Adams solver for ODE

Using Adams solver for 2- and 3- body problems

Severin Denisenko

October 8, 2024

Saint Petersburg State University

Table of contents

History

History

- Method called after British astronomer John Couch Adams.
- Described in work Bashforth, F. and Adams, J. C. Theories of Capillary Action. London: Cambridge University Press, 1883.
- But apparently was invented long before in 1855.
- After this was completely forgotten.
- Apparently A. N. Krylov revived works of Adams and Bashforth.

Formulation

Formulation

For the problem:

$$y' = f(x, y)$$
$$y(x_0) = y_0$$

Extrapolation (or Adams-Bashforth) method:

$$y_{n+1} = y_n + h \sum_{\lambda=0}^{k} u_\lambda f(x_{n-\lambda}, y_{n-\lambda}) \tag{1}$$

Where u_λ are calculated coefficients.

Formulation

For the Adams extrapolation method:

$$u_\lambda = \frac{(-1)^\lambda}{j!(k-\lambda-1)!} \int_0^1 \prod_{\substack{i=0 \\ i\neq\lambda}}^{k} (\nu + i) d\nu$$

The easiest way to calculate this integral is numerical.

Order of approximation

The Adams extrapolation method of order k have order of approximation $p = k$ (local errors is of order $O(h^{p+1})$).

For 2-step Adams method:

$$y_{n+2} = y_{n+1} + h((1-\lambda)f_{n+1} + \lambda f_n)$$

$$y_{n+2} = y(t_{n+1}) + h((1-\lambda)y'(t_{n+1}) + \lambda y'(t_n))$$

Order of approximation

Using Taylor's theorem expand $y'(t_n)$ at t_{n+1}:

$$y_{n+2} = y(t_{n+1}) + h((1-\lambda)y'(t_{n+1}) + \lambda(y'(t_{n+1}) - hy''(t_{n+1}) + O(h^2)))$$

$$y_{n+2} = y(t_{n+1}) + hy'(t_{n+1}) - \lambda h^2 y''(t_{n+1}) + O(h^3)$$

Since $\lambda = -1/2$:

$$y_{n+2} = y(t_{n+1}) + hy'(t_{n+1}) + \frac{1}{2}h^2 y''(t_{n+1}) + O(h^3)$$

End finaly:

$$y(t_{n+2}) - y_{n+2} = O(h^3)$$

Implementation

List of used software

- MPFR is a multiple-precision floating-point computation library
- mpreal is a c++ library for convenient use of MPFR
- Boost::uBLAS is a wrapper of BLAS for convenient use of procedures
- C++ of 20 standard
- CMake as a build system
- clang-format for auto-formatting of code
- address sanitizer for debugging of memory issues

2-body problem

Equation of motion

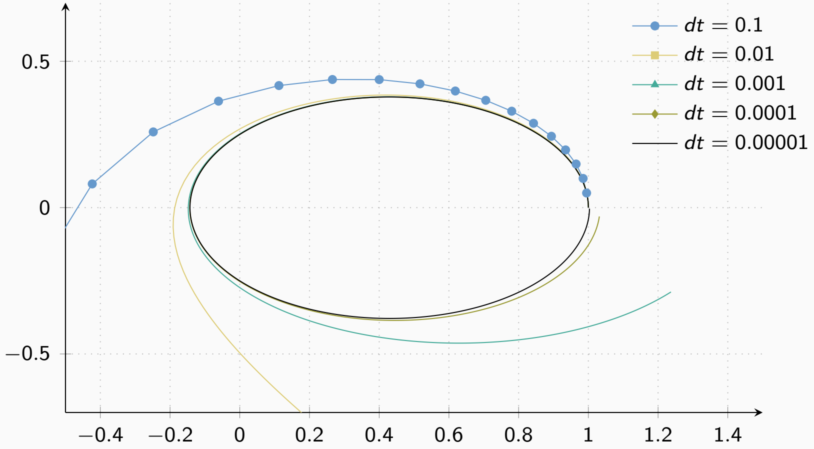
```
odes::ode_t ode = [](odes::real_t t,
                     odes::vector_t x)
-> odes::vector_t {
    odes::vector_t y(4);

    // pow is overrided for mpfr::mpreal
    odes::real_t z = pow(x[0] * x[0] + x[1] * x[1],
                         3.0 / 2.0);
    y[2] = -x[0] / z;
    y[3] = -x[1] / z;
    y[0] = x[2];
    y[1] = x[3];

    return y;
};
```

Equation of motion

```
odes::vector_t x0(4);
x0[0] = 1.0;
x0[1] = 0.0;
x0[2] = 0.0;
x0[3] = 0.5;
```

Solutions

First order



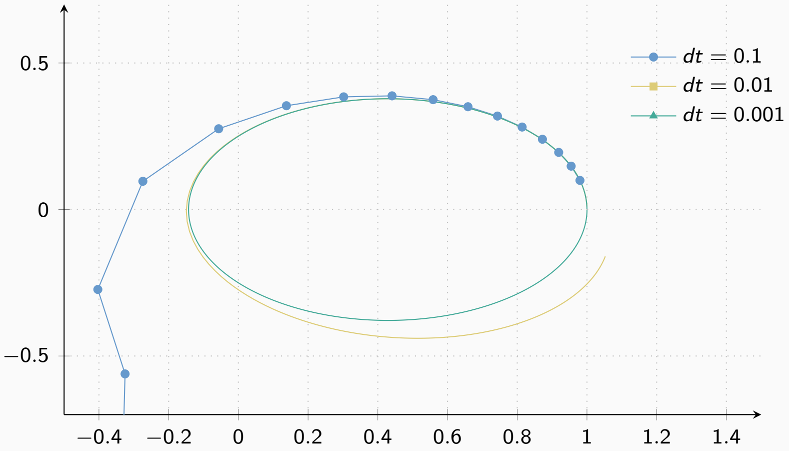
$dt = 0.1$
$dt = 0.01$
$dt = 0.001$
$dt = 0.0001$
$dt = 0.00001$
0.5
0
−0.5
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4

First order

Table 1: First order method precision

dt	x on t=T
0.1	-0.702892101
0.01	1.321601040
0.001	1.237004522
0.0001	1.031508929
0.00001	1.003246538
0.000001	1.000326558
0.0000001	1.000032663
exact	1.0

Second order

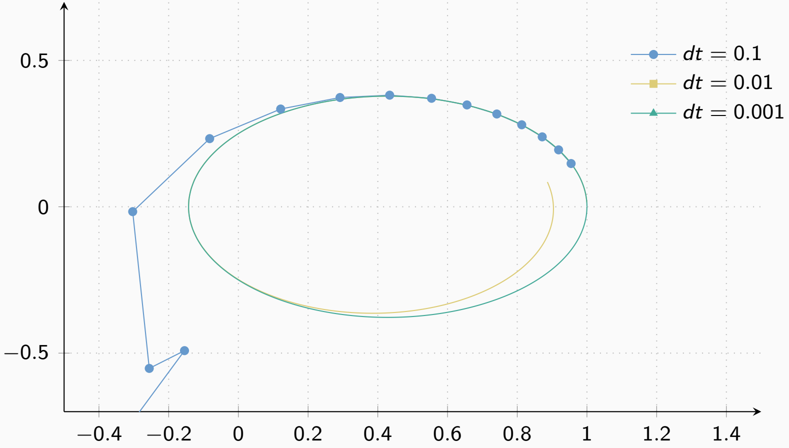


Second order

Table 2: Second order method precision

dt	x on t=T
0.1	-0.28799841125620
0.01	1.05254088205525
0.001	1.00007653064497
0.0001	1.00000007705246
0.00001	1.00000000004135
0.000001	0.99999999999949
exact	1.0

Third order



Third order

Table 3: Third order method precision

dt	x on t=T
0.1	-0.29895190046225
0.01	0.88646978773574
0.001	0.99988371313306
0.0001	0.99999988404660
0.00001	0.99999999984795
0.000001	0.99999999999988
0.0000001	1.00000000000005
exact	1.0

Fourth order (long double)

Table 4: Fourth order method precision

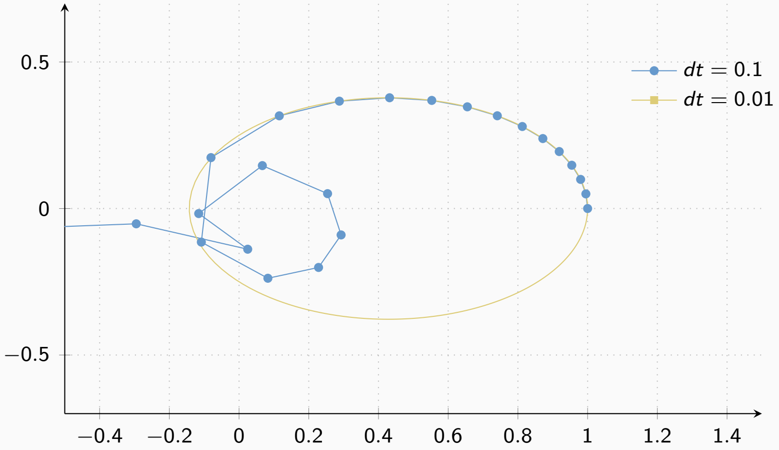
dt	x on t=T
0.1	0.88202009642157
0.01	0.97785887807003
0.001	0.99999932504978
0.0001	0.99999999982612
0.00001	0.99999999996357
0.000001	0.99999999999983
exact	1.0

Fourth order (mpfr)

Table 5: Fourth order method precision

dt	x on t=T
0.1	0.88204993803128
0.01	0.97785892721521
0.001	0.99999932554845
0.0001	0.99999999981791
0.00001	0.99999999995995
0.000001	1.00000000000000
exact	1.0

Runge-Kutta 4'th



Runge-Kutta 4'th

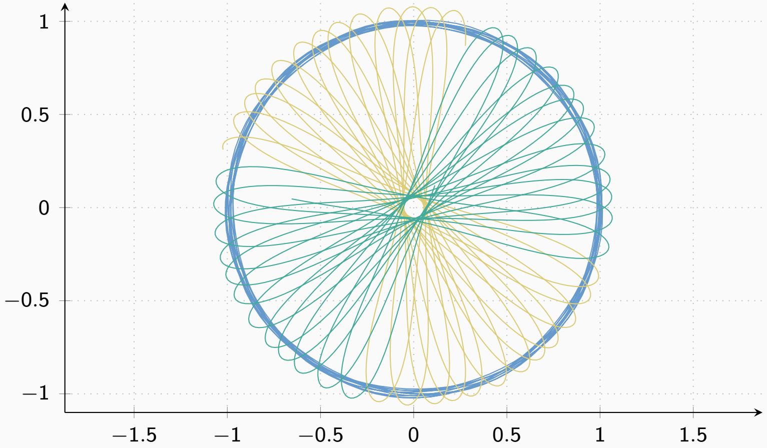
Table 6: Runge-Kutta 4'th method precision

dt	x on t=T
0.1	-4.39574629733722
0.01	0.99988122872003
0.001	0.99999957664443
0.0001	0.99999999981837
0.00001	0.99999999995897
0.000001	1.00000000000000
exact	1.0

Periodic 3-body solutions

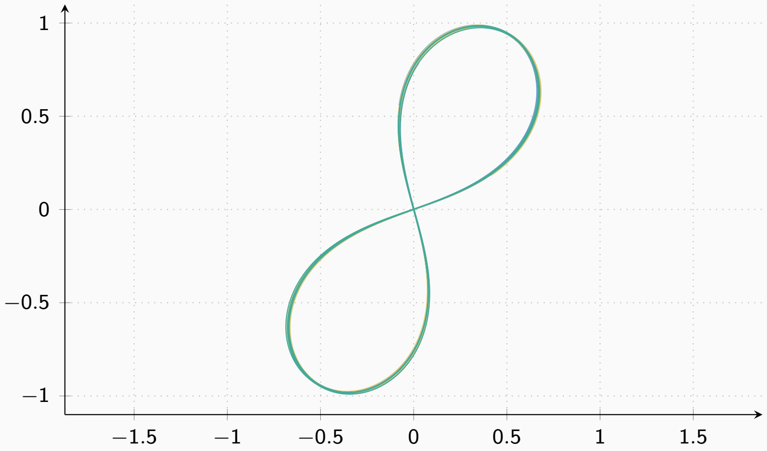
Three-body circled cross

T $= 24\ \pi$ [**?**]



Three-body figure-eight

$\mathsf{T} = 8\,\pi$



References i