HEM5 USER'S GUIDE

V. Brasey, Univ. Genève, Suisse, (1994)

1. The problem to be solved

The Fortran code HEM5 is based on the half-explicit method of order 5 described in Brasey (1992). It is suited to a direct integration of non-stiff equations of motion in descriptor form reduced to the index 2 formulation:

$$\begin{aligned} q' &= T(q,t)v && (1.1a)\\ M(q,t)v' &= f(q,v,u,t) - L(q,v,u,t)\lambda && (1.1b)\\ 0 &= G(q,t)v + k(q,t) && (1.1c)\\ u' &= d(q,v,u,\lambda,t), && (1.1d) \end{aligned}$$

with the following dimensions:

$$\begin{aligned} &\text{the position} \quad q(t) \in \mathbb{R}^{n_q},\\ &\text{the velocity} \quad v(t) \in \mathbb{R}^{n_v},\\ &\text{the variable of external dynamic} \quad u(t) \in \mathbb{R}^{n_u},\\ &\text{the Lagrange multipliers} \quad \lambda(t) \in \mathbb{R}^{n_l}, \end{aligned}$$

$$T(q,t) \in \mathbb{R}^{n_q} \times \mathbb{R}^{n_v}, \quad M(q,t) \in \mathbb{R}^{n_v} \times \mathbb{R}^{n_v},$$

$$f(q,v,u,t) \in \mathbb{R}^{n_v}, \quad L(q,v,u,t) \in \mathbb{R}^{n_v} \times \mathbb{R}^{n_l},$$

$$G(q,t) \in \mathbb{R}^{n_l} \times \mathbb{R}^{n_v}, \quad k(q,t) \in \mathbb{R}^{n_l}, \quad d(q,v,u,\lambda,t) \in \mathbb{R}^{n_u},$$

and where it is assumed that

$$\begin{pmatrix} M(q,t) & L(q,v,u,t) \\ G(q,t) & 0 \end{pmatrix} \tag{1.2}$$

is invertible in a neighbourhood of the solution.

Mechanical systems of the form (1.1) are obtained from the following general index 3 formulation:

$$\begin{aligned} q' &= T(q,t)v && (1.3a)\\ M(q,t)v' &= f(q,v,u,\beta,\gamma,t) - C^T(q,t)\beta - H^T(q,t)\gamma && (1.3b)\\ 0 &= H(q,t)v + h(q,t) && (1.3c)\\ 0 &= g(q,t) && (1.3d)\\ u' &= d(q,v,u,\beta,\gamma,t), && (1.3e) \end{aligned}$$

where

$$C = \frac{\partial g}{\partial q}(q) \cdot T(q,t) \tag{1.4}$$

and f depends linearly on (β, γ), i.e.

$$f(q, v, u, \beta, \gamma, t) = f_a(q, v, u, t) + f_b(q, v, u, t)\beta + f_c(q, v, u, t)\gamma. \tag{1.5}$$

Equation (1.3c) contains the non-holonomic constraints of index 2 and (1.3d) the holonomic index 3 constraints. A direct differentiation of these latter gives

$$0 = \frac{\partial g}{\partial q}(q)T(q, t)v + \frac{\partial g}{\partial t}(q, t). \tag{1.3f}$$

If we replace (1.3d) by (1.3f) and use the notations

$$G(q, t) = \begin{pmatrix} \frac{\partial g}{\partial q} T(q, t) \\ H(q, t) \end{pmatrix}, \quad f(q, v, u, t) = f_a(q, v, u, t), \quad \lambda = (\beta, \gamma)$$

$$L(q, v, u, t) = \Big(C^T(q, t) - f_b(q, v, u, t), H^T(q, t) - f_c(q, v, u, t) \Big), \quad k(q, t) = \begin{pmatrix} \frac{\partial g}{\partial t}(q, t) \\ h(q, t) \end{pmatrix},$$

we obtain a system of the form (1.1).

Remark 1.1. If f depends on (β, γ), then $L \neq G^T$.

Remark 1.2. The prescribed initial values

$$q(t_0) = q_0, \quad v(t_0) = v_0, \quad u(t_0) = u_0$$

must be consistent, i.e. they must satisfy condition (1.1c). If our system is obtained from (1.3), the position constraint (1.3d) must be also satisfied.

2. The numerical method

The half-explicit method can be implemented in such a way that only *linear* systems have to be solved. Application of this method to system (1.1) yields to:

$$\begin{aligned} Q_i' &= T_i V_i && (2.1a) \\ M_i V_i' &= f_i - L_i \Lambda_i && (2.1b) \\ 0 &= G_i V_i + k_i && (2.1c) \\ U_i' &= d_i, && (2.1d) \end{aligned}$$

where

$$Q_i = q_0 + h \sum_{j<i} a_{ij} Q_j', \quad V_i = v_0 + h \sum_{j<i} a_{ij} V_j', \quad U_i = u_0 + h \sum_{j<i} a_{ij} U_j' \tag{2.1e, f, g}$$

and with the simplified notations

$$T_i = T(Q_i, t_0 + c_i h), \quad M_i = M(Q_i, t_0 + c_i h), \quad G_i = G(Q_i, t_0 + c_i h),$$

$$k_i = k(Q_i, t_0 + c_i h), \quad d_i = d(Q_i, V_i, U_i, \Lambda_i, t_0 + c_i h),$$

$$f_i = f(Q_i, V_i, U_i, t_0 + c_i h) \quad \text{and} \quad L_i = L(Q_i, V_i, U_i, t_0 + c_i h).$$

For $i = 1$, equations (2.1e,f,g) give

$$Q_1 = q_0, \quad V_1 = v_0 \quad \text{and} \quad U_1 = u_0;$$

then assuming the initial values to be consistent, the constraints (2.1c) are automatically satisfied at the first stage of the method.

Relation (2.1e) now defines

$$Q_2 = q_0 + ha_{21} T(Q_1, t_0) V_1.$$

We next insert $V_2 = v_0 + ha_{21} V_1'$ into (2.1c) and together with (2.1b) we obtain a linear system for V_1', Λ_1 which can be solved. Afterwards, U_2 can be obtained with (2.1d) and (2.1g).

Proceeding this way, we get in the ith stage the following linear system (for $i = 1, \ldots, 8$ and denoting $a_{9i} = b_i$)

$$\begin{pmatrix} M_i & L_i \\ G_{i+1} & 0 \end{pmatrix} \begin{pmatrix} V_i' \\ \Lambda_i \end{pmatrix} = \begin{pmatrix} f_{ai} \\ r_i \end{pmatrix} \tag{2.2}$$

where

$$r_i = -\frac{G(Q_{i+1})}{ha_{i+1,i}} \big(v_0 + h \sum_{j=1}^{i-1} a_{i+1,j} V_j' + k_{i+1} \big). \tag{2.3}$$

The numerical solution at the end of a step is thus given by

$$q_1 = Q_9 = q_0 + h \sum_{i=1}^{8} b_i Q_i', \quad v_1 = V_9 = v_0 + h \sum_{i=1}^{8} b_i V_i', \quad u_1 = U_9 = u_0 + h \sum_{i=1}^{8} b_i U_i'. \tag{2.4}$$

The above algorithm does not require the positive definiteness of M_i. It can thus be applied to formulations of constrained mechanical systems (2.1) where M_i is singular but the matrix in (2.2) is still invertible.

Automatic step size selection

In order to obtain a code which automatically adjusts the step size in function of a prescribed tolerance of the local error, we dispose of an embedded formula of order 3, $\widehat{y}_1 = Y_8$ (see the construction of HEM5 in Brasey (1992)). Hence at the end of each step, the program computes together with the numerical solution y_1 a second approximation of lower order: $\widehat{y}_1$. A first error estimation is then possible by using

$$err_1 := \|y_1 - \widehat{y}_1\|_s = \mathcal{O}(h^4) \tag{2.5}$$

where the scaled norm is defined for $y \in \mathbb{R}^n$ by

$$\|y\|_s = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i}{sc_i} \right)^2}, \quad sc_i = Atol_i + \max(|y_{0i}|, |y_{1i}|) \cdot Rtol_i. \tag{2.6}$$

$Atol_i$ and $Rtol_i$ are tolerances prescribed by the user; $Atol_i = 0$ means that we consider relative error and $Rtol_i = 0$ that we consider absolute error. However, the numerical result is usually too accurate compared to the given tolerance. This is because y_1 is a fifth order approximation whereas $\widehat{y}_1$ is only a third order approximation. As proposed by Hairer, Nørsett & Wanner (1993, p.255), we consider in addition to $\widehat{y}_1$ a second order approximation $\widetilde{y}_1$ given by

$$\widetilde{y}_1 = y_0 + h\big(\frac{5}{2}Y_7 - \frac{3}{2}Y_8\big), \tag{2.7}$$

define

$$err_2 := \|y_1 - \widetilde{y}_1\|_s = \mathcal{O}(h^3) \tag{2.8}$$

and use

$$err = err_1 \cdot \frac{err_1}{err_1 + 0.01 \cdot err_2} = \mathcal{O}(h^5) \tag{2.9}$$

for step size selection.

For mechanical systems, we consider $y = (q, v, u)$ and $n = n_q + n_v + n_u$.

At the end of the current step, say the nth step, we can compute an optimal step size for the next step by the usual formula (see e.g. Hairer, Nørsett & Wanner (1993, p.168))

$$h_{new} = safe \cdot h_n \left(\frac{1}{err}\right)^{\frac{1}{5}}, \tag{2.10}$$

where $safe$ is a safety factor. If $err \leq 1$ then the current step is accepted and the integration is pursued with $h_{n+1} = h_{new}$ and y_1 as initial value, or else the step is rejected and the computation restarts at the nth step with y_0 and $h_n = h_{new}$.

Gustafsson proposed in his thesis (1992) a modification of the above procedure, which is based on a control theoric approach, the PI-control (Proportional-Integral control). He considers the error estimations of the two previous accepted steps, say err_n and err_{n-1}, and introduces tuning parameters with the aim that the step size control become more stable, particularly in the case of mildly stiff problems, when the step size is limited by numerical stability. (See the details in Gustafsson (1992) or Hairer & Wanner (1991, p. 31)).

It is proposed to take

$$h_{new} = h_n \cdot safe \cdot \frac{err_{n-1}^{\beta}}{err_n^{\alpha}}, \tag{2.11}$$

or to ensure that the step size change remains bounded,

$$h_{new} = h_n \cdot \min\Big(fac2, \max\big(fac1,\ safe \cdot \frac{err_{n-1}^{\beta}}{err_n^{\alpha}}\big)\Big), \tag{2.12}$$

where $fac1$, $fac2$ and $safe$ are safety factors and α and β are the tuning parameters (typical values are $fac1 = 0.2$, $fac2 = 10$, $safe = 0.9$).

The choice $\alpha = \frac{1}{p}$ and $\beta = 0$, where p is the order of the method, gives the classical step size control which seems be the most efficient for non-stiff problems. Another possibility, found by Gustafsson for the classical Runge-Kutta method DOPRI5 after numerical computations, also gives good results for HEM5 in the case of mildly stiff problems:

$$\alpha \approx \frac{0.7}{p}, \qquad \beta \approx \frac{0.4}{p}.$$

Projection on the manifold of the constraints

In order to ensure the position constraint $g(q,t) = 0$ (see equation (1.3d)), the code HEM5 has the option to project the numerical solution onto the manifold. Such projections are described e.g. by Alishenas in his thesis (1992) or by Lubich (1991). We follow here the method proposed by Lubich & al (1992) which seems very natural in our context, because it uses the same factorized matrix as in (2.2).

Warning: in formulation (1.1), we consider together holonomic and non-holonomic index 2 constraints. Since the non-holonomic constraints are not integrable, there are no corresponding index 3 constraints. In order to keep the right dimensions, we complete the vector $g(q,t) \in \mathbb{R}^{n_\beta}$ (see formulation (1.3)) with 0 and consider the extended vector $g_e(q,t) = (g(q,t), 0)^T \in \mathbb{R}^{n_l}$.

The projection yields the following method:

Step 1. Denote the numerical solution (2.4) at the time t_n by $\widetilde{q}_n$, $\widetilde{v}_n$ and $\widetilde{u}_n$.

Step 2. Projection onto the position constraints $g(q,t) = 0$. The projection is done via

$$\begin{aligned} q_n &= \widetilde{q}_n + T(\widetilde{q}_n, t_n)\nu \\ M(\widetilde{q}_n, t_n)\nu + L(Q_8, V_8, U_8, t_n)\mu &= 0 \\ 0 &= g(q_n, t_n) \\ 0 &= H(\widetilde{q}_n, t_n)\nu \end{aligned} ,$$

where (q_n, ν, μ) are the unknows. This non linear system is solved by Newton iterations using the factorized matrix occuring at the last stage of the current step:

$$\begin{pmatrix} M(Q_8, t_n) & L(Q_8, V_8, U_8, t_n) \\ G(\widetilde{q}_n, t_n) & 0 \end{pmatrix} \begin{pmatrix} \Delta\nu^k \\ \mu^k \end{pmatrix} = - \begin{pmatrix} M(\widetilde{q}_n, t_n)\nu^k \\ g_e(q^k, t_n) \end{pmatrix},$$

$$\nu^{k+1} = \nu^k + \Delta\nu^k, \quad q^{k+1} = q^k + T(\widetilde{q}_n, t_n)\Delta\nu^k,$$

with starting value $q^0 = \widetilde{q}_n$, $\nu^0 = 0$, until $\|T(\widetilde{q}_n, t_n)\Delta\nu^k\|_s \leq 10^{-2}$ for the scaled norm defined above. Take $q_n = q^{k+1}$.

Step 3. Projection onto the velocity constraint. Solve the linear system for ω:

$$\begin{pmatrix} M(q_n, t_n) & L(q_n, \widetilde{v}_n, \widetilde{u}_n, t_n) \\ G(q_n, t_n) & 0 \end{pmatrix} \begin{pmatrix} \omega \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -\big(k(q_n, t_n) + G(q_n, t_n)\widetilde{v}_n\big) \end{pmatrix}$$

and take

$$v_n = \widetilde{v}_n + \omega.$$

Remark 2.1. Due to the accuracy of the method, only very few Newton iterations are necessary. Furthermore these projections can also be done before the first step to ensure consistent initial values. However, if this first projection does not converge, the initial values provided by the user are too imprecise and the integration does not start.

Remark 2.2. If the Newton method does not converge after 5 iterations, the step is rejected and the step size is modified as follows:

$$h_{new} = 0.7 \cdot h_n$$

Acceleration and Lagrange multipliers

The acceleration and the Lagrange multipliers can be obtained with the same order of accuracy as the numerical solutions q and v by differentiating the constraints (1.1c) and by solving the following linear system

$$\begin{pmatrix} M(q,t) & L(q,v,u,t) \\ G(q,t) & 0 \end{pmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} = \begin{pmatrix} f_a(q,v,u,t) \\ r \end{pmatrix}$$

where

$$r = -\Big(G_q(q,t)\big(T(q,t)v, v\big) + G_t(q,t)v + k_q(q,t)T(q,t)v + k_t(q,t) \Big) \tag{2.13}$$

(This expression can either be provided by the user, or be computed numerically as described in the Sect.4 below.)

Dense output

To obtain the current solution also at points within each integration interval $[t, t+h]$, two dense output formulae can be selected. A detailed presentation of these formulas is given in Brasey (1994, Sect.II.6).

The first proposed dense output satisfies the algebraic constraints. **Warning**: if the function $d(q,v,u,\lambda,t)$ (see formula (1.1d)) depends on λ, this dense output can not be employed. Indeed, the variable Λ is never computed but only the product $hb_s(\theta) \cdot \Lambda$, where $b_s(\widehat{\theta})$ can vanish. In this case, $d(Q,V,U,\Lambda,t)$ can not be obtained.

The second dense output formula is based on Hermite interpolation. It can be selected without restriction, however it doesn't satisfy exactly the constraints.

3. Linear algebra

HEM5 requires at each stage the resolution of a linear system of the form (2.2). To account for possible sparsity of the occuring matrix, HEM5 offers in option the use of sparse solver routines from Harwell library (Duff, 1977).

Furthermore, the parts of the code which concern the linear algebra are clearly separated in two routines ASET and ASOL. They can be easily modified by the user who would like to use other factorization/resolution routines. ASET constructs and factorizes the matrix and ASOL solves the system (2.2) with the matrix decomposed in ASET. If the sparse linear algebra is switched on, an other routine, ACOPY, constructs the matrix in the form required by the sparse routines.

Full linear algebra routines

The first choice offered by HEM5 is the general LU-decomposition/substitution routines DEC/SOL written by Moler (1972) and well suited for little dimensional systems.

Ww propose as second choice the routines LAPACK (see Anderson & al, 1992) DGETRF which factorizes a general square matrix A and DGETRS which solves the system $Ax = b$.

MA28 - general sparse system solver

The routines of MA28PACK are designed to solve linear systems of the form $Ax = b$ where A is a square unsymmetric sparse matrix. Details of implementation and use are given in Duff (1977) and in the descriptions of the routines (head comments). Three routines (written in double precision) are directly called by HEM5:

MA28AD: Computes the pivot ordering and both symbolic and numerical LU factors of A. The nz non zero entries of the matrix are stored in the vector AVALUE(1..nz) and the indices in the integer vectors AINDEXR(1..nz) and AINDEXC(1..nz) as follows:

i

f AVALUE(k) $= a_{ij}$ then AINDEXR(k) $= i$ and AINDEXC(k) $= j$.

MA28BD: Computes the numerical LU factors of a matrix using the pivot ordering and symbolic LU factors computed by MA28AD. While A keeps the same pattern, only one initial call to MA28AD is necessary.

MA28CD: Solves the system $Ax = b$ using the LU factors computed by MA28AD or MA28BD.

4. User interface of HEM5

The calling sequence of HEM5 is the following

HEM5 (NQ, NV, NU, NL, FPROB,

T, Q, V, U, A, RLAM, TEND, H,

RTOL, ATOL, ITOL,

SOLOUT, IOUT,

WK, LWK, IWK, LIWK, IDID)

INPUT Arguments

NQ, NV, NU, NL: integer, they describe the dimensions n_q, n_v, n_u and n_l of the problem.

FPROB: external subroutine, it supplies all the functions and matrices describing the problem. Details are given below.

T, TEND: real, initial and final times, T $= t_0$, TEND $= t_{end}$.

Q,V,U: real array of dimension NQ, NV and NU respectively. They contain in input the initial values $q(t_0)$, $v(t_0)$ and $u(t_0)$.

H: real, initial step size guess (typical value: 1.d-03).

RTOL, ATOL: real array or scalar depending on wether ITOL is 0 or 1, relative and absolute tolerances.

ITOL: integer, indicates whether RTOL and ATOL are scalar (ITOL = 0), or array of dimension $n_q + n_v + n_u$ (ITOL = 1).

SOLOUT: external subroutine, it provides the solution (q, v, u) at selected internal points. It is called after every successful step if IOUT > 0. More details are given below.

IOUT: integer, it selects the dense output formula. More details are given below.

WK: real array of length LWK, it contains work spaces and tuning parameters to control performances of the code. For zero input default values are chosen.

WK(1) = UROUND, precision of the machine (default value: 1.d-16).

WK(2) = SAFE, safety factor for step size prediction (default value: 0.9d0).

WK(3) , *WK(4)* = FAC1, FAC2, parameters for step size selection (see formula (2.12), default values: 0.2d0 and 10.d0).

WK(5) = HMAX, maximal step size (default value: TEND - T).

WK(6) , *WK(7)* = ALPHA, BETA, parameters of Gustafsson strategy (see formula (2.12), default values: 1/5 and 0).

LWK: integer, length of array WK, minimal value: 19 + 27*NQ + 28*NV + 27*NU + 5*(NV+NL) + 4*NZA + 2*IXS + LL, where IXS, LDG, LDF, NMRC, NBLK and MODE are described below and

LL = 8*NL*NV + 4*(NL+NV)^2 if MODE = 0, 1, 2 or 3

LL = 6*LDG + 2*LDF + 4*NMRC^2*NBLK if MODE = 4 or 5

NZA = LDG + max(LDG,LDF) + NMRC^2*NBLK.

IWK: integer array of length LIWK, it contains work space, switching parameters and arguments used by sparse linear algebra routines:

IWK(11) = NMAX, maximal number of allowed steps (default value: 100000).

IWK(12) = IPCIV, an initial projection is performed if it is set to 1 (default value: 0).

IWK(13) = NBS, a projection is done every NBS accepted steps (default value: 0).

IWK(14) = MODE, it selects the linear algebra routines according to the problem formulation:

0 : full, $L = G^T$, DEC/SOL

1 : full, $L \neq G^T$, DEC/SOL

2 : full, $L = G^T$, DGTREF/DGETRS

3 : full, $L \neq G^T$, DGTREF/DGETRS

4 : sparse, $L = G^T$, MA28PACK

5 : sparse, $L \neq G^T$, MA28PACK.

IWK(15) = IACC, computes the acceleration and the Lagrange multipliers if it is set to 1 (default value: 0).

IWK(16) = IGII, computes numerically the expression r (see formula (2.13)) if it is set to 1 (default value: 0).

The following entries must be defined if the sparse mode is switched on:

IWK(1) , *IWK(2)* = LDG, LDF, maximal number of non zero elements of matrices GQ, resp FL (see the arguments of FPROB below).

IWK(21) , *IWK(22)* = NMRC, NBLK, size of a block and number of blocks of the mass matrix AM.

IWK(23) , *IWK(24)* = NPGP, NPFL, indicate whether GQ or FL change of pattern While GQ and FL keep the same pattern, NPGP and NPFL are set to 0 and the economical routine UMDRFA is called.

IWK(25) , *IWK(26)* = IS, IXS, sizes of IMEM and XMEM, common work space arrays for MA28PACK routines. For details we refer to the comments of these codes. (Default values 1000).

IWK(27) , *IWK(28)* = PRLEV, IO. The argument PRLEV controls the printing of error messages and the unit used for diagnostics is given by IO. For PRLEV equals to:

0 no diagnostics are printed,

-2 partial diagnostics are printed,

2 complete diagnostics are printed

(Default value for IO: 6).

LIWK: integer, length of array IWK, minimal value: 95 + 2*(NV+NL) + 2*IS + 12*LDG + 4*LDF + 4*NZA.

OUTPUT Arguments

T: real, current point of integration.

Q,V,U: real array, current values of $q(t)$, $v(t)$ and $u(t)$.

A,RLAM: real array, current values of $a(t)$ and $\lambda(t)$ if IWK(15) = 1.

IWK(31..38): integer array, statistics:

IWK(31) = NSTEP, number of total computed steps

IWK(32) = NACCPT, number of accepted steps

IWK(33) = NREJCT, number of rejected steps (after at least one step has been accepted)

IWK(34) = NFCN, number of f-evaluations

IWK(35) = NGQCN, number of G-evaluations

IWK(36) = NAMAT, number of M-evaluations

IWK(37) = NDEC, number of LU-decompositions

IWK(38) = NSOL, number of substitutions.

IDID: integer error indicator:

1: successful computation

-1: invalid input arguments

-2: NMAX too small

-3: step size becomes too small

-4: matrix in ASET is singular

-5: initial projection: no convergence.

FPROB

This subroutine is provided by the user. It describes all the functions and matrices of the problem and has the following calling sequence:

FPROB (IFCN, NQ, NV, NU, NL, LDG, LDF, LDA, NBLK, NMRC,

NPGP, NPFL, INDGR, INDGC, INDFLR, INDFLC,

T, Q, V, U, XL, G, GQ, F,

GQQ, GT, FL, QDOT, UDOT, AM)

INTEGER IFCN, NQ, NV, NU, NL, LDG, LDF, LDA, NBLK, NMRC, NPGP, NPFL,

INDGR(LDG), INDGC(LDG), INDFLR(LDF), INDFLC(LDF)

DOUBLE PRECISION T, Q(NQ), V(NV), U(NU), XL(NL), G(NL), F(NV),

GQQ(NL), GT(NL), QDOT(NQ), UDOT(NU)

In the full linear algebra mode GQ, FL and AM are matrices of dimension GQ(LDG, NV), FL(LDF, NL) and AM(LDA, NV) where LDG = NL, LDF = NV and LDA = NL+NV. If the sparse mode is selected, GQ and FL must be one dimensional arrays of length LDG, resp. LDF, and contain the non zero entries of GQ and FL. The indices are then stored in INDGR, INDGC, INDFLR and INDFLC as explained in Sect.III.3. The matrix AM is of dimension AM(LDA,NMRC) with LDA = NMRC*NBLK and is block-wise stored, i.e. only the NMRC first columns are non zero:

$$\text{if} \quad a_{ij} \in k\text{th} \quad \text{block of} M(q,t) \quad (1 \le k \le nblk) \quad \text{then} \quad \text{AM(i,j)} \; = a_{i,j+(k-1)\cdot nmrc}.$$

Example 3.1. For a diagonal matrix M of dimension $n_v \times n_v$, we have NMRC = 1 and NBLK = n_v whence the entries of M are stored in first column of M.

The correspondance between the abovementionned arrays and those of equation (1.1) is the following:

$$\begin{aligned}
\mathrm{Q,\ V,\ U,\ XL} &\sim q(t), v(t), u(t), \lambda(t)\\
\mathrm{GQ} &\sim G(q,t)\\
\mathrm{FL} &\sim L(q,v,u,t) \text{ if } L \neq G^T\\
\mathrm{G} &\sim g_e(q,t)\\
\mathrm{F} &\sim f(q,v,u,t)\\
\mathrm{GT} &\sim k(q,t)\\
\mathrm{QDOT} &\sim T(q,t)v\\
\mathrm{UDOT} &\sim d(q,v,u,\lambda,t)\\
\mathrm{AM} &\sim M(q,t)\\
\mathrm{GQQ} &\sim r \text{ (see formula } \quad (2.13)).
\end{aligned}$$

Warning: if $L(q,v,u,t) = G^T(q,t)$, then the entries of FL do not need to be defined since GQ already contains the whole information If the sparse mode is switched on, the integers NPGP and NPFL indicate a change of pattern of GQ and FL. They are set to 0 if no change occurs and they are set to 1 if else.

IFCN selects the expressions which must be evaluated in the current call to FPROB. There are 12 types of calls:

$$\begin{aligned}
\text{if IFCN equals to }\ 0 &\rightarrow \mathrm{UDOT}\\
1 &\rightarrow \mathrm{F,\ AM,\ QDOT,\ FL} \text{ (if used)}\\
2 &\rightarrow \mathrm{QDOT}\\
3 &\rightarrow \mathrm{GT}\\
4 &\rightarrow \mathrm{G}\\
5 &\rightarrow \mathrm{F,\ GII} \text{ (if analytically available)}\\
6 &\rightarrow \mathrm{GQ,\ GT}\\
7 &\rightarrow \mathrm{F,\ AM,\ GII} \text{ (if analytically available)}\\
8 &\rightarrow \mathrm{F,\ AM,\ FL} \text{ (if used)}\\
9 &\rightarrow \mathrm{AM}\\
10 &\rightarrow \mathrm{GT,\ GQ,\ AM,\ QDOT,\ FL} \text{ (if used)}\\
11 &\rightarrow \mathrm{GT,\ AM,\ GQ.}
\end{aligned}$$

SOLOUT

The dense output option is selected via the input integer IOUT. If IOUT is set to 1, 2 or 3, then after every accepted step, the subroutine SOLOUT is called. It has the following calling sequence:

SOLOUT (MODE, NSTEP, NQ, NV, NU, NL, LDG, LDF, LDA, LRDO, LIDO,

```
      FPROB, Q, V, U, DOWK, IDOWK )
   INTEGER MODE, NSTEP, NQ, NV, NU, NL, LDG, LDF, LDA,
      LRDO, LIDO, IDOWK(LIDO)
   DOUBLE PRECISION Q(NQ), V(NV), U(NU), DOWK(LRDO)
   EXTERNAL FPROB
```

DOWK and IDOWK are work space arrays, H and TOLD are available in DOWK(1) and DOWK(2) respectively.

The user has to choose the output points x within the interval [TOLD, TOLD+H] together with the component IDES of the solution that he desires on output (IDES= $1, \ldots, n_q + n_v + n_u$) and calls the dense output formula selected by IOUT:

if IOUT = 1 then call the function CT4:

CT4 (MODE, IDES, NQ, NV, NU, NL, LDG, LDF, LDA,
LRDO, LIDO, IDOWK, X, DOWK, FPROB)

if IOUT = 2 then call the function POL:

POL (MODE, IDES, NQ, NV, NU, NL, LDG, LDF, LDA,
LRDO, LIDO, IDOWK, X, DOWK, FPROB)

If only the current solution at the integration points is desired, then the user can IOUT = 3 and call neither CT4 nor POL from SOLOUT.

Example routines FPROB and SOLOUT are part of the package.

5. Program's structure

There are four user-callable subroutines: HEM5 which is the integration method and for consistent input values computes the numerical solution, CTQ4, CT4 and POL which are functions computing a prescribed component of the dense output solution at selected integration points. They are called in the subroutine SOLOUT.

Routines to be supplied by the user

DRIVER: main program.

FPROB: evaluations of functions and matrices describing the problem.

SOLOUT: selection of the dense output integration points and of the desired component of the solution. Call to one of the functions CTQ4, CTQ or POL in accordance to the value of IOUT.

HEM5 method routines

HEM5: user interface routine, input check, initialisation and work space distribution.

HCOR: numerical method and control of the options.

COEFF: coefficients of the method.

APROJ: projection of the numerical solution onto the position and the velocity manifolds.

GIINUM: computation of the expression r (formula (2.12)).

CPCT: compactification of several vectors and matrices to simplify the calling sequence of SOLOUT.

CMAT4: called by CT4, dense output formula respecting the index 2 constraint.

COPOL: called by POL, coefficients of the Hermite interpolation polynomial.

Dense output functions

They are directly called by the user via the subroutine SOLOUT:

CTQ4: simplified computations for the Q-components.

CT4: dense output respecting the index 2 constraint.

POL: dense output based on Hermite interpolation.

Linear algebra routines

ASET: construction and decomposition of the matrix, call to appropriate DEC, LAPACK or MA28PACK routines.

ACOPY: put a matrix in the form required by MA28PACK interface.

ASOL: resolution of a linear system with the matrix decomposed in ASET. Call to appropriate SOL, LAPACK or MA28PACK routines.

GPMULT: computation of the expression $fac \cdot (A \cdot v + g_t)$ where fac is a real, A a matrix, v and g_t vectors.

AMULT: computation of a matrix vector product.

GPM2: computation of the expression $fac \cdot (A - B) \cdot v$ where fac is a real, A and B matrices and v a vector.

References

T. Alishenas : *Zur numerischen Behandlung, Stabilisierung durch Projektion und Modellierung mechanischer Systeme mit Nebenbedingungen und Invarianten.* Ph.D. thesis, Royal Inst. of Tech., Stockholm, Sweden, (1992).

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greebaum, S. Hammarling, A. Mc Kenney, S. Ostrouchov & D. Sorensen : *LAPACK Users' Guide.* SIAM, Philadelphia, (1992).

V. Brasey : *A half-explicit Runge-Kutta method of order 5 for solving constrained mechanical systems.* Computing, Vol. 48, pp. 191-201, (1992).

V. Brasey : *Half-explicit methods for semi-explicit differential algebraic equations of index 2.* Ph.D. thesis, Univ. de Genève, Suisse, (1994).

V. Brasey & E. Hairer : *Half-explicit Runge-Kutta methods for differential-algebraic systems of index* 2. SIAM J. Numer. Anal., Vol. 30, pp. 538-552, (1993).

I.S. Duff : *MA28 - A set of FORTRAN Subroutines for Sparse Unsymmetric Linear Equations.* AERE Report R. 8730, HMSO, London (1977).

K. Gustafsson : *Control of error and convergence in ODE solvers.* Ph.D. thesis, Univ. of Lund, Sweden, (1992).

E. Hairer, S.P. Nørsett & G. Wanner : *Solving ordinary differential equations I. Nonstiff problems.* Second Revised Edition, Computational Mathematics, Vol. 8, Springer-Verlag, Berlin, (1993).

E. Hairer & G. Wanner : *Solving ordinary differential equations II. Stiff and differential-algebraic problems.* Computational Mathematics, Vol. 14, Springer-Verlag, Berlin, (1991).

Ch. Lubich : *Extrapolation integrators for constrained multibody systems.* Impact Comput. Sci. Eng., Vol. 3, pp. 213-234, (1991).

Ch. Lubich , U. Nowak , U. Pöhle & Ch. Engstler : *MEXX - Numerical software for integration of constrained mechanical multibody systems.* Preprint SC 92-12, Konrad-Zuse-Zentrum für Informationtechnik, Berlin (1992).

C.B. Moler : Algorithm 423, Linear Equation Solver, C.A.C.M. 15, p.274, (1972).