Programación: métodos de pasos múltiples explícitos de Adams–Bashforth para resolver EDO's con valores iniciales

Objetivos. Programar varios métodos de pasos múltiples de Adams–Bashforth para resolver ecuaciones diferenciales con condiciones iniciales.

Requisitos. Haber programado varios métodos de Runge–Kutta y el método de paso doble de Adams–Bashforth.

1. Ejemplos y algoritmos de las clases anteriores. En las clases anteriores ya hemos programado un ejemplo `f1` de función del lado derecho de ecuación diferenciales, un ejemplo `x1` de solución correspondiente, y algunos esquemas que realizan un paso usando un valor anterior de la función, incluso el método clásico de Runge–Kutta `rk41step`.

2. Idea de los métodos de Adams–Bashforth. Se calcula la función f (el lado derecho de la ecuación) en varios puntos (t_j, v_j) calculados previamente, y se utiliza una combinación lineal de estos valores de f para aproximar la pendiente actual. Por ejemplo, en el método de Adams–Bashforth de tres pasos,

$$v_j = v_{j-1} + h\left(\frac{23}{12}f(t_{j-1}, v_{j-1}) - \frac{4}{3}f(t_{j-2}, v_{j-2}) + \frac{5}{12}f(t_{j-3}, v_{j-3})\right). \tag{1}$$

Notamos que $f(t_5, v_5)$ se utiliza tanto en el cálculo de v_6 como en el cálculo de v_7 y de v_8, por eso guardaremos los valores $f(t_j, v_j)$ en un arreglo. Si empezamos con un valor dado v_0, entonces para el cálculo de v_1 y v_2 no podemos aplicar (1). Para el calculo de los primeros puntos utilizamos algún metodo de Runge–Kutta de orden grande, por ejemplo, el método de Runge–Kutta clásico de orden 4.

3. Una iteración del método de Adams–Bashforth del paso triple. Supongamos que el arreglo (columna) de tres componentes $f(t_{j-1}, v_{j-1}), f(t_{j-2}, v_{j-2}), f(t_{j-3}, v_{j-3})$ está dado como `reversed(fvalues[range(j - 3, j)])`. Entonces la fórmula (1) se puede realizar de la siguiente manera, usando el producto punto (renglón por columna):

```
abcoefs <- [23/12, -4/3, 5/12]
v[j] <- v[j - 1] + h * abcoefs * reversed(fvalues[range(j - 3, j)])
```

4. Fragmento invertido de un vector en el lenguaje Matlab/Octave. En el lenguaje Matlab/Octave se recomienda probar los siguientes comandos:

```
a = (11 : 19)'
a(7 : -1 : 4)
```

5. Ejercicio: entender los primeros pasos. Supongamos que los valores $f(t_j, v_j)$ se guardan en un arreglo `fvalues`. Determinar qué parte de este arreglo se necesita en cada uno de los primeros pasos del método de Adams–Bashforth con tres puntos anteriores:

```
abcoefs <- [???, ???, ???]
fvalues[0] <- f(t[0], v[0])
# empezamos el ciclo de preparación:
v[1] <- rk41step(f, t[???], v[???], h)
fvalues[1] <- f(t[???], v[???])
v[2] <- rk41step(f, t[???], v[???], h)
fvalues[2] <- f(t[???], v[???])
# empezamos el ciclo principal:
v[3] <- v[???] + h * abcoefs * reversed(fvalues[range(???, ???)]);
fvalues[3] <- ???
v[4] <- ???
fvalues[4] <- ???
```

6. Coeficientes de los métodos de Adams–Bashforth. En general,

$$v_j = v_{j-1} + \sum_{k=1}^{d} b_k f(t_{???}, v_{???}).$$

El método de Adams–Bashforth con un punto anterior coincide con el método de Euler. En este caso

$$b_1 = 1.$$

En el método de Adams–Bashforth con dos puntos anteriores los coeficientes son

$$b_1 = \frac{3}{2}, \quad b_2 = -\frac{1}{2}.$$

En el método de Adams–Bashforth con tres puntos anteriores los coeficientes son

$$???, \quad ???, \quad ???.$$

En el método de Adams–Bashforth con cuatro puntos anteriores los coeficientes son

$$\frac{55}{24}, \quad -\frac{59}{24}, \quad \frac{37}{24}, \quad -\frac{3}{8}.$$

En el método de Adams–Bashforth con cinco puntos anteriores los coeficientes son

$$\frac{1901}{720}, \quad -\frac{1387}{360}, \quad \frac{109}{30}, \quad -\frac{637}{360}, \quad \frac{251}{720}.$$

7. Función que realiza el método de Adams–Bashforth. El arreglo `abc` contiene los coeficientes para varios métodos de Adams–Bashforth. El argumento `d` de la función determina el orden del método. En el arreglo `fvalues` guardamos los valores $f(t_j, v_j)$. El argumento `firststepsformula` determina el el método que se utiliza para calcular los primeros puntos. Por ejemplo, el usuario puede llamar la siguiente función con `rk41step` en vez de `firststepsformula`.

```
defun adamsbashforth(f, tmin, tmax, x0, firststepsformula, d, n):
   # guardamos los coefs para varios metodos, elegimos el renglón d:
   abc = zeroarray(???, ???)
   abc[1, range(1)] <- [1]
   abc[2, range(2)] <- [3/2, -1/2]
   abc[3, range(3)] <- [???, ???, ???]
   abc[4, range(4)] <- [???, ???, ???, ???]
   abc[5, range(5)] <- [???, ???, ???, ???, ???]
   abcoefs <- abc[d, range(???)]
   # inicializamos otras variables y calculamos los primeros valores:
   h <- ???; t <- tmin + h * range(n + 1)
   v <- zeroarray(n + 1); v[0] <- ???
   fvalues <- zeroarray(n)
   fvalues[0] <- f(t[0], v[0])
   for j in range(???, ???):
      v[j] <- firststepsformula(f, ???, ???, ???)
      fvalues[j] <- ???
   # ahora todo esta preparado, empezamos el ciclo principal:
   for j in range(???, n + 1):
      v[j] <- v[???] + h * abcoefs * reversed(fvalues[range(???, ???)])
      fvalues[j] <- f(???, ???)
   (t, v)
```

En este pseudocódigo se supone que los índices empiezan desde `0` y que la expresión `range(5, 9)` regresa `[5, 6, 7, 8]`. Recordamos que en el lenguaje Matlab/Octave la expresión `5 : 9` regresa `[5, 6, 7, 8, 9]`, y para construir un vector columna de longitud 5 se escribe `zeros(5, 1)`.

8. Prueba.

```
defun testadamsbashforthmethod(d, n):
   (tmin, tmax, x0) <- (0, 1, 1)
   (t, xapprox) <- adamsbashforth(f1, ???, ???, ???, rk41step, d, n)
   maxerror <- max(abs(x1(t) - xapprox))
   maxerror
```

Ejecutamos las funciones programadas:

```
testadamsbashforthmethod(4, 100)
```