados, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neuronales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc.
Además también se dispone del programa Simulink que es un entorno gráfico interactivo con el que se puede analizar, modelizar y simular la dinámica de sistemas no lineales.

1. ALGEBRA MATRICIAL ELEMENTAL

No es necesario comprender el álgebra matricial para usar un programa de elementos finitos. Sin embargo, la explicación de los algoritmos y como se comportan los elementos puede expresarse convenientemente mediante matrices.
Una matriz contiene números dispuestos en filas y columnas. Una matriz será indicada usando letras mayúsculas en negritas y sus coeficientes serán indicados con dos subíndices asociados, respectivamente, con la fila y columna del coeficiente. Si una matriz tiene *m* filas y *n* columnas se dice que es de dimensión $m \times n$. Por ejemplo, la matriz **A** de dimensión *2x3* será:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Esta es una matriz rectangular. Si la matriz tuviera el mismo número *n* de filas y de columnas se dice que es una matriz *cuadrada* y *n* es llamado el *orden* de esta matriz. Por ejemplo, la siguiente matriz **B** es una matriz cuadrada de orden 3:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Ya se ha comentado que MATLAB es fundamentalmente un programa para cálculo matricial.
Inicialmente se utilizará MATLAB como ***programa interactivo***, en el que se irán definiendo las matrices, los vectores y las expresiones que los combinan y obteniendo los resultados sobre la marcha. Si estos resultados son asignados a

otras variables podrán ser utilizados posteriormente en otras expresiones. En este sentido MATLAB sería como una potente calculadora matricial (en realidad es esto y mucho más...).
Antes de tratar de hacer cálculos complicados, la primera tarea será aprender a introducir matrices y vectores desde el teclado. Más adelante se verán otras formas más potentes de definir matrices y vectores.

Definición de matrices desde teclado

Como en casi todos los lenguajes de programación, en MATLAB las matrices y vectores son ***variables*** que tienen ***nombres***. Ya se verá luego con más detalle las reglas que deben cumplir estos nombres. Por el momento se sugiere que se utilicen letras mayúsculas para matrices y minúsculas para vectores y escalares (MATLAB no exige esto, pero puede resultar útil).
Para definir una matriz *no hace falta establecer de antemano su tamaño* (de hecho, se puede definir un tamaño y cambiarlo posteriormente). MATLAB determina el número de filas y de columnas en función del número de elementos que se proporcionan (o se utilizan). ***Las matrices se definen por filas***; los elementos de una misma fila están separados por ***blancos*** o ***comas***, mientras que las filas están separadas por pulsaciones ***intro*** o por caracteres ***punto y coma*** (;). Por ejemplo, el siguiente comando define una matriz **A** de dimensión (3x3):

```
» A=[1 2 3; 4 5 6; 7 8 9]
```

La respuesta del programa es la siguiente:

```
A =
     1   2   3
     4   5   6
     7   8   9
```

A partir de este momento la matriz **A** está disponible para hacer cualquier tipo de operación con ella (además de valores numéricos, en la definición de una matriz o vector se pueden utilizar expresiones y funciones matemáticas).

Existen varios tipos de matrices cuadradas. Cuando los coeficientes b_{ij} de la matriz **B** satisfacen la relación

$$b_{ij} = b_{ji} \quad \text{para i, j = 1,2,3,…,n}$$

entonces la matriz **B** es llamada *matriz simétrica.* Si **B** es una matriz simétrica tendrá la siguiente estructura:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

Los coeficientes con índices iguales como b_{11}, b_{22},..., b_{nn} forman la *diagonal principal* de la matriz cuadrada. Cuando todos los coeficientes de una matriz cuadrada **B** son nulos, excepto para la diagonal principal, como por ejemplo

$$\mathbf{B} = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & b_{33} \end{bmatrix}$$

en este caso diremos que **B** es una *matriz diagonal.*

En MATLAB se accede a los elementos de un vector poniendo el índice entre paréntesis (por ejemplo *x(3)* ó *x(i)*). Los elementos de las matrices se acceden poniendo los dos índices entre paréntesis, separados por una coma (por ejemplo ***A****(1,2)* ó ***A****(i,j)*). Las matrices ***se almacenan por columnas*** (aunque ***se introduzcan por filas***, como se ha dicho antes), y teniendo en cuenta esto puede accederse a cualquier elemento de una matriz con un sólo subíndice. Por ejemplo, si **A** es una matriz (3x3) se obtiene el mismo valor escribiendo *A(1,2)* que escribiendo *A(4)*.

En particular si todos los coeficientes de una matriz diagonal valen 1, la matriz se denomina matriz *unidad* ó matriz *identidad* y se simboliza como **I**. Por ejemplo, la siguiente es una matriz identidad de orden 3:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matriz identidad se puede generar directamente:

```
» I = eye(3)
```

La respuesta del programa es la siguiente:

```
I =
     1  0  0
     0  1  0
     0  0  1
```

Otros tipos particulares de matrices cuadradas son las matrices triangulares que presentan todos sus coeficientes nulos por debajo ó por encima de la diagonal. Si todos los coeficientes por debajo de la diagonal son nulos tenemos una *matriz triangular superior*, por ejemplo:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \end{bmatrix}$$

Si todos los coeficientes por encima de la diagonal son nulos tenemos una *matriz triangular inferior*, por ejemplo:

$$\mathbf{B} = \begin{bmatrix} b_{11} & 0 & 0 \\ b_{12} & b_{22} & 0 \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

Si una matriz tiene una única columna ($n = 1$) se lo llama *vector columna* y lo notaremos con una letra minúscula en negrita y a sus coeficientes los indicamos con un único subíndice asociado al número de fila, así por ejemplo el vector columna **a** de m filas será

$$\mathbf{a} = \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{Bmatrix}$$

Si una matriz tiene una única fila (m = 1) se lo llama *vector fila* y nuevamente a sus coeficientes los indicamos con un único subíndice asociado al número de columna, así por ejemplo el vector fila **b** de n columnas será

$$\mathbf{b} = \{b_1 \quad b_2 \quad \cdots \quad b_n\}$$

De forma análoga a las matrices, es posible definir un ***vector fila*** en la forma siguiente (si los tres números están separados por *blancos* o *comas*, el resultado será un vector fila):

```
» b = [10 20 30]     % vector fila
b  =
      10   20   30
```

MATLAB considera ***comentarios*** todo lo que va desde el ***carácter tanto por ciento*** (%) hasta el final de la línea.

Por el contrario, si los números están separados por *intros* o *puntos y coma* (;) se obtendrá un ***vector columna***:

```
» a = [11 ;  12 ;  13]  % vector columna
a  =
      11
      12
      13
```

MATLAB tiene en cuenta la diferencia entre vectores fila y vectores columna.

La *transpuesta* de una matriz **A** es otra matriz con las filas y columnas intercambiadas que la indicamos con una letra T como índice superior, esto es, $\mathbf{A}^{\mathbf{T}}$. Si la matriz **A** es de dimensión $m \times n$, entonces su transpuesta es de dimensión $n \times m$. Así por ejemplo, para la matriz **A** dada tenemos

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix}$$

Una sencilla operación con **A** es hallar su ***matriz traspuesta***. En MATLAB el apóstrofo (') es el símbolo de *trasposición matricial*. Para calcular **A'** (traspuesta de **A**) basta teclear lo siguiente (se añade a continuación la respuesta del programa):

```
» A'
ans =
      1  4  7
      2  5  8
      3  6  9
```

Como el resultado de la operación no ha sido asignado a ninguna otra matriz, MATLAB utiliza un nombre de variable por defecto (*ans*, de *answer*), que contiene el resultado de la última operación. La variable *ans* puede ser utilizada como operando en la siguiente expresión que se introduzca. También podría haberse asignado el resultado a otra matriz llamada B:

```
» B=A'
B =
     1  4  7
     2  5  8
     3  6  9
```

Notemos que si una matriz cuadrada **A** es simétrica entonces debe ser igual a su transpuesta, esto es $\mathbf{A} = \mathbf{A}^T$.

La transpuesta de un vector columna es un *vector fila*, esto es, una matriz de una única fila

$$\mathbf{a}^T = \{a_1 \quad a_2 \quad \cdots \quad a_n\}$$

1.1 Suma y producto de matrices.

Si dos matrices **A** y **B** tienen la misma cantidad de filas y columnas, esto es tienen la misma dimensión, entonces se pueden sumar ó restar término por término

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \qquad \Rightarrow \qquad c_{ij} = a_{ij} + b_{ij}$$

El producto de una matriz **A** de m*x*p por una matriz **B** de p*x*n es una matriz **C** de m*x*n

$$\underset{mxp}{\mathbf{A}}\ \underset{pxn}{\mathbf{B}} = \underset{mxn}{\mathbf{C}} \qquad \text{donde} \qquad c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

Para que el producto **AB** sea posible las matrices **A** y **B** deben ser *conformables*, esto es, el número de columnas de **A** debe ser igual al número de filas de **B**. En general el producto de matrices no es conmutativo, esto es, $\mathbf{AB} \neq \mathbf{BA}$, inclusive en el caso de que ambas matrices sean cuadradas y de igual orden.

Se puede hacer el producto **B*A** (deberá resultar una matriz simétrica):

```
» B*A
ans =
     66   78   90
     78   93  108
     90  108  126
```

La transpuesta de un producto de matrices es igual al producto de las transpuestas en orden inverso.

$$\mathbf{C} = \mathbf{A}\,\mathbf{B} \qquad \text{entonces} \qquad \mathbf{C}^T = \mathbf{A}^T\,\mathbf{B}^T$$

Si **A** es un vector fila definido por la transpuesta de un vector columna **a** de dimensión *n* y **B** es un vector columna **b**, también de dimensión *n,* entonces su producto da como resultado un escalar que se llama producto escalar de los vectores **a**, **b** que viene definido como

$$\mathbf{a}^T\mathbf{b} = a_1b_1 + a_2b_2 + \cdots + a_nb_n$$

MATLAB tiene en cuenta la diferencia entre vectores fila y vectores columna. Por ejemplo, si se intenta sumar los vectores **a** y **b** se obtendrá el siguiente mensaje de error:

```
» a+b
??? Error using ==> +
Matrix dimensions must agree.
```

Estas dificultades desaparecen si se suma **b** con el vector transpuesto de **a**:

```
» a'+b
ans =
        21   32    43
```

Notemos que el producto escalar es conmutativo

$$\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$$

Si **B** es una matriz cuadrada y simétrica de dimensión $m\times m$ y **A** es una matriz de dimensión $m\times n$ entonces la matriz $\mathbf{C} = \mathbf{A}^\mathbf{T}\ \mathbf{B}\ \mathbf{A}$ es también una matriz cuadrada de dimensión $n\times n$ y simétrica, esto es

$$\underset{nxn}{\mathbf{C}} = \underset{nxm}{\mathbf{A}}^T \underset{mxm}{\mathbf{B}} \underset{nxm}{\mathbf{A}} = \underset{nxm}{\mathbf{A}}^T (\underset{mxm}{\mathbf{B}} \underset{nxm}{\mathbf{A}})$$

Si en el caso anterior la matriz **A** es un vector columna **a** entonces el resultado es un escalar c

$$c = \underset{1xm}{\mathbf{a}}^T \underset{mxm}{\mathbf{B}} \underset{1xm}{\mathbf{a}}$$

Si c es siempre positivo para cualquier vector arbitrario **a** entonces la matriz **B** se dice *definida positiva*. Esto vale también cuando **B** sea no simétrica.

El *particionado* de una matriz consiste en subdividir una matriz en submatrices, por ejemplo

$$\mathbf{A} = \left[\begin{array}{cc:ccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ \hdashline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{array}\right] = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

Dos matrices particionadas se pueden multiplicar entre sí considerando cada submatriz como si fueran escalares, pero los productos de submatrices deben ser conformables. Así, por ejemplo, si tenemos dos matrices **A**, **B** particionadas

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \\ \mathbf{A}_{31} & \mathbf{A}_{32} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}$$

su producto vale

$$\mathbf{A\,B}=\begin{bmatrix}\mathbf{A}_{11}\mathbf{B}_{11}+\mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12}+\mathbf{A}_{12}\mathbf{B}_{22}\\ \mathbf{A}_{21}\mathbf{B}_{11}+\mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{12}\mathbf{B}_{12}+\mathbf{A}_{22}\mathbf{B}_{22}\\ \mathbf{A}_{31}\mathbf{B}_{11}+\mathbf{A}_{32}\mathbf{B}_{21} & \mathbf{A}_{31}\mathbf{B}_{12}+\mathbf{A}_{32}\mathbf{B}_{22}\end{bmatrix}$$

donde hemos asumido que los productos de submatrices son conformables.

La *inversa* de una matriz cuadrada **A**, que la indicamos como $\mathbf{A}^{-1}$, cumple la propiedad:

$$\mathbf{A\,A}^{-1}=\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$$

Invertir una matriz es casi tan fácil como trasponerla. A continuación se va a definir una nueva matriz **A** -no singular- en la forma:

```
» A = [1 4 -3; 2 1 5; -2 5 3]
A =
   1  4  -3
   2  1   5
  -2  5   3
```

Ahora se va a calcular la inversa de **A** y el resultado se asignará a **B**. Para ello basta hacer uso de la función ***inv( )*** (la precisión o número de cifras con que se muestra el resultado se puede cambiar con el menú ***File/Preferences/General***):

```
» B = inv(A)
B =
   0.1803   0.2213  -0.1885
   0.1311   0.0246   0.0902
  -0.0984   0.1066   0.0574
```

Para comprobar que este resultado es correcto basta pre-multiplicar **A** por **B**;

```
» B*A
ans =
      1.0000   0.0000   0.0000
      0.0000   1.0000   0.0000
      0.0000   0.0000   1.0000
```

También se cumplen las siguientes relaciones:

$(\mathbf{AB})^{-1} = \mathbf{A}^{-1}\,\mathbf{B}^{-1}$

$(\mathbf{A}^{T})^{-1} = (\mathbf{A}^{-1})^{T}$

$(k\mathbf{A})^{-1} = (1/k)\,\mathbf{A}^{-1}$

Si $\mathbf{A}^{-1} = \mathbf{A}^{T}$ entonces A es una matriz *ortogonal*.

Una condición necesaria para que exista la inversa de una matriz es que dicha matriz no sea singular, es decir que el determinante de la matriz sea diferente de cero ($\det(\mathbf{A}) \neq 0$).

Se llama *determinante* de una matriz cuadrada **A** de orden n, y lo indicamos como $|\mathbf{A}|$, a la suma de todos los productos posibles de n de sus coeficientes de manera que en cada producto solo exista un elemento de cada fila y uno de cada columna. Cada uno de estos productos irá precedido por un signo más ó menos, según la siguiente regla: unimos por pares mediante segmentos de recta los elementos que intervienen en un determinado producto, si el número total de segmentos con pendiente hacia arriba y a la derecha es par, el producto llevará signo positivo, caso contrario llevará signo negativo. Así, por ejemplo, para una matriz de orden 3

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

su determinante vale

$$|\mathbf{A}| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

Una matriz se dice *singular* si su determinante es nulo.

El determinante de una matriz se calcula con la función ***det( )*** de la forma:

```
» det(A)
ans =
      -122
```

Un múltiplo escalar λ de una matriz **A** actúa sobre cada coeficiente de la matriz, esto es

$$\lambda\,\mathbf{A} = \lambda \begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & \cdots & \mathrm{a}_{1n} \\ \mathrm{a}_{21} & \mathrm{a}_{22} & \cdots & \mathrm{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{a}_{m1} & \mathrm{a}_{m2} & \cdots & \mathrm{a}_{mn} \end{bmatrix} = \begin{bmatrix} \lambda\mathrm{a}_{11} & \lambda\mathrm{a}_{12} & \cdots & \lambda\mathrm{a}_{1n} \\ \lambda\mathrm{a}_{21} & \lambda\mathrm{a}_{22} & \cdots & \lambda\mathrm{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda\mathrm{a}_{m1} & \lambda\mathrm{a}_{m2} & \cdots & \lambda\mathrm{a}_{mn} \end{bmatrix}$$

1.2 Integración y derivación de matrices.

Además, la integral de una matriz con respecto a un parámetro escalar, es una matriz que contiene las integrales de cada término, esto es

$$\int \mathbf{A}dt = \int \begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & \cdots & \mathrm{a}_{1n} \\ \mathrm{a}_{21} & \mathrm{a}_{22} & \cdots & \mathrm{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{a}_{m1} & \mathrm{a}_{m2} & \cdots & \mathrm{a}_{mn} \end{bmatrix} dt = \begin{bmatrix} \int \mathrm{a}_{11}dt & \int \mathrm{a}_{12}dt & \cdots & \int \mathrm{a}_{1n}dt \\ \int \mathrm{a}_{21}dt & \int \mathrm{a}_{22}dt & \cdots & \int \mathrm{a}_{2n}dt \\ \vdots & \vdots & \ddots & \vdots \\ \int \mathrm{a}_{m1}dt & \int \mathrm{a}_{m2}dt & \cdots & \int \mathrm{a}_{mn}dt \end{bmatrix}$$

Análogamente, la derivada de una matriz con respecto a una variable *x* es una matriz que contiene las derivadas de cada término, esto es

$$\frac{\partial \mathbf{A}}{\partial x} = \frac{\partial}{\partial x}\begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & \cdots & \mathrm{a}_{1n} \\ \mathrm{a}_{21} & \mathrm{a}_{22} & \cdots & \mathrm{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{a}_{\mathrm{m}1} & \mathrm{a}_{\mathrm{m}2} & \cdots & \mathrm{a}_{\mathrm{mn}} \end{bmatrix} = \begin{bmatrix} \frac{\partial a_{11}}{\partial x} & \frac{\partial a_{12}}{\partial x} & \cdots & \frac{\partial a_{1n}}{\partial x} \\ \frac{\partial a_{21}}{\partial x} & \frac{\partial a_{22}}{\partial x} & \cdots & \frac{\partial a_{2n}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_{m1}}{\partial x} & \frac{\partial a_{m2}}{\partial x} & \cdots & \frac{\partial a_{mn}}{\partial x} \end{bmatrix}$$

Uno de los toolbox más potentes que posee Matlab, es el *Symbolic*. Este permite realizar una gran cantidad de operaciones de tipo simbólico (como factorización, sustitución, derivadas, integrales y muchas más). Lo primero que se debe realizar es definir las variables simbólicas que se van a utilizar:

```
» syms a x
```

Después se definen las operaciones simbólicas, por ejemplo en este caso la matriz de transformación **R**, que se verá más adelante:

```
» R = [cos(a*x) sin(a*x) ; -sin(a*x) cos(a*x)]
R =
    [  cos(a*x)   sin(a*x)  ]
    [ -sin(a*x)   cos(a*x)  ]
» diff(A) %Calcula la derivada de la matriz A
ans =
    [  -sin(a*x)*a   cos(a*x)*a  ]
    [ -cos(a*x)*a    sin(a*x)*a  ]
» int(A) %Calcula la integral de la matriz A
ans =
    [  sin(a*x)/a      -1/a*cos(a*x)  ]
    [ 1/a*cos(a*x)    sin(a*x)/a  ]
```

Sea ϕ una cantidad escalar, entonces la derivada de ϕ con respecto a un vector **a** es

$$\frac{\partial \phi}{\partial \mathbf{a}} = \left\{ \frac{\partial \phi}{\partial a_1} \quad \frac{\partial \phi}{\partial a_2} \quad \cdots \quad \frac{\partial \phi}{\partial a_n} \right\}$$

Para el caso particular de la expresión *c,* se puede demostrar que

$$c = \underset{1xm}{\mathbf{a}}^T \underset{mxm}{\mathbf{B}} \underset{1xm}{\mathbf{a}}, \qquad \frac{\partial c}{\partial \mathbf{a}} = (\mathbf{B} + \mathbf{B}^T)\mathbf{a}$$

También se puede demostrar que para un producto escalar se cumple

$$\frac{\partial(\mathbf{a}^t \mathbf{b})}{\partial \mathbf{a}} = \mathbf{b}, \qquad \frac{\partial(\mathbf{a}^t \mathbf{b})}{\partial \mathbf{b}} = \mathbf{a}$$

Jacobiano
Si una matriz **f**(**x**) es una función vectorial de un vector n-dimencional **x**, entonces

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

La función *jacobian* puede ser utilizada para realizar esta operación.

```
» syms x1 x2
» F = 2*x1^2+3*x2+4;
» x = [x1 x2];
» D = jacobian(F,x)
D =
     [  4*x1,        3   ]
```

1.3 Rotación de vectores.
Un vector **v**, interpretado como una magnitud orientada, puede representarse en un sistema de coordenadas cartesianas mediante las coordenadas de su extremo, cuando su inicio coincide con el origen del sistema. (fig.1).

$$\mathbf{v} = \begin{Bmatrix} v_x \\ v_y \end{Bmatrix}$$

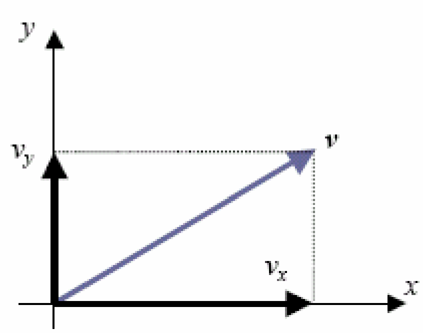

Figura 1. Componentes de un vector.

Supongamos que deseamos las componentes de este mismo vector pero respecto de otro sistema cartesiano rotado un ángulo α respecto del anterior (fig. 2).

$$\mathbf{v}' = \begin{Bmatrix} v'_x \\ v'_y \end{Bmatrix}$$

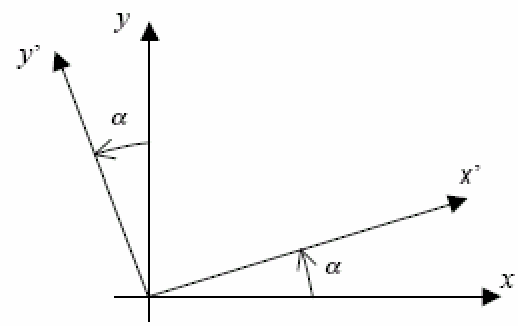

Figura 2. Sistema rotado de coordenadas.

Luego las componentes en el sistema *x*', *y*' del vector **v**' se pueden expresar como:

$$v'_x = v_x \cdot \cos(\alpha) + v_y \cdot \sin(\alpha)$$

$$v'_y = -v_x \cdot \sin(\alpha) + v_y \cdot \cos(\alpha)$$

En forma matricial esta relación se puede escribir como:

$$\begin{Bmatrix} v'_x \\ v'_y \end{Bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \end{Bmatrix}$$

o sea

$$\mathbf{v}' = \mathbf{R}\,\mathbf{v}$$

donde **R** es la *matriz de rotación*

$$\mathbf{R} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

La relación inversa viene dada por la inversa de la matriz de rotación, pues si multiplicamos a la ecuación $\mathbf{v}' = \mathbf{R}\,\mathbf{v}$ por $\mathbf{R}^{-1}$ y en virtud de la propiedad $\mathbf{A}\,\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ tenemos

$$\mathbf{v} = \mathbf{R}^{-1}\,\mathbf{v}'$$

Se puede demostrar que para sistemas cartesianos ortogonales la inversa de la matriz de rotación debe ser igual a su transpuesta, esto es:

$$\mathbf{R}^T = \mathbf{R}^{-1}$$

1.4 Rotación de matrices.

Consideremos la siguiente ecuación matricial

$$\mathbf{u} = \mathbf{A}\,\mathbf{v}$$

La matriz **A** puede interpretarse como una transformación lineal que a cada vector **v** de un sistema de coordenadas cartesianas lo transforma en un vector **u** expresado en el mismo sistema.

Nos interesa conocer como debería expresarse la matriz **A** si los vectores **v**, **u** estuvieran referidos a un sistema rotado de coordenadas, esto es

$$\mathbf{u}' = \mathbf{A}'\,\mathbf{v}'$$

Por la transformación de rotación tenemos

$$\mathbf{v}' = \mathbf{R}\,\mathbf{v}$$

$$\mathbf{u}' = \mathbf{R}\,\mathbf{u}$$

Substituyendo resulta

$$\mathbf{R}\,\mathbf{u} = \mathbf{A}'\mathbf{R}\,\mathbf{v}$$

Si premultiplicamos la ec. $\mathbf{u} = \mathbf{A}\,\mathbf{v}$ por $\mathbf{R}$ obtenemos

$$\mathbf{Ru} = \mathbf{RA}\,\mathbf{v}$$

Comparando estas dos últimas ecuaciones resulta

$$\mathbf{A}'\mathbf{R} = \mathbf{RA}$$

Luego, posmultiplicando esta ecuación por $\mathbf{R}^{-1}$ tenemos

$$\mathbf{A}'\mathbf{RR}^{-1} = \mathbf{A}' = \mathbf{RAR}^{-1}$$

y en virtud de la ec. $\mathbf{R}^T = \mathbf{R}^{-1}$ resulta

$$\mathbf{A}' = \mathbf{RAR}^T$$

Notemos que si premultiplicamos la ec. $\mathbf{A}'\mathbf{R} = \mathbf{RA}$ por $\mathbf{R}^{-1}$ obtenemos en virtud de la ec. $\mathbf{R}^T = \mathbf{R}^{-1}$

$$\mathbf{A} = \mathbf{R}^T\mathbf{A}'\mathbf{R}$$

que puede ser útil si conocemos la transformación en el sistema rotado.

1.5 Solución de sistemas de ecuaciones lineales.

La solución de muchos problemas de ingeniería puede reducirse a hallar la solución de uno ó varios sistemas algebraicos de n ecuaciones lineales con n incógnitas. Tales sistemas pueden escribirse como

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}$$

En forma matricial estas ecuaciones se pueden expresar como

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

Siendo **A** la matriz de los coeficientes del sistema de dimensión n x n, esto es

$$\mathbf{A}=\begin{bmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & \cdots & \mathrm{a}_{1n} \\ \mathrm{a}_{21} & \mathrm{a}_{22} & \cdots & \mathrm{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{a}_{\mathrm{n}1} & \mathrm{a}_{\mathrm{n}2} & \cdots & \mathrm{a}_{\mathrm{nn}} \end{bmatrix}$$

x es el vector de las incógnitas

$$\mathbf{x}=\begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}$$

y **b** es el vector de términos independientes

$$\mathbf{b}=\begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}$$

1.6 Métodos directos e iterativos.

La solución de muchos problemas de ingeniería puede reducirse a hallar la solución de uno En general, tenemos dos grandes familias de métodos para resolver estos sistemas de ecuaciones: métodos *directos* y métodos *iterativos*. Los métodos directos se caracterizan por obtener la solución del sistema en un número fijo de operaciones. Los métodos iterativos se basan en obtener la solución a partir de una solución aproximada inicial, la cual es sucesivamente mejorada, deteniéndose el proceso cuando la mejora obtenida es insignificante.

En general, los métodos usualmente empleados en los programas de elementos finitos son los directos. Los métodos iterativos presentan la ventaja frente a los directos de requerir mucha menor cantidad de memoria y almacenamiento que los métodos directos, y actualmente se están popularizando debido a ventajas de implementación en computadoras paralelas. Pero por otro lado presentan la desventaja de no poder asegurar a priori la convergencia hacia a la solución ni la velocidad de convergencia. Por este motivo prácticamente todos los programas comerciales utilizan métodos directos, ofreciendo algunos de ellos la variante de utilizar métodos iterativos a pedido del usuario pero que cambian a métodos directos si la velocidad de convergencia es muy lenta.

1.7 Método de eliminación de Gauss.

Entre los métodos directos, uno de los más populares es el método de eliminación de Gauss. Para analizar su funcionamiento consideremos el siguiente sistema de ecuaciones:

$$6x_1 + 3x_2 + 6x_6 = 30$$
$$2x_1 + 3x_2 + 3x_3 = 17$$
$$x_1 + 2x_2 + 2x_3 = 11$$

Si dividimos la primera ecuación por 6 tenemos

$$\begin{aligned} x_1 + 1/2\,x_2 + x_3 &= 5 \\ 2x_1 + 3x_2 + 3x_3 &= 17 \\ x_1 + 2x_2 + 2x_3 &= 11 \end{aligned}$$

Luego podemos eliminar x_1 de las dos últimas ecuaciones substituyendo por el valor de x_1 dado por la primera ecuación

$$x_1 = 5 - 1/2\,x_2 - x_3$$

obteniendo

$$\begin{aligned} x_1 + 1/2\,x_2 + x_3 &= 5 \\ 2x_2 + x_3 &= 7 \\ 3/2\,x_2 + x_3 &= 6 \end{aligned}$$

Dividiendo la segunda ecuación por 2 tenemos

$$\begin{aligned} x_1 + 1/2\,x_2 + x_3 &= 5 \\ x_2 + 1/2\,x_3 &= 7/2 \\ 3/2\,x_2 + x_3 &= 6 \end{aligned}$$

Luego podemos obtener x_2 de la segunda ecuación como

$$x_2 = 7/2 - 1/2\,x_3$$

y substituyendo en la tercera ecuación obtenemos

$$\begin{aligned} x_1 + 1/2\,x_2 + x_3 &= 5 \\ x_2 + 1/2\,x_3 &= 7/2 \\ 1/4\,x_3 &= 3/4 \end{aligned}$$

Esta parte del método de Gauss se denomina *proceso de eliminación*, pues se van eliminando incógnitas de las filas inferiores de cada ecuación eliminando de a una incógnita por vez.

Terminado este proceso la matriz del sistema se ha transformado en una matriz triangular superior, esto es, con coeficientes no nulos solo por encima ó dentro de la diagonal principal de la matriz.

Luego podemos obtener las incógnitas de a una por vez partiendo de la última fila y reemplazando en las filas superiores, obteniendo en este caso

$$\begin{aligned} x_3 &= 3 \\ x_2 &= 7/2 - 3/2 = 2 \\ x_1 &= 5 - 1 - 3 = 1 \end{aligned}$$

Este proceso se denomina de *retrosubstitución.*

Una condición necesaria para poder aplicar el método de Gauss es que la matriz del sistema no sea singular, es decir debe tener inversa definida.

Considérese el siguiente sistema de ecuaciones lineales ya presentado,

$$\mathbf{Ax} = \mathbf{b}$$

en donde **x** y **b** son vectores columna, y **A** una matriz cuadrada invertible. La resolución de este sistema de ecuaciones se puede escribir en las 2 formas siguientes (¡Atención a la 2ª forma, basada en la *barra invertida* (\), que puede resultar un poco extraña!):

x = inv(**A**)*__b__

x = **A\b**

Así pues, el operador *división-izquierda* por una matriz (barra invertida \) equivale a premultiplicar por la inversa de esa matriz.

1.8 Autovalores y autovectores de una matriz cuadrada.

Si para una matriz cuadrada **A** existe un vector **v** que satisface la siguiente ecuación

$$\mathbf{A\,v} = \lambda\,\mathbf{v}$$

entonces se dice que **v** es un autovector de **A** y λ es su autovector asociado. Se puede demostrar que para una matriz cuadrada de orden *n* tiene exactamente *n* autovectores y autovalores.

Se puede mostrar que si una matriz es singular debe tener al menos un autovalor nulo. Además, se puede mostrar que si una matriz es definida positiva todos sus autovalores deben ser estrictamente positivos. Por lo tanto los sistemas de ecuaciones cuyas matrices sean definidas positivas siempre tienen solución pues se puede asegurar que la matriz del sistema es siempre no singular.

Operaciones con matrices

MATLAB puede operar con matrices por medio de ***operadores*** y por medio de ***funciones***. Se han visto ya los operadores *suma* (+), *producto* (*) y *traspuesta* ('), así como la función *invertir* ***inv( )***.

Los operadores matriciales de MATLAB son los siguientes:

- \+ adición o suma
- – sustracción o resta
- * multiplicación
- ' traspuesta
- ^ potenciación
- \ división-izquierda
- / división-derecha
- .* producto elemento a elemento
- ./ y .\ división elemento a elemento
- .^ elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares, aunque con algunas diferencias5. Todos estos operadores son coherentes con las correspondientes operaciones matriciales: no se puede por ejemplo sumar matrices que no sean del mismo tamaño. Si los operadores no se usan de modo correcto se obtiene un mensaje de error.

Los operadores anteriores se pueden aplicar también de modo ***mixto***, es decir con un operando escalar y otro matricial. En este caso la operación con el escalar se aplica a cada uno de los elementos de la matriz.

Funciones Matemáticas Especiales	
abs(x)	Valor absoluto o magnitud de un numero complejo
angle(x)	Angulo de un numero complejo.
imag(x)	Parte imaginaria de un numero complejo.
real(x)	Parte real de un numero complejo.
round(x)	Redondea hacia el entero más próximo.
ceil(x)	Redondea hacia más infinito.
fix(x)	Redondea hacia cero.
floor(x)	Redondea hacia menos infinito.
conj(x)	Complejo conjugado.
exp(x)	Exponencial *e.*
log(x)	Logaritmo Natural.
log10(x)	Logaritmo Decimal.
rem(x,y)	Da el resto de x/y
sing(x)	Devuelve el signo del argumento.
sqrt(x)	Raíz cuadrada.
sin(x)	Seno (En radianes).
cos(x)	Coseno (En radianes).
tan(x)	Tangente.
asin(x)	Inversa de seno.
acos(x)	Inversa de coseno.
atan(x)	Inversa de Tangente.
sinh(x)	Seno hiperbólico.
cosh(x)	Coseno hiperbólico.
tanh(x)	Tangente hiperbólico.
asinh(x)	Inversa del seno hiperbólico.
acosh(x)	Inversa del coseno hiperbólico.
atanh(x)	Inversa de la tangente hiperbólica.
atan2(x)	Inversa de la tangente en los cuatro cuadrantes.

Operaciones con arrays	
Datos ilustrativos *a = [a1 a2 ... an] , b = [b1 b2 ... bn] y c = (c es un escalar)*	
Suma escalar	a + c = [a1+c a2+c ... an+c]
Multiplicación escalar	a * c = [a1*c a2*c ... an*c]
Suma de arrays	a + b = [a1+b1 a2+b2 ... an+bn]
Multiplicación de arrays	a.*b = [a1*b1 a2*b2 ... an*bn]
División de arrays	a./b = [a1/b1 a2/b2 ... an/bn]
Potencia de arrays	a.^c = [a1^c a2^c ... an^c]
	c.^a = [c^a1 c^a2 ... c^an]
	a.^b = [a1^b1 a2^b2 ... an^bn]

Funciones Principales del toolbox ‘Symbolic’	
Las operaciones aritméticas y matemáticas comunes pueden ser utilizadas con las variables simbólicas definidas.	
syms x1 x2 …	Define las variables x1, x2, etc en variables simbólicas.
double	Convierte a precisión doble.
expand	Expansión simbólica.
eval	Evalúa la expresión.
numden	Numerador y denominador de una expresión simbólica.
pretty	Muestra la expresión simbólica de forma ‘bonita’
simple	Busca la forma mas simple de una expresión simbólica.
simplify	Simplificación simbólica
symvar	Determina las variables simbólicas en una expresión.
symbfact	Análisis de factorización simbólica
symintro	Introducción al toolbox ‘Symbolic’

Ordenes y Funciones de Gestión	
demo	Corre programas de demostración
expo	Corre el programa de demostración MATLAB Expo
help	Documentación en línea.
info	Información sobre MATLAB
lasterr	Ultimo mensaje de error generado.
lookfor	Búsqueda de palabras clave en las entradas de help.
path	Control del camino de búsqueda de MATLAB.
subscribe	Suscribirse como usuario de MATLAB.
type	Lista una archivo M.
ver	Versiones actuales de MATLAB
version	Numero de la versión actual de MATLAB.
what	Listado de los archivos M, MAT y MEX del directorio.
whatsnew	Muestra los archivos README de MATLAB y del Toolboxes.
which	Localiza funciones y archivos.

Tomado de:

- Introducción al método de elementos finitos: Revisión del álgebra matricial. Dr. Ing. Claudio E. Jouglard. Universidad Tecnológica Nacional, Facultad Regional Buenos Aires.

- Aprenda Matlab 5.3 como si estuviera en Primero. Javier Garcia de Jalón, Jose Ignacio Rodrigrez, Alfonso Brazales. Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid.

- Symbolic Math Toolbox, User’s Guide. The MathWorks