J.G. Blom, H. Brunner

Discretized collocation and iterated collocation for nonlinear Volterra integral equations of the second kind

Department of Numerical Mathematics Report NM-R8618 July

Discretized Collocation and Iterated Collocation for Nonlinear Volterra Integral Equations of the Second Kind

J.G. Blom
Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

H. Brunner
Institut de Mathématiques, Université de Fribourg, CH-1700 Fribourg, Switzerland

In this paper a **FORTRAN** code is described for the approximate solution of systems of nonlinear Volterra integral equations of the second kind. The algorithm is based on polynomial spline collocation, possibly in combination with the corresponding iterated collocation. It exploits certain local superconvergence properties for the error estimation and the stepsize strategy.

1980 Mathematics subject classification: Primary:45-04, 65R20. Secondary:45D05, 45L10
Key Words & Phrases: systems of nonlinear Volterra integral equations of the second kind, approximation by polynomial splines, (iterated) collocation, local superconvergence, error estimations, variable stepsize, numerical methods, computer software.

Note: This paper is submitted for publication

1. Introduction

In this paper we describe the `FORTRAN 77` code `COLVI2` designed for the solution of a system of nonlinear Volterra integral equations of the second kind,

$$y(t) = g(t) + \int_0^t k(t,s,y(s))ds, \quad t \in I := [0,T]. \tag{1.1}$$

Here, g and k are continuous, real-valued functions, and $T>0$ is a given (finite) number; it will always be assumed that (1.1) possesses a unique solution $y \in C(I)$.

The theory underlying the method on which the present algorithm is based is given in[2] . In the following we present a brief summary of the main features of this method; although this is done for scalar equations (1.1), the extension to systems of integral equations of the form (1.1) is straightforward.

The solution y of (1.1) will be approximated in the space $S_{m-1}^{(-1)}(Z_N)$ of *polynomial spline functions* of degree $m-1$ which are allowed to be discontinuous at their knots $Z_N := \{t_n: n=1,...,N-1\}$ (defined by a given partition $\Pi_N : 0=t_0<t_1< \cdots <t_N=T$ of the interval I). Let u_n denote the restriction of the element $u \in S_{m-1}^{(-1)}(Z_N)$ to the subinterval $(t_n,t_{n+1}]$ (obviously, u_n is a polynomial of degree not exceeding $m-1$), and set $h_n := t_{n+1}-t_n$ $(n=0,...,N-1)$. The desired approximation u is found by *collocation* on the set of points

$$X(N) := \bigcup_{n=0}^{N-1} X_n, \tag{1.2a}$$

Report NM-R8618
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

where the subsets $\{X_n\}$ are defined by

$$X_n := \{t_{n,j} := t_n + c_j h_n : 0 \leqslant c_1 < \cdots < c_m \leqslant 1\}. \tag{1.2b}$$

The numbers $\{c_j\}$ will be referred to as collocation parameters. We require that u satisfy the given equation (1.1) on $X(N)$; i.e.,

$$u(t) = g(t) + \int_0^t k(t,s,u(s))ds \quad \text{for all } t \in X(N). \tag{1.3}$$

This collocation equation represents a recursive procedure for the successive computation of the restrictions $u_0, u_1, \ldots, u_{N-1}$ of u: we have, setting $t = t_{n,j} \in X_n$ in (1.3) and using an obvious substitution

$$u_n(t_{n,j}) = g(t_{n,j}) + F_n(t_{n,j}) + h_n \int_0^{c_j} k(t_{n,j}, t_n + \tau h_n, u_n(t_n + \tau h_n))d\tau, \tag{1.4a}$$

$j = 1,\ldots,m$, with lag term approximation F_n given by

$$F_n(t) := \sum_{i=0}^{n-1} h_i \int_0^1 k(t, t_i + \tau h_i, u_i(t_i + \tau h_i))d\tau, \quad n = 0,\ldots,N-1. \tag{1.4b}$$

Note that if the collocation parameters are such that $c_1 = 0$ and $c_m = 1$ then the resulting polynomial spline approximation u will be continuous on I; i.e.,

$$u \in S_{m-1}^{(-1)}(Z_N) \cap C(I) =: S_{m-1}^{(0)}(Z_N).$$

In general, a further discretization procedure will be necessary to render the collocation equation (1.4a) into a computationally feasible form, namely the approximation of the integrals occurring in (1.4a) and (1.4b) by suitable numerical quadrature processes. In the following we shall employ *s-point interpolatory quadrature formulas* based on the given collocation parameters $\{c_j\}$:

$$\hat{\Phi}_{n,i}^{(j)}[u_i] := \begin{cases} \sum_{l=1}^{s} w_l k(t_{n,j}, t_{i,l}, u_i(t_{i,l})), & \text{if } 0 \leqslant i \leqslant n-1, \\ \sum_{l=1}^{s} w_{j,l} k(t_{n,j}, t_n + c_j c_l h_n, u_n(t_n + c_j c_l h_n)), & \text{if } i = n \\ & (j = 1,\ldots,m,\ s = m \text{ or } m-1). \end{cases} \tag{1.5}$$

If s equals m the above are the usual interpolatory quadrature formulas based on all $\{c_j\}$. If $s = m-1$ the integrals are approximated by interpolatory quadrature rules based on only the first $m-1$ collocation parameters $\{c_j\}$, with $0 \leqslant c_1 < \cdots < c_{m-1} < c_m = 1$. The quadrature weights in (1.5) are thus determined by

$$w_l := \int_0^1 L_l^*(\tau)d\tau, \quad w_{j,l} := c_j w_l \quad (j = 1,\ldots,m,\ l = 1,\ldots,s) \tag{1.6a}$$

where

$$L_l^*(\tau) := \prod_{\substack{\kappa=1 \\ \kappa \neq l}}^{s} \frac{(\tau - c_\kappa)}{(c_l - c_\kappa)} \tag{1.6b}$$

denotes the lth Lagrange polynomial with respect to the (first s) collocation parameters $\{c_j\}$. The fully discretized version of (1.4a) then assumes the form

$$\hat{u}_n(t_{n,j}) = g(t_{n,j}) + \hat{F}_n(t_{n,j}) + h_n \hat{\Phi}_{n,n}^{(j)}[\hat{u}_n], \quad j = 1,\ldots,m \tag{1.7a}$$

where the *lag term approximation* $\hat{F}_n$ corresponding to (1.4b) is given by

$$\hat{F}_n(t_{n,j}) := \sum_{i=0}^{n-1} h_i \hat{\Phi}_{n,i}^{(j)}[\hat{u}_i], \quad n = 0,...,N-1. \tag{1.7b}$$

These equations define an approximation $\hat{u} \in S_{m-1}^{(-1)}(Z_N)$ which, due to the quadrature errors induced by the above quadrature formulas (1.5), will in general be different from the approximation u defined by the exact collocation equation (1.4a).

In order to compute $\hat{u}$ on the subinterval $(t_n, t_{n+1}]$ we have to solve a system of nonlinear equations in $\mathbb{R}^m$ for $\hat{Y}_{n,j} := \hat{u}_n(t_{n,j})$ $(j = 1,...,m)$. Once these values are known, the restriction $\hat{u}_n$ of $\hat{u}$ to this subinterval may be found by means of the Lagrange interpolation formula,

$$\hat{u}_n(t_n + \tau h_n) = \sum_{j=1}^{m} L_j(\tau) \hat{Y}_{n,j}, \quad t_n + \tau h_n \in (t_n, t_{n+1}], \tag{1.8a}$$

where

$$L_j(\tau) := \prod_{\substack{\kappa=1 \\ \kappa \neq j}}^{m} \frac{(\tau - c_\kappa)}{(c_j - c_\kappa)}. \tag{1.8b}$$

In particular, if $c_m = 1$, then we have $\hat{u}_n(t_{n+1}) = \hat{Y}_{n,m}$; otherwise,

$$\hat{u}_n(t_{n+1}) = \sum_{j=1}^{m} L_j(1) \hat{Y}_{n,j}.$$

For $t = t_n \in \bar{Z}_N := Z_N \bigcup \{T\}$ we shall also compute the so-called *iterated collocation approximation* $\hat{u}^I(t_n)$ corresponding to the collocation approximation $\hat{u}$: it is given by

$$\hat{u}^I(t_n) := g(t_n) + \sum_{i=0}^{n-1} h_i \sum_{l=1}^{m} w_l k(t_n, t_{i,l}, \hat{Y}_{i,l}). \tag{1.9}$$

Note that $\hat{u}^I(t_n) = \hat{u}(t_n)$ $(= \hat{Y}_{n-1,m})$ whenever $c_m = 1$.

The motivation for introducing the iterated collocation approximation lies in the fact that it leads to optimal local superconvergence on $\bar{Z}_N$ if the collocation parameters $\{c_j\}$ are the *Gauss points* (i.e., the zeros of $P_m(2s-1)$, where P_m is the Legendre polynomial of degree m). More precisely, while this choice of the $\{c_j\}$ yields only

$$\max_{t_n \in \bar{Z}_N} |y(t_n) - \hat{u}(t_n)| = O(N^{-m}),$$

we have

$$\max_{t_n \in \bar{Z}_N} |y(t_n) - \hat{u}^I(t_n)| = O(N^{-2m}).$$

If we take as collocation parameters $m-1$ *Gauss points* and add $c_m = 1$ and use the $m-1$ point interpolatory quadrature formulas we get:

$$\max_{t_n \in \bar{Z}_N} |y(t_n) - \hat{u}(t_n)| = O(N^{-(2m-2)}).$$

For the *Radau II points* (zeros of $P_{m-1}(2s-1) - P_m(2s-1)$) and for the *Lobatto points* (zeros of $s(s-1)P'_{m-1}(2s-1)$) we obtain, respectively,

$$\max_{t_n \in \bar{Z}_N} |y(t_n) - \hat{u}(t_n)| = O(N^{-(2m-1)})$$

and

$$\max_{t_n \in \bar{Z}_N} |y(t_n) - \hat{u}(t_n)| = O(N^{-(2m-2)}).$$

Recall that in the last three cases the iterated approximations at $t = t_n \in \overline{Z}_N$ coincide with the values $\hat{u}(t_n)$ since we have $c_m = 1$.

2. Error control

2.1. *Summary of the stepsize strategy*

Suppose we have computed an approximation to the solution in the interval $(t_n, t_{n+1}]$ using the formulas (1.7a,b) and (1.8a,b). The next thing is to estimate the truncation error made so far and to decide whether this error is acceptable.

In the code three different strategies have been implemented to limit $|y(t) - \hat{u}(t)|$ for all $t \in I$ to a user-prescribed tolerance *TOL*:

- based on a global error estimate in t_{n+1}:

$$GE_{n+1} := |y(t_{n+1}) - \hat{u}(t_{n+1})| \tag{2.1.1}$$

- based on the local error estimate in t_{n+1} (the ODE approach):

$$LE_{n+1} := \Big| \int_{t_n}^{t_{n+1}} k(t_{n+1}, s, y_n(s))ds - h_n \sum_{l=1}^{s} w_l k(t_{n+1}, t_{n,l}, \hat{Y}_{n,l}) \Big|, \quad \text{with} \tag{2.1.2}$$

$$y_n(t) := g(t) + \hat{F}_n(t) + \int_{t_n}^{t} k(t, s, y_n(s))ds.$$

With this type of error control one assumes that the global error in a point t is approximately equal to the sum of the local errors committed until t.

- based on a global error estimate for all $t \in [t_{n+1}, T]$. In this case the global error will be approximated by a sum of local errors in t in so far these local errors are known and by the last computed local error in t otherwise, i.e.:

Let

$$LE_{i+1}^{l} := \int_{t_i}^{t_{i+1}} k(t_l, s, y_i(s))ds - h_i \sum_{j=1}^{s} w_j k(t_l, t_{i,j}, \hat{Y}_{i,j}) \tag{2.1.3}$$

then

$$GE_{n+1}^{l} := \Big| \sum_{i=0}^{n-1} LE_{i+1}^{l} \Big| + (t_l - t_n) \frac{|LE_{n+1}^{l}|}{h_n}, \quad \text{for } t_{n+1} \leqslant t_l \leqslant T. \tag{2.1.4}$$

The current step is accepted if the global error in t_{n+1}, respectively the maximum of the global errors over $[t_{n+1}, T]$, is less than *TOL* and/or if the local error satisfies $LE_{n+1} \leqslant TOL.h_n / t_{n+1}$.

Regardless of the success of the last step the modification of the stepsize will be

$$h_{new} = h_n.(TOL / GE_{n+1})^{1/p}$$

if global error control in t_{n+1} is used. If the error is controlled over the interval $[t_{n+1}, T]$, the so-called uniform error control, the modification is

$$h_{new} = h_n.(TOL / \max GE_{n+1}^{l})^{1/q};$$

if local error control is used it reads:

$$h_{new} = h_n.(TOL.h_n / (t_{n+1}.LE_{n+1}))^{1/q}$$

Here q is the order of the quadrature formula and p the order of the collocation method (p differs from q only in the case of the *Gauss* collocation method). The idea behind these modifications is that we would like to take as large a stepsize as possible without letting the error exceed the specified tolerance.

It is important to notice that even global error control does not guarantee that the error can be limited as the integration advances, due to large errors in the lag term. Therefore a really robust, though for some problems too pessimistic, method will combine global error control with uniform error control; unless $p \ll q$, for such a combination already results in a pessimistic estimation of the new stepsize.

2.2. Implementation
In the above it is assumed that we have an approximation $\hat{u}$ in $(t_n, t_{n+1}]$ at hand. The computation of $\hat{u}$ involves the solution of the (nonlinear) system of collocation equations (1.7a,b). For this we use either functional iteration or Newton's method or a variant of it that uses only one evaluation of the Jacobian per step. Both need a decision as to when the corrector iteration process can be stopped. The system has to be solved to a somewhat greater accuracy than the user desired error bound. To achieve this we assume that the iteration process has converged if the norm of the corrector is less than *TOL* divided by an arbitrarily chosen factor 128 that results in an extra 7 bits of accuracy to cope with computational loss. For the computation of the reference solution this tolerance is tightened by a factor 10 to be ensured of a reliable error estimate.
The norm used throughout the program is the L_∞-norm with weights that depend on the error control strategy chosen by the user. By default the *i*-th component of an error vector is weighted by $1 / \max(1, |y^{(i)}|)$, but relative or absolute error control is also possible.

To estimate the error of the obtained approximation we need also a more accurate approximation that serves as reference solution. If the method with which $\hat{u}$ has been computed is based on collocation at the *Gauss* points and if we want to estimate the global error in t_{n+1} the obvious choice is to use *iterated collocation* to compute a reference solution. It is possible however, that this error estimate is very unreliable if the solution $y \in \pi_{m-1}$. We have built in the possibility to check this fact following the method as described in [2] and even to escape automatically to another method to compute the reference solution if it is decided that the solution behaves as a polynomial of degree less than *m*. If the approximation method is not based on Gauss collocation or if local error control is performed we use a higher order method with different collocation points to compute the reference solution with the stepsizes that were chosen when computing $\hat{u}$. If the stepsize strategy is based on local error control, eventually in combination with uniform error control over $[t_{n+1}, T]$, the lag term is approximated by Lagrangian interpolation using the lag terms already computed during the process of the computation of $\hat{u}$ (cf.[2]). This means a significant reduction of both the amount of computational work and of the amount of storage needed.

The stepsize choice as described in the previous subsection is not a very practical one. In reality one needs certain stepsize constraints both on the actual value of h_n and on the difference between two successive stepsizes. In the program the new stepsize after a success or a failure will be:

$$h_{new} = 0.9.h_n.R,$$

where *R* is normally the modification factor obtained by the rules in Section 2.1, but *R* can be restricted to ensure that:

$$0.5 \leq h_{new} / h_n \leq 2.0 .$$

The factor 0.9 has been added to get a conservative guess for the new stepsize. This is done because of the high cost of failed steps.
Furthermore the stepsize is restricted to minimum and maximum bounds `HMINN` and `HMAX`, which can be prescribed by the user but for which the code should provide defaults in case the user cannot supply satisfactory values. The lower bound imposed on h_n should ensure that the selected stepsize is at least large enough to make any advance at all in *t*. Moreover it is an empirical fact that the code never recovers from too small a stepsize (relative to other steps taken). Therefore, let `HMIN` be given

by the user or by default: HMIN=1E-5*HINIT, where HINIT is either chosen by the user or HINIT=min(T,1.0), then HMINN=max(HMIN,machine_precision times t_n). For the maximum stepsize it is much more difficult to provide a reasonable default value in the code. We chose an arbitrary 1.0, which moreover is the maximum value a stepsize can have to ensure that the check if $y \in \pi_{m-1}$ can be performed reliably.

If the error control fails even with the smallest allowed stepsize two possibilities are left, one is to issue an erratic exit to the calling program, the other is to relax the tolerance and go ahead. Both options have been implemented. The tolerance only will be relaxed until it is larger than 1.0.

Some remarks are to be made on the subject of uniform error control. Since it is impossible to estimate the global error in all $t \in [t_{n+1}, T]$ it is done so for T, T-HC, T-2HC, $\cdots$ and so on until t_{n+1} has been reached. The value HC can be specified by the user and is by default equal to HMAX. If T or HC are different in successive calls of COLVI2 the sum of local errors as needed in (2.1.4) is obviously not available. In that case the partial sum also will be based on the local error in t_{n+1} in the first step taken; this results in:

$$GE_{n+1}^{l} = (t_l - t_0).\frac{|LE_{n+1}^{l}|}{h_n}.$$

REMARK: All the constants used above are specified in PARAMETER statements in the FORTRAN code and can be easily altered by the user.

3. Description of the code

It was our objective to build a code that supplies by default an easy-to-use and robust method for solving Volterra integral equations of the second kind. In addition it should offer all kind of options by which the user can influence the kind of collocation methods and the error control strategy used if he either wishes to play with the different methods and stepsize strategies or knows specific facts about the problem that make it possible to use a more efficient method.

Moreover we tried to set up the code in such a way that it can be easily understood and adapted. To this goal an extensive documentation has been written of which the general part is gathered in a subroutine named COLDOC.

COLVI2 offers two default solvers for which a user needs to specify only the problem dependent functions and variables. The chosen collocation methods and error control strategies came out reliable and efficient in the testing we did in[2] . Both yield an estimation of the global error in the endpoint.

The first default offers an *8-point Gauss* collocation method combined with *iterated collocation* for the reference solution; the stepsize strategy is based on global error control. The code checks if the solution behaves as a polynomial of degree at most 7 and if so, uses from then on a collocation method based on 9 *Gauss* points + $c_{10} = 1$ with local and uniform error control. This escape method is the obvious one because the lag terms needed by the reference method can be computed by interpolation, so no reference solution values in the past nor the storage space for them will be needed. In case COLVI2 has escaped from iterated collocation as the method to compute the reference solution, the global error estimation in T will be based on the sum of the local errors in the endpoint T; if the escape was made in t_n the local errors in T from 0 to t_n will be approximated by the weighted local error in t_{n+1}.

The second default yields a *6-point Lobatto* method for the approximation combined with a *7-point Lobatto* to compute the reference solution. Again the stepsize strategy is based on global error control and, since the order of the quadrature formula is equal to that of the total method in the endpoint of a subinterval, also on uniform error control.

In both solvers the tolerance is relaxed if it cannot be met with the minimum stepsize.

The amount of working space needed by `COLVI2` can be computed by evaluating a rather complicated expression. However it is more convenient to have a shot at it and see how far the integration advances, since it is always possible to restart the integration from the last reached point.

3.1. Parameter description of `COLVI2`

The already mentioned subroutine `COLDOC` contains a general description of the user supplied parameters, the incorporated common blocks, an outline of the code and many other useful things to know when one wants to make an extensive use of `COLVI2`. Below we have copied that part of `COLDOC` that contains the parameter description and that gives a list of environmental constants and routines used from the IMSL library[1] that possibly need adaptation before invoking `COLVI2`.

```
C ---------------------------------------------------------------------I
C                                                                      I
C HOW TO USE:                                                          I
C ==========                                                           I
C INVOKE VIA A CALL OF THE SUBROUTINE "COLVI2". THE INPUT PARAMETERS   I
C ARE CHECKED ON LEGITIMACY AND CONSISTENCY; ERROR MESSAGES ARE        I
C WRITTEN TO A FILE (CF. PARAMETER "CNTRL(2)")                         I
C                                                                      I
C PARAMETERS:                                                          I
C ----------                                                           I
C NEQN   DIMENSION OF THE SYSTEM OF VOLTERRA INTEGRAL EQUATIONS        I
C G      SUBROUTINE G(T,GV); REAL T, GV(NEQN)                          I
C        EVALUATES THE FORCING TERM "G" OF THE VIE2 IN "T"             I
C        SHOULD BE DECLARED IN AN EXTERNAL STATEMENT IN THE CALLING    I
C        PROGRAM                                                       I
C KC     SUBROUTINE KC(T,S,Y,KV); REAL T,S, Y(NEQN),KV(NEQN)           I
C        EVALUATES THE KERNEL "K" OF THE VIE2.                         I
C        SHOULD BE DECLARED EXTERNAL IN THE CALLING PROGRAM            I
C DKCDY  SUBROUTINE DKCDY(T,S,Y,DKV); REAL T,S, Y(NEQN),DKV(NEQN,NEQN) I
C        EVALUATES THE JACOBIAN OF THE KERNEL W.R.T ITS THIRD ARGUMENT I
C        SHOULD BE DECLARED EXTERNAL IN THE CALLING PROGRAM            I
C        ONLY NEEDED IF NEWTON'S METHOD IS USED TO SOLVE THE SYSTEM OF I
C        COLLOCATION EQUATIONS; IF FUNCTIONAL ITERATION IS USED A      I
C        DUMMY ROUTINE SUFFICES.                                       I
C LINEAR LOGICAL VALUE.                                                I
C        "TRUE" INDICATES THAT THE KERNEL IS A LINEAR FUNCTION W.R.T   I
C        ITS THIRD ARGUMENT. NOT USED IF FUNCTIONAL ITER. IS EMPLOYED  I
C TO     REAL VALUE                                                    I
C        LEFT ENDPOINT OF THE INTEGRATION INTERVAL                     I
C TE     REAL VALUE                                                    I
C        RIGHT ENDPOINT OF THE INTEGRATION INTERVAL.                   I
C REQTOL REAL VALUE                                                    I
C        REQUESTED TOLERANCE FOR GLOBAL ERROR (NOT USED IN CASE OF     I
C        CONSTANT STEPSIZES)                                           I
C DEFOPT INTEGER VALUE.                                                I
C         0: USE NO DEFAULT SOLVERS                                    I
C        ?1: GAUSS 8  +  ITERATED GAUSS COLLOCATION;                   I
C            ESCAPE IN CASE SOLUTION IS A POLYNOMIAL OF DEGREE < 8  TO:I
C            GAUSS 8  +  [GAUSS 9 + (C10=1)]  WITH                     I
C            LOCAL + UNIFORM ERROR CONTROL                             I
```

```
C        ?2: LOBATTO 6  +  LOBATTO 7  WITH                            I
C            GLOBAL AND UNIFORM ERROR CONTROL                         I
C        WHERE "?" INDICATES THE METHOD OF CORRECTOR ITERATION (SEE   I
C        IOPT(5)).                                                    I
C        IF DEFOPT > 0, THE DEFAULTS FOR "IOPT", "OPT" AND "CNTRL"    I
C        VALUES ARE USED (IN SO FAR NOT SPECIFIED ABOVE)              I
C        IN THIS CASE THESE VECTORS DON'T HAVE TO BE INITIALIZED.     I
C IOPT   INTEGER VALUED OPTION VECTOR; (0): DEFAULT VALUES            I
C        1. KIND AND NUMBER OF COLLOCATION POINTS                     I
C           0: 8 POINT GAUSS COLLOCATION (ORDER = 8)                  I
C           OTHER: VALUE WITH DECIMAL EXPANSION "MC"                  I
C              WHERE "C" SPECIFIES THE KIND AND "M" THE # OF          I
C                  COLLOCATION POINTS.                                I
C              C    COLL. POINTS              M      GLOBAL ORDER     I
C              1        GAUSS               2<=M          M           I
C              2  (M-1) GAUSS + (CM=1)      3<=M        2M-2          I
C              3       LOBATTO              2<=M        2M-2          I
C              4        RADAU               2<=M        2M-1          I
C        2. STEPSIZE CHOICE                                           I
C           0: VARIABLE STEPSIZE                                      I
C           1: FIXED STEPSIZE                                         I
C        3. GLOBAL ERROR IN ENDPOINT REQUIRED?                        I
C           0: GLOBAL ERROR ESTIMATION IN "TE"                        I
C           1: NO GLOBAL ERROR ESTIMATION                             I
C        4. DEFINES ERROR WEIGHTS                                     I
C           0: MIXED ERROR (1 / MAX(1.0,!SOL!))                       I
C           1: ABSOLUTE ERROR (1.0)                                   I
C           2: RELATIVE ERROR (1 / !SOL!)                             I
C        5. INDICATES METHOD OF CORRECTOR ITERATION IN THE PROCESS OF I
C           SOLVING THE SYSTEM OF COLLOCATION EQUATIONS.              I
C           0: NEWTON'S METHOD; UPDATE JACOBIAN EACH NEWTON ITERATION I
C           1: NEWTON'S METHOD; EVALUATE JACOBIAN ONCE PER STEP       I
C           2: FUNCTIONAL ITERATION (NO "DKCDY" NEEDED!)              I
C        6. MAXIMUM # KERNEL EVALUATIONS ALLOWED.                     I
C           0: NO MAXIMUM                                             I
C        7. MAXIMUM # CPU-SECONDS ALLOWED.                            I
C           0: NO MAXIMUM                                             I
C        8. KIND AND NUMBER OF COLLOCATION POINTS FOR REFERENCE       I
C           SOLUTION. (NEEDS ONLY TO BE SPECIFIED IF IOPT(2)=0 OR     I
C           IOPT(3)=0)                                                I
C           0: 8 POINT ITERATED GAUSS COLLOCATION (ORDER = 16)        I
C           OTHER: VALUE WITH DECIMAL EXPANSION "MC" WHERE "C" AND "M" I
C              CAN HAVE THE VALUES AS SPECIFIED UNDER IOPT(1) WITH THE I
C              EXCEPTION THAT THE VALUE "M1", WITH 2<=M, INDICATES    I
C              M POINT ITERATED GAUSS COLLOCATION (ORDER = 2M).       I
C           NOTE: ITERATED COLLOCATION IS ONLY ALLOWED IN COMBINATION I
C           ----  WITH GAUSS COLLOCATION.                             I
C        9. STEPSIZE STRATEGY CONTROLLER (NEEDED ONLY IF IOPT(2)=0)   I
C           IF REF.SOL. IS COMPUTED BY ITERATED COLLOC.: A VALUE      I
C              WITH DECIMAL EXPANSION "PT" WHERE "P" CAN HAVE THE     I
C              VALUE 0,1 OR 2 AND "T" 0 OR 1.                         I
```

```
C              P=0: CHECK IF SOLUTION IS A POLYNOMIAL OF DEGREE < M,   I
C                   AND IF SO, ESCAPE TO LOCAL + UNIFORM ERROR CONTROL I
C                   WITH COMPUTATION OF THE REFERENCE SOLUTION BY      I
C                      (M+1) GAUSS + (C[M+2]=1)                        I
C                   IF IOPT(3)=0, THE GLOBAL ERROR IN "TE" WILL BE     I
C                   ESTIMATED BY THE SUM OF THE LOCAL ERRORS IN "TE"   I
C              P=1: CHECK IF SOLUTION IS A POLYNOMIAL OF DEGREE < M,   I
C                   IF SO, RETURN TO CALLING PROGRAM                   I
C              P=2: NO CHECK                                           I
C              T=0: INCREASE TOLERANCE BY A FACTOR "TOLREL" (SEE BELOW I
C                   UNDER "CONSTANTS USED") IF THE ERROR RESULTING     I
C                   FROM A STEP WITH VALUE "HMIN" IS GREATER THAN THE  I
C                   TOLERANCE                                          I
C              T=1: RETURN TO CALLING PROGRAM IF TOLERANCE CAN NOT BE  I
C                   SATISFIED                                          I
C           OTHERWISE: A VALUE WITH DECIMAL EXPANSION "PGUT", WHERE    I
C              "P" AND "T" ARE DESCRIBED ABOVE; "P" IS USED ONLY IF    I
C              THE APPROX. METHOD IS GAUSS COLLOCATION ,IF G=0 AND IF  I
C              ORDER REF.SOL. <= ORDER GAUSS QUADR. FORMULA.           I
C              IN CASE P=0 AND IF THE SOLUTION BEHAVES AS A POLYNOMIAL I
C              AN ESCAPE IS MADE TO  LOCAL+UNIFORM ERROR CONTROL WITH  I
C              COMPUTATION OF THE REF.SOL. BY THE SAME METHOD AS       I
C              BEFORE BUT WITH AN ADEQUATE # OF COLLOC. POINTS. IF     I
C              IOPT(3)=0, THE GLOBAL ERROR IN "TE" WILL BE ESTIMATED   I
C              BY THE SUM OF THE LOCAL ERRORS IN "TE"                  I
C              EACH OF THE DIGITS "G" AND "U" HAS THE VALUE 0 OR 1     I
C              G=0: STEPSIZE CONTROL USING GLOBAL ERROR                I
C              G=1: STEPSIZE CONTROL USING LOCAL ERROR                 I
C              U=0: MODIFIED STEPSIZE CONTROL; UNIFORM ERROR CONTROL   I
C                   OVER REMAINING INTERVAL                            I
C              U=1: NO MODIFICATION OF STEPSIZE CONTROL                I
C              NOTE: TO USE LOCAL OR UNIFORM ERROR CONTROL IN          I
C              ----  COMBINATION WITH AN M-POINT GAUSS QUADRATURE      I
C              FORMULA REQUIRES AN ORDER OF THE REFERENCE SOLUTION OF  I
C              AT LEAST 2M+1.                                          I
C NOTE:  BOTH THE VALUES IOPT(6) AND IOPT(7) CAN BE EXCEEDED BY THE    I
C ----   NUMBER OF KERNEL EVALUATIONS, RESP. CPU-SECONDS NEEDED TO     I
C        SOLVE THE SYSTEM OF COLLOCATION EQUATIONS IN AN INTERVAL.     I
C OPT    REAL VALUED OPTION VECTOR; (0.0): DEFAULT VALUES              I
C        1. INITIAL VALUE FOR STEPSIZE. IF IOPT(2)=1 THIS LENGTH       I
C           WILL REMAIN FIXED THROUGHOUT THE COMPUTATION AND SHOULD    I
C           NOT BE ZERO                                                I
C           0.0: DEFAULT VALUE. IF FIRST CALL: HINIT = MIN(TE-TO,1.0), I
C                OTHERWISE  HINIT IS SET TO THE GUESS OF THE LENGTH OF I
C                THE NEXT SUBINTERVAL MADE IN THE PREV. CALL OF COLVI2 I
C        2. MINIMUM STEPSIZE (NEEDED ONLY IF IOPT(2)=0)                I
C           0.0: DEFAULT VALUE  HMIN = MAX(SUNFLO,HINIT*HMINFC)        I
C                WITH HMINFC = 1E-5 (SEE BELOW UNDER "CONSTANTS USED") I
C           MIN. LENGTH OF N-TH SUBINTERVAL IS:                        I
C              HMINN = MAX(HMIN,SRELPR*!TN!)                           I
C                 WITH SUNFLO = SMALLEST F.P. NUMBER AND               I
```

```
C                     SRELPR = F.P. MACHINE PRECISION,                 I
C                     (SEE UNDER "MACHINE CONSTANTS")                  I
C        3. MAXIMUM STEPSIZE (NEEDED ONLY IF IOPT(2)=0)                I
C           0.0: DEFAULT VALUE  HMAX = 1.0                             I
C           IF "COLVI2" IS REQUIRED TO CHECK IF THE SOLUTION BEHAVES   I
C           AS A POLYNOMIAL  HMAX SHOULD BE <= 1.0.                    I
C        4. INTERVAL LENGTH HC (NEEDED ONLY IF UNIFORM ERROR CONTROL   I
C           IS REQUIRED); AT STEP "TN"  LOCAL ERR. CONTR. IS PERFORMED I
C           IN TE, (-HC), TN+HN.                                       I
C           0.0: DEFAULT VALUE  HC = HMAX                              I
C CNTRL  INTEGER VALUED CONTROL VECTOR; (0): DEFAULT VALUES            I
C        1. RE-ENTRY INDICATOR (CF. "GENERAL COMMENTS" SUB "ENTRY")    I
C           0: FIRST ENTRY                                             I
C           1: RE-ENTRY, NEW OPTIONS                                   I
C           2: RE-ENTRY AFTER SAVE, NEW OPTIONS                        I
C           3: RE-ENTRY, OLD OPTIONS                                   I
C           4: RE-ENTRY AFTER SAVE, OLD OPTIONS                        I
C        2. LOGICAL UNIT NUMBER OF FILE FOR ERROR MESSAGES             I
C           (CF. "GENERAL COMMENTS" SUB "ERROR MESSAGES")              I
C           0: ERROR MESSAGES ARE WRITTEN TO THE STANDARD OUTPUT FILE  I
C              (ADDRESSED BY THE "PRINT" STATEMENT)                    I
C           1 <= CNTRL(2) <= IMXLUN (SEE "MACHINE CONSTANTS")          I
C        3. CONTROL ON WRITING OF RESULTS IN ALL STEP POINTS           I
C           (CF. "GENERAL COMMENTS" SUB "WRITE ALL")                   I
C           0: NO INTERMEDIATE WRITING                                 I
C           1 <= CNTRL(2) <= IMXLUN : LOG. UNIT # OF FILE              I
C        4. INDICATOR TO SAVE VARIABLES FOR RE-ENTRY AFTER ERROR       I
C           (CF. "GENERAL COMMENTS" SUB "SAVE")                        I
C           0: NO SAVE                                                 I
C           1 <= CNTRL(2) <= IMXLUN : LOG. UNIT # OF FILE TO           I
C              SAVE COMMON BLOCKS, WKAREA, IOPT AND OPT FOR RE-ENTRY   I
C WKAREA REAL WKAREA(IW)                                               I
C        TEMPORARY WORKING STORAGE. ITS DIMENSION IS SPECIFIED BELOW;  I
C        HERE, A FEW SPECIFIC CASES ARE TREATED.                       I
C        NOTE THAT IT IS POSSIBLE TO JUST TAKE SOME VALUE FOR "IW",    I
C        SAY 2000, SET "CNTRL(4) > 0" AND LET THE PROGRAM RUN. IF THE  I
C        CODE LACKS WORKING SPACE IT DUMPS ALL NEEDED VARIABLES ON     I
C        FILE AND AN ERROR EXIT IS TAKEN. THE COMPUTATION CAN THEN BE  I
C        RESTARTED FROM THE LAST REACHED POINT BY A NEW CALLING        I
C        PROGRAM WHICH OFFERS MORE WORKING SPACE TO "COLVI2".          I
C        NOW THE PROMISED EXAMPLES:                                    I
C        LET M = # COLLOCATION PARAMETERS (SEE IOPT(1)),               I
C            S = # QUADRATURE POINTS (IF IOPT(1)=?2: M-1, OTHERWISE M) I
C            L = 2 IF IOPT(1)=?3, OTHERWISE 1                          I
C        AND DEFINE ANALOGOUSLY MR, SR, LR FROM IOPT(8).               I
C        LET     X = MAX(M,MR)                                         I
C              NHC = INT((TE-TO)/HC)                                   I
C          NIRVEC = 2.NEQN IF FUNCTIONAL ITERATION IS USED,            I
C                   OTHERWISE: NEQN.(1+NEQN)                           I
C       NWKSYS(N) = DIM.WKAREA SOLVER_FOR_LIN.SYS.OF_DIM.(N.NEQN)      I
C            F(N) = IF FUNCT.IT.: MAX(NIRVEC,2.N-N.NEQN),              I
```

```
C                     IF NEWTON'S METHOD IS USED AND THE JACOBIAN IS       I
C                             UPDATED EACH ITERATION:                      I
C                             (N.NEQN)**2 + MAX(NIRVEC,NWKSYS(N))          I
C                     IF NEWTON'S METH. IS USED WITHOUT UPDATING THE JAC.I
C                             (N.NEQN)**2 + NIRVEC+NWKSYS(N),              I
C               MAXNC = MAXIMUM # (SUCCESSFUL) STEPS POSSIBLE. THIS NUMBER I
C                     WILL BE CALCULATED BY "COLVI2" FROM THE USER         I
C                     SUPPLIED INPUT VALUES.                               I
C             THEN THE DIMENSION OF WKAREA SHOULD BE                       I
C             IF DEFOPT = ?1:                                              I
C                625+30.NEQN + (1+8.NEQN).MAXNC +                          I
C                (NO ESCAPE:) F(8)                                         I
C                (ESCAPE:)    831+(20+2.NHC).NEQN+F(10)                    I
C             IF DEFOPT = ?2:                                              I
C                429+(16+2.NHC).NEQN+F(6)+ (1+11.NEQN).MAXNC               I
C             IN CASE OF CONSTANT STEPSIZES, S=M,L=1, NO ERROR EST. IN "TE":I
C                1+3.M+M.M.M+(1+2.M).NEQN+F(M) + (1+M.NEQN).MAXNC          I
C             IN CASE OF ITERATED COLLOCATION WITHOUT ESCAPE:              I
C                1+6.M+M.M+M.M.M+(6+3.M).NEQN+F(M) + (1+M.NEQN).MAXNC      I
C             IN CASE OF LOC.+UNIF.ERR.CONTR.(NO GAUSS, NO ERR.EST.IN "TE"):I
C                1+4.X+2.X.X.X+(4+5.X+2.NHC).NEQN+F(X) +                   I
C                                             (1+(M-L+1).NEQN).MAXNC       I
C             IN CASE OF GLOBAL + UNIFORM ERROR CONTROL (NO GAUSS):        I
C                1+4.X+2.X.X.X+(4+2.X+2.NHC).NEQN+F(X) +                   I
C                                      (1+(M-L+1+MR-LR+1).NEQN).MAXNC      I
C IW          INTEGER VALUE                                                I
C             DIMENSION OF W AS DECLARED IN MAIN PROGRAM                   I
C TNC         REAL VALUE                                                   I
C             EXIT: RIGHT ENDPOINT OF LAST INTERVAL ON WHICH SOLUTION HAS  I
C                   BEEN COMPUTED (SHOULD BE EQUAL TO "TE").               I
C UE          REAL UE(NEQN)                                                I
C             EXIT: COMPUTED SOLUTION OF THE VIE2 AT "TE"                  I
C GEE         REAL GEE(NEQN)                                               I
C             EXIT: CONTAINS THE GLOBAL ERROR EST. AT "TE" IF EITHER       I
C                   IOPT(3)=0 OR IF GLOBAL OR UNIFORM ERROR CONTROL        I
C                   HAS BEEN SPECIFIED                                     I
C IERROR INTEGER VALUE                                                     I
C             ERROR COMPLETION CODE                                        I
C             IF AN ERROR HAS BEEN DETECTED, INFO IS WRITTEN TO THE        I
C             ERROR_MESSAGE_FILE (LOG. UNIT # : CNTRL(2))                  I
C               0: NO ERRORS                                               I
C               1: "DEFOPT" INCORRECT OR                                   I
C                  INITIAL FILE STATUS WRONG OF ERROR_MESSAGE_FILE         I
C                  INFO ON STANDARD OUTPUT FILE.                           I
C               2: INCORRECT INPUT.                                        I
C                  INFO ON ERROR_MESSAGE FILE.                             I
C               3: FAILED TO COMPUTE COLLOCATION PARAMETERS                I
C              11: FAILURE TO MEET TOLERANCE "REQTOL" WITH STEPSIZE "HMINN" I
C              12: WORKING STORAGE NEEDED EXCEEDS "IW" (VAR. STEPSIZE)     I
C              13: TOTAL # KERNEL EVALUATIONS USED > IOPT(6)               I
C              14: TOTAL # CPU-SECONDS USED > IOPT(7)                      I
```

```
C         15: POLYNOMIAL SOLUTION (GAUSS)                               I
C         16: TOLERANCE WOULD BE RELAXED TO A VALUE > 1.0               I
C         21: CORRECTOR ITER. PROCESS DID NOT CONVERGE WITHIN "MAXFIT"  I
C             (FUNCTIONAL ITERATION) OR "MAXNIT" (NEWTON'S METHOD)      I
C             ITERATIONS (CF. "CONSTANTS USED"). (FIXED STEPSIZE)       I
C         31: CORRECTOR ITER. PROCESS DID NOT CONVERGE WITHIN "MAXFIT"  I
C             (FUNCTIONAL ITERATION) OR "MAXNIT" (NEWTON'S METHOD)      I
C             ITERATIONS (CF. "CONSTANTS USED"). (WITHIN "YPOLM")       I
C        113: TOTAL # KERNEL EV. USED > IOPT(6) -+  WHILE COMP.         I
C        114: TOTAL # CPU-SEC. USED > IOPT(7) ---+  GLOB. ERR. EST. TE  I
C        OTHER: ERROR COMPLETION CODE FROM ONE OF THE IMSL ROUTINES.    I
C                                                                       I
C ------------------------------------------------------------------   I
C                                                                       I
C GENERAL COMMENTS:                                                     I
C ================                                                      I
C                                                                       I
C ERROR MESSAGES: ERROR MESSAGES GENERATED BY "COLVI2" ARE WRITTEN      I
C --------------  TO A SEQUENTIAL FORMATTED FILE. IF DEFOPT > 0 OR      I
C   CNTRL(2)=0 THE STANDARD OUTPUT FILE IS USED. OTHERWISE CNTRL(2)     I
C   DEFINES THE LOGICAL UNIT NUMBER OF THE FILE AND THIS FILE SHOULD    I
C   BE OPENED IN THE MAIN PROGRAM.                                      I
C                                                                       I
C THE NEXT THREE ALINEAS ARE IRRELEVANT WHEN "COLVI2" WILL BE CALLED    I
C WITH DEFOPT > 0.                                                      I
C                                                                       I
C SAVE: THE PACKAGE HAS THE OPTION (CNTRL(4)>0) TO SAVE ALL NECESSARY   I
C ----  VARIABLES ON A FILE IN CASE OF AN ERROR DURING THE              I
C   COMPUTATIONAL PROCESS, I.E. AFTER THE CONTROL AND INITIALIZATION     I
C   PHASE.                                                              I
C   THE COMMON BLOCK VARIABLES, WORKING STORAGE AND OPTION VECTORS      I
C   ARE WRITTEN TO A SEQUENTIAL UNFORMATTED FILE NAMED "COLSAV". THE    I
C   LOGICAL UNIT NUMBER USED IS "CNTRL(4)". IF THE FILE "COLSAV"        I
C   ALREADY EXISTS IT WILL BE OVERWRITTEN.                              I
C   THIS FILE SHOULD BE AVAILABLE IF "COLVI2" IS CALLED WITH            I
C   CNTRL(1)=2 OR 4.                                                    I
C   (SEE ALSO UNDER "CONSTANTS USED" AND "OTHER MACHINE DEPENDENCIES")  I
C                                                                       I
C ENTRY: "COLVI2" ACKNOWLEDGES A NUMBER OF DIFFERENT ENTRY OPTIONS.     I
C -----  THE FIRST TIME "COLVI2" IS CALLED TO SOLVE A SPECIFIC VIE2     I
C   CNTRL(1) SHOULD BE 0 AND ALL OPTION VECTORS SHOULD HAVE LEGITIMATE  I
C   VALUES.                                                             I
C     IT IS POSSIBLE TO CALL "COLVI2" A SECOND TIME IN THE SAME MAIN    I
C   PROGRAM TO CONTINUE THE PROCESS OF SOLVING THE VIE2 AFTER A NORMAL  I
C   EXIT OR AFTER "COLVI2" RETURNED WITH "IERROR" >= 10. IN THE LATTER  I
C   CASE THE USER SHOULD REACT APPROPRIATELY ON THE GIVEN ERROR.        I
C   THE ARRAY "WKAREA" SHOULD BE UNCHANGED OR COPIED INTO A NEW         I
C   WORKING STORAGE; "TNC" SHOULD BE UNALTERED.                         I
C   CNTRL(1)=1 INDICATES THAT NEW OPTION VECTORS HAVE BEEN DEFINED,     I
C   CNTRL(1)=3 THAT THE OLD OPTIONS SHOULD BE USED.                     I
C     A RE-ENTRY TO CONTINUE IN THE SAME OR A NEW JOB IS ALSO POSSIBLE  I
```

```
C   AFTER AN ERRATIC EXIT (WITH SAVE CONTROL ON), BOTH WITH NEW OPTION I
C   VECTORS (CNTRL(1)=2) AND WITH THE OLD OPTIONS (CNTRL(1)=4).        I
C   THE FILE "COLSAV" (SEE ABOVE) SHOULD BE AVAILABLE, AND "IW" SHOULD I
C   BE >= DIMENSION ARRAY "WKAREA" IN THE PREVIOUS CALL.               I
C   NOTE: IN CASE OF RE-ENTRY THE NUMBER OF COLLOCATION PARAMETERS AND I
C   ----  THE COLLOCATION METHOD TO APPROXIMATE THE SOLUTION (IOPT(1)) I
C   SHOULD BE THE SAME AS IN THE PREVIOUS CALL OF "COLVI2". IF GLOBAL  I
C   ERROR CONTROL IS USED THIS HOLDS ALSO FOR IOPT(8).                 I
C   IT IS NOT ALLOWED TO ASK FOR GLOBAL ERROR CONTROL IF THIS WAS NOT  I
C   USED IN THE PREVIOUS CALL OF COLVI2 .                              I
C   IT IS NOT POSSIBLE TO RE-ENTRY THE PROCESS OF COMPUTING THE GLOBAL I
C   ERROR ESTIMATE IN "TE" (HOWEVER, ONE CAN COPY THE CALL OF "SGEVI2" I
C   FROM "COLVI2").                                                    I
C   IF UNIFORM ERROR CONTROL IS REQUIRED AND IN THE PREVIOUS CALL THIS I
C   EITHER WAS NOT THE CASE, OR THE ENDPOINT "TE" OR THE VALUE OF      I
C   OPT(4) WERE DIFFERENT, THEN AN ESTIMATION OF THE SUM OF THE LOCAL  I
C   ERRORS IS MADE BASED ON THE LOCAL ERRORS IN THE FIRST POINT        I
C   COMPUTED.                                                          I
C                                                                      I
C WRITE ALL: IF CNTRL(3) IS NON-ZERO THE RESULTS IN EACH STEP POINT    I
C ---------  ARE WRITTEN TO A SEQUENTIAL FORMATTED FILE WITH LOGICAL   I
C   UNIT NUMBER CNTRL(3). THIS FILE SHOULD BE OPENED IN THE CALLING    I
C   PROGRAM.                                                           I
C   A SUBROUTINE YEXACT(T,YV) SHOULD EXIST AND DELIVER THE EXACT       I
C   SOLUTIONS AT "T" IN YV(1:NEQN). IF NO SOLUTION IS AVAILABLE        I
C   "YEXACT" STILL HAS TO BE PROVIDED AND SHOULD RETURN LEGITIMATE F.P.I
C   NUMBERS, SAY 0.0, IN YV.                                           I
C                                                                      I
C -------------------------------------------------------------------- I
C                                                                      I
C CONSTANTS USED: (OTHER THAN MACHINE CONSTANTS)                       I
C --------------                                                       I
C GSSFAC = 2.0,  FACTOR USED TO RELAX THE ORDER DEMAND IN THE CHECK ON I
C                POLYNOMIAL BEHAVIOR. LET UN2 BE THE SOLUTION COMPUTED I
C                OVER HALF THE N-TH INTERVAL THEN                      I
C                GEE(UN2)/GEE(UN) <= GSSFAC/(2**M) IMPLIES THAT THE    I
C                SOLUTION IS NOT A POLYNOMIAL OF DEGREE < M            I
C                ("YPOLM")                                             I
C HFAC   = 0.9,  REDUCTION FACTOR TO GET CONSERVATIVE GUESS OF THE     I
C                STEPSIZE ("SOLVI2")                                   I
C HFLFAC = 0.25, PENALTY REDUCTION FACTOR OF STEPSIZE                  I
C                ("SOLVI2")                                            I
C HMINFC = 1E-5, LIMIT FACTOR STEPSIZE: H(N)>=HINIT.HMINFC             I
C                ("CHKINI")                                            I
C HRLFAC = 2.0,  FACTOR TO LIMIT HNEW: 1/HRLFAC <= HNEW/HOLD <= HRLFAC I
C                ("SOLVI2")                                            I
C LSBITS = 128,  7 BITS FOR COMPUTATIONAL LOSS; USED IN TOLERANCE      I
C                MATTERS ("CHKINI","ESCRGS")                           I
C MAXFIT = 15,   MAX. NUMBER OF FUNCTIONAL ITERATIONS                  I
C                ("SOLSYS")                                            I
C MAXNIT = 10,   MAX. NUMBER OF NEWTON ITERATIONS                      I
C                ("SOLSYS")                                            I
```

```
C NPGESC = 2,     NUMBER OF CONSECUTIVE TIMES IT IS ALLOWED TO FIND      I
C                 POLYNOMIAL SOLUTION; IN CASE OF AUTOMATIC ESCAPE THE   I
C                 LAST NPGESC+1 STEPS ARE DISCARDED ("ESCRGS", "YPOLM")  I
C SAVFIL = 'COLSAV', FILE NAME ASSOCIATED WITH LOGICAL UNIT # CNTRL(4) I
C                 ("SAVALL", "RELOAD")                                   I
C TOLFRS = 0.1,   FACTOR BY WHICH THE TOLERANCE FOR THE CORRECTOR ITER.  I
C                 PROCESS TO SOLVE THE COLLOC.EQ. FOR THE APPROX. IS     I
C                 MULTIPLIED TO GET THE TOLERANCE FOR THE CORR.IT.PROC.  I
C                 TO COMPUTE THE REF.SOL. ("CHKINI", "SOLVI2")           I
C TOLMIN = LSBITS*SRELPR, MINIMUM TOLERANCE POSSIBLE (FOR "SRELPR" SEE   I
C                 BELOW) ("CHKINI","ESCRGS")                             I
C TOLMAX = 1.0,   MAXIMUM VALUE TO WHICH TOLERANCE MAY BE RELAXED        I
C                 ("SOLVI2")                                             I
C TOLREL = 4.0,   FACTOR TO RELAX THE TOLERANCE IN CASE IT CANNOT BE     I
C                 SATISFIED WITH MIN. STEPSIZE ("SOLVI2")                I
C                                                                        I
C -----------------------------------------------------------------      I
C                                                                        I
C !!!!!!!! IMPORTANT !!!!!!!!! IMPORTANT !!!!!!!!! IMPORTANT !!!!!!!! !!
C          =========           =========           =========          !!
C MACHINE CONSTANTS:                                                  !!
C ------------------                                                  !!
C TO REDEFINE THESE CONSTANTS CHANGE THE PARAMETER STATEMENTS IN THE  !!
C INDICATED MODULES.                                                  !!
C                                                                     !!
C IMXLUN LARGEST LOGICAL UNIT NUMBER ALLOWED BY THIS COMPILER.        !!
C        DEFINED AS 999; ANSI STANDARD: IOVFLO                        !!
C        ("CHKFIL")                                                   !!
C                                                                     !!
C THE CONSTANTS BELOW HAVE BEEN DEFINED FOR A MACHINE HAVING          !!
C F.P. NUMBERS WITH A 48 BITS MANTISSA (PLUS SIGN BIT) AND A 10 BITS  !!
C EXPONENT (PLUS SIGN BIT), AND INTEGER NUMBERS WITH AN EFFECTIVE     !!
C LENGTH OF 48 BITS.                                                  !!
C IOVFLO LARGEST INTEGER VALUE "I", SUCH THAT -I AND I ARE STORED     !!
C        REPRESENTABLE INTEGERS.                                      !!
C        ("CHKINI","WRIRES")                                          !!
C SRELPR SMALLEST REAL VALUE "X" FOR WHICH   1.0-X < 1.0 < 1.0+X      !!
C        (STORED VALUES)                                              !!
C        ("CHKINI","SOLVI2","ESCRGS","CHKWKA")                        !!
C SUNFLO SMALLEST REAL VALUE "X" SUCH THAT -X AND +X ARE STORED       !!
C        REPRESENTABLE F.P. NUMBERS)                                  !!
C        ("CHKINI")                                                   !!
C                                                                     !!
C OTHER MACHINE DEPENDENCIES:                                         !!
C ---------------------------                                         !!
C                                                                     !!
C _ TO SAVE THE NEEDED VARIABLES IN CASE OF A DETECTED ERROR AND IF   !!
C   CNTRL(4)>0 AN EXTERNAL FILE WITH THE NAME "COLSAV" IS USED.       !!
C   IF THIS IS NOT A LEGITIMATE FILE NAME CHANGE THE RELEVANT         !!
C   PARAMETER STATEMENTS IN SUBROUTINES "SAVALL" AND "RELOAD".        !!
C                                                                     !!
```

```
C _ SINCE THE PACKAGE HAS BEEN DEVELOPED ON A MACHINE WITH A RATHER     !!
C   LARGE WORD LENGTH, NO USE HAS BEEN MADE OF TYPE DOUBLE PRECISION    !!
C   TO REPRESENT FLOATING POINT NUMBERS. IF DOUBLE PRECISION IS         !!
C   REQUIRED CHANGE ALL TYPE REAL DECLARATIONS AND SPECIFICATIONS TO    !!
C   TYPE DOUBLE PRECISION. NO UNDECLARED OR UNSPECIFIED VARIABLES ARE   !!
C   USED. FOR F.P. INTRINSIC FUNCTIONS THE GENERIC NAMES ARE CHOSEN.    !!
C   THE ROUTINES "DECLUF", "SOLLUF" AND "ZERPOL" SHOULD BE ADJUSTED.    !!
C                                                                       !!
C _ TO MONITOR THE CPU-TIME USED THE SUBROUTINES "COLVI2", "SOLVI2",    !!
C   "SGEVI2" AND "CHKINI" INVOKE AN                                     !!
C      INTEGER FUNCTION NCPJOB ()                                       !!
C   THAT SHOULD RETURN THE NUMBER OF CPU SECONDS USED SINCE THE START   !!
C   OF THE JOB. "NCPJOB" CALLS A REAL FUNCTION "SECOND" THAT IS NOT     !!
C   ANSI STANDARD.                                                      !!
C                                                                       !!
C _ TO DECOMPOSE THE JACOBIAN AND TO SOLVE THE SYSTEM OF LINEAR         !!
C   EQUATIONS IN THE NEWTON PROCESS THE SUBROUTINES "DECLUF" AND        !!
C   "SOLLUF" ARE CALLED BY "SOLSYS". THE HEADERS ARE:                   !!
C      SUBROUTINE DECLUF (A, N, IA, WKAREA, IERROR)                     !!
C      INTEGER N, IA, IERROR                                            !!
C      REAL A(IA,*), WKAREA(*)                                          !!
C   AND                                                                 !!
C      SUBROUTINE SOLLUF (A, N, IA, B, WKAREA, IERROR)                  !!
C      INTEGER N, IA, IERROR                                            !!
C      REAL A(IA,*), B(*), WKAREA(*)                                    !!
C   "DECLUF" INVOKES THE SUBROUTINE "LUDATF" AND "SOLLUF" INVOKES       !!
C   "LUELMF", BOTH FROM THE IMSL LIBRARY.                               !!
C   IF A NEW SYSTEM SOLVER NEEDS MORE WORK SPACE SOME STATEMENTS        !!
C   HAVE TO BE ALTERED IN ROUTINES "CHKINI" AND "ESCRGS".               !!
C                                                                       !!
C _ TO DETERMINE THE HIGHER ORDER COLLOCATION PARAMETERS THE ROUTINE    !!
C   "ZERPOL" IS CALLED:                                                 !!
C      SUBROUTINE ZERPOL (C, N, Z, S, IERROR)                           !!
C      COMPLEX Z(N)                                                     !!
C      INTEGER N, IERROR                                                !!
C      REAL C(0:N), S(N)                                                !!
C   THAT SHOULD RETURN IN "S" THE SORTED REAL ZEROS OF THE POLYNOMIAL   !!
C   C(0).Z**N + C(1).Z**(N-1) +...+ C(N-1).Z + C(N) = 0.                !!
C   "ZERPOL" CALLS THE SUBROUTINE "ZPOLR" OF THE IMSL LIBRARY.          !!
C                                                                       !!
C !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!
C                                                                        I
C -------------------------------------------------------------------- I
```

4. Use of the package

Suppose we want to solve the system of Volterra equations given in[3] :

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \frac{1}{100} \cdot \begin{bmatrix} e^{-21/20.t} \\ 1+(10-e^{-t})e^{-t/20} \end{bmatrix} + \int_0^t A.v ds$$

over the interval [0,50], where

$$A = \begin{bmatrix} e^{\frac{21}{20}(s-t)} & 0 \\ (1-e^{s-t})e^{\frac{s-t}{20}} & \frac{1}{1000}e^{\frac{s-t}{20}} \end{bmatrix}, \text{ and } \quad \mathbf{v} = \begin{bmatrix} 3y_1(1-y_1-y_2) \\ 1-y_1-y_2 \end{bmatrix}.$$

We have to provide the subroutines:

```
SUBROUTINE G(T,GV)
REAL T, GV(2)
GV(1) = EXP(-21*T/20)/100
GV(2) = (1+(10-EXP(-T))*EXP(-T/20))/100
   - - - - - - -

SUBROUTINE KC(T,S,Y,KV)
REAL T, S, Y(2), KV(2)
   - - - - - - -
KV(1) = A11*V1 + A12*V2
KV(2) = A21*V1 + A22*V2
   - - - - - - -
```

The problem dependent variables and the desired endpoint and tolerance yield the statements:

```
NEQN   = 2
LINEAR = .FALSE.
TO     = 0.0
TE     = 50.0
REQTOL = 1E-4
```

The driver program that is supplied with the package uses six different parameter choices to solve this problem. The first and second show the most simple way to use `COLVI2`; i.e., `DEFOPT`=21 or `DEFOPT`=22. The system of collocation equations will be solved by functional iteration so only a dummy `SUBROUTINE DKCDY` is required. Moreover, since `DEFOPT`$\neq$0, the `IOPT`, `OPT` and `CNTRL` arrays need not to be initialized. So the only input parameters that still need specification are `WKAREA` and `IW`. In the third and fourth pass Newton's method will be used to solve the collocation system, still with the default parameter choices (`DEFOPT`=1 or `DEFOPT`=2). Now we have to supply the complete

```
SUBROUTINE DKCDY(T,S,Y,DKV)
REAL T, S, Y(2), DKV(2,2)
   - - - - - - -
DKV(1,1) = A11*(3-6*Y(1)-3*Y(2)) - A12
DKV(1,2) = A11*(-3*Y(1))         - A12
DKV(2,1) = A21*(3-6*Y(1)-3*Y(2)) - A22
DKV(2,2) = A21*(-3*Y(1))         - A22
   - - - - - - -
```

From the results in Table 4.1 it is obvious that the default solvers are much too powerful for this problem since the approximated solutions in the endpoint are almost exact (if Newton's method is used) and in view of the fact that all subintervals were of maximum length. Note that although the number of kernel evaluations in the first two passes is larger than in pass 4 and 5 the computational time used is less, since not too much corrector iterations are needed to solve the collocation system up to the required tolerance.

Next let us try some modifications in the parameter settings. First we take all the options the same as when `DEFOPT=2` but allow as a maximum stepsize 10.0. This gives us the following array values:

```
IOPT :  (63, 0, 0, 0, 0, 0, 0, 73, 0),
 OPT :  (0.0, 0.0, 10.0, 1.0),
CNTRL:  (0, 0, 0, 0).
```

Note that we have to set `OPT(4)` to 1.0, otherwise it would be taken equal to `OPT(3)` (=10.0). These small changes give a drastic improvement of the efficiency, without affecting the reliability as can be seen from Table 4.1.

Lastly we try a very low order method, namely a 2-*point Gauss* collocation combined with *iterated collocation* to compute the reference solution. The process controlling arrays are now valued as follows:

```
IOPT :  (21, 0, 0, 0, 0, 0, 0, 21, 0),
 OPT :  (0.0, 0.0, 0.0, 0.0),
CNTRL:  (0, 0, 0, 0).
```

From the messages given by `COLVI2` we can conclude that both solutions become very smooth after a while since y_2 is reported to behave as a polynomial of degree <2 in the points 24.317 and 34.808 and y_1 in 27.791 and in the two consecutive subinterval endpoints 40.082 and 40.821. As a consequence of these last two observations, `COLVI2` assumes that the reference solution is not necessarily more accurate than the approximation and reports that from then on it will use a 3-point Gauss + $c_4 = 1$ collocation method to compute the reference solution and local and uniform error control as a basis for the stepsize strategy. The global error estimate vector in the endpoint that is returned, is therefore based on the computed local errors committed during the integration from $t = 39.326$ up to $t = 50.0$ and therefore it cannot be assumed to be very reliable.

Pass	GEE	GE	N	#F	#kev	$\hat{u}(25)$
1	-5.78E-8	-1.16E-7	50	0	126232	.0510787322
	-8.11E-9	3.51E-8				.5982258510
2	-9.64E-8	-1.07E-7	50	0	114010	.0510787235
	9.84E-8	9.88E-8				.5982259093
3	-3.47E-13	-6.76e-13	50	0	106904	.0510786952
	-4.97E-14	6.87E-12				.5982261634
4	-1.85E-11	-1.87E-11	50	0	101230	.0510786951
	5.94E-11	6.00E-11				.5982261629
5	6.27E-6	6.44E-6	11	0	10716	.0511141940
	-3.09E-5	-3.24E-5				.5981278007
6	-3.89E-7	-8.99E-6	135	7	59100	.0510835776
	4.74E-5	5.47E-6				.5981845056

TABLE 4.1. Results for 6 different parameter settings

The abbreviations in the table header mean:

GEE	global error estimate in $T = 50.0$
GE	real global error
N	number of intervals used
#F	number of failed steps
#kev	total number of kernel evaluations needed.

The results in Table 4.1 were extracted from the common blocks used in `COLVI2` by a subroutine `SUMARY` which is also incorporated. The value of the approximate solution in $t = 25.0$, as listed in the last column, was obtained by the call:

```
     T = 25.0
     CALL COMPUH (T, NEQN, T0, WKAREA, UH25)
```

after `COLVI2` had returned to the calling program. `COMPUH` computes the approximate value in any desired point of the interval using the Lagrange interpolation formula (1.8b).

For examples on more sophisticated use of `COLVI2` e.g., how to use the *save* and *re-entry* options, we refer to the driver programs forming part of the code.

REFERENCES

1. R.L. ANDERSON ET ALII (1985). *Problem-solving software system for mathematical and statistical FORTRAN programming,* IMSL user's manual.
2. J.G. BLOM and H. BRUNNER (1985). The numerical solution of nonlinear Volterra integral equations of the second kind by collocation and iterated collocation methods, *Report NM-R8522, Centre for Mathematics and Computer Science, Amsterdam (to appear in SIAM J. Sci. Statist. Comput.).*
3. H.W. HETHCOTE and D.W. TUDOR (1980). Integral equation models for endemic infectious diseases, *J.Math.Biol.*, 9, 37-47.