1 symplectic_ode

1.1 Introduction to symplectic_ode

For a hamiltonian of the form F(p) + G(q), the function `symplectic_ode` numerically solves Hamilton's equations of motion $p' = -dH/dq, q' = dH/dp$, where H is the hamiltonian. The method preserves the Poisson bracket of p and q. One time step is accomplished with the loop

```
for k = 1 ... n
   q <-  q + c(k)*diff(H,p)*dt ;update momentum p
   p <-  p - d(k)*diff(H,q)*dt ;use updated position q
```

where $c1, c2, \ldots, cn$ and $d1, d2, \ldots, dn$ are constants and H is the hamiltonian.

This code has built-in methods for the symplectic Euler, the Verlet method, and third and fourth order methods that are due to Ronald Ruth. For a complete description of these methods, see https://en.wikipedia.org/wiki/Symplectic_integrator. Additionally, there is a fifth order method that is due to Kostas Tselios and T. E. Simos.

The function `symplectic_ode` creates and compiles functions for updating p & q. The arguments to these functions are modedeclared to have type given by an optional argument (defaults to float). Of course, hand coding of these functions could increase speed or accuracy, but the automatically generated functions are convenient.

Unlike adaptive methods such as RK45, `symplectic_ode` uses a fixed step size. Generally symplectic methods that use an adaptive step size lose their advantages over non-symplectic methods.

1.2 Definitions for symplectic_ode

`poisson_bracket` *poisson_bracket*(*f*, *g*, *p*, *q*) [Function]

poisson_bracket(*f*, *g*, [*p1*,. . . , *pn*], [*q1*,. . . , *qn*])

Compute the Poisson bracket of the expressions f and g with respect to the canonical coordinates p and q (or *p1*, *p2*,. . . , *pn* and *p1*, *p2*,. . . , *pn*).

Examples:

```
(%i1) load("symplectic_ode")$
(%i2) poisson_bracket(p,p^2/2+q^4,p,q);
(%o2) -4*q^3

(%i3) poisson_bracket(q,p^2/2+q^4,p,q);
(%o3) p

(%i4) poisson_bracket(q1,p1^2/2+p2^2/2+q1^4+q2^4,[p1,p2],[q1,q2]);
(%o4) p1
```

```
(%i5) poisson_bracket(p1,p1^2/2+p2^2/2+q1^4+q2^4,[p1,p2],[q1,q2]);
(%o5) -4*q1^3
```

`symplectic_ode` *symplectic_ode*(*ham,p,q,po,qo,dt,N*) [Function]

symplectic_ode(ham,p,q,po,qo,dt,N,method) symplectic_ode(ham,p,q,po,qo,dt,N,method,type)

Numerically solve Hamilton's equations of motion using a symplectic method. Specifically:

- The hamiltonian is the Maxima expression *ham* that depends on the canonical coordinates p and q. The hamiltonian must be time independent. The method is symplectic when the hamiltonian is separable; that is when it has the form `f(p) + g(q)`.
- The canonical coordinates are p and q. The arguments p and q should be symbols or equal length lists of symbols.
- The arguments *po* and *q0* are the initial values of p and q, respectively. These should be expressions or equal length lists of expressions. Generally, the values of *po* and *q0* should be numbers. When the optional argument *type* is float, the code attempts to convert the values of *po* and *qo* into floating point numbers; when this isn't possible, the code signals an error.
- *dt* is the fixed time step.
- N is the number of time steps.
- The optional argument *method* determines the integration method. It must be either symplectic_euler (default), verlet, symplectic_third_order, symplectic_fourth_order, or symplectic_fifth_order. For an explanation of these methods, see https://en.wikipedia.org/wiki/Symplectic_integrator.
- The optional argument *type* determines the value for mode_declare for various automatically generated functions. The value *type* must be one of float (default), rational, or any (no type). Since *float* is a Maxima option variable, the *type* variable should be quoted, especially for type *float*.

For both the scalar case (both p and q are mapatoms) and the nonscalar case (both p and q are lists of mapatoms), `symplectic_ode` returns a list of two lists. For the scalar case, the first list is a list of the values of p at the times `0, dt, 2*dt,..., N*dt` and similarly for the second list. For a nonscalar case, the first list is a list of the form $[p1, p2, \ldots, pn]$ at the times `0, dt, 2*dt,..., N*dt`.

Examples:

```
(%i2) load("symplectic_ode")$
(%i3) symplectic_ode(p^2/2 + q^4/4,p,q,1,0,1/10,2);
(%o3) [[1.0,1.0,0.9999],[0.0,0.1,0.19999]]
```

```
(%i4) symplectic_ode(p^2/2 + q^4/4,[p],[q],[1],[0],1/10,2);
(%o4) [[[1.0],[1.0],[0.9999]],[[0.0],[0.1],[0.19999]]]

(%i5) symplectic_ode(p^2/2 + q^4/4,p,q,1,0,1/10,2,verlet);
(%o5) [[1.0,0.9999875,0.9996500084374297],[0.0,0.099999375,0.1999812504218715]]

(%i6) symplectic_ode(p^2/2 + q^4/4,p,q,1.0b0,0.0b0, 0.1b0,2,verlet,'any);
(%o6) [[1.0b0,9.999875b-1,9.996500084374297b-1],[0.0b0,9.9999375b-2,1.99981250421
```

Appendix A Function and variable index

P

S

(Index is nonexistent)