Mapeo de datos adquiridos en variables de MATLAB

María del Mar Sanz Lluch

Borja Bordel Sánchez

Marina Pérez Jiménez

MATLAB aplicado a la instrumentación electrónica

Departamento de Electrónica Física (UPM)

PROGRAMA

- El lenguaje M
- Tipos de datos en M: tipos básicos y arrays
- Arrays dispersos
- Tablas
- Arrays categóricos
- Manejadores de funciones
- Objetos

EL LENGUAJE M

- Una vez adquiridos los datos desde uno o varios dispositivos externos, es preciso construir con ellos las estructuras de datos necesarias para su procesamiento
- En MATLAB el lenguaje que aporta la necesaria capacidad de programación es el lenguaje M
- Aunque es posible emplear C, C++ y Java estas tecnologías no son nativas de MATLAB

EL LENGUAJE M

- M es el lenguaje de programación con el que se codifican los algoritmos que se ejecutan en el entorno MATLAB
- Es muy anterior a la creación del propio MATLAB
  - Nace en 1970 como un recubrimiento de alto nivel sobre librerías FORTRAN de cálculo numérico
- No debe confundirse con M#
  - Para programación de sistemas Microsoft

EL LENGUAJE M

- Ejemplo de código escrito en M

```
% --- Executes during object creation, after setting all properties.
function selector_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to selector (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called

    % Hint: popupmenu controls usually have a white background on Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    seleccion = get(handles.selector,'Value');
    switch seleccion
        case 1
            warndlg('Debe seleccionar un proceso para arrancar la simulación','Error');
        case 2
            corazonMain();
        case 3
            vanDerPolMain();
        case 4
            brussleatorMain();
        case 5
    end
end

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to axes1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    empty - handles not created until after all CreateFcns called
```

EL LENGUAJE M

- Tradicionalmente M ha sido un lenguaje interpretado
  - Se habla de “scripts MATLAB”
- Desde la versión 8.5 de M (asociada a MATLAB R2015a) las rutinas se traducen de forma dinámica a código máquina nativo
  - Compilación *Just in time*, “en tiempo de ejecución” o “justo a tiempo” (voz latinoamericana)

EL LENGUAJE M

- Se trata de un lenguaje multiparadigma, pues admite:
  - Programación imperativa: Se basa en dar instrucciones al ordenador de como hacer las cosas en forma de algoritmos
  - Programación orientada a objetos: Está basada en el paradigma imperativo, pero encapsula elementos denominados “objetos” que incluyen tanto variables como funciones

EL LENGUAJE M

- Programación orientada a arrays: Está basada en el paradigma imperativo, en el que las operaciones habituales son las de tipo matricial, no escalar.
- Programación declarativa: Está basado en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones
  - En concreto admite programación funcional, en el que las reglas y propiedades vienen en forma de predicado matemático

EL LENGUAJE M

- Además admite programación dinámica
  - El script se puede modificar en tiempo de ejecución
- M es además débilmente tipado…
  - No es preciso declarar el tipo de las variables empleadas

M

```
A = 0;
```

Java

```
int A = 0;
```

EL LENGUAJE M

- … y permite tipado dinámico
  - El tipo de un dato puede variar de forma dinámica

M

```
A = 0;
A = [0 0];
```

Java

```
int A = 0;
int [] A = [0 0];
```

Error en compilación

Illegal start of expression

Variable A is already defined

EL LENGUAJE M

- El paso de variables entre funciones sigue el modelo de “copia vaga” *(copy-on-write o lazy-copy)*
  - Las variables se pasan por referencia, pero si se modifica su valor se realiza una copia temporal (que es la que se actualiza)
  - Sólo se puede modificar el valor de una variable en el mismo nivel en el que se declara

EL LENGUAJE M

```
function [B] = myfunction (A)
        B = 2*A;
end
```

Sólo se consulta el valor de A en memoria

EL LENGUAJE M

```
function [B] = myfunction (A)
        A = A/2;
        B = A%2;
end
```

Se hace una copia local de A para esta función, que desaparece al terminar el bloque. La actualización de A no se conserva

EL LENGUAJE M

- ¿Cómo se puede actualizar una variable en una subrutina?
  - Sólo es recomendable si es imprescindible
    - P. ej. Por problemas de disponibilidad de memoria para mantener copias locales
  - Se deben aplicar las llamadas “funciones in-place”
  - Lo veremos más adelante (Tema 4)

EL LENGUAJE M

- La alternativa es devolver una nueva variable con el valor actualizado

```
function [B, C] = myfunction (A)
        C = A/2;
        B = C%2;
end
```

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- En M los datos se clasifican en dos grandes grupos
  - Básicos
  - Arrays
- Los tipos básicos sólo almacenan un dato
- Los arrays son colecciones de tipos básicos, que pueden ser tratadas como una única entidad
  - Se asocian con las matrices matemáticas

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- En M todos los operadores pueden ser aplicados indistintamente a tipos básicos, arrays o ambos
  - El resultado, sin embargo, puede variar según el tipo de datos que intervengan

```
A = B*C;
%Producto habitual si B y C son tipos básicos
%Producto matricial si B y C son arrays
```

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- Al margen de los tipos básicos y de los arrays, existen otros tipos de datos
  - Tablas
  - Arrays categóricos
  - Manejadores de funciones
  - Objetos
- Sobre estos tipos se definen operaciones específicas, que no pueden aplicarse a otros tipos

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

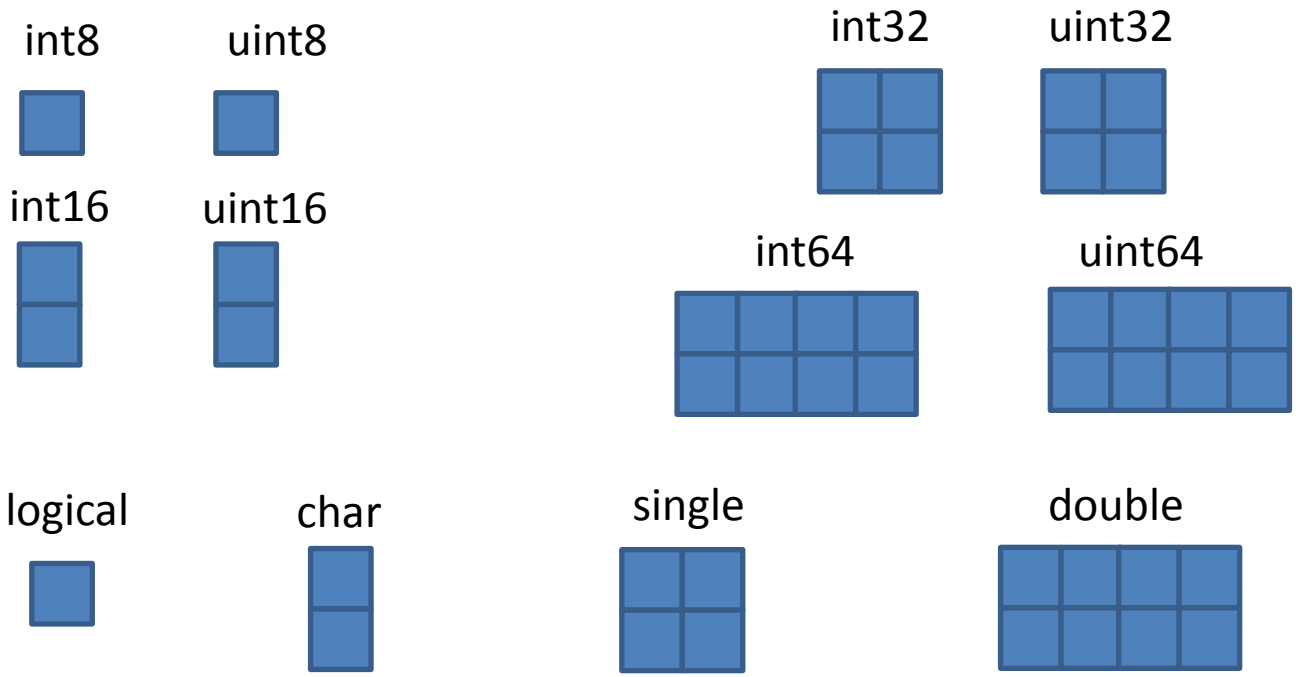
- Los tipos básicos se caracterizan porque todo el espacio reservado se ocupa por el valor de la variable
- Hay 12 tipos básicos
  - int8 : Entero con signo de 8 bits (-2^7 a 2^7)
  - uint8 : Entero sin signo de 8 bits (0 a 2^8)
  - int16, uint16, int32, uint32, int64, uint64
  - logical: Ocupan 8 bits. Codifican el valor 0 ó 1

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- char: ocupa 16 bits. Representa un carácter Unicode
- single: Datos numéricos en precisión simple (número decimales)
- double: Datos numéricos en doble precisión (número decimales)

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- Resumen de tipos básicos en M



TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- Los arrays ocupan en memoria el espacio correspondiente a la suma de las variables básicas que los componen, más una cabecera
- Hay 3 tipos de array
  - Vectores y matrices: Arrays n-dimensionales de tipos básicos. Añaden una cabecera de 112 bytes

```
Vector = [0 0];
Matriz = [0 0 ; 0 0];
```

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- Arrays de celdas: Arrays de celdas indexadas, donde cada celda contiene un vector o matriz de dimensiones cualesquiera. Añaden una cabecera de 112 bytes (adicional a la que acompaña a cada vector o matriz).

```
Celdas = {[0 0]};
```

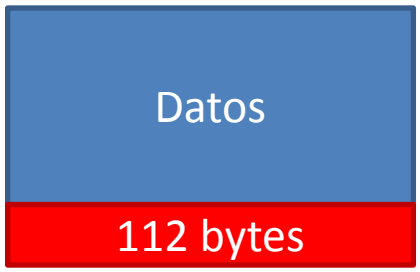
TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

- Estructuras: Conjuntos de variables que pueden ser referenciadas mediante un nombre. Añaden una cabecera de 112 bytes general, 64 bytes por cada campo (para almacenar el nombre), y la cabecera (si la tiene) asociada a cada una de las variables almacenadas

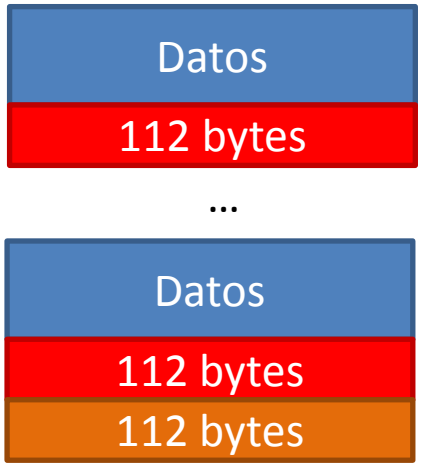
```
Estructura = struct('Vector', [0 0]);
```

TIPOS DE DATOS EN M: TIPOS BÁSICOS Y ARRAYS

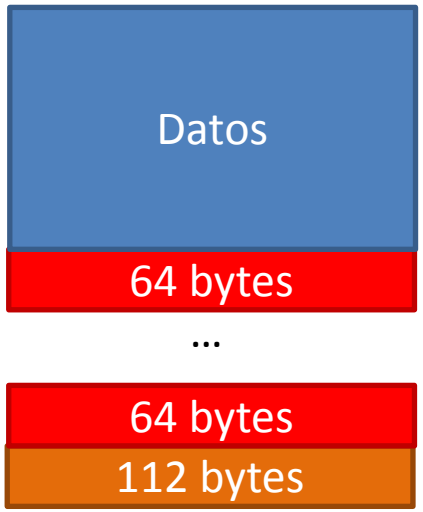
- Resumen de arrays



Vectores y matrices



Arrays de celdas



Estructuras

ARRAYS DISPERSOS

- Se trata de un tipo especial de vectores y matrices de gran tamaño en los que la mayoría de los elementos son nulos
- MATLAB optimiza el uso de memoria si un array se declara como *sparse* (disperso)
  - El tamaño ocupado es inferior a la suma de todas las variables básicas que componen el array

ARRAYS DISPERSOS

- Se puede crear una matriz dispersa con la función

```
dispersa = sparse(filas, columnas);
```

- La función devuelve una matriz nula, en la que se deben modificar las posiciones que deban ser distintas de cero

ARRAYS DISPERSOS

- Existe un límite de posiciones que pueden ser no nulas en un array disperso
- Típicamente
  - La mitad del total en arquitecturas de 64 bits
  - Dos terceras partes en arquitecturas de 32 bits
- MATLAB no verifica este límite, pero por encima de él trabajar con arrays dispersos en muy ineficiente

TABLAS

- Conjunto de conjuntos de datos.
- Se pueden mezclar diferentes tipos
  - MATLAB mantiene metadatos para soportarlo
- Permite indexación flexible, ordenación automática, fusión de conjuntos, etc.

	1	2	3	4	5	6	7	8	9	10	11
1	0.8147	0.1622	0.6443	0.0596	0.4229	0.5822	0.8507	0.5590	0.6837	0.9879	0.6312
2	0.9058	0.7943	0.3786	0.6820	0.0942	0.5407	0.5606	0.8541	0.1321	0.1704	0.3551
3	0.1270	0.3112	0.8116	0.0424	0.5985	0.8699	0.9296	0.3479	0.7227	0.2578	0.9970
4	0.9134	0.5285	0.5328	0.0714	0.4709	0.2648	0.6967	0.4460	0.1104	0.3968	0.2242
5	0.6324	0.1656	0.3507	0.5216	0.6959	0.3181	0.5828	0.0542	0.1175	0.0740	0.6525
6	0.0975	0.6020	0.9390	0.0967	0.6999	0.1192	0.8154	0.1771	0.6407	0.6841	0.6050
7	0.2785	0.2630	0.8759	0.8181	0.6385	0.9398	0.8790	0.6628	0.3288	0.4024	0.3872
8	0.5469	0.6541	0.5502	0.8175	0.0336	0.6456	0.9889	0.3308	0.6538	0.9828	0.1422
9	0.9575	0.6892	0.6225	0.7224	0.0688	0.4795	5.2238e-04	0.8985	0.7491	0.4022	0.0251
10	0.9649	0.7482	0.5870	0.1499	0.3196	0.6393	0.8654	0.1182	0.5832	0.6207	0.4211
11	0.1576	0.4505	0.2077	0.6596	0.5309	0.5447	0.6126	0.9884	0.7400	0.1544	0.1841
12	0.9706	0.0838	0.3012	0.5186	0.6544	0.6473	0.9900	0.5400	0.2348	0.3813	0.7258
13	0.9572	0.2290	0.4709	0.9730	0.4076	0.5439	0.5277	0.7069	0.7350	0.1611	0.3704
14	0.4854	0.9133	0.2305	0.6490	0.8200	0.7210	0.4795	0.9995	0.9706	0.7581	0.8416
15	0.8003	0.1524	0.8443	0.8003	0.7184	0.5225	0.8013	0.2878	0.8669	0.8711	0.7342
16	0.1419	0.8258	0.1948	0.4538	0.9686	0.9937	0.2278	0.4145	0.0862	0.3508	0.5710
17	0.4218	0.5383	0.2259	0.4324	0.5313	0.2187	0.4981	0.4648	0.3664	0.6855	0.1769
18	0.9157	0.9961	0.1707	0.8253	0.3251	0.1058	0.9009	0.7640	0.3692	0.2941	0.9574
19	0.7922	0.0782	0.2277	0.0835	0.1056	0.1097	0.5747	0.8182	0.6850	0.5306	0.2653

TABLAS

- Las tablas están pensadas para ser visualizadas, por lo que incluyen encabezado, propiedades de alineamiento, título, etc.
- La clase *ModelAdvisor.Table* encapsula todas las operaciones (incluyendo la creación) que pueden realizarse sobe tablas
  - *http://es.mathworks.com/help/slvnv/ref/modeladvisor.table-class.html*

TABLAS

- Ejemplo de creación de una tabla

```
tabla = ModelAdvisor.Table(filas, columnas);
```

- Se puede modificar el contenido de una posición mediante la función *setEntry*

```
Tabla.setEntry(fila, columna, valor);
```

ARRAYS CATAGÓRICOS

- Vectores o matrices en los que cada posición toma un valor no numérico de entre un conjunto finito de valores posibles
  - P.ej. {“Bueno”, “Malo”, “Pésimo”}
- Es mucho más eficiente que un array tradicional de cadenas de caracteres
- Sobre este tipo de datos pueden realizarse operaciones lógicas como si fueran arrays numéricos

ARRAYS CATAGÓRICOS

- Los arrays categóricos sólo están disponibles en el *toolbox Statistics and Machine Learning Toolbox*
- El conjunto de valores posibles puede ser:
  - Ordenado: Si se trata de un conjunto de “niveles”
  - No ordenado: En cualquier otro caso
- La forma de crear el array categórico depende del tipo de conjunto de valores elegido

ARRAYS CATAGÓRICOS

- Los arrays categóricos definidos sobre conjuntos no ordenados se llaman **nominales**

```
categorico = nominal({‘el1’, ‘el2’, …, ‘elN’});
```

- Los arrays categóricos definidos sobre conjuntos no ordenados se llaman **ordinales**

```
categorico = ordinal({‘o1’, ‘o2’, …, ‘oN’});
```

ARRAYS CATAGÓRICOS

- Las posiciones arrays categóricos ordinales admiten comparaciones de tipo lógico
  - Mayor que
  - Menor que
  - Igual
  - Etc.
- Sobre los arrays categóricos nominales sólo se define la operación “igual”

MANEJADORES DE FUNCIONES

- Son punteros a funciones obtenidos, por ejemplo, mediante la definición de funciones anónimas
  - Las veremos en el Tema 4
- En M, el operador “dirección de” es @
  - No confundir con & empleado en C y C++

MANEJADORES DE FUNCIONES

- Sobre este nuevo tipo de datos se definen sólo cinco operaciones
  - La estudiamos a continuación
- Evaluación de una función

```
[out1, out2, …, outN] = feval(manejador, in1, in2, …, inN);
```

MANEJADORES DE FUNCIONES

- Convertir un manejador en una cadena de caracteres

```
cadena = fun2str(manejador);
```

- Convertir una cadena de caracteres en un manejador

```
manejador = str2fun(cadena);
```

MANEJADORES DE FUNCIONES

- Obtener manejadores a todas las funciones locales del entorno

```
manejadores = localfunctions();
```

- Obtener información sobre la función a la que apunta un manejador

```
info = functions(manejador);
```

OBJETOS

- Objetos: Se distinguen dos subtipos
  - Objetos de usuario: Creados a partir de clases creadas en M
  - Objetos Java: Creados a partir de clases Java
- La programación orientada a objetos es útil en escenarios donde los datos van acompañados de mucha meta-información

OBJETOS

- MATLAB, sin embargo, sigue siendo mucho más eficiente cuando se emplean algoritmos matemáticos
- En este tema no vamos a profundizar en este aspecto del lenguaje M
- Existe una página de formación oficial de Mathworks
  - http://es.mathworks.com/discovery/object-oriented-programming.html