Introducción a Octave

para ciencias aplicadas e ingeniería

Unidad 3

Daniel Millán, Nicolas Muzi, Eduardo Rodríguez
San Rafael, Argentina, Abril–Mayo de 2020

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE CIENCIAS APLICADAS A LA INDUSTRIA

CONICET

MoCCAi
MODELADO COMPUTACIONAL EN CIENCIAS APLICADAS E INGENIERÍA

Unidad 3
Funciones de biblioteca

- Octave tiene un gran número de funciones incorporadas, cuyos aspectos más relevantes describiremos en esta Unidad.

- Ciertas funciones vienen incorporadas en el propio código fuente y son particularmente rápidas y eficientes, mientras que otras es posible que sean creadas por el propio usuario.

Unidad 3
Funciones de biblioteca

1. Características generales de las funciones de Octave.
2. Funciones matemáticas elementales que operan de modo escalar y que actúan sobre vectores/matrices.
3. Funciones matriciales elementales y especiales.
4. Factorización y/o descomposición matricial.
5. Más operadores relacionales vectores/matrices.
6. Otras funciones que actúan sobre vectores/matrices.
7. Funciones para cálculos con polinomios.
8. Funciones anónimas @f

1. Características generales de las funciones de Octave.

- En Octave hay diversos tipos de funciones. Una posible clasificación según su finalidad es:
  - Funciones matemáticas elementales.
  - Funciones especiales.
  - Funciones matriciales elementales.
  - Funciones matriciales especiales.
  - Funciones para descomposición y/o factorización de matrices.
  - Funciones para análisis estadístico de datos.
  - Funciones para manejo de conjunto de datos.
  - Funciones para análisis de polinomios e interpolación.
  - Funciones para resolución de ecs. diferenciales ordinarias.
  - Resolución de ecuaciones no-lineales y optimización.
  - Integración numérica.
  - Funciones para procesamiento de señales, sonido e imagen.
  - Geometría (qhull library).
  - ...

1. Características generales de las funciones de Octave.

- Una función tiene ***nombre***, ***valor de retorno*** y ***argumentos***.
- Una función *se "llama"* utilizando su nombre en una expresión o utilizándolo como un comando más.
- Las funciones se pueden definir en ficheros de texto ***.m**

Ejemplo: Llamadas a funciones

```
>> x = [1,-0.1,pi,0.7,-1/3];
>> [maximo, posmax] = max(x);
>> r = sqrt(2^2+3^2) + eps;
>> a = cos(pi/3) - sin(pi/6);
```

✓ Los ***nombres*** de las funciones se han puesto en negrita.

✓ Los ***argumentos*** de cada función van a continuación del nombre entre paréntesis (y separados por comas si hay más de uno).

✓ Los ***valores de retorno*** son el resultado de la función y sustituyen a esta en la expresión donde la función aparece.

1. Características generales de las funciones de Octave.

- Suma/resta de una matriz con un escalar, consiste en sumar/restar el escalar a todos los elementos de la matriz.
- En Octave las funciones pueden tener ***valores de retorno matriciales múltiples***.
- Los valores de retorno ***se recogen entre corchetes***, separados por comas.

Ejemplo: comprobar la sentencia anterior

```
>> A = magic(5)
>> [V, D] = eig(A) %vectores y valores propios de A
>> [xmax, imax] = max(V)
>> xmax = max(x)
```

- Los ***argumentos*** de las funciones pueden ser expresiones y llamadas a otra función.
- Las funciones no modifican las variables que se pasan como argumento, a no ser que se incluyan también como valores de retorno. Se pasan ***por valor***, ***no por referencia*** (es decir se realiza una copia).

1. Características generales de las funciones de Octave.

- **WARNING!** los nombres de las funciones
  ***NO* son palabras reservadas**.
  - Por ejemplo es posible crear una variable llamada ***sin*** o ***cos***, que oculte las funciones correspondientes.
  - Para poder acceder a las funciones originales hay que eliminar (***clear***) las variables del mismo nombre que las ocultan (visibles en el Espacio de Trabajo).

Ejercicio: interprete las siguientes operaciones

```
>> cos = cos(pi/3);
>> sin = sin(pi/3);
>> disp(sqrt(2)*[cos, sin]);
>> clear
>> disp(sqrt(2)*[cos, sin]);
```

1. Características generales de las funciones de Octave.

- Existen funciones que no precisan ser llamadas con argumentos o no llevan paréntesis, por lo que a simple vista no siempre son fáciles de distinguir de las variables y constantes predefinidas.

```
>> help
>> help sin
>> help(sin)
```

Ejemplo: ***clc*** es una función sin argumentos ¿otras?

- **Constantes matemáticas**, variables con valores predefinidos pero que pueden recibir argumentos: ***pi***, ***e***, ***I***, ***eps***, ***Inf***, ***NaN***.

Ejemplo: verificamos el comportamiento cuando se pasan argumentos a las constantes predefinidas.

```
>> pi(2)
>> e(2)
>> I(2)
```

2. Funciones matemáticas elementales que operan de modo **escalar**

- Estas comprenden las funciones matemáticas algebraicas y trascendentales, así como otras funciones básicas.
- Cuando se aplican a una matriz actúan sobre cada elemento de la matriz como si se tratase de un escalar.
- Por tanto, se aplican de la misma forma a escalares, vectores y matrices.
- Algunas de las funciones de este grupo son las siguientes:
  - **sqrt**(x) raíz cuadrada, **pow2**(x) calcula **2.^x**[i] de cada elemento de x
  - **sin**(x) seno, **cos**(x) coseno, **tan**(x) tangente
  - **asin**(x) arco seno, **acos**(x) arco coseno
  - **atan**(x) arco tangente (ángulo entre -π/2 y +π/2)
  - **atan2**(y,x) arco tangente (ángulo entre -π y +π); se le pasan 2 argumentos, proporcionales al seno y coseno
  - **sinh**(x), **cosh**(x), **tanh**(x) seno, coseno y tan hiperbólico
  - **log**(x) logaritmo natural, **exp**(x) función exponencial

2. Funciones que operan sobre **vectores.**

- Las siguientes funciones ***sólo actúan sobre vectores***:
  - **min**(x)/**max**(x) mínimo/maximo elemento de un vector
  - **sum**(x) suma de los elementos de un vector
  - **cumsum**(x) devuelve el vector suma acumulativa de los elementos de un vector
  - **mean**(x) valor medio de los elementos de un vector
  - **std**(x) desviación típica
  - **prod**(x) producto de los elementos de un vector
  - **cumprod**(x) devuelve el vector producto acumulativo de los elementos de un vector
  - [y,i]=**sort**(x) ordenación ascendente de los elementos de **x**

- NOTA: en realidad estas funciones ***se pueden aplicar también a matrices***, pero en ese caso ***se aplican por separado a cada columna de la matriz***, dando como valor de retorno un vector fila de resultado.

3. Funciones que operan sobre **matrices.**

- **Funciones matriciales elementales:**
- B = A**'** calcula la traspuesta (conjugada) de la matriz **A**
- B = A**.'** calcula la traspuesta (sin conjugar) de la matriz **A**
- v = **poly**(A) devuelve un vector **v** con los coeficientes del polinomio característico de la matriz cuadrada **A**
- t = **trace**(A) devuelve la traza **t** (suma de los elementos de la diagonal) de una matriz cuadrada **A**
- [m,n] = **size**(A) devuelve el número de filas **m** y de columnas **n** de una matriz rectangular **A**
- n = **size**(A) devuelve el tamaño de una matriz cuadrada **A**
- nf = **size** (A,**1**) devuelve el número de filas de **A**
- nc = **size** (A,**2**) devuelve el número de columnas de **A**

3. Funciones que operan sobre **matrices.**

- Las funciones ***exp()***, ***sqrt()*** y ***log()*** se aplican ***elemento a elemento*** a las matrices y/o vectores.

- Existen otras funciones similares que se aplican a una matriz como una única entidad.

Funciones matriciales especiales:

- **expm**(A) si $\mathbf{A}=\mathbf{XDX}^T$ ⇒ expm(**A**) = **X***diag(exp(diag(**D**)))***X'**
- **sqrtm**(A) devuelve una matriz que multiplicada por sí misma da la matriz **A**
- **logm**() es la función recíproca de expm(**A**)

3. Funciones que operan sobre **matrices.**

- Aunque no pertenece a esta familia de funciones, el ***operador potencia*** (**^**) está emparentado con ellas:

➢ A^n está definida si **A** es cuadrada y **n** es un número real.
- Si **n** es entero, el resultado se calcula por multiplicaciones sucesivas de A.
- Si **n** es real, el resultado se calcula como:

A^n = **X** * **D**.^n * inv(**X**)

siendo [**X**,**D**]=eig(**A**).

Ejemplo: Verifique 2^**A** = **X***2^**D**/**X**.

4. Funciones de factorización y/o descompocisión matricial.

1) **Basadas en factorización triangular (eliminación de Gauss)**

- U = **chol**(A) descomposición de Cholesky de matriz simétrica y definida positiva. El resultado es una matriz **U** triangular superior tal que **A** = **U'*U**
- [L,U] = **lu**(A) descomposición de Crout (**A** = **LU**). La matriz **L** es una permutación de una matriz triangular inferior
- B = **inv**(A) calcula la inversa de **A**. Equivale a **B**=inv(**U**)*inv(**L**)
- d = **det**(A) devuelve el determinante **d** de la matriz cuadrada **A**. Equivale a d=det(**L**)*det(**U**)
- [E,xc] = **rref**(A) reducción a forma de escalón con un vector **xc** que da una posible base del espacio de columnas de **A**
- c = **rcond**(A) devuelve una estimación del recíproco de la condición numérica de la matriz **A** basada en la norma-1. Si el resultado es próximo a 1 la matriz **A** está bien condicionada;
si es próximo a 0 no lo está.

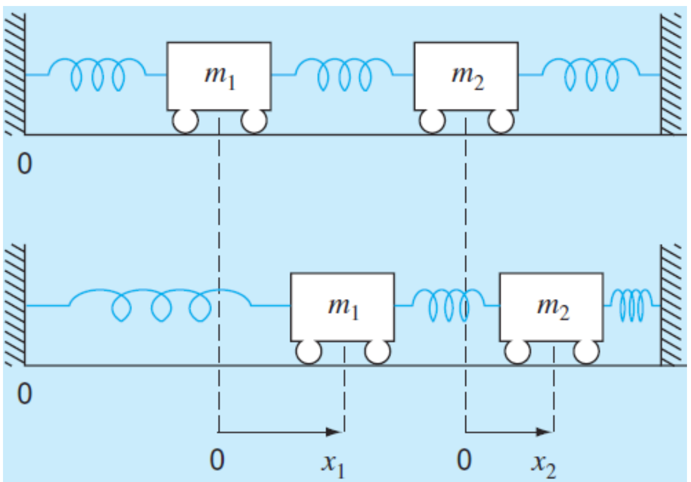
4. Funciones de factorización y/o descompocisión matricial.

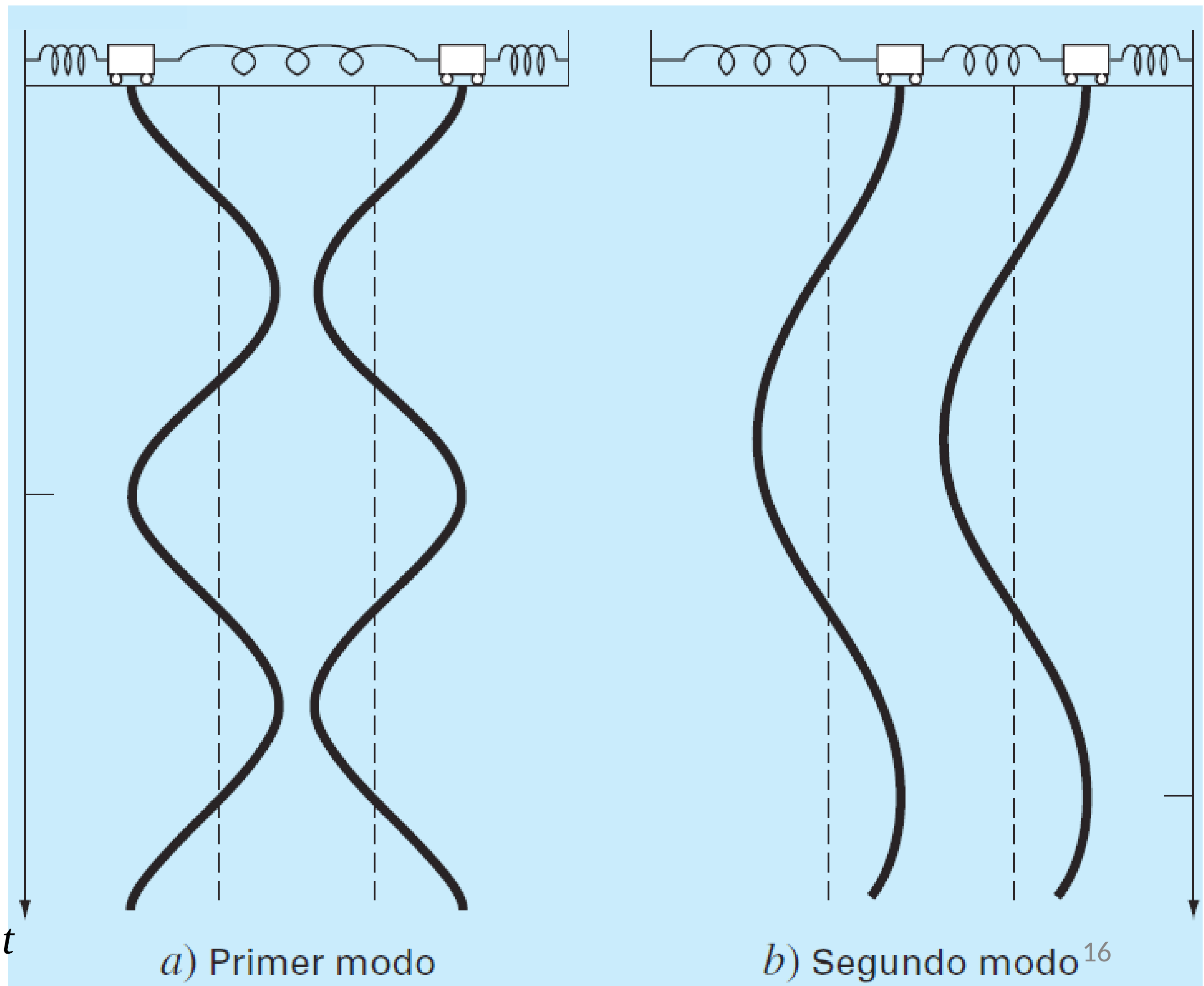
2) Basadas en el cálculo de valores y vectores propios

- El cálculo de los valores propios y de los vectores propios de una matriz simétrica tiene gran importancia en las matemáticas y en la ingeniería.
- Los problemas de *valores propios*, o característicos o eigenvalores, constituyen una clase especial de problemas con valores en la frontera, que son comunes en el contexto de problemas de ingeniería que implican vibraciones, elasticidad y otros sistemas oscilantes.
- Cabe destacar, el problema de la diagonalización de una matriz, el cálculo de los momentos de inercia y de los ejes principales de inercia de un sólido rígido, o de las frecuencias propias de oscilación de un sistema oscilante.

4. Funciones de factorización y/o descompocisión matricial.

2) Basadas en el cálculo de valores y vectores propios





Chapra y Canale, Métodos Numéricos Para Ingeniería, Cap.27, 5ta Ed, 2007.

4. Funciones de factorización y/o descompocisión matricial.

2) Basadas en el cálculo de valores y vectores propios

- [X,D] = **eig**(A) vectores propios (columnas de **X**) y valores propios (diagonal de **D**) de una matriz cuadrada **A**. Con frecuencia el resultado posee números complejos si **A** no es simétrica.

- [X,D] = **eig**(A,B) vectores propios (columnas de **X**) y valores propios (diagonal de **D**) de dos matrices cuadradas **A** y **B** (**Ax** = λ**Bx**).
  Los vectores propios están normalizados de modo que **X'*B*X=I**.

  Cuando **A** es simétrica y **B** es simétrica y definida positiva se puede utilizar [**X**,**D**] = eig(**A**,**B**,'chol').

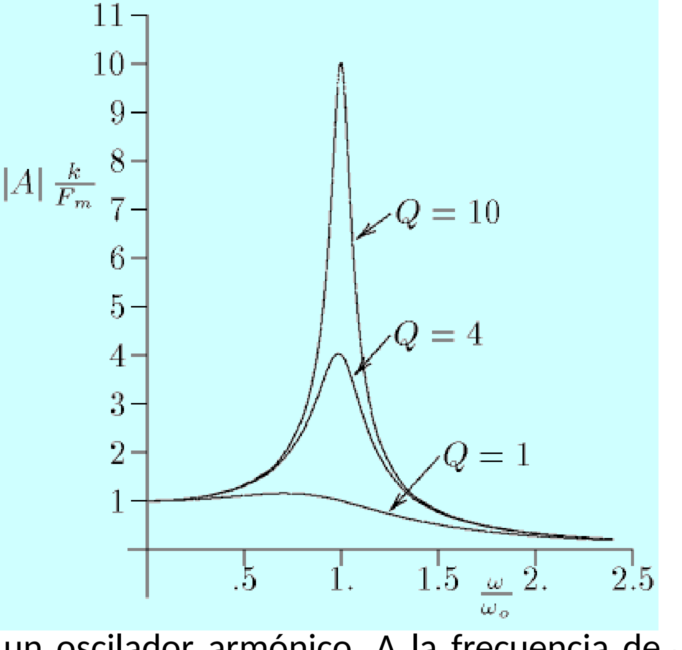
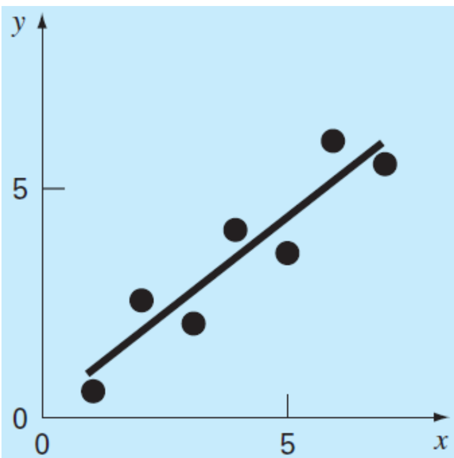


Figura: Respuesta en frecuencia de un oscilador armónico. A la frecuencia de resonancia, la amplitud es Q veces más grande que a muy baja frecuencia. [Wiki]

4. Funciones de factorización y/o descompocisión matricial.

3) Basadas en la descomposición QR

- [Q,R] = **qr**(A) descomposición **A=QR** de una matriz rectangular.



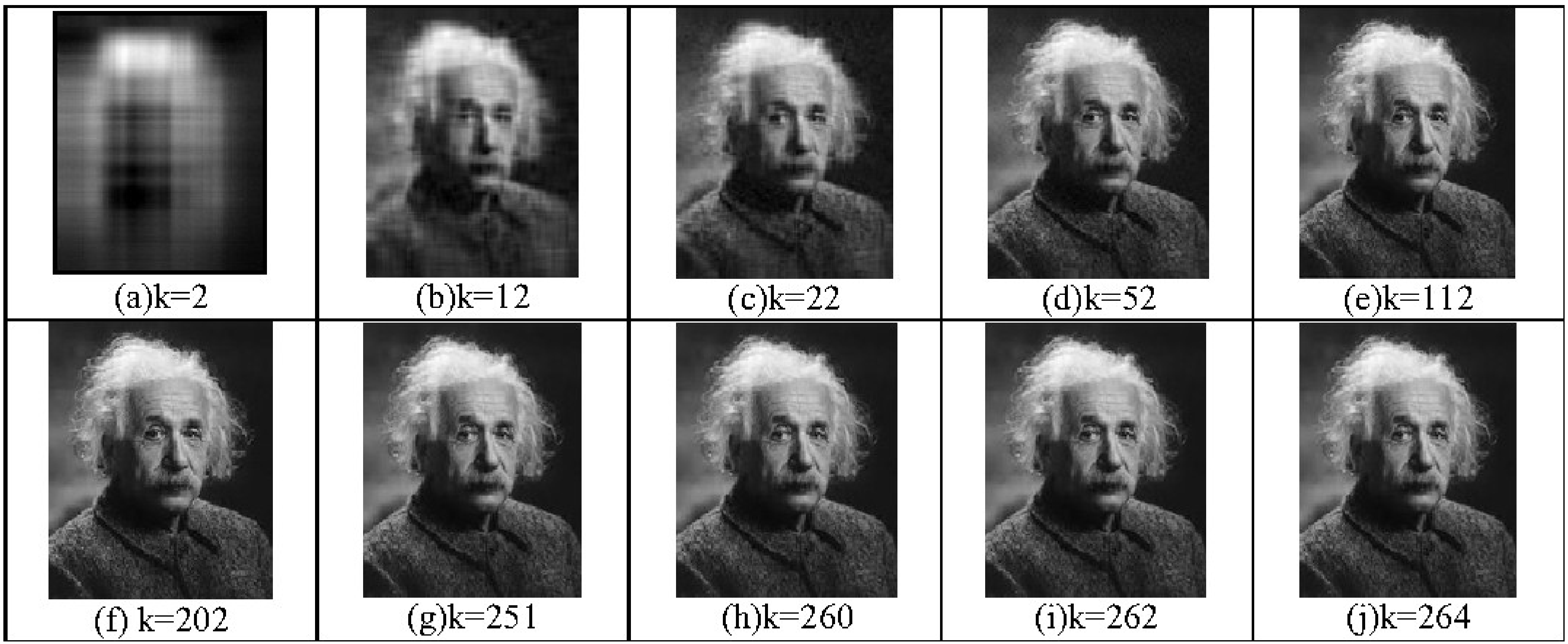
Se utiliza para sistemas con más ecuaciones que incógnitas ($m>n$). **Q** es una matriz ortogonal (**Q'*Q=I**) aunque **A** no lo sea. **R** es una matriz triangular superior, elementos diagonales de **R** pueden no ser positivos.
Se emplea para resolver **problemas de ajuste por mínimos cuadrados (Estadística, Álgebra Lineal).**

- [Q,R,P]=**qr**(A) factorización QR con pivotamiento por columnas. La matriz **P** es una matriz de permutación tal que **A*P=Q*R**.
- B = **null**(A) devuelve una base ortonormal del subespacio nulo (**Ax** = 0) de la matriz rectangular **A**.
- Q = **orth**(A) devuelve una base ortonormal del espacio de columnas de **A**. El número de columnas de **Q** es el rango de **A**.

4. Funciones de factorización y/o descompocisión matricial.

4) Basadas en la descomposición de valores singulares

- [U,D,V] = **svd**(A) descomposición de valor singular de una matriz rectangular (**A=U*D*V'**). **U** y **V** son matrices ortonormales. **D** es diagonal y contiene los valores singulares.



Aishwarya et al. **Lossy image compression using SVD coding algorithm**, International Conference WiSPNET, **2016**

Análisis de falla de rodamientos/rulemanes

Zheng et al., Applied Sciences **2017**, doi: [10.3390/app7111117](https://doi.org/10.3390/app7111117)

Estadística

Matemática IV

Mecánica Racional

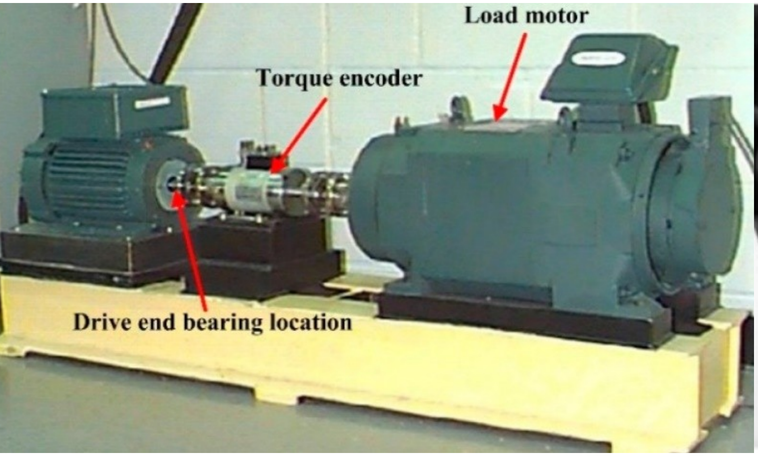
Métodos Numéricos

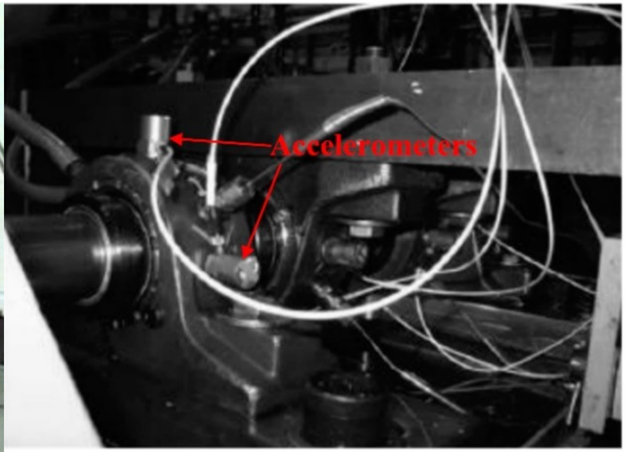
Laboratorio I

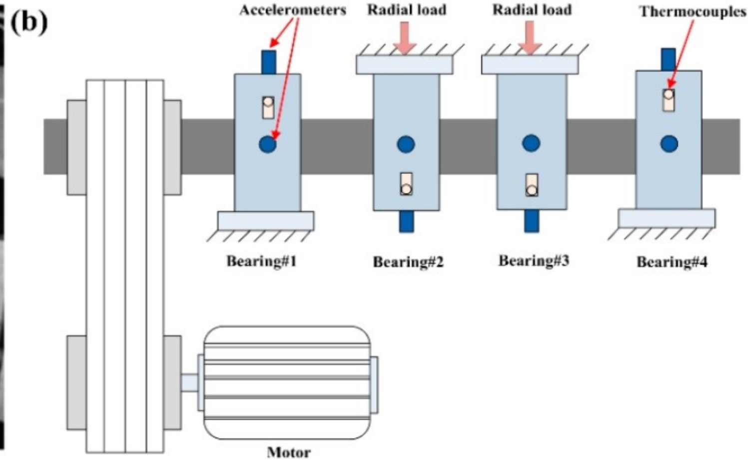
Electrotecnia

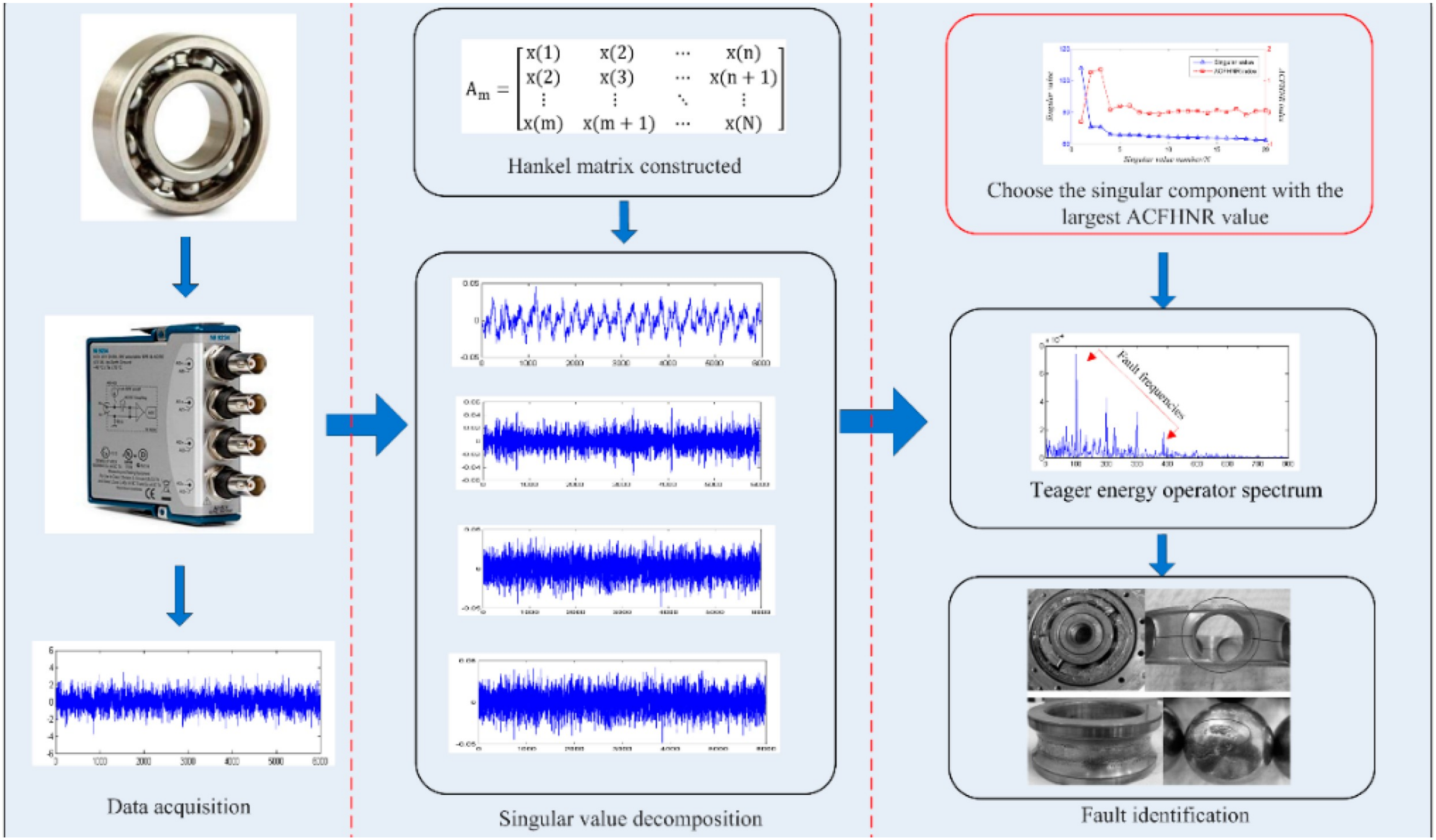
Análisis de falla de rodamientos/rulemanes

Zheng et al., *Applied Sciences* **2017**, doi: [10.3390/app7111117](https://doi.org/10.3390/app7111117)



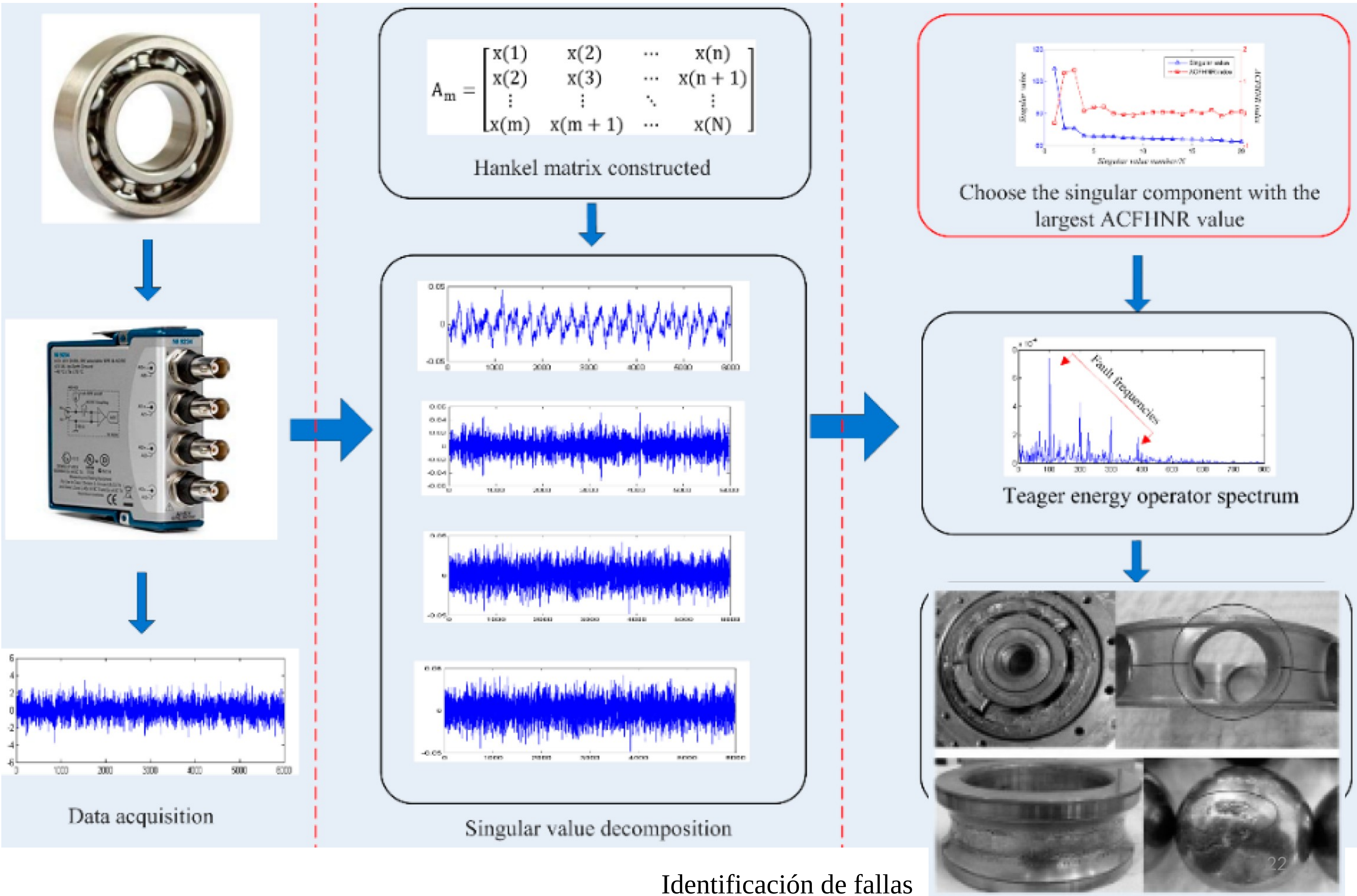






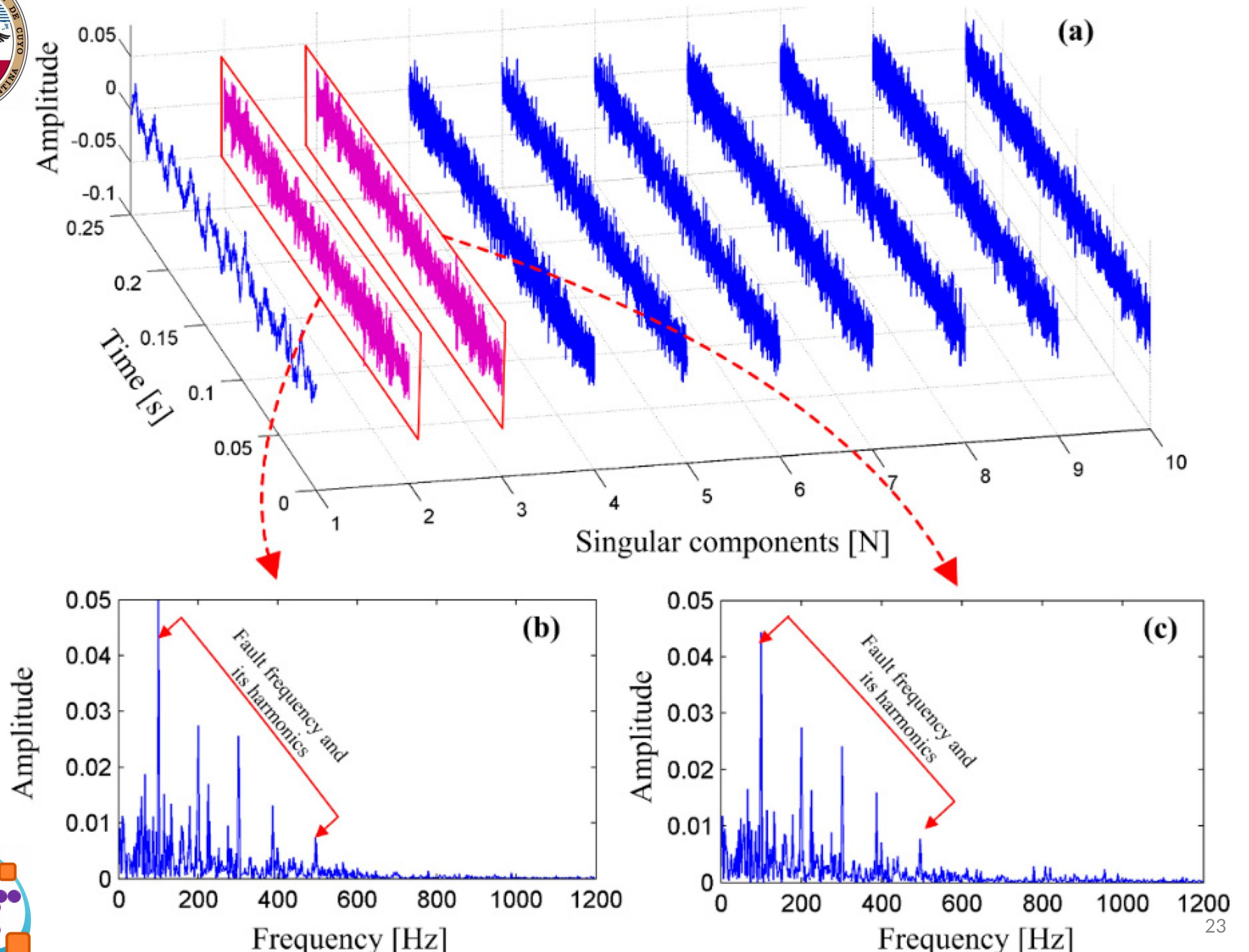
Análisis de falla de rodamientos/rulemanes

Zheng et al., *Applied Sciences* **2017**, doi: [10.3390/app7111117](https://doi.org/10.3390/app7111117)



Análisis de falla de rodamientos/rulemanes

Zheng et al., *Applied Sciences* **2017**, doi: [10.3390/app7111117](https://doi.org/10.3390/app7111117)



4. Funciones de factorización y/o descompocisión matricial.

4) Basadas en la descomposición de valores singulares

- nor = **norm**(A) calcula la norma-2 de **A** (mayor valor singular).
- menos operaciones aritméticas que la función ***norm***.
- c = **cond**(A) condición numérica de la matriz **A**. Cociente entre el máx y el mín valor singular. **cond**(A) da una idea del error que se obtiene al resolver un sistema de ecs lineales: log(cond) indica el número de cifras significativas que se pierden.
- r = **rank**(A) calcula el rango **r** de una matriz rectangular **A.**
- B = **pinv**(A) calcula la pseudo-inversa de una matriz rectangular **A.**
- nor = **normest**(A) calcula de forma aproximada la norma-2 con
- c = **condest**(A) estimación por defecto de la condición numérica de **A** con la norma-1. Función mucho más económica que ***cond***.

4. Funciones de factorización y/o descompocisión matricial.

5) Cálculo del rango y normas

- El rango se calcula implícitamente (sin que el usuario lo pida) al ejecutar las funciones ***rref(A)***, ***orth(A)***, ***null(A)*** y ***pinv(A)***. La función ***rank(A)*** está basada en ***pinv(A)***. Con ***pinv(A)*** se utiliza la descomposición SVD, que es el método más fiable y más caro.
- Normas de vectores:
  - ✓ **norm**(x,p) norma-p, es decir ***sum(abs(x)^p)^(1/p)***.
  - ✓ **norm**(x) norma-2 ó euclídea; equivale al módulo o ***norm(x,2)***.
  - ✓ **norm**(x,inf) norma-∞, es decir ***max(abs(x))***.
  - ✓ **norm**(x,1) norma-1, es decir ***sum(abs(x))***.
- Normas de matrices:
  - **norm**(A) norma-2, máximo valor singular de **A**, ***max(svd(A))***.
  - **normes**t(A) estimación de la norma-2. Útil para matrices grandes.
  - **norm**(A,1) norma-1 de **A**, máxima suma de valores absolutos por columnas, es decir: ***max(sum(abs(A)))***.
  - **norm**(A,inf) norma-∞ de **A**, máxima suma de valores absolutos por filas, es decir: ***max(sum(abs(A')))***.

5. Más sobre operadores relacionales: vectores/matrices

- Los operadores relacionales vistos previamente (**<**, **>**, **<=**, **>=**, **==** y **~=**) actúan entre dos matrices/vectores del mismo tamaño, el resultado es otra matriz/vector de ese mismo tamaño conteniendo 1 y 0, (***true*** o ***false***).
- Las matrices "binarias" no se almacenan en memoria ni se asignan a variables, se procesan sobre la marcha. Octave dispone de varias funciones para ello. Cualquier valor ≠ 0 equivale a ***true***, mientras que 0 equivale a ***false***.

5. Más sobre operadores relacionales: vectores/matrices

- Algunas de estas funciones son:
- **any**(x) función vectorial; chequea si ***alguno*** de los elementos del vector **x** cumple una dada condición (≠ 0). Devuelve 1 ó 0.
- **any**(A) se aplica por separado a cada columna de la matriz **A**. El resultado es un vector de unos y ceros.
- **all**(x) función vectorial; chequea si *todos* los elementos del vector **x** cumplen una condición. Devuelve un 1 ó 0.
- **all**(A) se aplica por separado a cada columna de la matriz **A**. El resultado es un vector de unos y ceros.
- **find**(x) busca índices correspondientes a elementos de vectores que cumplen una determinada condición. El resultado es un vector con los índices que cumplen la condición.
- **find**(A) cuando esta función se aplica a una matriz la considera como un vector con una columna detrás de otra, de la 1ª a la última, **A(:)**.

5. Más sobre operadores relacionales: vectores/matrices

Ejemplo: Veamos como funcionan algunas de ellos

```
>> A=magic(3)
>> m=find(A>4)
>> A(m)=10*ones(size(m))
>> any(A==3)
>> any(ans)
>> all(all(A))
```

- La función ***isequal(A, B)*** devuelve *uno* si las matrices son idénticas y *cero* si no lo son.

6. Otras funciones que actúan sobre vectores/matrices

- Las siguientes funciones pueden actuar sobre vectores y matrices, y sirven para chequear ciertas condiciones:
- **exist**('var') comprueba si el nombre **var** existe como variable, función, directorio, fichero, etc.
- **isnan**(A) chequea si hay valores ***NaN*** en **A**, devolviendo una matriz de unos y ceros del mismo tamaño que **A**.
- **isinf**(A) chequea si hay valores ***Inf*** en **A**, devolviendo una matriz de unos y ceros del mismo tamaño que **A**.
- **isfinite**(A) chequea si los valores de **A** son finitos.
- **isempty**(A) chequea si un vector o matriz está vacío o tiene tamaño nulo.
- **ischar**() chequea si una variable es una cadena de caracteres.
- **isglobal**() chequea si una variable es global.
- **issparse**() chequea si una matriz es dispersa (*sparse*, es decir, con un gran número de elementos cero).

6. Otras funciones que actúan sobre vectores/matrices

- A continuación se presentan algunos **ejemplos** de uso de estas funciones en combinación con otras vistas previamente.
- Se desea eliminar un ***NaN*** de un vector:

```
>> x=[1 2 3 4 0/0 6]
>> i=find(isnan(x))
>> x=x(find(~isnan(x)))
```

- Otras posibles formas de eliminar el ***NaN***:

```
>> x=x(~isnan(x))
>> x(isnan(x))=[]
```

- La siguiente sentencia elimina las filas de una matriz que contienen algún ***NaN***:

```
>> A(any(isnan(A)'), :)=[]
```

7. Funciones para cálculos con polinomios

- Para Octave un polinomio se puede definir mediante un vector de coeficientes. Por ejemplo, el polinomio:

  $$\boldsymbol{x^4 - 8x^2 + 6x - 10 = 0}$$

  se puede representar mediante el vector [1, 0, -8, 6, -10].

- Octave puede realizar diversas operaciones sobre un polinomio, como por ejemplo evaluarlo para un determinado valor de **x** (función ***polyval()***) y calcular las raíces (función ***roots()***).

Ejemplo:

```
>> pol=[1 0 -8 6 -10]
>> roots(pol)
>> polyval(pol,1)
```

7. Funciones para cálculos con polinomios

- Algunas funciones orientadas al cálculo con polinomios:
  - **poly**(A) polinomio característico de la matriz **A**
  - **roots**(pol) raíces del polinomio **pol**
  - **polyval**(pol,x) evaluación del polinomio **pol** para el valor de **x**. Si **x** es un vector, **pol** se evalúa para cada elemento de **x**
  - **conv**(p1,p2) producto de convolución de dos polinomios **p1** y **p2**
  - [c,r]=**deconv**(p,q) división del polinomio **p** por el polinomio **q**. En **c** se devuelve el cociente y en **r** el resto de la división
  - **residue**(p1,p2) descompone el cociente entre **p1** y **p2** en suma de fracciones simples (ver ***>>help residue***)
  - **polyder**(pol) calcula la derivada de un polinomio
  - **polyder**(p1,p2) calcula la derivada de producto de polinomios
  - **polyfit**(x,y,n) calcula los coeficientes de un polinomio **p(x)** de grado **n** que se ajusta a los datos **p(x(i)) ~= y(i)**, mínimo error cuadrático medio.
  - **interp1**(xp,yp,x) calcula el valor interpolado para la abscisa **x** a partir de un conjunto de puntos dado por los vectores **xp** e **yp**.

8. Funciones anónimas @

- Las funciones anónimas @ constituyen una forma muy flexible de crear funciones sobre la marcha, bien en la línea de órdenes, bien en una línea cualquiera de una función o de un fichero ***.m**.
- La forma general de las funciones anónimas es la siguiente:

```
fhandle = @(argumentos) expresión;
```

- Después de ser creada, la función anónima puede ser llamada a través del ***fhandle*** seguido de la lista de argumentos actuales entre paréntesis, o también puede ser pasada a otra función como argumento, también por medio del ***fhandle***.

Ejemplo: Calcular el valor del seno del doble del ángulo:

fhandle(theta) = sen(2*theta)

```
senAngDoble = @(ang) 2*sin(ang).*cos(ang);
senAngDoble([pi/4,pi/3])
ans: [1, sqrt(3)/2]
```

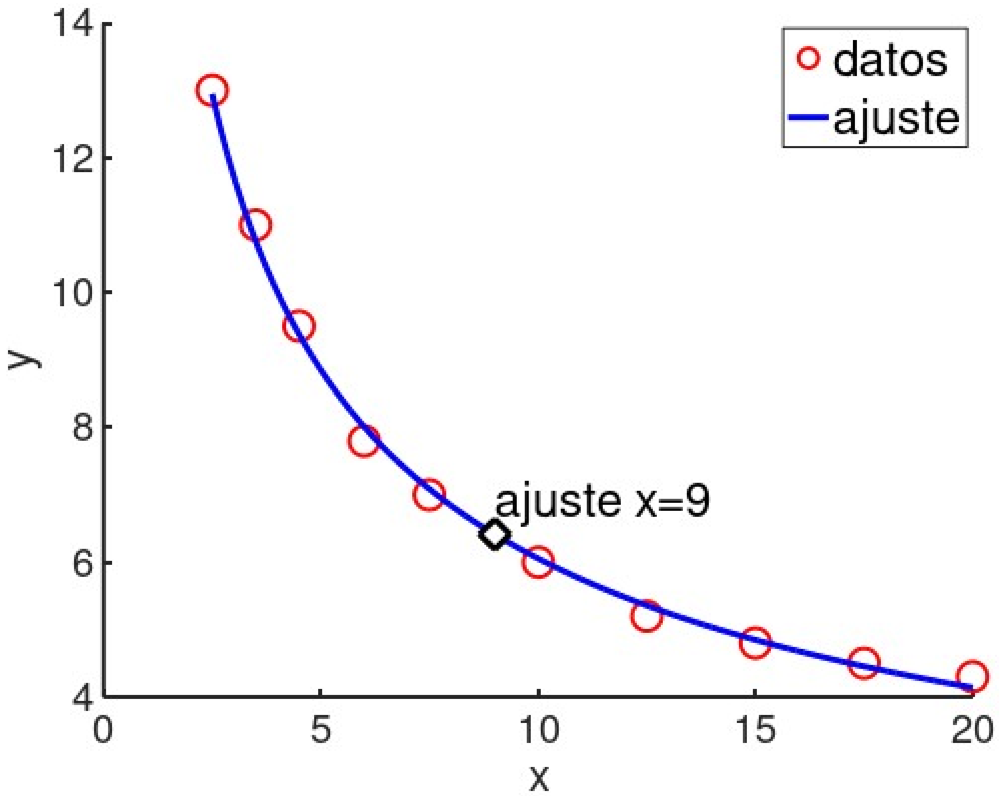
8. Funciones anónimas @

Ejemplo:

(i) Ajuste los datos siguientes con el modelo de potencias y = a x^b.

(ii) Use la ecuacion de potencias resultante para hacer el pronostico de y en x = 9.

x	y
2.5	13.0
3.5	11.0
4.5	9.5
6.0	7.8
7.5	7.0
10.0	6.0
12.5	5.2
15.0	4.8
17.5	4.5
20.0	4.3



Ayuda: utilizar el *script* **U3_ej_ajuste_potencias.m** subido a la web del curso.

Bonus track Funciones de biblioteca

- Existen además funciones definidas en ficheros ***.m** y ***.oct** que vienen con el propio programa o que han sido aportadas por usuarios del mismo.
- Los archivos **oct** son piezas de código C++ que se han compilado con la API Octave (Application Programming Interface).
- Octave proporciona el comando mkoctfile para construir archivos **oct** a partir de código fuente en C, C++ o Fortran.
- Para que Octave encuentre una determinada función de usuario dicho archivo-M debe estar en el directorio actual de trabajo o en el search path.

Bonus track Funciones de biblioteca

- Octave incluye una interfaz para permitir el uso de archivos ***.mex** y para compartir código compilado entre Octave y MATLAB.
- Dado que los ***.mex** emplean subrutinas, funciones y procedimientos de la estructura interna de MATLAB (diferentes de Octave), un archivo ***.mex** nunca poseerá el mismo rendimiento en Octave que el archivo ***.oct** equivalente.
- Por ej. cuando se invoca una función **mex-file**, para pasar las variables a las funciones **mex** se emplean un número significativo de copias adicionales de bloques de memoria.
- Se recomienda que cualquier nuevo código en C, C++ o Fortran se escriba con la interfaz **oct-file** para generar el archivo **oct** mediante la orden mkoctfile.

Introducción a Octave

para ciencias aplicadas e ingeniería

San Rafael, Argentina 2020

UNCUYO UNIVERSIDAD NACIONAL DE CUYO | FACULTAD DE CIENCIAS APLICADAS A LA INDUSTRIA

CONICET

MoCCAi MODELADO COMPUTACIONAL EN CIENCIAS APLICADAS E INGENIERÍA