15 Integration of ordinary differential equations

Functions:

rk4: integrate one step of ode's by 4th order Runge-Kutta
rkdumb: integrate ode's by 4th order Runge-Kutta
rkqc: integrate one step with accuracy monitoring
odeint: integrate ode's with accuracy monitoring
mmid: integrate ode's with the modified midpoint method
bsstep: integrate one step - Bulirsch-Stoer method

---

rk4

Description:

This function integrates one step of a system of ODE's using a fourth order Runge-Kutta method.

rk4(y, dydx, x, h, derivs) ⟶ yout

Where:

y is a real list representing the initial values of the dependent variables at x,
dydx is a real list representing initial values of the derivatives at x,
x is the initial value of the independent variable,
h is a real number representing the length of the interval of integration,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), \quad 1 \leq i \leq n$$

to be integrated,
yout is a list of floating point numbers representing the values of the dependent variables at the end of the interval.

Pre-variables:

rk4_nmax [10] an upper bound for the number of functions.

Example:

The system of 4 ordinary differential equations described by the right hand sides f1, f2, f3, and f4 have a solution interms of Bessel functions given by y(x). This system is used to test rk4 and other ODE solvers below.

```
-> y(x) := [bessj0(x),bessj1(x),bessj(2,x),bessj(3,x)];
-> f1(x,y1,y2,y3,y4) := -y2;
-> f2(x,y1,y2,y3,y4) := y1 - y2/x;
-> f3(x,y1,y2,y3,y4) := y2 - 2*y3/x;
-> f4(x,y1,y2,y3,y4) := y3 - 3*y4/x;
-> derivs : [f1,f2,f3,f4];
-> rk4: integrate one step of ode's by 4th order Runge-Kutta #
-> Choose Bessel functions for example: #
-> x : 1.0;
-> y(x)?
[0.765197684, 0.440050586, 0.114903485, 0.126596837]
-> dydx : [-proj(y(x),2), proj(y(x),1)-proj(y(x),2),
            proj(y(x),2)-2.0*proj(y(x),3), proj(y(x),3)-3.0*proj(y(x),4)]?
[-0.440050586, 0.325147098, 0.210243616, -0.264887025]
-> rk4(y(x),dydx,x,0.2,derivs)?
[0.671133418, 0.498289905, 0.159351031, 0.094848635]
-> Should get: #
-> y(1.2)?
[0.671132718, 0.498289070, 0.159349030, 0.083772293]
-> rk4(y(x),dydx,x,0.4,derivs)?
[0.566878968, 0.541970805, 0.207395193, 0.090113648]
-> Should get: #
-> y(1.4)?
[0.566855132, 0.541947710, 0.207355897, 0.081363027]
```

rkdumb

Description:

This function integrates a system of ODE's across a range using a fourth order Runge-Kutta method.

$$\textbf{rkdumb(vstart, x1, x2, nstep, derivs)} \longrightarrow [\textbf{xx}, \textbf{y}]$$

Where:

vstart is a list of real numbers representing the initial values of the dependent variables,
x1 is a real number representing the value of the independent variable at the start of the integration,
x2 is a real number representing the value of the independent variable at the end of the integration,
nstep is the number of equal increment steps used in the integration from $x1$ to $x2$,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), 1 \leq i \leq n$$

to be integrated,
xx is a floating point list represnting the values of the independent variable used,
y is a floating point matrix such the the i'th column represents the values of the dependent variables at the i'th value of x used.

Pre-variables:

rkdumb_nmax [10] the maximum number of functions,
rkdumb_nstpmx [200] the maximum number of points to be stored.

Example:

In this example the system of equations defined with the example for `rk4` above is integrated across [0.0, 1.4].

```
-> rkdumb: integrate ode's by 4th order Runge-Kutta #
-> x1 : 1.0;
-> x2 : 1.4;
-> vstart : y(x1)?
[0.765197684, 0.440050586, 0.114903485, 0.126596837]
-> nstep : 4;
-> rkdumb(vstart,x1,x2,nstep,derivs)?
[[1.000000000, 1.099999994, 1.199999988, 1.299999982, 1.399999976],
| 0.765197684  0.719622038  0.671132785  0.620086050  0.566855197 |
|                                                                 |
| 0.440050586  0.470902422  0.498289113  0.522023332  0.541947829 |
|                                                                 |]
| 0.114903485  0.136564232  0.159349145  0.183026857  0.207356078 |
|                                                                 |
| 0.126596837  0.106111795  0.094816739  0.089855483  0.089505516 |
-> Compare with: #
-> transpose(matrix(y(1),y(1.1),y(1.2),y(1.3),y(1.4)))?
| 0.765197684  0.719622005  0.671132718  0.620086012  0.566855132 |
|                                                                 |
| 0.440050586  0.470902402  0.498289070  0.522023237  0.541947710 |
|                                                                 |
| 0.114903485  0.136564159  0.159349030  0.183026689  0.207355897 |
|                                                                 |
| 0.126596837  0.093647450  0.083772293  0.080967684  0.081363027 |
```

rkqc

Description:

This function integrates one step of a sytem of ODE's using a Runge Kutta method with accuracy monitoring.

rkqc(y, dydx, x, htry, eps, yscal, derivs) $\longrightarrow$ [**y, x, hdid, hnext**]

Where:

y is a list of real numbers representing the initial values of the dependent variables,
dydx is a real list representing initial values of the derivatives at x,
x is the initial value of the independent variable,
htry is a real number representing the initial step size to be attempted,
eps is a real number being the required accuracy,
yscal a real list representing a vector of values against which the error is to be scaled,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), 1 \leq i \leq n$$

to be integrated,
hdid is a floating point number representing the stepsize which was used,
hnext is a floating point number representing the estimated stepsize for the next integration interval.

Pre-variables:

rkqc_nmax [10] the maximum number of equations,
rkqc_fcor [0.066d0] a floating point number used in the mopping up of the 5'th order truncation error,
rkqc_safety [0.9d0] a floating point number used in the calculation of *hnext*,
rkqc_errcon [6.0d-4] a floating point number used in the calculation of *hnext*.

Example:

In this example the system of equations defined in the example with `rk4` above is integrated a number of times with increasing accuracy.

```
-> rkqc: integrate one step of ode's with accuracy monitoring #
-> x : 1.0;
-> ys : y(x)?
[0.765197684, 0.440050586, 0.114903485, 0.126596837]
-> dydx : [-proj(ys,2), proj(ys,1)-proj(ys,2), proj(ys,2)-2.0*proj(ys,3),
  proj(ys,3)-3.0*proj(ys,4)]?
[-0.440050586, 0.325147098, 0.210243616, -0.264887025]
-> htry : 0.1;
-> eps : exp(-x)?
0.367879450
-> yscal : [1.0, 1.0, 1.0, 1.0];
-> rkqc(ys,dydx,x,htry,eps,yscal,derivs)?
[[0.719622015, 0.470902395, 0.136564154, 0.106110359], 1.100000001,
0.100000001, 0.400000006]
-> Compare to: #
-> y(1.1)?
[0.719622005, 0.470902402, 0.136564159, 0.093647450]
-> for i to 5 do sprog(eps : exp(-1.0*i),
  print([eps,rkqc(ys,dydx,x,htry,eps,yscal,derivs)]));
[0.367879450, [[0.719622015, 0.470902395, 0.136564154, 0.106110359],
1.100000001, 0.100000001, 0.400000006]]
[0.135335280, [[0.719622015, 0.470902395, 0.136564154, 0.106110359],
1.100000001, 0.100000001, 0.400000006]]
[0.049787067, [[0.719622015, 0.470902395, 0.136564154, 0.106110359],
1.100000001, 0.100000001, 0.400000006]]
[0.018315640, [[0.719622015, 0.470902395, 0.136564154, 0.106110359],
1.100000001, 0.100000001, 0.400000006]]
[0.006737947, [[0.719622015, 0.470902395, 0.136564154, 0.106110359],
1.100000001, 0.100000001, 0.400000006]]
```

odeint

Description:

This function integrates a system of ODE's across a range using a Runge-Kutta or Bulirsh-Stoer step with accuracy monitoring.

odeint(ystart, x1, x2, eps, h1, hmin, derivs)
$\longrightarrow$ **[ystart, xp, yp, kount, nok, nbad]**

Where:

ystart is a list of real numbers representing the initial values of the dependent variables on input and on output the final values of the variables,
x1 is a real number representing the value of the independent variable at the start of the integration,
x2 is a real number representing the value of the independent variable at the end of the integration,
eps is a real number representing the required accuracy,
h1 is a real number representing a guess for the first step size,
hmin is a real number representing the minimum allowed stepsize,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), 1 \leq i \leq n$$

to be integrated,
xp is a floating point list of length *kount* representing the points which have been chosen,
yp is a *nvar* by *kount* floating point matrix such the the i'th row represents the values of the dependent variables at the i'th value of x used,
kount is the number of results stored,
nok is the number of results for which the stepsize used equals the stepsize guessed,
nbad is the number of results for which the stepsize used is not equal to the stepsize guessed.

Pre-variables:

odeint_step ["rkqc"] the type of step to be used - the other option is "bsstep",
odeint_maxstp [1000] the maximum number of steps which may be taken,
odeint_nmax [10] the maximum number of equations,
odeint_tiny [1.0d-30] a small floating point number,
odeint_kmax [200] storage stops after this number is exceeded,
odeint_dxsav [0.01d0] results are stored only after the step exceeds this number.

Example:

In this example the system of equations defined with rk4 above is integrated across a range [1,1.4] with accuracy monitoring.

```
-> x1 : 1.0;
-> x2 : 1.4;
-> ystart : y(x1)?
[0.7651977, 0.4400506, 0.1149035, 0.1265968]
-> eps : 0.01;
-> h1 : 0.1;
-> hmin : 0.0;
-> odeint(ystart,x1,x2,eps,h1,hmin,derivs)?
[[0.5668551, 0.5419478, 0.2073560, 0.0895029], [0.0000000, 1.0000000,
            | 0.0000000  0.7651977  0.5668551 |
            |                                 |
            | 0.0000000  0.4400506  0.5419478 |
1.4000000], |                                 |, 3, 2, 0]
            | 0.0000000  0.1149035  0.2073560 |
            |                                 |
            | 0.0000000  0.1265968  0.0895029 |
-> Compare to: #
-> y(1.4)?
[0.5668551, 0.5419477, 0.2073559, 0.0813630]
```

mmid

Description:

This function integrates a system of ODE's using the modified mid-point method.

mmid(y, dydx, xs, htot, nstep, derivs) $\longrightarrow$ yout

Where:

y is a list of real numbers representing the initial values of the dependent variables,
dydx is a real list representing initial values of the derivatives at x,
xs is the initial value of the independent variable,
htot is a real number representing the total step to be taken,
nstep is the number of substeps to be used,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), 1 \leq i \leq n$$

to be integrated,
yout is a list of floating point numbers representing the values of the dependent variables at the end of the interval.

Pre-variables:

mmid_nmax [10] the maximum number of equations.

Example:

The system of equations defined with `rk4` above is integrated across a range using the function `mmid`.

```
-> xs : 1.0;
-> ys : y(xs)?
[0.765197684, 0.440050586, 0.114903485, 0.126596837]
-> htot : 0.5;
-> dydx : [-proj(ys,2), proj(ys,1)-proj(ys,2), proj(ys,2)-2.0*proj(ys,3),
  proj(ys,3)-3.0*proj(ys,4)]?
[-0.440050586, 0.325147098, 0.210243616, -0.264887025]
-> nstep : 5;
-> mmid(ys,dydx,xs,htot,nstep,derivs)?
[0.511557162, 0.557758289, 0.232422599, 0.093302978]
-> Compare to: #
-> y(1.5)?
[0.511827671, 0.557936508, 0.232087679, 0.083555546]
```

bsstep

Description:

This function integrates a system of ODE's using a Bulirsch-Stoer type step and either rational or polynomial interpolation.

bsstep(y, dydx, x, htry, eps, yscal, derivs) $\longrightarrow$ **[y, x, hdid, hnext]**

Where:

y is a list of real numbers representing the initial values of the dependent variables,
dydx is a real list representing initial values of the derivatives at x,
x is the initial value of the independent variable,
htry is a real number representing the initial stepsize to be attempted,
eps is a real number being the required accuracy,
yscal a real list representing a vector of values against which the error is to be scaled,
derivs is a list $[f_1, ..., f_n]$ of the names of Senac functions representing the right hand sides of the system of equations

$$y_i' = f_i(x, y_1, ..., y_n), \quad 1 \leq i \leq n$$

to be integrated,
hdid is a floating point number representing the stepsize accomplished,
hnext is a floating point number representing an estimate of the stepsize to be used for the next integration.

Pre-variables:

bsstep_interp ["rzextr"] is a string representing the method of interpolation to be used - the other allowed value is "pzextr",
bsstep_nmax [10] the maximum number of equations,
bsstep_imax [11] the maximum sequential index of the number of substeps (where the value 11 corresponds to 96 substeps),
bsstep_nuse [7] the substep level after which earlier information is discarded when attempting this number of steps.
bsstep_ncol [7] maximum expected value of *nuse* passed to `rzextr` or `pzextr`.

Example:

The system of equations defined with rk4 above is integrated using bsstep and the results compared with the true solution.

```
-> float_format(5,"f");
-> x : 1.0;
-> ys : y(x)?
[0.76520, 0.44005, 0.11490, 0.12660]
-> dydx : [-proj(ys,2), proj(ys,1)-proj(ys,2), proj(ys,2)-2.0*proj(ys,3),
  proj(ys,3)-3.0*proj(ys,4)]?
[-0.44005, 0.32515, 0.21024, -0.26489]
-> htry : 0.1;
-> eps : exp(-x)?
0.36788
-> yscal : [1.0, 1.0, 1.0, 1.0];
-> bsstep(ys,dydx,x,htry,eps,yscal,derivs)?
[[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]
-> Compare to: #
-> y(1.1)?
[0.71962, 0.47090, 0.13656, 0.09365]
-> for i to 5 do sprog(eps : exp(-1.0*i),
  print([eps,bsstep(ys,dydx,x,htry,eps,yscal,derivs)]));
[0.36788, [[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]]
[0.13534, [[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]]
[0.04979, [[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]]
[0.01832, [[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]]
[0.00674, [[0.71962, 0.47090, 0.13656, 0.10611], 1.10000, 0.10000, 0.20000]]
```