Klasszikus ODE megoldási eljárások pontosságának összehasonlítása a harmonikus oscillátor példáján

Bartha Ferenc, SZTE, 2002. március

```
> restart;
```

A differenciálegyenlet: $equ := \frac{\partial}{\partial t} x(t) = v(t), \frac{\partial}{\partial t} v(t) = -x(t)$

Kezdöfeltételek: $ini := x(0) = 0, v(0) = 1$

Analitikus megoldás: $fun := (t, k) \rightarrow \sin(t)$

Vizsgáljuk a megoldásokat az $x = [0, 3\pi]$ *intervallumon*, 40 *pontban*

```
n:=40:dh:=(3*Pi-0.)/n:
hpnt:=array(1..n+1,[seq(i*dh,i=0..n)]):
```

Választható klasszikus formulák:

- **foreuler** is the *forward Euler* method specified by the equation:

```
Y[n+1] = Y[n] + h*f(t[n], Y[n])
```

- **heunform** is the *Heun* formula (also known as the trapezoidal rule, or the improved Euler method), as specified by the equation:

```
Y[n+1] = Y[n] + (h/2)*(f(t[n],Y[n]) + f(t[n+1],Y[n+1]))
```

- **impoly** is the *improved polygon* method (also known as the modified Euler method), as specified by the equation:

```
Y[n+1] = Y[n] + h*(f(t[n]+h/2, Y[n]+(h/2)*f(t[n],Y[n])))
```

- **rk2** is the *second-order classical Runge-Kutta* method, as specified by:

```
k1     = f(t[n], Y[n])
k2     = f(t[n]+h, Y[n]+h*k1)
Y[n+1] = Y[n] + (h/2)*(k1+k2)
```

- **rk3** is the *third-order classical Runge-Kutta* method, as specified by:

```
k1     = f(t[n], Y[n])
k2     = f(t[n]+(h/2), Y[n]+(h/2)*k1)
k3     = f(t[n]+h, Y[n]+(2*h/3)*k2)
Y[n+1] = Y[n] + (h/6)*(k1+4*k2+k3)
```

- **rk4** is the *fourth-order classical Runge-Kutta* method, as specified by:

```
k1     = f(t[n], Y[n])
k2     = f(t[n]+h/2, Y[n]+(h/2)*k1)
k3     = f(t[n]+h/2, Y[n]+(h/2)*k2)
k4     = f(t[n]+h, Y[n]+h*k3)
Y[n+1] = Y[n] + (h/6)*(k1+2*k2+2*k3+k4)
```

  This is not to be confused with **method=rkf45**, which uses a *Fehlberg fourth-fifth order Runge-Kutta method.*

- **adambash** is the *Adams-Bashford* method (a "predictor" method), as specified by:

```
Y[n+1] = Y[n] + (h/24) * (55*f(t[n],Y[n]) - 59*f(t[n-1],Y[n-1])
                        + 37*f(t[n-2],Y[n-2]) - 9*f(t[n-3],Y[n-3]))
```

- **abmoulton** is the *Adams-Bashford-Moulton* method (a "predictor-corrector" method), as specified by:

```
Y[n+1] = Y[n] + (h/24) * (9*f(t[n+1],Y[n+1]) + 19*f(t[n],Y[n])
```

```
- 5*f(t[n-1],Y[n-1]) + f(t[n-2],Y[n-2]))
```

```
> meth1:='foreuler';meth2:='abmoulton';h:=1/10;
```

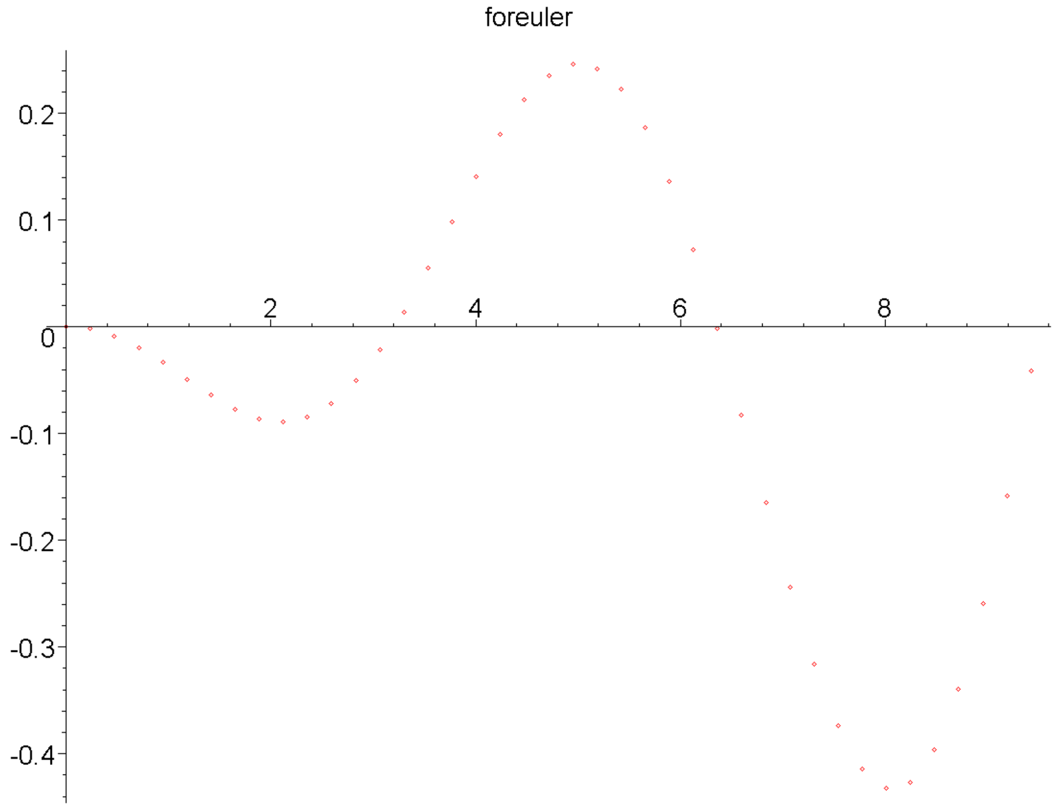
$$meth1 := foreuler$$

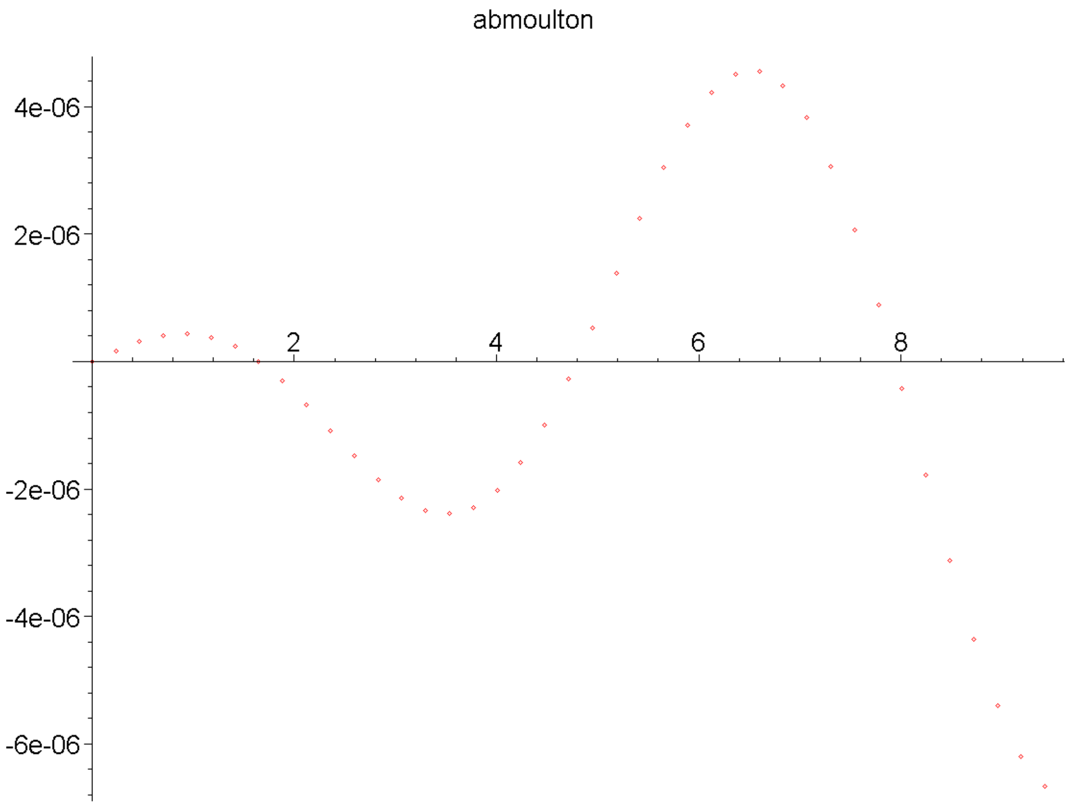
$$meth2 := abmoulton$$

$$h := \frac{1}{10}$$

```
>  dsol1 := dsolve([equ,ini], type=numeric,
method=classical[meth1],
 output=hpnt,stepsize=h):
 pl1:=dsol1[2,1]:

 dsol2 := dsolve([equ,ini], type=numeric,
method=classical[rk4],
 output=hpnt,stepsize=h):
 pl2:=dsol2[2,1]:
> plot({[pl1[s,1],fun(pl1[s,1],1.)-pl1[s,3]] $s=1..n+1},
style=point,
title=meth1);
> plot({[pl2[s,1],fun(pl2[s,1],1.)-pl2[s,3]] $s=1..n+1},
style=point,
title=meth2);
```




abmoulton
4e-06
2e-06
-2e-06
-4e-06
-6e-06
2
4
6
8

>

>