NUMERICAL RECIPES
Webnote No. 20, Rev. 1

Routine Implementing an Eighth-order Runge-Kutta Method

Here is the routine `StepperDopr853`. It uses a set of constants, which are provided by deriving the class from the class `Dopr853_constants`, also listed below.

stepperdopr853.h

```
template <class D>
struct StepperDopr853 : StepperBase, Dopr853_constants {
```
Dormand-Prince eighth-order Runge-Kutta step with monitoring of local truncation error to ensure accuracy and adjust stepsize. Only important differences from `StepperDopr5` are commented.
```
    typedef D Dtype;
    VecDoub yerr2;                              Use a second error estimator.
    VecDoub k2,k3,k4,k5,k6,k7,k8,k9,k10;
    VecDoub rcont1,rcont2,rcont3,rcont4,rcont5,rcont6,rcont7,rcont8;
    StepperDopr853(VecDoub_IO &yy, VecDoub_IO &dydxx, Doub &xx,
        const Doub atoll, const Doub rtoll, bool dens);
    void step(const Doub htry,D &derivs);
    void dy(const Doub h,D &derivs);
    void prepare_dense(const Doub h,VecDoub_I &dydxnew,D &derivs);
    Doub dense_out(const Int i, const Doub x, const Doub h);
    Doub error(const Doub h);
    struct Controller {
        Doub hnext,errold;
        bool reject;
        Controller();
        bool success(const Doub err, Doub &h);
    };
    Controller con;
};
template <class D>
StepperDopr853<D>::StepperDopr853(VecDoub_IO &yy,VecDoub_IO &dydxx,Doub &xx,
    const Doub atoll,const Doub rtoll,bool dens) :
    StepperBase(yy,dydxx,xx,atoll,rtoll,dens),yerr2(n),k2(n),k3(n),k4(n),
    k5(n),k6(n),k7(n),k8(n),k9(n),k10(n),rcont1(n),rcont2(n),rcont3(n),
    rcont4(n),rcont5(n),rcont6(n),rcont7(n),rcont8(n) {
    EPS=numeric_limits<Doub>::epsilon();
}
template <class D>
void StepperDopr853<D>::step(const Doub htry,D &derivs) {
```
This routine is essentially the same as the one in `StepperDopr5` except that `derivs` is called here rather than in `dy` because this method does not use FSAL.
```
    VecDoub dydxnew(n);
    Doub h=htry;
    for (;;) {
        dy(h,derivs);
        Doub err=error(h);
        if (con.success(err,h)) break;
        if (abs(h) <= abs(x)*EPS)
            throw("stepsize underflow in StepperDopr853");
    }
    derivs(x+h,yout,dydxnew);
    if (dense)
        prepare_dense(h,dydxnew,derivs);
    dydx=dydxnew;
    y=yout;
    xold=x;
    x += (hdid=h);
    hnext=con.hnext;
}
template <class D>
void StepperDopr853<D>::dy(const Doub h,D &derivs) {
    VecDoub ytemp(n);
    Int i;
    for (i=0;i<n;i++)                          Twelve stages.
        ytemp[i]=y[i]+h*a21*dydx[i];
    derivs(x+c2*h,ytemp,k2);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a31*dydx[i]+a32*k2[i]);
    derivs(x+c3*h,ytemp,k3);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a41*dydx[i]+a43*k3[i]);
    derivs(x+c4*h,ytemp,k4);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a51*dydx[i]+a53*k3[i]+a54*k4[i]);
    derivs(x+c5*h,ytemp,k5);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a61*dydx[i]+a64*k4[i]+a65*k5[i]);
    derivs(x+c6*h,ytemp,k6);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a71*dydx[i]+a74*k4[i]+a75*k5[i]+a76*k6[i]);
    derivs(x+c7*h,ytemp,k7);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a81*dydx[i]+a84*k4[i]+a85*k5[i]+a86*k6[i]+a87*k7[i]);
    derivs(x+c8*h,ytemp,k8);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a91*dydx[i]+a94*k4[i]+a95*k5[i]+a96*k6[i]+a97*k7[i]+
            a98*k8[i]);
    derivs(x+c9*h,ytemp,k9);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a101*dydx[i]+a104*k4[i]+a105*k5[i]+a106*k6[i]+
            a107*k7[i]+a108*k8[i]+a109*k9[i]);
    derivs(x+c10*h,ytemp,k10);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a111*dydx[i]+a114*k4[i]+a115*k5[i]+a116*k6[i]+
            a117*k7[i]+a118*k8[i]+a119*k9[i]+a1110*k10[i]);
    derivs(x+c11*h,ytemp,k2);
    Doub xph=x+h;
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a121*dydx[i]+a124*k4[i]+a125*k5[i]+a126*k6[i]+
            a127*k7[i]+a128*k8[i]+a129*k9[i]+a1210*k10[i]+a1211*k2[i]);
    derivs(xph,ytemp,k3);
    for (i=0;i<n;i++) {
        k4[i]=b1*dydx[i]+b6*k6[i]+b7*k7[i]+b8*k8[i]+b9*k9[i]+b10*k10[i]+
            b11*k2[i]+b12*k3[i];
        yout[i]=y[i]+h*k4[i];
    }
    for (i=0;i<n;i++) {                        Two error estimators.
        yerr[i]=k4[i]-bhh1*dydx[i]-bhh2*k9[i]-bhh3*k3[i];
        yerr2[i]=er1*dydx[i]+er6*k6[i]+er7*k7[i]+er8*k8[i]+er9*k9[i]+
             er10*k10[i]+er11*k2[i]+er12*k3[i];
    }
}
template <class D>
```

```
void StepperDopr853<D>::prepare_dense(const Doub h,VecDoub_I &dydxnew,
    D &derivs) {
    Int i;
    Doub ydiff,bspl;
    VecDoub ytemp(n);
    for (i=0;i<n;i++) {
        rcont1[i]=y[i];
        ydiff=yout[i]-y[i];
        rcont2[i]=ydiff;
        bspl=h*dydx[i]-ydiff;
        rcont3[i]=bspl;
        rcont4[i]=ydiff-h*dydxnew[i]-bspl;
        rcont5[i]=d41*dydx[i]+d46*k6[i]+d47*k7[i]+d48*k8[i]+
            d49*k9[i]+d410*k10[i]+d411*k2[i]+d412*k3[i];
        rcont6[i]=d51*dydx[i]+d56*k6[i]+d57*k7[i]+d58*k8[i]+
            d59*k9[i]+d510*k10[i]+d511*k2[i]+d512*k3[i];
        rcont7[i]=d61*dydx[i]+d66*k6[i]+d67*k7[i]+d68*k8[i]+
            d69*k9[i]+d610*k10[i]+d611*k2[i]+d612*k3[i];
        rcont8[i]=d71*dydx[i]+d76*k6[i]+d77*k7[i]+d78*k8[i]+
            d79*k9[i]+d710*k10[i]+d711*k2[i]+d712*k3[i];
    }
    for (i=0;i<n;i++)                          The three extra function evaluations.
        ytemp[i]=y[i]+h*(a141*dydx[i]+a147*k7[i]+a148*k8[i]+a149*k9[i]+
            a1410*k10[i]+a1411*k2[i]+a1412*k3[i]+a1413*dydxnew[i]);
    derivs(x+c14*h,ytemp,k10);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a151*dydx[i]+a156*k6[i]+a157*k7[i]+a158*k8[i]+
            a1511*k2[i]+a1512*k3[i]+a1513*dydxnew[i]+a1514*k10[i]);
    derivs(x+c15*h,ytemp,k2);
    for (i=0;i<n;i++)
        ytemp[i]=y[i]+h*(a161*dydx[i]+a166*k6[i]+a167*k7[i]+a168*k8[i]+
            a169*k9[i]+a1613*dydxnew[i]+a1614*k10[i]+a1615*k2[i]);
    derivs(x+c16*h,ytemp,k3);
    for (i=0;i<n;i++)
    {
        rcont5[i]=h*(rcont5[i]+d413*dydxnew[i]+d414*k10[i]+d415*k2[i]+d416*k3[i]);
        rcont6[i]=h*(rcont6[i]+d513*dydxnew[i]+d514*k10[i]+d515*k2[i]+d516*k3[i]);
        rcont7[i]=h*(rcont7[i]+d613*dydxnew[i]+d614*k10[i]+d615*k2[i]+d616*k3[i]);
        rcont8[i]=h*(rcont8[i]+d713*dydxnew[i]+d714*k10[i]+d715*k2[i]+d716*k3[i]);
    }
}
template <class D>
Doub StepperDopr853<D>::dense_out(const Int i,const Doub x,const Doub h) {
    Doub s=(x-xold)/h;
    Doub s1=1.0-s;
    return rcont1[i]+s*(rcont2[i]+s1*(rcont3[i]+s*(rcont4[i]+s1*(rcont5[i]+
        s*(rcont6[i]+s1*(rcont7[i]+s*rcont8[i]))))));
}
template <class D>
Doub StepperDopr853<D>::error(const Doub h) {
    Doub err=0.0,err2=0.0,sk,deno;
    for (Int i=0;i<n;i++) {
        sk=atol+rtol*MAX(abs(y[i]),abs(yout[i]));
        err2 += SQR(yerr[i]/sk);
        err += SQR(yerr2[i]/sk);
    }
    deno=err+0.01*err2;
    if (deno <= 0.0)
        deno=1.0;
    return abs(h)*err*sqrt(1.0/(n*deno));  The factor of h is here because it was omit-
}                                              ted when yerr and yerr2 were formed.
template <class D>
StepperDopr853<D>::Controller::Controller() : reject(false), errold(1.0e-4) {}
template <class D>
```

```
bool StepperDopr853<D>::Controller::success(const Doub err, Doub &h) {
```
Same controller as StepperDopr5 except different values of the constants.
```
    static const Doub beta=0.0,alpha=1.0/8.0-beta*0.2,safe=0.9,minscale=0.333,
        maxscale=6.0;
    Doub scale;
    if (err <= 1.0) {
        if (err == 0.0)
            scale=maxscale;
        else {
            scale=safe*pow(err,-alpha)*pow(errold,beta);
            if (scale<minscale) scale=minscale;
            if (scale>maxscale) scale=maxscale;
        }
        if (reject)
            hnext=h*MIN(scale,1.0);
        else
            hnext=h*scale;
        errold=MAX(err,1.0e-4);
        reject=false;
        return true;
    } else {
        scale=MAX(safe*pow(err,-alpha),minscale);
        h *= scale;
        reject=true;
        return false;
    }
}
```

Here is the class Dopr853_constants:

stepperdopr853.h
```
struct Dopr853_constants {
```
Constants for the Dormand-Prince 853 method.
```
    static const Doub c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c14,c15,c16,
        b1,b6,b7,b8,b9,b10,b11,b12,bhh1,bhh2,bhh3,
        er1,er6,er7,er8,er9,er10,er11,er12,
        a21,a31,a32,a41,a43,a51,a53,a54,a61,a64,a65,a71,a74,a75,a76,
        a81,a84,a85,a86,a87,a91,a94,a95,a96,a97,a98,a101,a104,a105,
        a106,a107,a108,a109,a111,a114,a115,a116,a117,a118,a119,a1110,
        a121,a124,a125,a126,a127,a128,a129,a1210,a1211,a141,a147,a148,
        a149,a1410,a1411,a1412,a1413,a151,a156,a157,a158,a1511,a1512,
        a1513,a1514,a161,a166,a167,a168,a169,a1613,a1614,a1615,
        d41,d46,d47,d48,d49,d410,d411,d412,d413,d414,d415,d416,d51,d56,
        d57,d58,d59,d510,d511,d512,d513,d514,d515,d516,d61,d66,d67,d68,
        d69,d610,d611,d612,d613,d614,d615,d616,d71,d76,d77,d78,d79,
        d710,d711,d712,d713,d714,d715,d716;
};
const Doub Dopr853_constants::c2  = 0.526001519587677318785587544488e-01;
const Doub Dopr853_constants::c3  = 0.789002279381515978178381316732e-01;
const Doub Dopr853_constants::c4  = 0.118350341907227396726757197510e+00;
const Doub Dopr853_constants::c5  = 0.281649658092772603273242802490e+00;
const Doub Dopr853_constants::c6  = 0.333333333333333333333333333333e+00;
const Doub Dopr853_constants::c7  = 0.25e+00;
const Doub Dopr853_constants::c8  = 0.307692307692307692307692307692e+00;
const Doub Dopr853_constants::c9  = 0.651282051282051282051282051282e+00;
const Doub Dopr853_constants::c10 = 0.6e+00;
const Doub Dopr853_constants::c11 = 0.857142857142857142857142857142e+00;
const Doub Dopr853_constants::c14 = 0.1e+00;
const Doub Dopr853_constants::c15 = 0.2e+00;
const Doub Dopr853_constants::c16 = 0.777777777777777777777777777778e+00;

const Doub Dopr853_constants::b1 =   5.42937341165687622380535766363e-2;
const Doub Dopr853_constants::b6 =   4.45031289275240888144113950566e0;
const Doub Dopr853_constants::b7 =   1.89151789931450038304281599044e0;
const Doub Dopr853_constants::b8 =  -5.80120396001058478146721142270e0;
const Doub Dopr853_constants::b9 =   3.11164366957819894408916062370e-1;
const Doub Dopr853_constants::b10 = -1.52160949662516078556178806805e-1;
const Doub Dopr853_constants::b11 =  2.01365400804030348374776537501e-1;
const Doub Dopr853_constants::b12 =  4.47106157277725905176885569043e-2;

const Doub Dopr853_constants::bhh1 = 0.244094488188976377952755905512e+00;
const Doub Dopr853_constants::bhh2 = 0.733846688281611857341361741547e+00;
const Doub Dopr853_constants::bhh3 = 0.220588235294117647058823529412e-01;

const Doub Dopr853_constants::er1  =  0.1312004499419488073250102996e-01;
const Doub Dopr853_constants::er6  = -0.1225156446376204440720569753e+01;
const Doub Dopr853_constants::er7  = -0.4957589496572501915214079952e+00;
const Doub Dopr853_constants::er8  =  0.1664377182454986536961530415e+01;
const Doub Dopr853_constants::er9  = -0.3503288487499736816886487290e+00;
const Doub Dopr853_constants::er10 =  0.3341791187130174790297318841e+00;
const Doub Dopr853_constants::er11 =  0.8192320648511571246570742613e-01;
const Doub Dopr853_constants::er12 = -0.2235530786388629525884427845e-01;

const Doub Dopr853_constants::a21 =    5.26001519587677318785587544488e-2;
const Doub Dopr853_constants::a31 =    1.97250569845378994544595329183e-2;
const Doub Dopr853_constants::a32 =    5.91751709536136983633785987549e-2;
const Doub Dopr853_constants::a41 =    2.95875854768068491816892993775e-2;
const Doub Dopr853_constants::a43 =    8.87627564304205475450678981324e-2;
const Doub Dopr853_constants::a51 =    2.41365134159266685502369798665e-1;
const Doub Dopr853_constants::a53 =   -8.84549479328286085344864962717e-1;
const Doub Dopr853_constants::a54 =    9.24834003261792003115737966543e-1;
const Doub Dopr853_constants::a61 =    3.7037037037037037037037037037e-2;
const Doub Dopr853_constants::a64 =    1.70828608729473871279604482173e-1;
const Doub Dopr853_constants::a65 =    1.25467687566822425016691814123e-1;
const Doub Dopr853_constants::a71 =    3.7109375e-2;
const Doub Dopr853_constants::a74 =    1.70252211019544039314978060272e-1;
const Doub Dopr853_constants::a75 =    6.02165389804559606850219397283e-2;
const Doub Dopr853_constants::a76 =   -1.7578125e-2;

const Doub Dopr853_constants::a81 =    3.70920001185047927108779319836e-2;
const Doub Dopr853_constants::a84 =    1.70383925712239993810214054705e-1;
const Doub Dopr853_constants::a85 =    1.07262030446373284651809199168e-1;
const Doub Dopr853_constants::a86 =   -1.53194377486244017527936158236e-2;
const Doub Dopr853_constants::a87 =    8.27378916381402288758473766002e-3;
const Doub Dopr853_constants::a91 =    6.24110958716075717114429577812e-1;
const Doub Dopr853_constants::a94 =   -3.36089262944694129406857109825e0;
const Doub Dopr853_constants::a95 =   -8.68219346841726006818189891453e-1;
const Doub Dopr853_constants::a96 =    2.75920996994467083049415600797e1;
const Doub Dopr853_constants::a97 =    2.01540675504778934086186788979e1;
const Doub Dopr853_constants::a98 =   -4.34898841810699588477366255144e1;
const Doub Dopr853_constants::a101 =   4.77662536438264365890433908527e-1;
const Doub Dopr853_constants::a104 =  -2.48811461997166764192642586468e0;
const Doub Dopr853_constants::a105 =  -5.90290826836842996371446475743e-1;
const Doub Dopr853_constants::a106 =   2.12300514481811942347288949897e1;
const Doub Dopr853_constants::a107 =   1.52792336328824235832596922938e1;
const Doub Dopr853_constants::a108 =  -3.32882109689848629194453265587e1;
const Doub Dopr853_constants::a109 =  -2.03312017085086261358222928593e-2;

const Doub Dopr853_constants::a111 =  -9.3714243008598732571704021658e-1;
const Doub Dopr853_constants::a114 =   5.18637242884406370830023853209e0;
const Doub Dopr853_constants::a115 =   1.09143734899672957818500254654e0;
const Doub Dopr853_constants::a116 =  -8.14978701074692612513997267357e0;
const Doub Dopr853_constants::a117 =  -1.85200656599969598641566180701e1;
const Doub Dopr853_constants::a118 =   2.27394870993505042818970056734e1;
const Doub Dopr853_constants::a119 =   2.49360555267965238987089396762e0;
const Doub Dopr853_constants::a1110 = -3.0467644718982195003823669022e0;
const Doub Dopr853_constants::a121 =   2.27331014751653820792359768449e0;
const Doub Dopr853_constants::a124 =  -1.05344954667372501984066689879e1;
const Doub Dopr853_constants::a125 =  -2.00087205822486249909675718444e0;
const Doub Dopr853_constants::a126 =  -1.79589318631187989172765950534e1;
const Doub Dopr853_constants::a127 =   2.79488845294199600508499808837e1;
const Doub Dopr853_constants::a128 =  -2.85899827713502369474065508674e0;
const Doub Dopr853_constants::a129 =  -8.87285693353062954433549289258e0;
const Doub Dopr853_constants::a1210 =  1.23605671757943030647266201528e1;
const Doub Dopr853_constants::a1211 =  6.43392746015763530355970484046e-1;

const Doub Dopr853_constants::a141 =  5.61675022830479523392909219681e-2;
const Doub Dopr853_constants::a147 =  2.53500210216624811088794765333e-1;
const Doub Dopr853_constants::a148 = -2.46239037470802489917441475441e-1;
const Doub Dopr853_constants::a149 = -1.24191423263816360469010140626e-1;
const Doub Dopr853_constants::a1410 =  1.53291798278765697312063226850e-1;
const Doub Dopr853_constants::a1411 =  8.20105229563468988491666602057e-3;
const Doub Dopr853_constants::a1412 =  7.56789766054569976138603589584e-3;
const Doub Dopr853_constants::a1413 = -8.298e-3;

const Doub Dopr853_constants::a151 =  3.18346481635021405060768473261e-2;
const Doub Dopr853_constants::a156 =  2.83009096723667755288322961402e-2;
const Doub Dopr853_constants::a157 =  5.35419883074385676223797384372e-2;
const Doub Dopr853_constants::a158 = -5.49237485713909884646569340306e-2;
const Doub Dopr853_constants::a1511 = -1.08347328697249322858509316994e-4;
const Doub Dopr853_constants::a1512 =  3.82571090835658412954920192323e-4;
const Doub Dopr853_constants::a1513 = -3.40465008687404560802977114492e-4;
const Doub Dopr853_constants::a1514 =  1.41312443674632500278074618366e-1;
const Doub Dopr853_constants::a161 = -4.28896301583791923408573538692e-1;
const Doub Dopr853_constants::a166 = -4.69762141536116384314449447206e0;
const Doub Dopr853_constants::a167 =  7.68342119606259904184240953878e0;
const Doub Dopr853_constants::a168 =  4.06898981839711007970213554331e0;
const Doub Dopr853_constants::a169 =  3.56727187455281109270669543021e-1;
const Doub Dopr853_constants::a1613 = -1.39902416515901462129418009734e-3;
const Doub Dopr853_constants::a1614 =  2.94751478915277233895562721490e0;
const Doub Dopr853_constants::a1615 = -9.15095847217987001081870187138e0;

const Doub Dopr853_constants::d41  = -0.84289382761090128651353491142e+01;
const Doub Dopr853_constants::d46  =  0.56671495351937776962531783590e+00;
const Doub Dopr853_constants::d47  = -0.30689499459498916912797304727e+01;
const Doub Dopr853_constants::d48  =  0.23846676565120698287728149680e+01;
const Doub Dopr853_constants::d49  =  0.21170345824450282767155149946e+01;
const Doub Dopr853_constants::d410 = -0.87139158377797299206789907490e+00;
const Doub Dopr853_constants::d411 =  0.22404374302607882758541771650e+01;
const Doub Dopr853_constants::d412 =  0.63157877876946881815570249290e+00;
const Doub Dopr853_constants::d413 = -0.88990336451333310820698117400e-01;
const Doub Dopr853_constants::d414 =  0.18148505520854727256656404962e+02;
const Doub Dopr853_constants::d415 = -0.91946323924783554000451984436e+01;
const Doub Dopr853_constants::d416 = -0.44360363875948939664310572000e+01;

const Doub Dopr853_constants::d51  =  0.10427508642579134603413151009e+02;
const Doub Dopr853_constants::d56  =  0.24228349177525818288430175319e+03;
const Doub Dopr853_constants::d57  =  0.16520045171727028198505394887e+03;
const Doub Dopr853_constants::d58  = -0.37454675472269020279518312152e+03;
const Doub Dopr853_constants::d59  = -0.22113666853125306036270938578e+02;
const Doub Dopr853_constants::d510 =  0.77334326684722638389603898808e+01;
const Doub Dopr853_constants::d511 = -0.30674084731089398182061213626e+02;
const Doub Dopr853_constants::d512 = -0.93321305264302278729567221706e+01;
const Doub Dopr853_constants::d513 =  0.15697238121770843886131091075e+02;
const Doub Dopr853_constants::d514 = -0.31139403219565177677282850411e+02;
const Doub Dopr853_constants::d515 = -0.93529243588444783865713862664e+01;
const Doub Dopr853_constants::d516 =  0.35816841486394083752465898540e+02;

const Doub Dopr853_constants::d61 =  0.19985053242002433820987653617e+02;
const Doub Dopr853_constants::d66 = -0.38703730874935176555105901742e+03;
const Doub Dopr853_constants::d67 = -0.18917813819516756882830838328e+03;
const Doub Dopr853_constants::d68 =  0.52780815920542364900561016686e+03;
const Doub Dopr853_constants::d69 = -0.11573902539959630126141871134e+02;
const Doub Dopr853_constants::d610 =  0.68812326946963000169666922661e+01;
const Doub Dopr853_constants::d611 = -0.10006050966910838403183860980e+01;
const Doub Dopr853_constants::d612 =  0.77771377980534432092869265740e+00;
const Doub Dopr853_constants::d613 = -0.27782057523535084065932004339e+01;
const Doub Dopr853_constants::d614 = -0.60196695231264120758267380846e+02;
const Doub Dopr853_constants::d615 =  0.84320405506677161018159903784e+02;
const Doub Dopr853_constants::d616 =  0.11992291136182789328035130030e+02;

const Doub Dopr853_constants::d71  = -0.25693933462703749003312586129e+02;
const Doub Dopr853_constants::d76  = -0.15418974869023643374053993627e+03;
const Doub Dopr853_constants::d77  = -0.23152937917604549567536039109e+03;
const Doub Dopr853_constants::d78  =  0.35763911791061412378285349910e+03;
const Doub Dopr853_constants::d79  =  0.93405324183624310003907691704e+02;
const Doub Dopr853_constants::d710 = -0.37458323136451633156875139351e+02;
const Doub Dopr853_constants::d711 =  0.10409964950896230045147246184e+03;
const Doub Dopr853_constants::d712 =  0.29840293426660503123344363579e+02;
const Doub Dopr853_constants::d713 = -0.43533456590011143754432175058e+02;
const Doub Dopr853_constants::d714 =  0.96324553959188282948394950600e+02;
const Doub Dopr853_constants::d715 = -0.39177261675615439165231486172e+02;
const Doub Dopr853_constants::d716 = -0.14972683625798562581422125276e+03;
```