Programación: operación axpy

Objetivos. Programar una función que calcule $\alpha x + y$, donde $\alpha \in \mathbb{R}$, $x, y \in \mathbb{R}^n$.

Requisitos. Programación de funciones con arreglos y ciclos `for`; creación de vectores pseudoaleatorios; medición del tiempo de ejecución.

1. Problema: operación axpy. En algún lenguaje de programación escribir una función que calcule el vector $\alpha x + y$, donde $\alpha \in \mathbb{R}$, $x, y \in \mathbb{R}^n$.

Entrada: $\alpha \in \mathbb{R}$, $x, y \in \mathbb{R}^n$.

Condición que debe cumplir la entrada: los vectores dados x, y son de la misma longitud. No tiene que verificar esta condición.

Salida: el vector $\alpha x + y$.

Solución en el lenguaje de Matlab (guardar en el archivo `axpy.m`):

```
function r = axpy(a, x, y),
   n = length(x);
   r = zeros(n, 1);
   for j = 1 : n,
      r(j) = a * x(j) + y(j);
   end
end
```

2. Análisis de complejidad. Calcular el número de operaciones de multiplicación en el algoritmo anterior. La respuesta es un polinomio de la variable n. Calcular el número de operaciones de adición.

3. Pruebas con vectores pequeños. Escribir un programa que llame a la función del ejercicio anterior, aplicándola a los siguientes datos:

$$\alpha = 5, \qquad x = \begin{bmatrix} 3 \\ -4 \\ 1 \end{bmatrix}, \qquad y = \begin{bmatrix} 2 \\ 5 \\ -2 \end{bmatrix}.$$

Nuestra función `axpy` debe regresar el mismo resultado que la expresión `alpha * x + y`. Solución en el lenguaje de Matlab (guardar en el archivo `testaxpy1.m`):

```
function [] = testaxpy1(),
   alpha = 5; x = [3; -4; 1]; y = [2; 5; -2];
   display(axpy(alpha, x, y));
   display(alpha * x + y);
end
```

4. Prueba con un vector corto pseudoaleatorio.

```
function [] = testaxpy2(),
   alpha =  2 * rand() - 1;
   x = 2 * rand(n, 1) - ones(n, 1);
   y = 2 * rand(n, 1) - ones(n, 1);
   display(alpha);
   display(x);
   display(y);
   p = axpy(alpha, x, y);
   display(p);
   q = alpha * x - y;
   display(norm(p - q));
end
```

5. Pruebas con vectores pseudoaleatorios largos. En algún lenguaje de programación escribir un programa que genere vectores de tamaños grandes: $n = 10^4$, luego $n = 10^5$, luego $n = 10^6$, aplique a estos vectores la función del Ejercicio 1 y mida el tiempo de ejecución. Las computadoras contemporaneas realizan una operación de tipo `axpy` muy rápidamente, aún cuando n es grande. Para medir mejor los intervalos de tiempo, repetimos cada prueba 10 veces. Solución en Matlab (guardar en el archivo `testaxpy3.m`):

```
function [] = testaxpy3(),
   nrep = 10;
   for n = [10000, 100000, 1000000],
      display(n);
      alpha =  2 * rand() - 1;
      x = 2 * rand(n, 1) - ones(n, 1);
      y = 2 * rand(n, 1) - ones(n, 1);
      t1 = cputime();
      for rep = 1 : nrep,
         p = axpy(alpha, x, y);
      end
      t2 = cputime();
      display((t2 - t1) / nrep);
   end
end
```

¿Cómo se cambia el tiempo de ejecución cuando n se multiplica por 10?