NAG Toolbox for MATLAB

d02xk

1 Purpose

d02xk interpolates components of the solution of a system of first-order ordinary differential equations from information provided by the integrators in sub-chapter D02M/N. It provides C^1 interpolation suitable for general use.

2 Syntax

```
[sol, ifail] = d02xk(xsol, m, ysav, acor, neq, x, nqu, hu, h, 'sdysav', sdysav)
```

3 Description

d02xk evaluates the first m components of the solution of a system of ordinary differential equations at any point using C^1 polynomial interpolation based on information generated by the integrator. This information must be passed unchanged to d02xk. d02xk should not normally be used to extrapolate outside the range of values obtained from the above functions.

It may be used with the D02N functions only when the BDF integration method is being employed (setup function d02nv), provided the Petzold error test was not selected.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **xsol – double scalar**

The point at which the first m components of the solution are to be evaluated. **xsol** should not be an extrapolation point, that is **xsol** should satisfy $(\mathbf{xsol} - \mathbf{x}) \times \mathbf{hu} \leq 0.0$. Extrapolation is permitted but not recommended.

2: **m – int32 scalar**

the number of components of the solution whose values at **xsol** are required. The first m components are evaluated.

Constraint: $1 \leq \mathbf{m} \leq \mathbf{neq}$.

3: **ysav**(**ldysav,sdysav**) **– double array**

ldysav, the first dimension of the array, must be at least 1.

The values provided in the parameter **ysav** on return from the integrator.

4: **acor**(**ldysav**) **– double array**

ldysav, the first dimension of the array, must be at least 1.

The value returned in position $(\mathbf{ldysav} + 50 + i)$, for $i = 1, 2, \ldots, \mathbf{neq}$, of the parameter **rwork** returned by the integrator. If one of the forward communication D02N functions is being employed and d02xk is to be used in (sub)program **monitr**, then **acor**(i) must contain the value given in position $(i, 2)$ of the **monitr** parameter **acor**, for $i = 1, 2, \ldots, \mathbf{neq}$ (e.g., see d02nb).

5: **neq – int32 scalar**

The value used for the parameter **neq** when calling the integrator.

Constraint: $1 \leq \mathbf{neq} \leq \mathbf{ldysav}$.

6: **x – double scalar**

The latest value at which the solution has been computed, as provided in the parameter **tcur** on return from the optional output d02ny.

7: **nqu – int32 scalar**

The order of the method used up to the latest value at which the solution has been computed, as provided in the parameter **nqu** on return from the optional output d02ny.

Constraint: $\mathbf{nqu} \geq 1$.

8: **hu – double scalar**

The last successful step used, that is the step used in the integration to get to **x**, as provided in the parameter **hu** on return from the optional output d02ny.

9: **h – double scalar**

The next step size to be attempted in the integration, as provided in the parameter **h** on return from the optional output d02ny.

5.2 Optional Input Parameters

1: **sdysav – int32 scalar**

Default: The dimension of the array **ysav**.

the value used for the parameter **sdysav** when calling the integrator.

Constraint: $\mathbf{sdysav} \geq \mathbf{nqu} + 1$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldysav

5.4 Output Parameters

1: **sol(m) – double array**

The calculated value of the ith component of the solution at **xsol**, for $i = 1, 2, \ldots, m$.

2: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{m} < 1$,
or $\mathbf{neq} < 1$,
or $\mathbf{ldysav} < 1$,
or $\mathbf{neq} > \mathbf{ldysav}$,
or $\mathbf{m} > \mathbf{neq}$,
or $\mathbf{nqu} < 1$,

or **sdysav** $<$ **nqu** $+ 1$,
or the BDF integrator was not previously used,
or the Petzold error test, if applicable, was used.

ifail $= 2$

On entry, **hu** $= 0.0$ or **h** $= 0.0$. This error can only occur if **h** and **hu** have been changed by you or possibly if the integrator has failed before calling d02xk.

ifail $= 3$

d02xk has been called for extrapolation. Before returning with this error exit, the value of the solution at **xsol** is calculated and placed in **sol**.

7 Accuracy

The solution values returned will be of a similar accuracy to those computed by the integrator.

8 Further Comments

d02xk provides a C^1 interpolant and as such is ideal for most applications, for example for tabulation and root-finding. In general d02xk should be preferred to d02xj for interpolation as the latter provides only a C^0 interpolant. d02xj is the natural interpolant employed by the BDF method and it is supplied only to permit you to reproduce the internal values used by the integrator.

9 Example

```
neq = int32(3);
t = 0;
tout = 10;
xout = 2;
iout = int32(2);
y = [1; 0; 0];
ydot = [0; 0; 0];
rwork = zeros(62,1);
rtol = [0.0001];
atol = [1e-07];
itol = int32(1);
inform = zeros(23,1,'int32');
ysave = zeros(3, 6);
wkjac = zeros(12,1);
jacpvt = [int32(0)];
imon = int32(8185424);
inln = int32(8184256);
ires = int32(1);
irevcm = int32(0);
itask = int32(1);
itrace = int32(0);

lacorb=int32(53);
lacor1 = lacorb + 1;
lacor2 = lacorb + 2;
lacor3 = lacorb + 3;
lsavrb = int32(56);
lsavr1 = lsavrb + 1;
lsavr2 = lsavrb + 2;
lsavr3 = lsavrb + 3;
[const, rwork, ifail] = ...
    d02nv(int32(3), int32(6), int32(5), 'Newton', false, zeros(6), ...
    0, 1e-10, 10, 0, int32(200), int32(5), 'Average-L2', rwork);
[rwork,  ifail]  =  d02ns(int32(3),  int32(3),  'Numerical',  int32(12),
rwork);
```

```
fprintf('\n      x           y(1)          y(2)          y(3)\n');
fprintf(' %8.3f %13.5f %13.5f %13.5f\n', t, y(1), y(2), y(3));

[t, y, ydot, rwork, inform, ysave, wkjac, jacpvt, imon, inln, ires,
irevcm, ifail] = ...
    d02nm(neq, t, tout, y, ydot, rwork, rtol, atol, itol, inform, ysave,
...
    wkjac, jacpvt, imon, inln, ires, irevcm, itask, itrace);
while (irevcm ~= 0)
  if (irevcm == 1  ||  irevcm == 3)
%   equivalent to fcn evaluation in forward communication routines
    rwork(lsavr1) = -0.04*y(1) + 1.0d4*y(2)*y(3);
    rwork(lsavr2) = 0.04*y(1) - 1.0d4*y(2)*y(3) - 3.0d7*y(2)*y(2);
    rwork(lsavr3) = 3.0d7*y(2)*y(2);
  elseif (irevcm == 4)
%   equivalent to fcn evaluation in forward communication routines
    rwork(lacor1) = -0.04*y(1) + 1.0d4*y(2)*y(3);
    rwork(lacor2) = 0.04*y(1) - 1.0d4*y(2)*y(3) - 3.0d7*y(2)*y(2);
    rwork(lacor3) = 3.0d7*y(2)*y(2);
  elseif (irevcm == 5)
%   equivalent to fcn evaluation in forward communication routines
    ydot(1) = -0.04*y(1) + 1.0d4*y(2)*y(3);
    ydot(2) = 0.04*y(1) - 1.0d4*y(2)*y(3) - 3.0d7*y(2)*y(2);
    ydot(3) = 3.0d7*y(2)*y(2);
  elseif (irevcm == 9)
%   equivalent to monitr call in forward communication routines
    if (imon == 1)
      tc = rwork(19);
      hlast = rwork(15);
      hnext = rwork(16);
      nqu = int32(rwork(10));
      while (iout < 6 && tc-hlast < xout && xout <= tc)
        [sol, ifail] = ...
            d02xk(xout, neq, ysave, rwork(lacor1:lacor1+neq-1), ...
                  neq, tc, nqu, hlast, hnext);
        if (ifail ~= 0)
          imon = -2;
        else
           fprintf(' %8.3f %13.5f %13.5f %13.5f\n', xout, sol(1), sol(2),
sol(3));
          iout = iout + 1;
          xout = double(iout)*2.0;
        end
      end
    end
  end
   [t, y, ydot, rwork, inform, ysave, wkjac, jacpvt, imon, inln, ires,
irevcm, ifail] = ...
    d02nm(neq, t, tout, y, ydot, rwork, rtol, atol, itol, inform, ...
    ysave, wkjac, jacpvt, imon, inln, ires, irevcm, itask, itrace);
end
```

```
x           y(1)          y(2)          y(3)
0.000       1.00000       0.00000       0.00000
2.000       0.94161       0.00003       0.05836
    6.000       0.87926       0.00002       0.12072
    8.000       0.85854       0.00002       0.14144
   10.000       0.84135       0.00002       0.15863
```