

## Implementación de Filtros Digitales con Sistemas Computacionales Interactivos

*Barbosa, Lucrecia - Kleisinger, Gretchen H. - del Valle, Eduardo E. - Monzón, Jorge E.*

Departamento de Ingeniería Eléctrica - Facultad de Ciencias Exactas - UNNE.  
9 de Julio 1449 - (3400) Corrientes - Argentina.  
Teléfono: +54 (03783) 423126 - Fax: +54 (03783) 474930 - Email: jemonzon@exa.unne.edu.ar

### ANTECEDENTES

El desarrollo de sistemas computacionales interactivos –que representan a la vez lenguajes de programación de alto nivel– para el cálculo científico y técnico general, ha permitido la solución de muchos problemas numéricos en una fracción del tiempo que llevaría escribir un programa en lenguajes tales como C, Fortran, Basic o Pascal.

Al igual que con los lenguajes de programación convencionales, en los sistemas computacionales interactivos se puede definir variables, trabajar con operadores lógicos y relacionales, disponer de funciones predefinidas (por ejemplo las funciones matemáticas) u operar con las estructuras de control de flujo. Adicionalmente, estos sistemas en general ofrecen una destacada capacidad gráfica para la visualización de los datos e incluyen una importante biblioteca de subrutinas, por lo que resultan muy eficientes para el desarrollo de variadas tareas de procesamiento numérico.

El objetivo del presente trabajo es mostrar las facilidades proporcionadas por uno de estos sistemas computacionales interactivos –como alternativa frente a lenguajes convencionales de programación–, en la implementación de filtros digitales para el procesamiento de bioseñales.

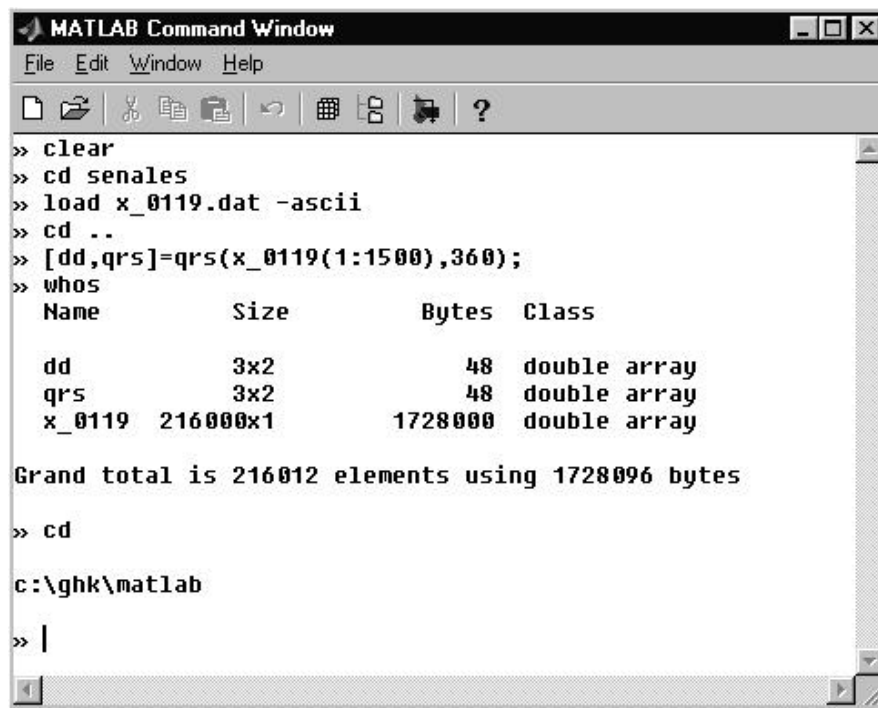
### MATERIALES Y METODOS

**Bioseñales.** Son registros electrocardiográficos extraídos de la base de datos de arritmias cardíacas del MIT-BIH (1999). Las muestras están digitalizadas a 11 bits, con una velocidad de captura de 360 Hz (Moody, 1997)

**Hardware.** Procesador Intel Pentium I de 166 MHz, 32 Mb de RAM, bajo sistema operativo Windows'95.

**Software.** Utilizamos el sistema MATLAB (*MATrix LABoratory*; The Mathworks, Natick, Massachusetts), cuyo elemento básico de datos es una matriz que no requiere dimensionamiento. Como lenguaje de programación, permite construir herramientas re-usables, denominadas archivos M (*M-files*). El software dispone de gran capacidad gráfica y viene acompañado de una variedad de herramientas accesorias (*Toolbox*) para aplicaciones específicas (i.e., procesamiento de imágenes, control no lineal, análisis espectral, redes neuronales)(Hanselman y Littlefield, 1995).

La Figura 1 ilustra la ventana de comando que aparece al ejecutar el programa. Mediante esta ventana, es posible impartir las instrucciones al programa y realizar todas las operaciones, incluyendo el llamado a los M-files o a las funciones. La ventana muestra además la estructura dimensional asignada a las matrices de datos.



```

MATLAB Command Window
File Edit Window Help
[Icons]
>> clear
>> cd senales
>> load x_0119.dat -ascii
>> cd ..
>> [dd,qrs]=qrs(x_0119(1:1500),360);
>> whos
Name          Size          Bytes  Class
dd             3x2             48  double array
qrs            3x2             48  double array
x_0119        216000x1       1728000  double array

Grand total is 216012 elements using 1728096 bytes

>> cd

c:\ghk\matlab

>> |

```

Figura 1. Ventana de comando

*M-file*. Es un archivo simple de texto, de extensión *m*, que contiene comandos que MATLAB interpreta y ejecuta como si hubiesen sido tipeados en la ventana de comando. Los comandos dentro del M-file tienen acceso a todas las variables, del entorno de trabajo de MATLAB.

*Función M-file*. El sistema permite crear funciones propias del usuario. La función M-file es un tipo especial de M-file, que se comunica con el espacio de trabajo de MATLAB solamente a través de aquellas variables que han sido definidas como entrada o generadas como salida de la función.

## DISCUSION DE RESULTADOS

A los fines de evaluar el desempeño del sistema computacional en la realización de filtros digitales para el procesamiento de señales electrocardiográficas, con MATLAB implementamos el clásico algoritmo de detección del complejo QRS propuesto por Pan y Tompkins (1985) y cuyas reglas fueron investigadas por Hamilton y Tompkins (1986). La detección se realiza analizando la pendiente, la amplitud y el ancho del complejo. El algoritmo consta de varias etapas: (a) un filtro pasabanda compuesto por filtros pasa-bajos y pasa-altos, para aislar al QRS, cuya energía se centra alrededor de los 17 Hz; (b) un filtro derivador para detectar los frentes abruptos característicos del complejo QRS; (c) conversión de la curva de diferenciación a excursiones positivas únicamente; (d) un filtro integrador de ventana, que suma las áreas bajo la curva obtenida en (c) considerando un intervalo de 150 ms, y luego avanza un intervalo de muestreo para integrar la próxima ventana, cuyo ancho permite incluir las mayores duraciones de QRS patológicos, pero evita la superposición del complejo con la onda T. Se detecta el QRS y se mide su duración. El algoritmo actualiza el umbral del filtro pasabanda como del integrador, según el nivel de pico de la señal y del ruido superpuesto. El proceso concluye con la determinación de si el evento detectado es en verdad un QRS.

```
f=figure(1);
set(f,'color',[1 1 1])
subplot(3,2,1), plot(s,'k','linewidth',1.5)

% lowpass
B=[1 0 0 0 0 -2 0 0 0 0 1];
A=[1 -2 1];
s1=filter(B,A,s);
figure(1),subplot(3,2,2),plot(s1,'k','linewidth',1.5)

% high pass
B=zeros(1,33);
B(1)=1; B(33)=-1;
A=[1 -1];
C=zeros(1,17); C(17)=1;
s2=filter(C,1,s1)-filter(B,A,s1)/32;
subplot(3,2,3),plot(s2,'k','linewidth',1.5)
```

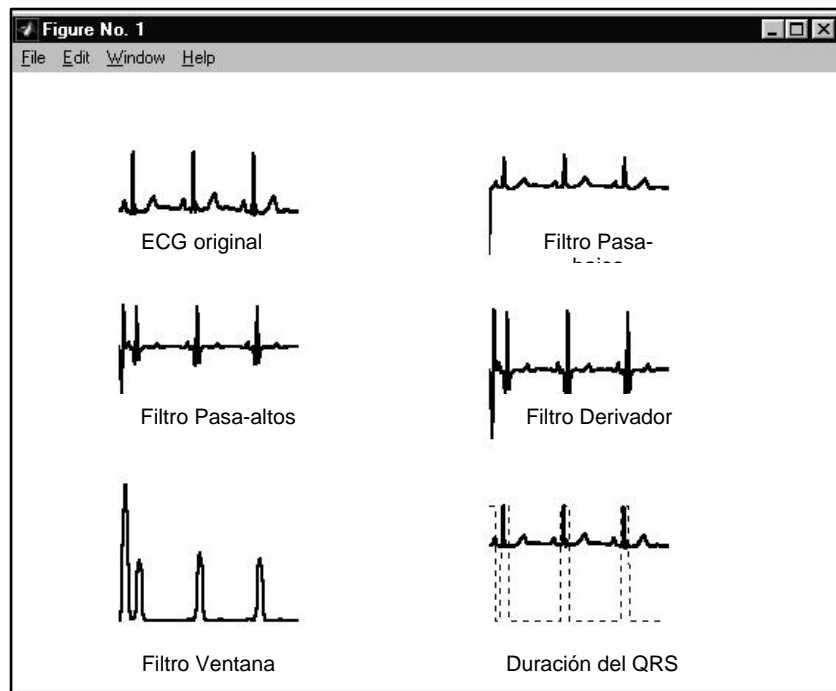
```
% derivative
B=[0.2 0.1 0 -0.1 -0.2];
s3=filter(B,A,s2);
subplot(3,2,4),plot(s3,'k','linewidth',1.5)

% moving integral
N=30;delay=30;
B=ones(1,N)/N;
s4=s3.^2;
s4=filter(B,1,s4);
subplot(3,2,5),plot(s4,'k','linewidth',1.5)

%Title('Filtro Ventana')
det=s4>(0.125*(max(s4)/2));
det(1:delay)=[];
subplot(3,2,6), plot(s,'k','linewidth',1.5)
subplot(3,2,6), hold on,plot(max(s)*det,'k:')
plot(s,'k','linewidth',1.5)
```

**Figura 2.** Códigos empleados para implementar los distintos filtros del algoritmo

La Figura 2 muestra un extracto del código empleado para implementar los distintos filtros del algoritmo. Se observa que la instrucción de ejecución de un filtro está dada por la función *FILTER*, cuyo parámetro de entrada no es otro que la función transferencia, interpretada por MATLAB en el momento de aplicar el filtro. La Figura 3 corresponde a la visualización gráfica de MATLAB, en la que se ilustran las señales obtenidas luego de la aplicación de cada uno de los filtros implementados.



**Figura 3.** Señal de ECG procesada con filtros digitales

## CONCLUSIONES

El sistema y lenguaje MATLAB simplifica los cálculos del álgebra lineal. El enfoque matricial de su estructura permite diversas maneras de procesamiento de los arreglos numéricos de datos (arrays). Se dispone también de las herramientas necesarias para resolver algunos problemas en los que resulta difícil integrar, diferenciar o determinar valores específicos de funciones que requieran aproximaciones numéricas de la solución deseada.

(a)	<pre>% Filtro Pasa Bajo B=[1 0 0 0 0 0 -2 0 0 0 0 0 1]; A=[1 -2 1]; s1=filter(B,A,s);</pre>
(b)	<pre>pasa_bajo(dato) int dato; {     static int y1=0, y2=0, x[26], n=12;     int y0;      x[n] = x[n + 13] = dato;     y0 = (y1&lt;&lt;1)-y2 + x[n] - (x[n + 6]&lt;&lt;1) + x[n + 12];     y2 = y1;     y1 = y0;     y0 &gt;&gt;= 5;     if (--n &lt; 0) n=12;     return(y0); }</pre>

**Figura 4.** Comandos para la implementación de un filtro pasabajos.  
(a) con MATLAB; (b) En C.

La Figura 4 (a) muestra la simplicidad de la implementación de un filtro digital pasabajos con MATLAB, en comparación con el mayor número de comandos en la implementación utilizando el lenguaje C (Fig. 4(b)). La diferencia radica en que MATLAB opera directamente con la función transferencia del filtro (en el dominio  $z$ ), mientras que con C debe implementarse la ecuación de diferencias del filtro, en el dominio del tiempo.

## BIBLIOGRAFIA

- Hamilton PS, Tompkins WJ**, 1986. Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database. *IEEE Transactions on Biomedical Engineering*, vol. BME-35, pp. 1157-65
- Hanselman D, Littlefield B**, 1995. *MATLAB User's Guide*. Englewood Cliffs, NJ, Prentice-Hall.
- MIT-BIH**, 1999. Massachusetts Institute of Technology-Beth Israel Hospital Database Distribution, MIT, 77 Massachusetts Avenue, Room 20A-113, Cambridge, MA 02139.
- Moody GB**, 1997. *ECG Database Applications Guide*, Tenth Edition. Harvard-MIT Division of Health Sciences, Biomedical Engineering Center.
- Pan J, Tompkins WJ**, 1985. A real-time QRS detection algorithm. *IEEE Transactions on Biomedical Engineering*, vol. BME-32, pp. 230-236.