

En esta tercera sesión de prácticas veremos,

1. cómo se diseñan y usan funciones en Octave, mostrando una aplicación sobre el método de Gauss-Seidel (cuyo algoritmo que ya se vio en la segunda sesión);
2. un *script* en el que se implementa un algoritmo del método de Gram-Schmidt para la ortogonalización de vectores.

3.1. Diseño de funciones (gauss_seidel.m)

Como cualquier otro programa de cálculo numérico, Octave tiene muchas funciones predefinidas. Esto no impide que nosotros podamos definir nuestras propias funciones.

En general, la estructura de una función es la siguiente:

```
function [resultados] = nombre_función (lista_argumentos)
    cuerpo: cálculos y tareas a realizar
endfunction
```

donde `lista_argumentos` es la lista de datos o argumentos, separados por comas, sobre los que la función actuará, `nombre_función` es la cadena de caracteres que nosotros asignamos al comando creado, y al cual tendremos que apelar para ejecutarlo posteriormente, y `cuerpo` constituye propiamente la función, y consiste en el conjunto de todos cálculos o tareas a realizar sobre los argumentos y que dan lugar a `resultados`, es decir, el listado de los valores, separados por comas, que devolverá como resultado la función. Las versiones más modernas de Octave dejan, en `cuerpo`, espacios a la izquierda para las líneas que lo componen (indentación); pero en las versiones antiguas se usa la barra espaciadora para llevar a cabo esta acción (¡nunca el tabulador!). La lista `resultados` se puede omitir en aquellas funciones que no devuelvan valores o devuelvan sólo uno. Por último, `endfunction` determina que hemos concluido la definición de nuestra función.

Todos estos elementos se pueden ver en el archivo `gauss_seidel.m`. Es importante tener en cuenta que el nombre del archivo debe ser exactamente el mismo que el nombre de la función. Si no fuera así, Octave no sabrá usar nuestra función.

También es muy conveniente incluir algunos comentarios que expliquen cómo opera nuestra función. De esta forma cualquier persona, en cualquier momento, tendrá acceso a información que le ayudará a entender para qué sirve la función definida. Por cierto, todos los comentarios incluidos tras `function` y la primera línea en blanco (completamente) u orden a ejecutar serán considerados como la ayuda de la función. De hecho, si ejecutamos en Octave la orden

```
>> help gauss_seidel
```

entonces obtenemos, entre otras cosas, dichos comentarios.

En la función `gauss_seidel` está implementado el método de Gauss-Seidel para un sistema de ecuaciones con el mismo número de ecuaciones e incógnitas. Además, se presupone que la matriz `A` de coeficientes del sistema es invertible y, además, que ningún elemento de la diagonal es nulo.

Por cierto, los argumentos `tol` y `maxiter` tienen asignados valores preestablecidos, por lo que la función `gauss_seidel` se puede usar proporcionando solo los valores de `A`, `b` y `x0` (o sea, `gauss_seidel` con solo tres argumentos). Si deseamos otros valores de `tol` y `maxiter`, basta con usar `gauss_seidel` con cinco argumentos.

3.2. Uso de funciones (Archivo3_usando_gauss_seidel.m)

En el *script* Archivo3_usando_gauss_seidel.m se ve cómo actúa la función `gauss_seidel` definida en la sección anterior. De paso se recuerda, mediante varios ensayos, cómo operan los comandos `'` y `.'` sobre matrices y vectores.

3.3. Método de Gram-Schmidt (Archivo3_Gram_Schmidt.m)

En el *script* Archivo3_Gram_Schmidt.m se ha implementado el método de Gram-Schmidt (para vectores de \mathbb{R}^n) haciendo uso de la fórmula vista en teoría. Recordemos que, si $B = \{v_1, \dots, v_n\}$ es una base de \mathbb{R}^n , entonces los vectores $\{u_1, u_2, \dots, u_n\}$ definidos por

$$\begin{aligned} u_1 &= v_1, \\ u_2 &= v_2 - \frac{\langle v_2, u_1 \rangle}{\|u_1\|^2} u_1, \\ u_3 &= v_3 - \frac{\langle v_3, u_1 \rangle}{\|u_1\|^2} u_1 - \frac{\langle v_3, u_2 \rangle}{\|u_2\|^2} u_2, \\ &\vdots \\ u_n &= v_n - \frac{\langle v_n, u_1 \rangle}{\|u_1\|^2} u_1 - \frac{\langle v_n, u_2 \rangle}{\|u_2\|^2} u_2 - \dots - \frac{\langle v_n, u_{n-1} \rangle}{\|u_{n-1}\|^2} u_{n-1}, \end{aligned}$$

forman una base ortogonal de \mathbb{R}^n .

Sobre el doble bucle que se ha diseñado para implantar el algoritmo, decir que con el bucle exterior “llamamos” a cada nuevo vector, mientras que con el bucle interno se va modificando el nuevo vector paso a paso (esto es, una resta en cada paso) hasta completar la fórmula que lo define.

3.4. Cambio de base (Archivo3_cambio_base.m)

En el *script* Archivo3_cambio_base.m se explica cómo se deben realizar los cambios de base cuando, por comodidad, los vectores vienen dados por las filas de una cierta matriz. De nuevo hacemos uso del comando `.'` para la transposición de matrices.