

FRB–FORTRAN routines for the exact computation of free rigid body motions

ELENA CELLEDONI

Department of Mathematical Sciences, NTNU

ANTONELLA ZANNA

Department of Mathematics, University of Bergen

We present two algorithms and their corresponding FORTRAN routines for the exact computation of free rigid body motions. The methods use the same description of the angular momentum part \mathbf{m} by Jacobi elliptic functions, and suitably chosen frames for the attitude matrix/quaternion Q/q respectively. The frame transformation requires the computation of elliptic integrals of the third kind. Implementation and usage of the routines are described, and some examples of drivers are included. Accuracy and performance are also tested to provide reliable numerical results.

Categories and Subject Descriptors: G.4 [Mathematical Software]: Algorithm design and analysis; G.1.7 [Numerical Analysis]: Ordinary differential equations—One step (single-step) methods; G.1.10 [Applications]: Rigid bodies

General Terms: Numerical methods, rigid bodies

Additional Key Words and Phrases: Rigid body, Jacobi elliptic integrals, splitting methods, attitude rotation.

1. INTRODUCTION

Due to the recent interest in computing rigid body dynamics using Jacobi elliptic functions and integrals we present here FORTRAN subroutines for use in the computation of problems of rigid bodies subject to external forces. We consider both the equations for the angular momentum (in the body frame), and the equations for the reconstruction of the frame itself (kinematic equations).

The free rigid body plays an important role in physics and engineering [Whittaker 1937] [Leimkuhler and Reich 2004]. In fact rigid body integrators are used as building blocks in problems of celestial mechanics, molecular dynamics, etc. [Morton et al. 1974], [Touma and Wisdom 1994], [Dullweber et al. 1997].

The exact solution of the free rigid body equations can be successfully employed in connection with splitting methods allowing the use of fairly large step sizes of integration [Celledoni and Säfstöm 2006], [Celledoni et al. 2008], [van Zon and

Author’s address: E. Celledoni, Department of Mathematical Sciences, NTNU, 7491 Trondheim, Norway, (Elena.Celledoni@math.ntnu.no).

A. Zanna, Department of Mathematics, University of Bergen, Johannes Brunsgt. 12, 5008 Bergen, Norway, (Antonella.Zanna@math.uib.no).

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2008 ACM 0098-3500/2008/1200-0111 \$5.00

Schofield 2007b], [van Zon and Schofield 2007a].

A typical application is the case of a rigid body subjected to gravity or to an external torque. Exact analytic solution of such rigid body equations (including the kinematic equations) are known, but only for special choices of the inertia tensor [Romano 2008].

1.1 The problem and the exact solution

We consider the motion of a rigid body with a fixed point (free rigid body, FRB)

$$\dot{\mathbf{m}} = \mathbf{m} \times I^{-1}\mathbf{m}, \quad (1)$$

$$\dot{Q} = Q \widehat{I^{-1}\mathbf{m}}, \quad (2)$$

where $\mathbf{m} = (m_1, m_2, m_3)^T$ is the body representative of the angular momentum vector and $I = \text{diag}(I_1, I_2, I_3)$ is the inertia tensor. Here \times denotes the vector product in \mathbf{R}^3 and the hat-map $\widehat{\cdot}: \mathbf{R}^3 \rightarrow \mathfrak{so}(3)$ is defined as

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \mapsto \widehat{\mathbf{v}} = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}$$

and satisfies $\widehat{\mathbf{v}}\mathbf{u} = \mathbf{v} \times \mathbf{u}$ for all $\mathbf{u}, \mathbf{v} \in \mathbf{R}^3$.

Equation (1) is Euler equation (written for the angular momentum rather than for the angular velocity $\boldsymbol{\omega} = I^{-1}\mathbf{m}$), while (2) is the equation of the attitude rotation of the free rigid body, transforming a chosen orthogonal frame fixed in space (say the canonical frame in \mathbf{R}^3 , $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) into a chosen orthogonal frame attached to the body, both having the origin in the body fixed point. The kinetic energy of this system is

$$T = \frac{m_1^2}{2I_1} + \frac{m_2^2}{2I_2} + \frac{m_3^2}{2I_3},$$

and the equations are Hamiltonian. The three components of the (spatial angular momentum) vector $Q\mathbf{m}$ are constants of motion. In particular, the norm of the (body) angular momentum, $G = \|\mathbf{m}\|$, is a constant of motion.

Equations (1) and (2) can be explicitly integrated in terms of elliptic functions. The integration is done in two steps. First, Euler equation (1) is integrated to give $\mathbf{m}(t)$. Then, equation (2) becomes a time dependent linear equation for $Q(t)$, which can be integrated exploiting the constancy of the spatial angular momentum vector.

Note that, due to the obvious $\text{SO}(3)$ -symmetry and scaling invariance of equations (1) and (2), we may restrict ourselves to describe their solutions with initial conditions (Q_0, \mathbf{m}_0) at $t = t_0$ such that

$$Q_0 = 1, \quad \|\mathbf{m}_0\| = 1.$$

(Abusing notation, we denote the identity matrix by 1).

From now on, we assume that the three moments of inertia I_1, I_2, I_3 are pairwise distinct and we order them in ascending order, $I_1 < I_2 < I_3$.

1.2 Solution of the Euler equation

The expression of the solutions of (1) involve the three Jacobi elliptic functions sn, cn and dn, whose definition is recalled in the Appendix. As mentioned, we consider only solutions with unit norm. Given T , define the positive constants

$$I_{jh} = |I_j - I_h|, \quad \Delta_j = |1 - 2TI_j|, \quad B_{jh} = \left(\frac{I_j \Delta_h}{I_{jh}} \right)^{1/2}$$

for $j, h = 1, 2, 3, j \neq h$, and

$$k = \left(\frac{\Delta_1 I_{32}}{\Delta_3 I_{21}} \right)^{1/2}, \quad \lambda_1 = \left(\frac{\Delta_1 I_{23}}{I_1 I_2 I_3} \right)^{1/2}, \quad \lambda_3 = \left(\frac{\Delta_3 I_{12}}{I_1 I_2 I_3} \right)^{1/2}. \quad (3)$$

Let $\mathbf{m}(t)$ be a solution of Euler equation (1) with unit norm and energy T . We distinguish three cases.

(i) If $2TI_2 > 1 > 2TI_1$, then

$$\mathbf{m}(t) = \left(\sigma B_{13} \operatorname{dn}(\lambda t - \nu, k), B_{21} \operatorname{sn}(\lambda t - \nu, k), B_{31} \operatorname{cn}(\lambda t - \nu, k) \right)^T, \quad (4)$$

with $\lambda = \sigma \lambda_3$, for some $\nu \in \mathbf{R}$ and $\sigma = \pm 1$.

(ii) If $2TI_2 < 1 < 2TI_3$, then

$$\mathbf{m}(t) = \left(B_{13} \operatorname{cn}(\lambda t - \nu, k^{-1}), B_{23} \operatorname{sn}(\lambda t - \nu, k^{-1}), \sigma B_{31} \operatorname{dn}(\lambda t - \nu, k^{-1}) \right)^T, \quad (5)$$

with $\lambda = \sigma \lambda_1$, for some $\nu \in \mathbf{R}$ and $\sigma = \pm 1$.

(iii) If $2TI_2 = 1$ and $\mathbf{m}(0)$ is not an equilibrium, then

$$\mathbf{m}(t) = \left(\sigma' B_{13} \operatorname{sech}(\lambda t - \nu), \tanh(\lambda t - \nu), \sigma' B_{31} \operatorname{sech}(\lambda t - \nu) \right)^T,$$

with $\lambda = \sigma \lambda_3$, for some $\nu \in \mathbf{R}$, $\sigma = \pm 1$ and $\sigma' = \pm 1$.

One can easily verify by differentiation that these expressions satisfy the Euler equations, see e.g. [Lawden 1989]. Solutions of We included the case when $2TI_2 = 1$ for completeness, the occurrence of these solutions in numerical computations is quite rare. Note that in the first two cases the phase ν can be taken modulo the period of the Jacobi elliptic functions.

Remark Solutions with norm G are obtained from the given formulas of with the substitutions $\mathbf{m} \mapsto G\mathbf{m}$ and $T \mapsto T/G^2$.

1.3 Integration of the rotation matrix

We will follow the presentation of [Celledoni et al. 2008]. Here and in the following we denote by a dot the Euclidean scalar product in \mathbf{R}^3 (and later on also in \mathbf{R}^4). Moreover, we use the inner product

$$\langle A, B \rangle := \frac{1}{2} \operatorname{tr}(A^T B)$$

on the space of 3×3 skew-symmetric matrices. Note that $\langle \widehat{\mathbf{u}}, \widehat{\mathbf{v}} \rangle = \mathbf{u} \cdot \mathbf{v}$ for all $\mathbf{u}, \mathbf{v} \in \mathbf{R}^3$.

It is convenient to factorize the attitude matrix $Q(t)$ in the product

$$Q(t) = P(t_0)^T Y(t) P(t) \quad (6)$$

with $P(t), Y(t) \in \text{SO}(3)$ such that

$$P(t)\mathbf{m}(t) = \mathbf{e}_3 \quad \text{and} \quad Y(t)\mathbf{e}_3 = \mathbf{e}_3 \quad \forall t, \quad Y(t_0) = \mathbb{1}. \quad (7)$$

Note that any unit vector $\mathbf{v}(t)$ orthogonal to $\mathbf{m}(t)$ can be used to construct the matrix $P(t) = [\mathbf{v}(t), \mathbf{w}(t), \mathbf{m}(t)]^T$, where $\mathbf{w}(t) = \mathbf{m}(t) \times \mathbf{v}(t)$, and $P(t)$ is therefore not unique.

Assuming a $P(t)$ satisfying (7) is fixed, we can write $Y(t) = \exp(\psi(t)\hat{\mathbf{e}}_3)$ where $\psi(t)$ is the rotation angle. One can show that

$$\psi(t) = \int_{t_0}^t (2T + \mathbf{w}(s) \cdot \dot{\mathbf{v}}(s)) ds \pmod{2\pi}. \quad (8)$$

Since $\|\mathbf{m}(t)\| = 1$ implies that $\dot{\mathbf{m}}(t)$ is orthogonal to $\mathbf{m}(t)$, a possible choice is that of taking $\mathbf{v}(t)$ aligned with $\dot{\mathbf{m}}(t)$.

The expression of the angle $\psi(t)$ corresponding to this choice and with $\mathbf{m}(t)$ as in (4) with unit norm and energy T such that $2TI_2 > 1 > 2TI_1$, is

$$\psi(t) = 2T(t - t_0) + \frac{\Delta_2}{\lambda I_2} \left[\Pi(\text{am}(\lambda t - \nu), n, k) - \Pi(\text{am}(\lambda t_0 - \nu), n, k) \right] \quad (9)$$

with k, λ and ν as in (4) and $n = B_{23}^{-1}$.

The expression of ψ uses the elliptic integral of the third kind, Π , and the amplitude function am , whose definitions are recalled in the Appendix.

Remark If $2TI_3 > 1 > 2TI_2$ then ψ is as in (9) with k replaced by k^{-1} , with λ and ν as in case (ii) of subsection 1.2, and with $n = B_{21}^{-1}$.

This algorithm equals that of [Lawden 1989], except for the sign of ψ . A similar algorithm is given in [Cushman 2000].

For another choice of P we refer to [van Zon and Schofield 2007a] where $P(t) = [\mathbf{v}(t), \mathbf{w}(t), \mathbf{m}(t)]^T$ is such that $\mathbf{v} = \frac{\mathbf{m} \times \mathbf{e}_3}{\|\mathbf{m} \times \mathbf{e}_3\|}$ and $\mathbf{w} = \frac{\mathbf{m} \times \mathbf{v}}{\|\mathbf{m} \times \mathbf{v}\|}$.

1.4 Formulation in quaternions

Sometimes it might be convenient to represent the equations (1) and (2) in quaternions. The quaternions (of unit norm) are the elements of the unit sphere of \mathbf{R}^4 ,

$$S^3 = \{q = (q_0, \mathbf{q}) \in \mathbf{R} \times \mathbf{R}^3 \mid q_0^2 + \|\mathbf{q}\|^2 = 1\},$$

equipped with a Lie-group structure with product

$$(p_0, \mathbf{p})(q_0, \mathbf{q}) := (p_0 q_0 - \mathbf{p} \cdot \mathbf{q}, p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q}),$$

identity $e = (1, \mathbf{0})$ and the inverse of $(q_0, \mathbf{q}) \in S^3$ is $q^{-1} = (q_0, -\mathbf{q})$. The quaternion product extends to \mathbf{R}^4 .

The ‘Euler–Rodriguez’ map $\mathcal{E} : S^3 \rightarrow \text{SO}(3)$ defined by

$$\mathcal{E}(q) = \mathbb{1} + 2q_0 \hat{\mathbf{q}} + 2\hat{\mathbf{q}}^2 \quad (10)$$

transforms any quaternion of unit norm into a rotation on \mathbf{R}^3 and is onto. Note that $\mathcal{E}(q) = \mathcal{E}(-q)$.

The Lie algebra of S^3 is

$$\mathfrak{s}^3 = T_e S^3 = \{u = (0, \mathbf{u}) : \mathbf{u} \in \mathbf{R}^3\}.$$

If \mathbf{m} is a solution of the Euler equations (1) and $q \in S^3$ then $Q = \mathcal{E}(q)$ is a solution of of Arnold equation (2) if and only if

$$\dot{q} = \frac{1}{2}q\omega \quad (11)$$

with $\omega = (0, I^{-1}\mathbf{m})$.

The analogous of the Euler equations (1) in S^3 is $\dot{m} = \frac{1}{2}(m\omega - \omega m)$.

As in section 1.3 we consider a factorization of the solution q of (11) as follows

$$q(t) = p(t_0)^{-1} y(t) p(t) \quad (12)$$

and the conditions, in parallel to (7), are

$$p(t)m(t)p(t)^{-1} = e_3 \quad \text{and} \quad y(t)e_3y(t)^{-1} = e_3 \quad \forall t, \quad y(t_0) = e. \quad (13)$$

Let \mathbf{m} be a solution of the Euler equations with unit norm. Then, a $q(t)$ of the form (12) satisfies (11) and $q(t_0) = e$ if and only if $y(t) = (\cos \frac{\psi(t)}{2}, \mathbf{e}_3 \sin \frac{\psi(t)}{2})$ with

$$\psi(t) = \int_{t_0}^t (2T + 2e_3 \cdot p(s)\dot{p}(s)^{-1}) ds \quad (\text{mod } 2\pi). \quad (14)$$

Remark It is possible to show that a quaternion p satisfying (13) is determined, up to the sign, once the square of the 4-th component, p_3^2 , and the relative signs of p_0 and p_3 are known, [Celledoni et al. 2008]. Assuming \mathbf{m} and p_3^2 known, one has the following dependence among the components of p ,

$$p_0^2 = \frac{1 + m_3}{2} - p_3^2, \quad p_1 = \frac{p_3 m_1 + p_0 m_2}{1 + m_3}, \quad p_2 = \frac{p_3 m_2 - p_0 m_1}{1 + m_3}. \quad (15)$$

Assume \mathbf{m} is a solution of the Euler equations with unit norm and $2TI_2 > 1 > 2TI_1$ then for different choices of p satisfying (13) we obtain different angles ψ . In what follows we distinguish three cases giving rise to three different algorithms.

(1) If we choose

$$p(t) = \frac{1}{\sqrt{2}} \left(\sqrt{1 + m_3(t)}, \frac{m_2(t)}{\sqrt{1 + m_3(t)}}, -\frac{m_1(t)}{\sqrt{1 + m_3(t)}}, 0 \right),$$

then the corresponding angle is

$$\psi(t) = \frac{t - t_0}{I_3} + \frac{I_{31}}{I_1 I_3 \lambda} [\Pi(\varphi(t), n, k) + f(t) - \Pi(\varphi(t_0), n, k) - f(t_0)], \quad (16)$$

where $\varphi(s) = \text{am}(\lambda s - \nu, k)$ with λ , k and ν as in (4), $n = -(B_{31}/B_{13})^2$ and

$$f(s) = B_{21}^{-1} B_{13} B_{31} \arctan(B_{13}^{-1} B_{21} \text{sd}(\lambda s - \nu, k)).$$

This is a rescaled version of the algorithm presented in [Kosenko 1998].

(2) Taking

$$p_3^2 = \frac{1 + m_3}{4} + \frac{I_{32}m_2}{4I_2I_3\|\dot{\mathbf{m}}\|} (m_3 - B_{32}), \quad \text{sign}(p_0p_3) = \text{sign}\left(m_1m_3\frac{I_{31}}{I_1I_3} + m_1\frac{1 - 2TI_1}{I_1}\right)$$

and determining (p_0, p_1, p_2) according to (15), we retrieve the angle ψ of (9), and we obtain the quaternion formulation of the algorithm of section 1.3.

(3) Taking

$$p_3^2 = \frac{1 + m_3}{4} - \frac{m_2(1 + m_3)}{4\sqrt{1 - m_3^2}}, \quad \text{sign}(p_0p_3) = \text{sign}(m_3m_2 - m_1)$$

and determining (p_0, p_1, p_2) according to (15), the rotation angle we obtain is

$$\psi = \int_{t_0}^t \frac{2TI_3 - m_3^2}{I_3(1 - m_3^2)} ds = \frac{t - t_0}{I_3} + \frac{I_{31}}{\lambda I_3 I_1} (\Pi(\text{am}(\lambda t - \nu), n, k) - \Pi(\text{am}(\lambda t_0 - \nu), n, k))$$

with $n = -B_{31}^{-2}B_{13}^{-2}$.

This produces a quaternion version of the algorithm based on rotation matrices recently considered by van Zon and Schofield [van Zon and Schofield 2007a].

2. THE ALGORITHMS

In this sections we present an overview of the algorithms that perform the FRB computations in the interval $[t_0, t_0 + h]$. The parameters k, λ_1, λ_3 are defined in (3). The two algorithms differ in the choice of the frame $P(t)$, for matrices, and $p(t)$, for quaternions in the updates (6) and (12).

Algorithm 1. Rigid body with rotation by matrices

- (1) Set up the initial parameters
- (2) **if** $\lambda_3 = 0$ or $\lambda_1 = 0$ (special case) **then**
- (3) Update rotation matrix Q by a 2-dim rotation
- (4) **else** (generic motion)
- (5) Compute k
- (6) **if** $k < 1$ **then**
- (7) Compute parameters ν, λ, n as in (4)
- (8) Compute frame P at $t = t_0$
- (9) Compute the angular momentum \mathbf{m} and $\dot{\mathbf{m}}$ at $t = h$
- (10) **else**
- (11) Set $k \mapsto 1/k$.
- (12) Compute parameters ν, λ, n as in (5)
- (13) Compute frame P at $t = t_0$
- (14) Compute the angular momentum \mathbf{m} and $\dot{\mathbf{m}}$ at $t = t_0 + h$
- (15) **end if**
- (16) Compute angle ψ by (9)
- (17) Compute new frame at P at $t = t_0 + h$ using angle ψ
- (18) Update attitude Q using frame P at t_0 and $t_0 + h$ and input Q_0 by (6).

(19) **end if**

Algorithm 2. Rigid body with rotation by quaternions

- (1) Set up the initial parameters
 - (2) **if** $\lambda_3 = 0$ or $\lambda_1 = 0$ (special case) **then**
 - (3) Update rotation quaternion q by a 2-dim rotation
 - (4) **else** (generic motion)
 - (5) Compute k
 - (6) **if** $k < 1$ **then**
 - (7) Compute parameters ν, λ, n as in (4)
 - (8) Compute frame at p at $t = t_0$
 - (9) Compute the angular momentum \mathbf{m} at $t = t_0 + h$
 - (10) Compute angle ψ as in (16)
 - (11) Compute new frame p at $t = t_0 + h$ using angle ψ
 - (12) Update attitude q using frame at t_0 and $t_0 + h$ and input q_0 by (12).
 - (13) **else**
 - (14) Set $k \mapsto 1/k$.
 - (15) Compute parameters ν, λ, n as in (5)
 - (16) Compute frame p at $t = t_0$
 - (17) Compute the angular momentum \mathbf{m} at $t = t_0 + h$
 - (18) Compute angle ψ as in (16)
 - (19) Compute new frame p at $t = t_0 + h$ using angle ψ
 - (20) Update attitude q using frame p at t_0 and $t_0 + h$ and input q_0 by (12).
 - (21) **end if**
 - (22) **end if**
-

Specifically, for each algorithm, a suitable choice for the frame $P(t)$ (resp. $p(t)$) is made by means of the momentum. For the rotation matrix, we choose P based on \mathbf{m} and its derivative, see section 1.3. For the quaternion algorithm, we make the choice described in case (1), see 1.4. The angle $\psi(t)$, whose computation will be described later, defines a planar rotation $Y(t)$ (resp. $y(t)$) given as $Y(t) = \exp(\psi(t)\hat{\mathbf{e}}_3)$ which is a Givens rotation, and $y(t) = (\cos(\psi(t)/2), \mathbf{e}_3 \sin(\psi(t)/2))$. Finally the three rotations in (6) and (12) are composed, either with two 3×3 matrix products (one involving a Givens rotation), or with two quaternions products.

3. USAGE AND IMPLEMENTATION

The fundamental FORTRAN routines that implement Algorithms 1 and 2 are

```
subroutine frb_step(Piout, Qout, h, aInert, Piin, Qin, Np)
subroutine quat_step(Piout, qout, h, aInert, Piin, qin, Np)
```

where $\mathbf{Piout} = \mathbf{m}(h)$, $\mathbf{Piin} = \mathbf{m}_0$ are the angular momentum vectors out and in, normalized to unit vectors, $\mathbf{Qout} = Q(h)$, $\mathbf{Qin} = Q_0$ are the 3×3 attitude matrices out and in, $\mathbf{qout} = q(h)$, $\mathbf{qin} = q_0$ are the quaternions (4 dimensional vector) out and in, h is the time step of integration and finally \mathbf{aInert} is a 3-dimensional vector, with elements $I_1 < I_2 < I_3$. \mathbf{Np} is an integer parameter related to the elliptic integral of the third kind. The value $\mathbf{Np} = 0$ corresponds to exact integrals, while other values $1 \leq \mathbf{Np} \leq 10$ correspond to quadrature of order $p = 2\mathbf{Np}$ (see below for details).

A typical application of the routines would be through some splitting method. Assume that the original interval of integration is $[0, T_{\text{fin}}]$ and that this interval is divided in \mathbf{NSteps} sub-intervals with stepsize hi . In each of these sub-intervals, the flow of the system $F = A + B$ is approximated by a splitting method, which requires the alternation of two (or more) elementary flows A and B for a certain number of internal steps $\mathbf{NInternalSteps}$. Either A or B might be free rigid body sub-problems. The stepsizes h_{ij} must obey the requirement $\sum_j h_{ij} = hi$.

A typical driver for the subroutines would look like:

```

Initialization
do i=1:NSteps
  do j= 1:NInternalSteps
    other computations
    frb computations (Piout, Qout, hij, aInert, Piin, Qin, Np)
    other computations
  enddo
enddo
Finalization

```

Fig. 1. A typical driver for equations with a free rigid body component solved with a splitting method

3.1 Rescaling of the inputs and reordering of the inertia moments

Algorithms 1 and 2 assume $\|\mathbf{m}_0\| = 1$. To treat the general case, it suffices to rescale both the angular momentum and time: under the change of variable $\mathbf{m} \mapsto \mathbf{m}/G$, $t \mapsto Gt$, equation (1) remains invariant. Therefore we scale \mathbf{m} to $\mathbf{m}/\|\mathbf{m}_0\|$ ($G = \|\mathbf{m}_0\|$) and h to $\|\mathbf{m}_0\|h$ prior and soon after the computational routine (see figure 2).

```

...
do j= 1:NInternalSteps
  other computations
  Compute  $G = \|\mathbf{Piin}\|$ , set  $\mathbf{Piin} := \mathbf{Piin}/G$ ,  $h_{ijG} := G * h_{ij}$ 
  frb computations (Piout, Qout, hijG, aInert, Piin, Qin, Np)
  Set back  $\mathbf{Piout} := G * \mathbf{Piout}$ 
  other computations
enddo
...

```

Fig. 2. How to adjust time stepping in order to deal with non-normalized inertia moments

We also assume that the inertia moments are sorted in increasing order, $I_1 < I_2 < I_3$. If this is not the case, one needs to find a permutation matrix \mathcal{P} ($\mathcal{P}^{-1} = \mathcal{P}^T$) that sorts the inertia moments $J = \mathcal{P}I\mathcal{P}^{-1}$, $J_1 < J_2 < J_3$. As permutation matrices are orthogonal, they have property that $\mathcal{P}(\mathbf{a} \times \mathbf{b}) = \mathcal{P}\mathbf{a} \times \mathcal{P}\mathbf{b}$, and this implies that $\mathbf{n} = \mathcal{P}\mathbf{m}$ obeys the equation

$$\dot{\mathbf{n}} = \mathbf{n} \times J^{-1}\mathbf{n}$$

By a similar token, $\mathcal{P}(\widehat{I^{-1}\mathbf{m}})\mathcal{P}^{-1} = J^{-1}(\widehat{\mathcal{P}\mathbf{m}})$, hence the matrix $R = Q\mathcal{P}^T$ obeys the differential equation

$$\dot{R} = R\widehat{J^{-1}\mathbf{n}}.$$

When necessary, the permutation (that shuffles the rows of \mathbf{m} , and, correspondingly, the columns of Q) is performed before/after the computational if the inertia moment change with time or when the components of the vector fields are easy to permute.

```

...
do j= 1:NInternalSteps
  other computations
  Set i1 := 1, i2 := 2, i3 := 3
  if NOT(aInert(1) < aInert(2) < aInert(3)) then
    Find indices i1, i2, i3 such that aInert(i1) < aInert(i2) < aInert(i3).
    Set Piin := Piin([i1, i2, i3]), aInert := aInert([i1, i2, i3])
    Set Qin := Qin(:, [i1, i2, i3])
  end if
  Compute G = ||Piin||, set Piin := Piin/G, hijG := G * hij
  frb computations (Piout, Qout, hijG, aInert, Piin, Qin, Np)
  Set back Piout := G * Piout
  Set back Qout := Qout(:, [i1, i2, i3])
  Set back Piout := Piout([i1, i2, i3]), aInert := aInert([i1, i2, i3])
  other computations
enddo
...

```

Fig. 3. A permutation that shuffles the rows of \mathbf{m} is applied before and after the call to the subroutine that performs the rigid body computations. Such permutation is important when the inertia moments change with time. When the inertia moments are constants with time, it often suffices to apply the permutation and its inverse at the beginning and end of the computations.

When the inertia moments do not change with time, the permutation can be applied once and for all in the initialization procedure.

3.2 Elliptic integrals of the first kind and Jacobi elliptic functions for the update of the angular momentum

In order to update the momentum solution of the Euler equations (1) (see also lines (9), (14) of Algorithm 1 and (9), (17) of Algorithm 2), it is necessary to compute elliptic integrals of the first kind and the Jacobi elliptic functions sn , cn , dn .

Elliptic integrals of the first kind can be computed very rapidly by using algorithms based on the arithmetic geometric mean (AGM) sequence and the ascending/descending Landen transformations [Abramowitz and Stegun 1992].

As an example, consider the formulae for the momentum vector in (4): once we have computed the constants k , B_{13} , B_{21} , B_{31} , λ using the inertia moments and the constants of motion we have to determine the phase ν and the sign $\sigma (= \pm 1)$. This is done using the initial condition $\mathbf{m}(t_0) = \mathbf{m}_0$. The computation of ν requires the calculation of an elliptic integral of the first kind.

Once the phase ν is computed, the Jacobi elliptic functions $\text{sn}(\lambda t - \nu, k)$, $\text{cn}(\lambda t - \nu, k)$ and $\text{dn}(\lambda t - \nu, k)$ should be evaluated.

Both these tasks require the computation of the AGM sequence with the same initial conditions, and the computation of two Landen transformations (one ascending and one descending) to compute the elliptic integral and the amplitude for the elliptic functions. These tasks are implemented in the subroutine `EgellpjX.step` described here in the sequel, see also [Celledoni and Säfstöm 2006].

We proceed as follows. Assume $\{a_n\}_n$, $\{b_n\}_n$, $\{c_n\}_n$ is the AGM sequence with $a_0 = 1$, $b_0 = \sqrt{1 - k^2}$, $c_0 = k$ and $a_{n+1} = \frac{a_n + b_n}{2}$, $b_{n+1} = \sqrt{a_n b_n}$, $c_{n+1} = \frac{a_n - b_n}{2}$. In this sequence $c_n \rightarrow 0$ as $n \rightarrow \infty$, and we denote with N the index such that $c_N \leq \text{eps}$, (with eps the machine epsilon). Given ϕ_0 , the transformation

$$\tan(\phi_{n+1} - \phi_n) = \frac{b_n}{a_n} \tan(a_n), \quad (17)$$

is such that

$$\lim_{n \rightarrow \infty} \frac{\phi_n}{2^n a_n} = \int_0^{\phi_0} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}, \quad (18)$$

so

$$\frac{\phi_N}{2^N a_N} \approx \int_0^{\phi_0} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}. \quad (19)$$

Using the components of the initial condition $\mathbf{m}(t_0)$ we obtain the equations

$$\sigma \text{dn}(\lambda t_0 - \nu) = \frac{m_1(t_0)}{B_{13}}, \quad \text{sn}(\lambda t_0 - \nu) = \frac{m_2(t_0)}{B_{21}}, \quad \text{cn}(\lambda t_0 - \nu) = \frac{m_3(t_0)}{B_{31}}.$$

Since we have

$$\text{sn}(\lambda t_0 - \nu) = \sin(\text{am}(\lambda t_0 - \nu)), \quad \text{cn}(\lambda t_0 - \nu) = \cos(\text{am}(\lambda t_0 - \nu)),$$

see the definitions in the appendix, as $m_2(t_0)$ and $m_3(t_0)$ are known, we can compute the amplitude $\text{am}(\lambda t_0 - \nu)$. We will denote its value by φ_0 in what follows. From $m_1(t_0)$ we find σ . The map am can be inverted (see 21), we have

$$\lambda t_0 - \nu = \int_0^{\varphi_0} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}.$$

Now we can use (17) with $\phi_0 := \varphi_0$ and (19), to approximate the right hand side of the last equation and compute $\lambda t_0 - \nu$. Finally the argument where sn , cn and dn are to be evaluated is simply $\lambda t - \nu = \lambda t_0 - \nu + \lambda h$.

To proceed at the numerical evaluation of the Jacobi elliptic functions one first approximates the amplitude $\varphi_1 := \text{am}(\lambda t - \nu)$ using the descending transformation

$$\sin(2\phi_{n-1} - \phi_n) = \frac{c_n}{a_n} \sin(\phi_n), \quad n = N, N - 1, \dots, 0,$$

starting with $\phi_N = 2^N a_N(\lambda t - \nu)$, and obtaining $\varphi_1 = \phi_0$. Finally $\text{sn}(\lambda t - \nu) = \sin(\varphi_1)$, $\text{cn}(\lambda t - \nu) = \cos(\varphi_1)$ and $\text{dn}(\lambda t - \nu) = \sqrt{1 - k^2 \sin(\varphi_1)^2}$.

An alternative way of proceeding would be to use addition formulae for the Jacobi elliptic functions, [Byrd and Friedman 1971], and rewrite the momentum components as products of Jacobi elliptic functions depending separately on ν and λt . This approach is interesting when we want to compute also the derivative of the exact solution with respect to the initial condition (tangent map). See for example [Bates and Fassò 2007] for the use of the tangent map of the flow of the Euler top in numerical simulations. However such strategy might require more evaluations of elliptic functions per time step compared to the present algorithm.

3.3 Elliptic integrals of the third kind for the update of the attitude

In order to update the angle ψ in (9) and (16) (see also line (16) of Algorithm 1 and lines (10) and (18) of Algorithm 2), it is necessary to compute elliptic integrals of the third kind.

The same techniques used for the elliptic integrals of the first kind can be used also for the elliptic integral of the third kind, but their performance is not so uniform, and other algorithms are preferred instead. Our implementation in the FORTRAN subroutines `frb_step` and `quat_step` makes use of Carlson's algorithms `rf`, `rfj`, and `rc` [Carlson and Notis 1981] that have been acclaimed to produce accurate values for large sets of parameters. These methods are among the most common routines for elliptic integrals of the third kind in several scientific libraries, see for instance [Press et al. 1996]. There exist other implementations based on theta functions; these are used for instance by [van Zon and Schofield 2007a].

An alternative to the exact computation of the elliptic integral of the third kind is the approximation by a quadrature method. Unless quadrature is performed to machine accuracy, the resulting methods will be *semi-exact*, in the sense that, though Q is approximated only to the order of the underlying method, by construction, they integrate the angular momentum exactly, they preserve $Q\mathbf{m}$ (because of the properties of the matrix P in Proposition 2.2 of [Celledoni et al. 2008]), and they are time-symmetric if the underlying quadrature formula is symmetric.

Other authors [van Zon and Schofield 2007b] approximate the integral

$$\int_{u_0}^u \frac{ds}{1 - n \text{sn}^2 s}$$

by a quadrature based on Hermite interpolation, as the function sn and its derivative can be easily computed at the end points of the interval. Alternatively, one can write the same integral in the Legendre form:

$$\int_{\text{am}(u_0)}^{\text{am}(u)} \frac{d\theta}{(1 - n \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}. \quad (20)$$

This format is convenient when using quadrature formulas because it requires tabulating the sine function in the quadrature nodes instead of $\text{sn}(\lambda(t - \nu))$. Thus, (20) can be approximated as

$$\int_{\text{am}(u_0)}^{\text{am}(u)} f(\theta) d\theta \approx \sum_{i=1}^p b_i f(\varphi_0 + a_i \Delta\varphi),$$

where $\Delta\varphi = \text{am}(u) - \text{am}(u_0)$ and b_i and a_i are weights and nodes of a quadrature formula, respectively. Our package includes a subroutine `eint_gauss.f` that uses Gaussian quadrature (i.e., quadrature based on orthogonal polynomials), because of its high order. In particular, Gauss–Legendre quadrature with p points attains the maximal quadrature order $2p$. Numerical experiments [Celledoni et al. 2008] indicate that this approximation is very effective. With respect to the exact methods, the semiexact methods obtained with this approach are more directly comparable to the preprocessed discrete Moser–Veselov methods of [Hairer and Vilmart 2006], see also [McLachlan and Zanna 2005].

3.4 Fortran components

The main FORTRAN computational subroutines are `frb_step` and `quat_step` and are invoked as

```
call frb_step(Piout, Qout, h, aInert, Piin, Qin, Np)
call quat_step(Piout, qout, h, aInert, Piin, qin, Np)
```

with the same meaning for the symbols as in §3. The subroutines `frb_step` and `quat_step` include the following components:

`EgellipX_step` : Computes the momentum using the AGM and ascending/descending Landen transformation, see section 3.2 for a closer description.

`ellint_pi` : ($Np = 0$). Computes the elliptic integral of the third kind between φ_1 and φ_2 as difference of two incomplete elliptic integrals between 0 and φ_2 and 0 and φ_1 . If the angles φ_i , $i = 1, 2$, are larger than $\pi/2$, the complete elliptic integral is also computed. The integrals are computed by calling the computational routine `ellpi`.

`ellpi` : Computes incomplete elliptic integrals between 0 and $\varphi_i \leq \pi/2$, $i = 1, 2$, using `rf`, `rj`, `rc`.

`eint_gauss` : ($1 \leq Np \leq 10$). Computes the elliptic integrals of the third kind from 0 to φ using Gaussian quadrature with $1 \leq p = np \leq 10$ nodes (order $2p$). This approximation is suitable only if the time-step of integration is not too large.

Driver examples:

`driver_example0` : This is an example of computations of a free rigid body in the interval $[0, T_{\text{fin}}]$ (single step of length $h = T_{\text{fin}}$). The input data are loaded from the data file `init_example0.dat` and written in the output file `out_example0.dat`. This driver uses the computational routine `frb_step.f`.

`driver_example1` : This is an example of computations of a heavy top in the interval $[0, T_{\text{fin}}]$ with step-size h and a sixth-order splitting method [Blanes and Moan 2002]. The input data (including initial condition, initial attitude Q , inertia moments, \mathbf{u}_0 , and the coefficients of the splitting methods) are loaded from the data file `init_example1.dat` and the results written in output file `out_example1.dat`. This driver uses the computational routine `frb_step.f`.

`driver_example2` : This is similar to `driver_example0`, except that it uses quaternions (`quat_step.f`) instead of rotations (`frb_step.f`). The input data are in `init_example2.dat` and the output data are written to `out_example2.dat`.

`driver_example3` : This performs a free rigid body integration in the interval $[0, T_{\text{fin}}]$ with N steps of length h ($T_{\text{fin}} = Nh$) using Gaussian quadrature of order $p = 2Np$. The input data are loaded from the data file `init_example3.dat` and written in the output file `out_example3.dat`. This driver uses the computational routine `frb_step.f`.

3.5 Portability: compilation requirements

This package is compatible with FORTRAN 95 (`f95`, `g95`, `gfortran` compilers) and FORTRAN 77 (`f77`, `g77`). To compile, the main computational routines `frb_step/quat_step` are linked to a driver. As an example,

```
gfortran --optimize -o driver_example1 driver_example1.f frb_step.f
```

compiles the computational subroutine with the driver `driver_example1.f` (included) to the executable file `driver_example1`. Compilation of all the driver examples can be made using the `make` command, which, by default, uses the `f95`, `g95` command.

4. TEST EXAMPLES

4.1 Accuracy

The accuracy of numerical methods for the approximation of rigid bodies depends on the inertial matrix I and the initial condition \mathbf{m}_0 for the angular momentum, see [Fassò 2003; Celledoni et al. 2008]. We perform an accuracy test of the exact computational routines (`frb_step`, `quat_step`, through the drivers `driver_example0` and `driver_example2`) using the same approach as [Celledoni et al. 2008]: we consider values of the form $I_1/I_3 < I_2/I_3 < 1$ for the inertia matrix (other values correspond to time-reparametrization) and we visualize the average number of significant digits in the triangle

$$\mathcal{T} = \{(x, y) \in \mathbf{R}^2 : 0 < 1 - y \leq x < y < 1\}$$

where $x = I_1/I_3$ and $y = I_2/I_3$ (see Figure 4) in the following manner. We construct a discretization of this triangle by superimposing a rectangular grid (100 points in the x direction and 50 in the y direction). For each point (x, y) in the interior of the triangle, we solve 20 initial value problems with initial condition \mathbf{m}_0 in the first octant. The initial Q is the identity matrix (identity quaternion resp.). This set of initial parameters is identical for both methods. Thereafter, we compute the average \log_{10} of the error for each method (whose absolute value corresponds to the number of significant digits). The “reference” solution is computed with MATLAB’s `ode45`, setting both absolute and relative error to machine accuracy. The results of the experiments are shown in Figure 5, computed with a single integration step size $h = 1$. The conclusion is that the computational routines provide accurate results (order of machine accuracy) for large portions of the data sets. The “reference” solution is not fully reliable in the top left corner of the triangle, due to the properties of the problem.

4.2 Performance

As far as performance is concerned, we have tested the computational time of the exact routines versus the class of `dmv` methods [Hairer and Vilmart 2006] (in

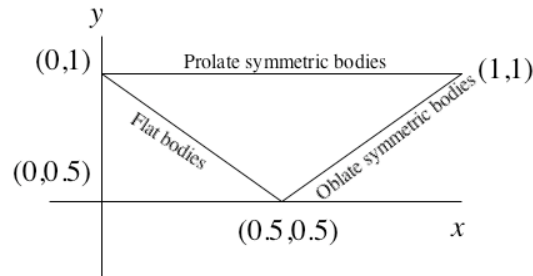


Fig. 4. Parametrization domain for the matrix of inertia. x -axis: I_1/I_3 , y -axis: I_2/I_3 .

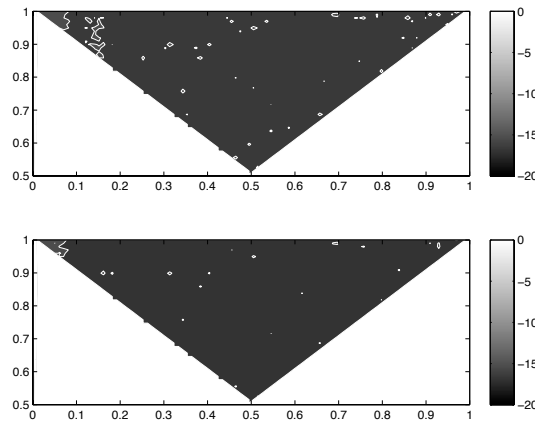


Fig. 5. Average \log_{10} error for the various values of the matrix of inertia with step size $h = 1$. Comparison of exact methods. Top: Matrix case. Bottom: Quaternion case.

FORTTRAN), that approximate to given order the equations (1)-(2) using modified inertia moments. To our knowledge, these methods are among the most efficient routines for the numerical approximation of free rigid bodies. The integration of (1)-(2) is performed in the interval $[0, 10]$ with several step sizes h , performing $N = \lfloor 10/h \rfloor$ steps (even though the exact methods would give the exact solution with a single step).

The cost of the exact algorithms varies between 8 and 20 times the cost of the approximate `dmv` algorithms (of order 6, 8, 10), and this indicates that the exact methods indeed can be competitive when large step sizes are used. We have also tested the semi-exact variants, for which the computation of the elliptic integral of third kind (20) is replaced by Gaussian quadrature (`eint_gauss` routine). In these cases, the cost is reduced to about a third, and this is reasonable, as the cost of a Gaussian quadrature corresponds approximatively to the cost of a elliptic integral of the third kind, while the exact method requires the computation of three of those.

More extensive testing of the the semi-exact methods and comparison with the

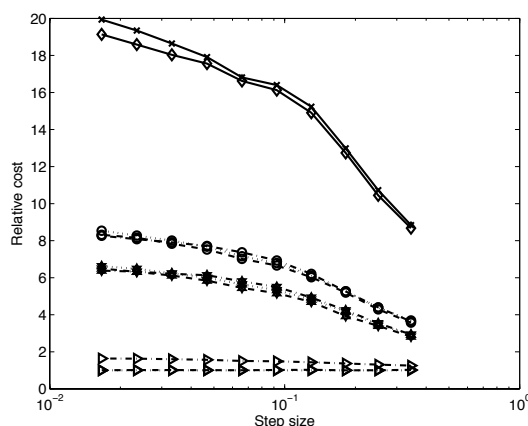


Fig. 6. Average CPU time (y -axis) versus step size (x -axis) for the exact routine based on rotations (solid line and diamonds) and on quaternions (solid line and crosses) relative to the cheapest `dmrv` method (line $y = 1$) for several values of the step size. The exact methods are 8-20 times more expensive than the approximate `dmrv` methods (triangles). The other classes of methods are the semi-exact methods (stars for rotation matrices and circles for quaternions). See text for details.

`dmrv` methods, as well as application to heavy bodies, satellite simulations and molecular dynamics can be found in [Celledoni et al. 2008].

4.3 Application in splitting methods

In the last example we illustrate the application of our algorithms to splitting methods. We consider the problem

$$\begin{aligned}\dot{\mathbf{m}} &= \mathbf{m} \times I^{-1}\mathbf{m} + \mathbf{f}(Q) \\ \dot{Q} &= Q\widehat{I^{-1}\mathbf{m}}\end{aligned}$$

where $\mathbf{f}(Q) = [u(2), -u(1), 0]^T$, with $\mathbf{u} = Q^T\mathbf{u}_0$. This problem is split in rigid body motion plus torque, which we solve by a sixth-order splitting method [Blanes and Moan 2002]. The integration is performed in the interval $[0, 5.0e+04]$, with time step $h = 0.5$ and initial conditions

$$\mathbf{m}_0 = \begin{bmatrix} -3.4790957088547336e-01 \\ -1.9822914599675923e-01 \\ -9.1633189192763642e-01 \end{bmatrix}, \quad \mathbf{u}_0 = \begin{bmatrix} 9.5586303547238536e-05 \\ 4.8777318247201465e-04 \\ -8.6772148817192390e-04 \end{bmatrix},$$

inertia moments $I_1 = 1, I_2 = 1.0126869887825154, I_3 = 3.3062374224730378$ and initial attitude $Q_0 = I$, the identity matrix (data in `init_example1.dat`). The driver of this example is `driver_example1`. This is an example of perturbed rigid body motion, as $\|\mathbf{u}_0\|_2 = 10^{-3}$. The numerical trajectory (see Figure 7) stays nicely bounded over very long time. A close-up on the trajectory reveals that the trajectory evolves on a torus close to the unperturbed problem. The braids are a consequence of the fact that the integration time is an integer multiple of the step size h .

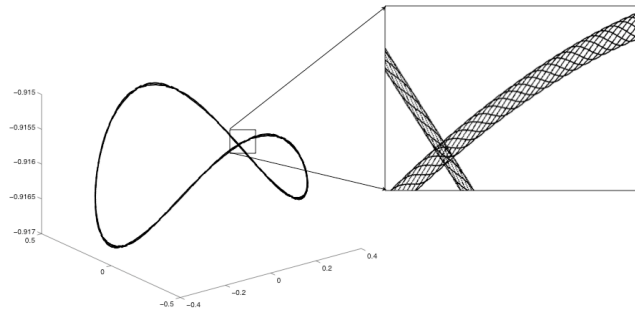


Fig. 7. The vector $\mathbf{m}_i \approx \mathbf{m}(ih)$ for the perturbed rigid body motion over long time integration and close-up to a section of the trajectory. The trajectory stays nicely bounded over very long time. A close-up reveals that the orbit is on a torus close to the unperturbed problem.

5. CONCLUSION

In this paper we have described two algorithms, with associated FORTRAN sub-routines, for the exact computation of free rigid body motions. The algorithms rely on computation of elliptic functions, which we compute using iteration algorithm that always converge, whereas standard iterative procedures might require good starting values for convergence.

The FORTRAN routines are meant for use within splitting methods, and are relevant in mechanics, celestial mechanics, molecular biology and applications where it is possible to decompose the underlying dynamics in a free rigid body motion plus a torsion/force component.

The cost of the proposed routines is higher than other existing methods (a factor between 8 and 20, with respect to highly efficient methods for rigid body dynamics hitherto [Hairer and Vilmart 2006]). However, this is not a major problem, as typically the calculation of inter-body forces (order N^2 force evaluations for N bodies) is much more demanding than the rigid body part (order N rigid body evaluations).

Extensive numerical tests [Celledoni et al. 2008] have indicated that the use of exact routines produces better results especially in regimes where the error of the underlying splitting is not so large. If the splitting error is large, this error would typically subsume errors that come, for instance, by less accurate rigid body simulations, and the gain of using the exact routines becomes less evident. For this reason, the proposed methods are particularly indicated for nearly-integrable problems, in which case we expect superior numerical results with respect to other numerical procedures.

Appendices

Jacobi elliptic functions

We collect here a few facts about the elliptic functions we use in the article. Given $0 \leq k < 1$, the function

$$\varphi \mapsto F(\varphi, k) := \int_0^\varphi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}} \quad (21)$$

is called (incomplete) *elliptic integral of the first type* with modulus k and is a diffeomorphism $\mathbf{R} \rightarrow \mathbf{R}$. Its inverse $F(\cdot, k)$ is an odd function

$$\text{am}(\cdot, k) : \mathbf{R} \rightarrow \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$$

which is called *amplitude* of modulus k . The *Jacobi elliptic functions* sn and cn of modulus k are the functions $\mathbf{R} \rightarrow [-1, 1]$ defined as

$$\text{sn}(u, k) = \sin(\text{am}(u, k)), \quad \text{cn}(u, k) = \cos(\text{am}(u, k))$$

and are periodic of period $4K(k)$, where $K(k) = F(\frac{\pi}{2}, k)$ (the so called complete elliptic integral of the first type of modulus k). Moreover,

$$\text{dn}(u, k) = \sqrt{1 - k^2 \text{sn}(u, k)^2}, \quad \text{sd}(u, k) = \frac{\text{sn}(u, k)}{\text{dn}(u, k)}.$$

For given k , the u -derivatives of these functions satisfy $\text{sn}' = \text{cn} \text{dn}$, $\text{cn}' = -\text{sn} \text{dn}$ and $\text{dn}' = -k^2 \text{sn} \text{cn}$.

The (incomplete) *elliptic integral of the third kind* with modulus $0 < k \leq 1$ and parameter $n \in \mathbf{R}$ is the function $\Pi(\cdot, n, k) : (-\frac{\pi}{2}, \frac{\pi}{2}) \rightarrow \mathbf{R}$ defined by

$$\Pi(\varphi, n, k) := \int_0^\varphi \frac{d\theta}{(1 - n \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}, \quad (22)$$

(Legendre form), or equivalently

$$\Pi(\varphi, n, k) = \int_0^{F(\varphi, k)} \frac{ds}{1 - n \text{sn}(s, k)^2}.$$

Coefficients of the splitting schemes

Given the differential equation

$$y' = F(y) = A(y) + B(y),$$

denote by $\varphi_\tau^{[F]}$ the flow of the vector-field F from time t to time $t + \tau$. Given a numerical approximations $y^{(j)} \approx y(t_j)$, we consider symmetric splitting schemes of the type

$$y^{(j+1)} = \varphi_{a_1 h}^{[A]} \circ \varphi_{b_1 h}^{[B]} \circ \varphi_{a_2 h}^{[A]} \circ \cdots \circ \varphi_{a_m h}^{[A]} \circ \cdots \circ \varphi_{b_1 h}^{[B]} \circ \varphi_{a_1 h}^{[A]} y^{(j)},$$

where $h = t_{j+1} - t_j$. A typical splitting is obtained separating the contributions arising from the kinetic (A) and potential (B) energy of the system. For this reason, (twice) the number s of the coefficients b_i is called the *stage number* of the splitting method. The effective error is defined as $E_f = s \sqrt{\|\mathbf{c}\|_2}$, where \mathbf{c} is the

vector of coefficients of the elementary differentials of the leading error term and p is the order of the method. We refer to [Blanes and Moan 2002; McLachlan and Quispel 2002] for background and notation.

For completeness, we report the coefficients of the method used in the accompanying examples:

RKN6₁₄^a (order 6, 14 stages, effective error $E_f = 0.63$):

$$\begin{aligned}
 a_1 &= 0.0378593198406116, & b_1 &= 0.09171915262446165, \\
 a_2 &= 0.102635633102435, & b_2 &= 0.183983170005006, \\
 a_3 &= -0.0258678882665587, & b_3 &= -0.05653436583288827, \\
 a_4 &= 0.314241403071477, & b_4 &= 0.004914688774712854, \\
 a_5 &= -0.130144459517415, & b_5 &= 0.143761127168358, \\
 a_6 &= 0.106417700369543, & b_6 &= 0.328567693746804, \\
 a_7 &= -0.00879424312851058, & b_7 &= 1/2 - (b_1 + \dots + b_6), \\
 a_8 &= 1 - 2(a_1 + \dots + a_7).
 \end{aligned} \tag{23}$$

REFERENCES

- ABRAMOWITZ, M. AND STEGUN, I. A. 1992. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. National Bureau of Standards Applied Mathematics Series, 55, vol. 55. Reprint of the 1972 edition. Dover Publications, Inc., New York.
- BATES, L. AND FASSÒ, F. 2007. The conjugate locus for the Euler top. I. The axisymmetric case. *Int. Math. Forum* 2, 2109–2139.
- BLANES, S. AND MOAN, P. C. 2002. Practical symplectic partitioned Runge–Kutta and Runge–Kutta–Nyström methods. *J. Comp. Appl. Math.* 142, 2, 313–330.
- BYRD, P. AND FRIEDMAN, M. 1971. *Handbook of elliptic integrals for engineers and scientists.*, Second edition ed. Die Grundlehren der mathematischen Wissenschaften, Band 67. Springer-Verlag, New York-Heidelberg.
- CARLSON, B. C. AND NOTIS, E. M. 1981. Algorithm 577: Algorithms for incomplete elliptic integrals [S21]. *ACM Transactions on Mathematical Software* 7, 3 (Sept.), 398–403.
- CELLEDONI, E., FASSÒ, F., SÄFSTÖM, N., AND ZANNA, A. 2008. The exact computation of the free rigid body motion and its use in splitting methods. *SIAM J. Sci. Comp.* 30, 2084–2112.
- CELLEDONI, E. AND SÄFSTÖM, N. 2006. Efficient time-symmetric simulation of torqued rigid bodies using Jacobi elliptic functions. *Journal of Physics A* 39, 5463–5478.
- CUSHMAN, R. 2000. No polar coordinates. Lecture notes, MASIE summer school, Peyresq, France, September 2–16.
- DULLWEBER, A., LEIMKUHNER, B., AND MCLACHLAN, R. 1997. Symplectic splitting methods for rigid body molecular dynamics. *J. Chem. Phys.* 107, 5840–5851.
- FASSÒ, F. 2003. Comparison of splitting algorithm for the rigid body. *J. Comput. Phys.* 189, 2, 527–538.
- HAIRER, E. AND VILMART, G. 2006. Preprocessed discrete Moser–Veselov algorithm for the full dynamics of a rigid body. *J. Phys. A* 39, 13225–13235.
- KOSENKO, I. I. 1998. Integration of the equations of a rotational motion of rigid body in quaternion algebra. The Euler case. *J. Appl. Maths Mechs* 62, 2, 193–200.
- LAWDEN, D. F. 1989. *Elliptic functions and applications*. Applied Mathematical Sciences, vol. 80. Springer-Verlag, New York.
- LEIMKUHNER, B. AND REICH, S. 2004. *Simulating Hamiltonian Dynamics*, First edition ed. Cambridge Monographs on Applied and Computational Mathematics, vol. 14. Cambridge University Press.
- MCLACHLAN, R. I. AND QUIPSEL, G. R. W. 2002. Splitting methods. *Acta Numer.* 11, 341–434.
- MCLACHLAN, R. I. AND ZANNA, A. 2005. The discrete Moser–Veselov algorithm for the free rigid body, revisited. *Found. of Comp. Math.* 5, 1, 87–123.
- ACM Transactions on Mathematical Software, Vol. xx, No. xx, xx 2008.

- MORTON, S. H., JUNKINS, J. L., AND BLANTON, J. N. 1974. Analytical solutions for Euler parameters. *Celestial Mech.* 10, 287–301.
- PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1996. *Numerical recipes in Fortran 77 and Fortran 90*. Cambridge University Press, Cambridge. The art of scientific and parallel computing, Second edition.
- ROMANO, M. 2008. Exact analytic solution for the rotation of a rigid body having spherical ellipsoid of inertia and subjected to a constant torque. *Cel. Mech. Dyn. Astr.* 100, 181–189.
- TOUMA, J. AND WISDOM, J. 1994. Lie–Poisson integrators for rigid body dynamics in the solar system. *Astr. J.* 107, 1189–1202.
- VAN ZON, R. AND SCHOFIELD, J. 2007a. Numerical implementation of the exact dynamics of free rigid bodies. *J. of Comput. Phys.* 225, 145–164.
- VAN ZON, R. AND SCHOFIELD, J. 2007b. Symplectic algorithms for simulations of rigid body systems using the exact solution of free motion. *Phys. Rev. E* 75.
- WHITTAKER, E. T. 1937. *A Treatise on the Analytical Dynamics of Particles and Rigid Bodies*, 4th ed. Cambridge University Press.

Received xxxx 2008; xxxxx accepted xxxx

THIS DOCUMENT IS THE ONLINE-ONLY APPENDIX TO:

FRB-FORTRAN routines for the exact computation of free rigid body motions

ELENA CELLEDONI

Department of Mathematical Sciences, NTNU

ANTONELLA ZANNA

Department of Mathematics, University of Bergen

ACM Transactions on Mathematical Software, Vol. xx, No. xx, xx 2008, Pages 111–129.

The contents of the electronic appendix is written after the references and the “received” environment. The electronic appendix is started by an `\elecappendix` command:

```
\elecappendix
```

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2008 ACM 0098-3500/2008/1200-0111 \$5.00