Journal of Engineering Mathematics **26**: 315–337, 1992.
© 1992 *Kluwer Academic Publishers. Printed in the Netherlands.*

Stable numerical integration of dynamical systems subject to equality state-space constraints

A.A. TEN DAM

Department of Mathematical Models and Methods, National Aerospace Laboratory NLR, P.O. Box 90502, 1006 BM Amsterdam, The Netherlands

Received 1 October 1990; accepted 1 March 1991

Key words: DAE, constraints, numerical integration

Abstract. Numerical simulation of constrained dynamical systems is known to exhibit stability problems even when the unconstrained system can be simulated in a stable manner. We show that not the constraints themselves, but the transformation of the continuous set of equations to a discrete set of equations is the true source of the stability problem. A new theory is presented that allows for stable numerical integration of constrained dynamical systems. The derived numerical methods are robust with respect to errors in the initial conditions and stable with respect to errors made during the integration process. As a consequence, perturbations in the initial conditions are allowed. The new theory is extended to the case of constrained mechanical systems. Some numerical results obtained when implementing the numerical method here developed are shown.

1. Introduction

Mechanical systems can be described by a second-order Ordinary Differential Equation (ODE). Solving the ODE yields the time-evolution of the state-variables that represent the internal behaviour of the mechanical system. Mechanical systems are often composed of subsystems. Instead of deriving an ODE that describes the whole system one can model each subsystem separately. However, in this set up relations between subsystems are not automatically satisfied. To compensate this, dependencies between state-variables must be added to the ODEs that describe the dynamic behaviour of the individual subsystems. The dependencies between state-variables are modelled as, additional, constraint equations. The combination of ODE and constraint equations gives rise to a Differential Algebraic Equation (DAE). The advantage of a DAE description can readily be seen by taking the case where subsystems are to be designed; exchanging models for subsystems is a necessity. This is clearly easier to do in case of a DAE description compared to the case where a new ODE must be derived that describes the complete constrained mechanical system.

For the evaluation of mathematical models one often has to take resort to simulation studies [1]. For simulation studies to be carried out it is necessary to have numerical algorithms that allow stable numerical integration. Simulation of dynamical systems described by a set of differential and algebraic equations gives rise to specific stability problems with respect to time-integration [2]. In this paper we will concentrate on the numerical solution of dynamical systems with equality state-space constraints.

In the present literature with respect to numerical integration DAEs are characterized by their index [3]. The index can be viewed upon as a measure of how far a DAE is from being an ODE. Constrained mechanical systems often have an index equal to three. So far, index-three problems have resisted efforts at their direct solution by (available) ODE

methods [4]. As a result, for constrained mechanical systems, stable numerical algorithms are few and impose restrictions on the kind of constraints that can be simulated [5].

Even if the unconstrained system can be simulated in a stable manner the addition of constraints leads to unstable numerical behaviour for the chosen formulation. So, it could be concluded that the numerical difficulties arise from the additional constraint equation(s). However, we will show that not the constraints themselves, but rather the transformation of the continuous set of equations to a discrete set of equations is the true source of the stability problem. Furthermore, it will be shown that there exists no need to introduce the index terminology to characterize difficulties with respect to numerical integration of DAEs. In fact, we show that use of the appropriate formulation automatically results in stable numerical integration of differential algebraic equations. This result will be extended to the case of constrained mechanical systems. It will also be shown that one can use well-known and well-documented standard ODE solvers, commonly accepted by engineers, to solve the constrained equations of motion in a stable manner. It should be mentioned that a similar stabilization technique, for index-one systems with linear, stationary constraints where the equations are solved with the aid of the Forward–Euler integration method, has successfully been applied in the area of computational fluid dynamics [6].

The remainder of this paper is as follows. In Section 2 the classical formulation and numerical solution of constrained mechanical systems is discussed. Also some assumptions are stated that are used throughout this paper. In Section 3 a solution to the problem of stable numerical integration of constrained dynamical systems is given. A new theory is presented that allows stable numerical integration of the corresponding DAEs. The results of Section 3 are extended to constrained mechanical systems in Section 4. In that section the numerical solution of this kind of systems is discussed. Simulation results are presented in Section 5. The conclusions are stated in Section 6.

2. Classical solution of mechanical systems with equality state-space constraints

In this section we describe the classical formulation of constrained mechanical systems. Details can be found in [7]. Consider a mechanical system that is composed of a number of subsystems. Interconnecting the subsystems yields the overall, constrained mechanical, system. Let k denote the number of subsystems and let $x_i \in \mathbf{R}^{n_i}$ denote the position coordinates of subsystem i, and $n := \Sigma_{i=1}^{k} n_i$. Then a compact description of the equations of motion (where the dot denotes the time derivative) is given by:

$$M(x)\ddot{x} = B(x, \dot{x}) + F_c \,. \tag{2.1}$$

Here x represents the positions of all involved subsystems, $x = [x_1^T, \ldots, x_k^T]^T$, with x_i the position vector of system i; M represents the inertia matrix of the composed system, $M = \text{diag}[M_1, \ldots, M_k]$, with M_i the inertia matrix of system i; B represents the Coriolis, gravitational and centrifugal force/torque vector, $B = [B_1^T, \ldots, B_k^T]^T$, with B_i the Coriolis, gravitational and centrifugal vector acting on system i; and F_c represents the control force/torque vector acting on the composed system, $F_c = [F_{c_1}^T, \ldots, F_{c_k}^T]^T$, with F_{c_i} the control force/torque vector of system i.

ASSUMPTION A
$\forall x_i \in \mathbf{R}^{n_i}$: $M_i(x_i)$ is positive-definite, $i = 1 \cdots k$.

Assumption A implies that we are dealing with a true ODE in the sense that for each x_i an ordinary differential equation is present. Later on this assumption will be relaxed.

If there are no additional restrictions, one is left with k independent systems. Relations between subsystems impose restrictions on the composed system. These relations are called constraints and are modelled as constraint equations. Usually the constraint equations are of lower order than the order of the ODEs that describe the behaviour of the subsystems. For instance, the equation

$$P(x, t) = 0\,, \tag{2.2}$$

with $P: \mathbf{R}^{n+1} \to \mathbf{R}^m$, $m \leqq n$, $t \in T$, represents a constraint equation on position level. Differentiation of Eq. (2.2) yields:

$$C(x, t)\dot{x} = d(x, t)\,, \tag{2.3}$$

where $C(x, t) := P_x(x, t)$ and $d(x, t) := -P_t(x, t)$. The matrix C is called the constraint Jacobian matrix.

ASSUMPTION B
$\forall (x, t) \in \mathbf{R}^{n+1}$ such that $P(x, t) = 0$: $\operatorname{rank}(C(x, t)) = m$, $m \leqq n$.

Assumption B implies that the constraint Jacobian matrix C has full row-rank, i.e. there are no redundant constraints. The constrained dynamical system can be looked upon as a dynamical system with its behaviour restricted by the constraints. This restriction now is the fundamental source of difficulties in the formulation and numerical solution of constrained dynamical systems. If constraint equations can be modelled as in Eq. (2.2) they are called holonomic constraints. Constraints as in Eq. (2.3) but not integrable to a form as in Eq. (2.2) are called nonholonomic constraints. In this paper we mainly consider holonomic constraints. The results, however, can be extended to nonholonomic constraints in a straightforward manner with the aid of the results presented in [7].

The classical way to formulate the equations that describe constrained mechanical systems is by introducing a (classical) Lagrange multiplier, denoted by λ. The resulting equations then read [7]:

$$M\ddot{x} = B + F_c + C^T\lambda\,, \tag{2.4a}$$

$$0 = P(x, t)\,. \tag{2.4b}$$

Where, for notational convenience, we have deleted (part of) the arguments. In classical mechanics the expression $C^T\lambda$ is identified with the constraint force. Note that the constraint force is added to an already given formulation. We refer to Eq. (2.4) as the classical formulation of constrained mechanical systems.

In order to find the time-evolution of the state of the system one usually has to take resort to simulation studies. The classical way to do this is first to solve the above equations for λ, second to discretize the resulting equations and third to solve these discrete equations by a numerical algorithm. This sequence of steps is known to exhibit stability problems. This can best be shown by performing simulations where the discrete equations are obtained in the

classical manner. Throughout this paper we will use a simple example, a planar pendulum shown in Fig. 1, to illustrate the concepts.

In Fig. 2 the results of such a simulation are depicted. The initial position and initial velocity are chosen such that the constraint equations are not exactly satisfied. The solid line represents a part of the unit-circle: the constraint. It can clearly be seen that the simulated behaviour (dashed line) does not satisfy the constraint and that the constraint violation

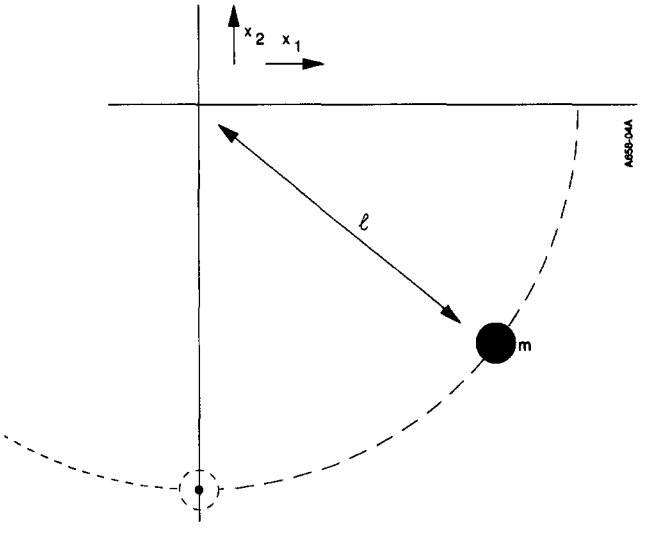

Fig. 1. A single planar pendulum modelled as a free falling point-mass that is constrained to a circle with radius *l*.

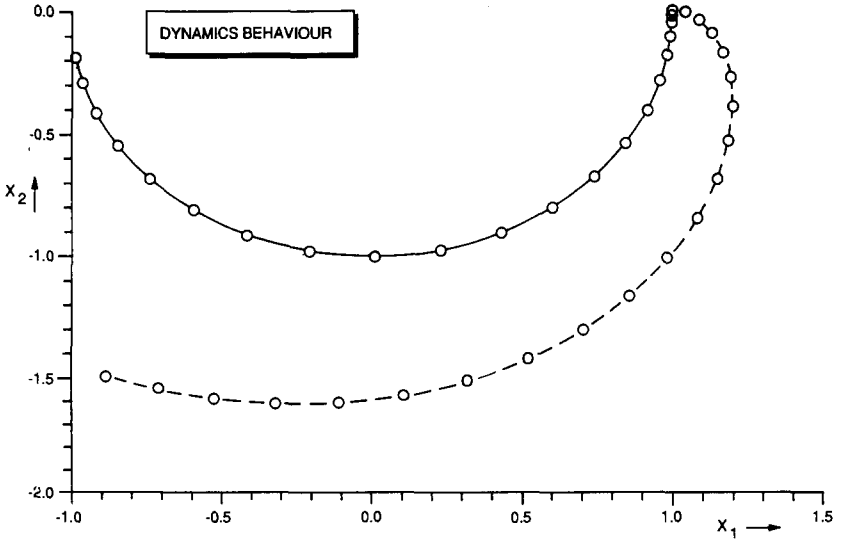

Fig. 2. The dynamics behaviour obtained from simulations performed in the classical manner (dashed line) and the model developed in this paper (solid line). The initial conditions do not satisfy the constraint equations. The solid line also represents the analytic solution. It can be seen that the results of the simulation performed in the classical manner are far from the analytic solution.

increases with time as well. (Details with respect to these simulation results are given in Section 5.)

The example shows that numerical simulation of the classical formulation of a constrained mechanical system indeed exhibits severe stability problems. It is very important, however, to treat the constraint equations in an accurate and reliable way. Simulation of multibody systems is an activity with a long history. As a result several computational procedures have been proposed to overcome the stability problems. These include techniques where a distinction is made between dependent and independent variables (a solution is sought through singular-value-decomposition), equilibrium correction strategies [8], penalty formulations [9], coordinate partitioning methods [10], predictor/corrector algorithms [11] and the differential algebraic approach [12]. Some remarks with respect to the differential algebraic approach can be found in the next section. In engineering practice, the constraint stabilization technique presented by Baumgarte [8], is often applied because it is conceptually simple and easy to implement. This technique can be derived by looking only at the constraint equations. So, the term constraint stabilization is well chosen. Differentiating Eq. (2.2) twice yields

$$\ddot{P}(x, t) = 0 . \tag{2.5}$$

From the control literature it is well known that a numerical solution of Eq. (2.5) can be unstable; that is, it can lead to values of P and $\dot{P}$ that are far from the desired value zero. From control theory it follows that the modified acceleration equation:

$$\ddot{P} + 2\alpha\dot{P} + \beta^2 P = 0 , \tag{2.6}$$

is (asymptotically) stable for $\alpha > 0$ [13]. (The additional terms in Eq. (2.6) can be seen to act as a proportional/derivative control with gains equal to 2α and β^2.) Baumgarte also presented the proportional/integral counterpart, well known from control theory, for the asymptotic stabilization of holonomic constraints [14].

One problem can readily be seen from the formulation of the stabilization technique: how to choose the coefficients α and β. Since the stabilization term can be interpreted as a proportional/derivative control law, it is noted that the use of the stabilization term shifts the poles of the system and thus alters the dynamical behaviour of the system. The choice of α and β is merely a matter of how fast we want to damp out the constraint violations. (Large values of α and β lead to high-gain feedback laws.) Note that the choice $\alpha = \beta$ yields a critically damped system. It is this choice that is commonly used when Baumgarte's technique is applied. In [15] the gains are related to the stepsize with which the numerical algorithm is applied. There it is remarked that this particular choice of gains tends to damp out constraint violations faster than any other choice, but accumulation of (integration) errors cannot be prevented. Furthermore, decreasing the stepsize results in larger gains. As a result the damping terms dominate the numerical solution process of Eq. (2.6): they make the system become numerically stiff. In spite of this drawback, the constraint stabilization technique is often applied since it avoids iterative solution of algebraic constraints and hence no additional computation time is necessary. This in contrast to for instance a predictor/corrector algorithm. These algorithms usually require an iteration process to obtain values within a certain predefined error level: a number of corrector steps must be applied. This iteration process is known to increase computation time considerably.

3. Stable numerical integration of dynamical systems described by differential algebraic equations: theory

3.1. Introduction

Up to now in a DAE approach difficulties with respect to stable numerical integration are related to the index of a DAE. An introduction to the notion of the index of a DAE can be found in [16]. Several definitions of the index exist, and general agreement is not yet reached with respect to the most appropriate one. However, one can say that mechanical systems often have the index-three property. So far index-three problems have resisted efforts at their direct solution by available ODE solvers. Most of the research with respect to numerical algorithms designed especially for DAE systems is dedicated to implicit integration methods, more specifically to Backward Differential Formulas [17] and to implicit Runge–Kutta methods [18, 19]. Hence finding the numerical solution involves an iteration process.

We show that there exists no need to introduce the index terminology to characterize difficulties with respect to numerical integration of DAEs. Hence we do not need to worry about which definition of index is best. In fact, we do not use the notion of index at all.

In order to demonstrate the theory here developed in a straightforward manner, and to prevent an iteration process from entering the discussion, we restrict ourselves to numerical integration by well-known explicit ODE-solvers. It can be seen, however, that the results are also valid in case implicit ODE-solvers are used.

To show where the actual stability problem originates we will discuss a constrained dynamical system, described by the following DAE:

$$G(x)\dot{x} = B(x) + F\,, \tag{3.1a}$$

$$0 = P(x, t)\,. \tag{3.1b}$$

Here G and B are assumed to be known. F represents the control. Throughout this paper we assume that the controls take their value in $\mathbf{R}^n$, notation $F \in (\mathbf{R}^n)^T$. Note that we allow non-linear system equations. In addition we make the following assumption.

ASSUMPTION A′
$\forall x \in \mathbf{R}^n$: $G(x)$ is non-singular.

This assumption can be seen to be a relaxation of assumption A, and implies that we are still dealing with a true ODE in Eq. (3.1a). We consider the system in Eq. (3.1) as the given problem formulation: no additional variables are added later on (as done in case of the constraint force in the classical formulation of mechanical system).

3.2. Formulation of dynamical systems described by a differential algebraic equation

Given a true ODE (that satisfies a Lipschitz condition) and arbitrary initial conditions a unique solution exists. However, in the case of constrained mechanical systems there may not be a trajectory that also satisfies the constraints without applying a control law, even if the set of initial conditions is consistent with these constraints. In [7] we showed that the

freedom offered by control can be used to formulate the equations that describe constrained dynamical systems. For this we first introduce the following notation:

$$S := \{(x, t) \in \mathbf{R}^{n+1} \mid P(x, t) = 0\} .$$

S defines the manifold to which the dynamical behaviour of the system is (to be) restricted. The point we want to emphasize here is that we look upon a constrained dynamical system as a dynamical system that was originally free and that is, given the equations that describe the free behaviour, subsequently restricted in some way by constraints. This indicates, in our point of view, a restriction to the control that can be applied to the dynamical system. The relation between the control F, the trajectory that results if F is applied to the dynamical system in Eq. (3.1a), and the manifold S must now be exploited. The result will be a number of equivalent formulations of the dynamical system given in Eq. (3.1). As in section 2 we denote the partial derivative of P with respect to x by the matrix C. We first give the following

DEFINITION 3.1 (applicable control) [7]. Let $G(x)$ and $B(x)$ be given matrices. Then a control F is called applicable if and only if from $(x(t_0), t_0) \in S$ and $G\dot{x} = B + F$ it follows that $(x(t), t) \in S$, $\forall t \geqq t_0$.

Details with respect to this definition can be found in [7]. For the present it suffices to remark that with the aid of Definition 3.1 a number of equivalent formulations can be given.

THEOREM 3.2 (formulation of dynamical systems described by a DAE) [7]. *Let*

(*a*) *Assumptions A' and B hold,*

(*b*) *the matrix* $Z(x, t)$ *be such that* $\forall (x, t) \in S$: $rank(CZC^T) = rank(C)$, *and*

(*c*) x_0 *and* t_0 *be any vectors such that* $(x_0, t_0) \in S$.

Then the following formulations of a dynamical system with equality state-space constraints are equivalent:

(*i*) $\exists F \in (\mathbf{R}^n)^T$ *such that* $x(t)$ *is a trajectory of the constrained dynamical system in Eq. (3.1) with* $x(t_0) = x_0$,

(*ii*) $\exists F_c \in (\mathbf{R}^n)^T$ *such that* $x(t)$ *is a trajectory of the dynamical system*:

$$G\dot{x} = B + F_c + GZC^T\lambda_Z ,$$

$$\lambda_Z = -(CZC^T)^{-1}(CG^{-1}(B + F_c) - d) ,$$

with $x(t_0) = x_0$,

(*iii*) $\exists F_c \in (\mathbf{R}^n)^T$, $\exists \lambda_Z \in (\mathbf{R}^m)^T$ *such that* $x(t)$ *is a trajectory of the dynamical system*:

$$G\dot{x} = B + F_c + GZC^T\lambda_Z ,$$

$$0 = P(x, t) ,$$

with $x(t_0) = x_0$.

The proof of Theorem 3.2 can be found in [7].

The parameter Z can be used for optimization purposes and is referred to as a weighting matrix. We will call λ_Z a generalized Lagrange multiplier. This name is motivated by the

Lagrange multiplier that is present in the classical formulation of mechanical systems. In our case, however, the Lagrange multiplier also depends on the parameter Z. With the aid of the above theorem we now show the actual origin of the stability problems (as discussed in Section 2).

3.3. Solving differential algebraic equations, or, how simple things can go wrong

When the time-evolution of a DAE is sought one usually must apply a numerical algorithm. Since we are dealing with numerical integration routines we must make the following assumptions.

ASSUMPTION B′
$(\exists \varepsilon > 0)$ such that $(\forall \delta$ with $0 \leqq \delta \leqq \varepsilon)$: $\forall (x, t)$ with $|P(x, t)| \leqq \delta \Rightarrow \operatorname{rank}(C(x, t)) = m$, $m \leqq n$.

Assumption B′ reduces to assumption B for $\delta = 0$, and is necessary for a correct (numerical) formulation of our problem.

ASSUMPTION C
Let R denote the stability region of the numerical method under consideration. Let $\sigma(.)$ denote the collection of eigenvalues of the controlled system. Then $\forall \nu \in \sigma(.)$, choose Δt such that $\nu \, \Delta t \in R$.

Since all continuous formulations in Theorem 3.2 are equivalent one can choose any one of them to find the time-evolution of the DAE. In the formulation as given in Theorem 3.2(ii) the constraint equations are not explicitly present. Indeed, it is this formulation (with $Z = G^{-1}$) that is commonly used to obtain a discrete set of equations. Since the constraints are not explicitly present in this formulation it seems to be a good basis in our search for a numerical solution. Unfortunately, this is not the case as will be demonstrated next.

Consider the case where one applies the Forward–Euler integration method to the formulation in Theorem 3.2(ii) in order to find the time-evolution of a constrained dynamical system. Then one can state:

$$x_{n+1} = x_n + \Delta t(G^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}) , \tag{3.2a}$$

with

$$\lambda_{nZ} := -(C_n Z_n C_n^T)^{-1}(C_n G_n^{-1}(B_n + F_{cn}) - d_n) . \tag{3.2b}$$

However, it can be shown that this combination leads to error accumulation once an error is made in the calculation of λ_{nZ}, or if one starts with initial conditions that are not on the manifold S. This will be proven next.

Let ε_n denote the error made in the calculation of λ_{nZ} at time t_n. Then the position constraint-violation at time t_{n+1} can be calculated. Notation: $P_n := P(x(t_n), t_n) := P(x_n, t_n)$. With neglect of higher-order terms we obtain:

$$
\begin{aligned}
P(x_{n+1}, t_{n+1}) &\overset{F-E}{=} P(x_n + \Delta t(G_n^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}),\ t_n + \Delta t) \\
&\overset{\text{Taylor}}{=} P_n + P_x\, \Delta t(G_n^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}) + P_{t,n}\, \Delta t \\
&= P_n + \Delta t\, C_n(G_n^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}) - \Delta t\, d_n \\
&\overset{(3.2b)}{=} P_n + \Delta t\, C_n G_n^{-1}(B_n + F_{cn}) - \Delta t\, d_n \\
&\quad - \Delta t\, C_n Z_n C_n^T((C_n Z_n C_n^T)^{-1}(C_n G_n^{-1}(B_n + F_{cn}) - d_n) - \varepsilon_n) \\
&= P_n + \Delta t\, C_n Z_n C_n^T \varepsilon_n\ .
\end{aligned}
$$

Of course there are other sources of error as well, for instance, finite word-length of the computer. The above derivation shows that, with the Forward–Euler integration method,

$$P_{n+1} = P_n + \Delta t\, C_n Z_n C_n^T \varepsilon_n\ . \tag{3.3}$$

Since this is a recursive formula it can be reduced to:

$$P_{n+1} = P_0 + \sum_{i=0}^{n} \Delta t\, C_i Z_i C_i^T \varepsilon_i\ . \tag{3.4}$$

From Eq. (3.3) we see that once an error is made in the calculation of the generalized Lagrange multiplier the solution is not on the constraint manifold. Let us examine what this means for fixed error level ε. For simplicity reasons we also assume C and Z to be constant matrices. In this special case we obtain from Eq. (3.4):

$$P_{n+1} = P_0 + tCZC^T\varepsilon\ . \tag{3.5}$$

Note that it makes no sense letting $\Delta t \to 0$. And when $t \to \infty$, for example because one is interested in an equilibrium solution, one has $P(x, t) \to \infty$. Even if no error is made in the initial conditions, i.e. the initial conditions are on the manifold S, the solution does not satisfy the constraint equation throughout the integration interval.

OBSERVATION 3.3

In the numerical solution of a constrained dynamical system two sources of error are noted:

(i) initial conditions. The initial conditions should be such that $(x(t_0), t_0) \in S$.

(ii) numerical errors. These include, amongst others, errors due to finite word-length, calculation errors and integration errors. As a special example of a calculation error we mention the possible error made in the calculation of the generalized Lagrange multiplier (as discussed above).

Each of these error sources can give rise to error accumulation. And if one is dealing with digital computers, one should keep in mind that machine zero ('0') is not identical to zero (0). As a result of this fact we have analytically (with A a linear operator),

$$(Ax = 0) \Leftrightarrow (A\dot{x} = 0 \text{ and } Ax(0) = 0)\ . \tag{3.6}$$

However, numerically we have no equivalence between their discrete counterparts (with Forward–Euler):

$$(Ax_{n+1} = \text{`0'}) \not\Leftrightarrow (Ax_{n+1} - Ax_n = \text{`0' and } Ax_0 = \text{`0'}) . \tag{3.7}$$

From Eq. (3.7) it follows that, even if the initial conditions are correct, one cannot guarantee that at the next time-step the solution satisfies all equations. Of course, in the case where the initial conditions are incorrect surely no such guarantee can be given. This implies that one should treat the algebraic equations in an implicit manner. Furthermore, in mechanical systems with position constraints the initial values of dependent velocities must also be stated correctly. This type of initial condition is referred to as hidden initial condition. One can find all hidden initial conditions analytically. However, if one starts with a large number of systems, and hence one can introduce a large number of independent constraints, one has to find and state a large number of initial conditions. In real world applications starting conditions may have been obtained from sensor information. Usually, sensor information contains noise, i.e. the information is only accurate to a certain degree. An important conclusion therefore is that it is often not possible to obtain exact initial conditions; approximate values are the best one can get. Although use of exact initial conditions is necessary to obtain a correct analytical solution, for the numerical solution however, one has to deal with perturbations. Consequently, any (numerical) solving procedure should be sufficiently robust with respect to deviations from the values of the exact (hidden) initial values, and by Observation 3.3(ii), it should also be stable with respect to errors made during the integration process.

3.4. Solving differential algebraic equations, or, how problems are avoided

Careful examination reveals that once an error is made in the solution for the generalized Lagrange multiplier error amplification cannot be prevented if Eq. (3.2b) is used. In fact, one could say that error accumulation results from the use of incorrect formulas. Equation (3.2b) alone is simply not the correct discrete formula to prevent error accumulation. Fortunately we can state a remedy. Starting with the appropriate formulation we obtain a numerical method that has the property that it is robust with respect to errors in the initial conditions (Observation 3.3(i)), and that it is stable with respect to errors made during numerical integration (Observation 3.3(ii)) [20]. Furthermore, this numerical method does not involve any (additional) iteration process.

Opposed to the sequence of steps that is applied in solving a DAE in the classical manner, as outlined in the previous section, we propose a different sequence. This sequence of steps reads:

- start with the appropriate set of continuous equations. (The meaning of the word 'appropriate' will be explained shortly);
- discretise the set of continuous equations;
- solve the discrete set of equations for the Lagrange multiplier;
- combine and solve the resulting equations.

Note that, in contrast to the classical way, in this sequence the equations are solved for the Lagrange multiplier only after they have been discretised. We remark that this sequence of steps can be applied successfully to more problems [21].

Assume, for simplicity reasons, that the matrices C, d, and Z are constant. Consider the discrete DAE, obtained from Theorem 3.2(iii), by application of Forward–Euler:

$$x_{n+1} - x_n = \Delta t(G_n^{-1}(B_n + F_{cn}) + ZC^T\lambda_{nZ}) , \tag{3.8a}$$

$$0 = P_{n+1} . \tag{3.8b}$$

Note that we now treat the constraint equation in an implicit manner.

The idea now is to obtain a discrete formula for λ_{nZ} directly from Eq. (3.8), rather than using the continuous expression in Theorem 3.2(ii). To distinguish the Lagrange multiplier in Eq. (3.2b) from the discrete generalized Lagrange multiplier here derived we denote the latter by λ_{nZ}^d. This implicitly means that the expression for the discrete generalized Lagrange multiplier depends on the integration method used to find the time-evolution of a DAE. However, we will show that one single expression for the discrete generalized Lagrange multiplier can be used in combination with a number of different integration methods.

PROPOSITION 3.4 (discrete generalized Lagrange multiplier). *Let assumption A', B' and C hold. Let the matrices C, d and Z be constant. And let the matrix $Z(x, t)$ be such that $\forall(x, t) \in S$: $rank(CZC^T) = rank(C)$. Then with the Forward–Euler integration method the discrete generalized Lagrange multiplier λ_{nZ}^d takes the following form*:

$$\lambda_{nZ}^d = -(CZC^T)^{-1}(CG_n^{-1}(B_n + F_{cn}) - d + P_n/\Delta t) , \tag{3.9}$$

Proof. From Eq. (3.8a) we obtain: $x_{n+1} = x_n + \Delta t(G_n^{-1}(B_n + F_{cn}) + ZC^T\lambda_{nZ}^d)$. Substitution of this equation in Eq. (3.8b) gives

$$\begin{aligned} P_{n+1} &= P(x_n + \Delta t(G_n^{-1}(B_n + F_{cn}) + Z_nC^T\lambda_{nZ}^d),\ t_n + \Delta t) \\ &= P(x_n, t_n) + P_x\,\Delta t(G_n^{-1}(B_n + F_{cn}) + Z_nC^T\lambda_{nZ}^d) + P_{t,n}\,\Delta t \\ &= P_n + \Delta t\, CG_n^{-1}(B_n + F_{cn}) + \Delta t\, CZC^T\lambda_{nZ}^d - \Delta t\, d . \end{aligned}$$

We demand that at t_{n+1} the constraint equation on position level holds, so from $0 = P_{n+1}$, it follows that $\Delta t\, CZC^T\lambda_{nZ}^d = -P_n - \Delta t\, CG_n^{-1}(B_n + F_{cn}) + \Delta t\, d$. Due to our assumptions CZC^T is non-singular, so $\lambda_{nZ}^d = -(CZC^T)^{-1}(CG_n^{-1}(B_n + F_{cn}) - d + P_n/\Delta t)$.
End of proof of Proposition 3.4. ■

If we compare Eq. (3.2b) with Eq. (3.9) we have

$$CZC^T\lambda_{nZ}^d = CZC^T\lambda_{nZ} - P_n/\Delta t . \tag{3.10}$$

As a result we have that the equivalent continuous formulations of Theorem 3.2 do not yield equivalent discrete formulations. This at first sight surprising result is the true origin of the stability problems discussed in Section 2. By use of the appropriate continuous formulation, i.e. the formulation in which the constraint is still present, stable numerical integration can be attained. In Section 3.3 we showed that with the classical Lagrange multiplier error accumulation cannot be avoided. Performing the same analysis that lead to Eq. (3.5) but now with λ_Z^d instead of λ_Z gives (for constant C, Z and ε)

$$P_{n+1} = \Delta t\, CZC^T\varepsilon\,.$$

No error accumulation can take place now. Indeed, if $\Delta t \to 0$ one has $P_{n+1} \to 0$, as desired.

Another interesting result of Proposition 3.4 is that the expression in Eq. (3.9) for the discrete generalized Lagrange multiplier λ_{nZ}^d, although it is derived with the Forward–Euler integration method and for linear constraints, is useful in combination with other integration routines and for non-linear constraints as well. Furthermore, it is not the accuracy of the derivation in Proposition 3.4 that is important. The mere solution of the discrete Lagrange multiplier on the interval $[t_n, t_{n+1}]$ from

$$CZC^T\lambda_Z^d = d - CG^{-1}(B + F_c) - P_n/\Delta t \tag{3.11}$$

in the set-points needed by the ODE-solver and the subsequent use of this Lagrange multiplier in combination with the characteristics of the ODE solver determines the accuracy of the numerical solution of the DAE.

Clearly Eq. (3.11) can be solved in a number of ways. For example, one can apply a singular value decomposition, one can invert the leading matrix, or one can solve the equation with an iteration process [22]. In the latter case the iteration process is ended once the desired accuracy is attained. Usually this desired accuracy is related to the order of the numerical integration routine used to obtain a solution for the ODE in Eq. (3.8a). We remark that these numerical integration routines yield only approximations of the analytical solution and are in this sense a source of errors by themselves. The most important observation we make here is that Eq. (3.11) can be solved for λ_Z^d with the accuracy one needs. In the extreme, one can solve the equation with 'machine accuracy'.

For future reference we identify the term $P_n/\Delta t$ as a compensation term. This name is motivated by the observation that P_n represents not only the value of the constraint but at the same time represents the constraint violation. So the presence of the term $P_n/\Delta t$ in Eq. (3.9) can be interpreted as 'compensation' of errors made at time t_n. Note however, that this compensation term is not added in a heuristic manner but instead follows from a precise mathematical derivation.

We would like to emphasize that the use of the discrete generalized Lagrange multiplier λ_{nZ}^d, i.e. including the compensation term, does not yield numerically stiff equations when the time step is reduced. This can be seen from Eq. (3.8a). In that equation the Lagrange multiplier is multiplied again with Δt. As a result, the term Δt in the denominator of the compensation term is cancelled. Note also that application of the discrete generalized Lagrange multiplier does not alter the dynamics of the system.

3.5. Stable numerical integration of dynamical systems described by differential algebraic equations

We now return to the case of nonlinear constraints, i.e. the case where the constraint Jacobian matrix C is state and time dependent. Let the continuous set of constrained equations of motions be given by:

$$G\dot{x} = B + F_c + GZC^T\lambda_Z\,, \tag{3.12a}$$

$$0 = P(x, t)\,. \tag{3.12b}$$

Due to the nonlinear character of the constraints we need one additional

ASSUMPTION D
The time-step Δt and the constraint Jacobian matrix C satisfy the conditions on the interval $[t_n, t_{n+1}]$, for all n, for all steps of the Newton–Raphson process to converge. These conditions can be found in [23].

We are pursuing an easy to verify mathematical condition that relates Assumption C and Assumption D. For real-world constraints, however, the time-step that validates Assumption C, usually also validates Assumption D.

THEOREM 3.5 (stable numerical integration of DAEs with Forward–Euler). *The system given by Eq. (3.12) can be solved in a stable manner with Forward–Euler if*:
(*i*) *assumptions A', B', C and D hold,*
(*ii*) *we define the discrete generalized Lagrange multiplier on the interval* $[t_n, t_{n+1}]$ *as in Eq. (3.11), and*
(*iii*) *the error in the initial condition is such that* $P_0 = O(\Delta t)$.
Furthermore, if the discrete generalized Lagrange multiplier is solved with sufficient accuracy, one has: $P(x_{n+1}, t_{n+1}) = \text{'0'} + O((\Delta t)^2)$.

Proof.

$$
\begin{aligned}
P_{n+1} &= P(x_{n+1}, t_{n+1}) = P(x_n + \Delta t(G_n^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}^d),\ t_n + \Delta t) \\
&= P(x_n, t_n) + C_n\, \Delta t(G_n^{-1}(B_n + F_{cn}) + Z_n C_n^T \lambda_{nZ}^d) - d_n\, \Delta t + O((\Delta t)^2) \\
&= P_n + \Delta t\, C_n G_n^{-1}(B_n + F_{cn}) + \Delta t\, C_n Z_n C_n^T \lambda_{nZ}^d - \Delta t\, d_n + O((\Delta t)^2) \\
&\quad \text{(assuming the Lagrange multiplier is solved with machine accuracy)} \\
&= P_n + \Delta t\, C_n G_n^{-1}(B_n + F_{cn}) + \Delta t\, C_n Z_n C_n^T (C_n Z_n C_n^T)^{-1}(d_n - C_n G_n^{-1}(B_n + F_{cn}) \\
&\quad - P_n/\Delta t) - \Delta t\, d_n + \text{'0'} + O((\Delta t)^2) = P_n - P_n + \text{'0'} + O((\Delta t)^2) \\
&= \text{'0'} + O((\Delta t)^2)\,.
\end{aligned}
$$

End of proof of Theorem 3.5. ■

Assumption (iii) (with the notation adapted from [24]) is to validate the Taylor approximation in the proof. What if this assumption does not hold and the initial constraint violation is significant? This case is also covered by the use of the discrete generalized Lagrange multiplier.

Evaluation of P_{n+1} using the formulas for Forward–Euler and λ_{nZ}^d yields: $P_{n+1} = P(x_n - Z_n C_n^T (C_n Z_n C_n^T)^{-1} P_n + O(\Delta t))$. In this formula the first step of a Newton–Raphson iteration process can be recognized. This is where Assumption D explicitly comes into focus. For other ODE-solvers the situation is more complicated as intermediate points enter the discussion. Simulation results, however, clearly show that even for large initial violations the error is largely reduced within a few time-steps in case of a higher-order ODE-solver (see Section 5). Since with linear constraints no Taylor approximation is necessary, only one integration step is needed to reduce the constraint violation. And, if desired, for the discrete solution accuracy up to machine zero can be attained, independent of the applied ODE-solver! Conditions under which convergence of the numerical algorithm is attained, given

incorrect initial values, is an ongoing research item. Use of the discrete generalized Lagrange multiplier and a standard ODE-solver yields numerical methods that offer error reduction and avoidance of error accumulation in one single algorithm.

Use of the discrete generalized Lagrange multiplier as solved from Eq. (3.11) yields, in the case of nonlinear constraints, a discrete solution of the DAE that satisfies the nonlinear constraint equation up to a certain accuracy. This means that if some derivative of the constraint equation is linear, say the kth, one can take a kth order ODE-solver in order to obtain $P_{n+1} = \text{'0'}$. Application of a lower-order numerical method, say of order p, still yields $P_{n+1} = \text{'0'} + O((\Delta t)^{p+1})$. This observation, of course, only holds for the discrete solution. In the linear case the solution x_{n+1} is such that $P_{n+1} = \text{'0'}$. The difference between the discrete solution and the exact solution is still related through the order of the truncation error of the ODE-solver.

In [20] theorems are presented for a number of explicit ODE-solvers, and we give

CONJECTURE 3.6 (stable numerical integration of DAEs with Runge–Kutta-k). The system given by Eq. (3.12) can be solved in a stable manner with a Runge–Kutta-k formula if:

(i) assumptions A′, B′, C and D hold,
(ii) we define the discrete generalized Lagrange multiplier on the interval $[t_n, t_{n+1}]$ as in Eq. (3.11), and
(iii) the error in the initial condition is such that $P_0 = O(\Delta t)$.

Furthermore, if the discrete generalized Lagrange multiplier is solved with sufficient accuracy, one has: $P(x_{n+1}, t_{n+1}) = \text{'0'} + O((\Delta t)^{k+1})$ after k steps.

For $k = 2$ and $k = 4$ the proof can be found in [20]. Also the case of explicit multi-step methods is discussed in [20].

Summarizing our results so far: for explicit integration methods the expression for the discrete generalized Lagrange multiplier suffices to obtain a stable numerical algorithm. No need exists to stabilize the integration process any further.

4. Stable numerical integration of mechanical systems subject to equality state-space constraints: theory

The motion of a constrained mechanical system is to be restricted to the manifold defined by the constraint equations. We have demonstrated in Section 3 that one can use the freedom offered by the control to accomplish this in case of a DAE in combination with an algebraic constraint. The point we want to emphasize here is that a constrained mechanical system can be (re)formulated to fit into this framework. This gives the opportunity to use the theory developed in Section 3. This in turn leads to algorithms that allow stable numerical integration of the equations that describe constrained mechanical systems.

Recall the constrained equation of motions for a mechanical system with state-space constraints from Section 2:

$$M\ddot{x} = B + F\,, \tag{4.1a}$$

as the second-order ODE describing the mechanical system, and

$$0 = P(x, t)\,, \tag{4.1b}$$

as the constraint equation.

Our first goal is to find a continuous time description of constrained mechanical systems in which no elimination of dependent variables has taken place. In order to be able to compare the results derived in this section with the classical formulation in Eq. (2.4) we will denote the control by F instead of F_c as done in Section 2. An equivalent first-order description of the system in Eq. (4.1) is given by:

$$\dot{x} = y\,, \tag{4.2a}$$

$$M\dot{y} = B + F\,, \tag{4.2b}$$

$$0 = P(x, t)\,, \tag{4.2c}$$

$$0 = Cy - d\,. \tag{4.2d}$$

Recall that C, d, denote the partial derivative of P with respect to x, t, respectively. Note that in this formulation Eq. (4.2d) is redundant as it directly follows from (4.2a) and (4.2c). However, the presence of this equation is essential for our purposes.

We must redefine the constraint manifold S as:

$$S := \{(x, y, t) \in \mathbf{R}^{2n+1} \mid P(x, t) = 0 \text{ and } C(x, t)y = d(x, t)\}\,.$$

Next we present the (partial) analogue of Theorem 3.2 for constrained mechanical systems. The original and complete version can be found in [7].

THEOREM 4.1 (formulation of constrained mechanical system) [7]. *Let*

(*a*) *Assumptions A and B hold,*

(*b*) *X and Y be any matrices such that $\forall (x, y, t) \in S$, $\operatorname{rank}(CXC^T) = \operatorname{rank}(C) = \operatorname{rank}(CYC^T)$, and*

(*c*) *x_0, y_0, and t_0 be any vectors such that $(x_0, y_0, t_0) \in S$.*

Then the following formulations of a mechanical system with equality state-space constraints are equivalent:

(*i*) *$\exists F \in (\mathbf{R}^n)^T$ such that $(x(t), y(t))$ is a trajectory of the dynamical system in Eq. (4.2) with $(x(t_0), y(t_0)) = (x_0, y_0)$,*

(*ii*) *$\exists F_c \in (\mathbf{R}^n)^T$, $\exists \mu_X \in (\mathbf{R}^m)^T$ and $\exists \lambda_Y \in (\mathbf{R}^m)^T$ such that $(x(t), y(t))$ is a trajectory of the dynamical system:*

$$\dot{x} = y + XC^T\mu_X\,,$$

$$M\dot{y} = B + F_c + MYC^T\lambda_Y\,,$$

$$0 = P(x, t)\,,$$

$$0 = Cy - d\,,$$

with $(x(t_0), y(t_0)) = (x_0, y_0)$.

The proof of Theorem 4.1 can be found in [7].

The presence of λ_Y in formulation 4.1(ii) is not that surprising since in the classical formulation also a Lagrange multiplier is present, although less general since there is no free matrix Y (see Eq. (2.4)). However, the generalized Lagrange multiplier μ_X is new and no (simplified) analogous Lagrange multiplier is present in the classical approach. The question now is: how did we obtain this generalized Lagrange multiplier and how did we obtain the expression $XC^T\mu_X$? Again, this expression is not obtained in a heuristic manner but follows from a precise and non-trivial mathematical derivation. The actual derivation is beyond the scope of this paper and for details we refer to [7]. Still some remarks can be made. The basic idea is to first introduce an auxiliary control at position level. For this it is necessary to have a first-order formulation. Next we apply Theorem 3.2 and subsequently remove the degrees of freedom introduced by the control at position level. The resulting equations now contain the generalized Lagrange multiplier μ_X (with the free parameter X). From the proof of Theorem 4.1 it follows that analtyically $\mu_X = 0$. However, due to integration errors (see Observation 3.3), this is not true for the numerical solution. This can be used to compensate for these errors, and enables one to find the solution of constrained mechanical systems in a stable manner.

The idea, again, is to obtain expressions for the discrete generalized Lagrange multipliers by first discretizing the continuous set of constrained equations of motion. And if we treat the ODE in an explicit manner and the constraint equations in an implicit manner the following can be obtained. The expressions for the discrete generalized Lagrange multipliers on the interval $[t_n, t_{n+1}]$ become:

$$\mu_X^d = -(CXC^T)^{-1}(Cy - d + P_n/\Delta t)\,, \tag{4.3a}$$

$$\lambda_Y^d = -(CYC^T)^{-1}(CM^{-1}(B + F_c) + \dot{C}y - \dot{d} + (C_n y_n - d_n)/\Delta t)\,. \tag{4.3b}$$

We have obtained two compensation terms, namely $P_n/\Delta t$ in Eq. (4.3a) and $((C_n y_n - d_n)/\Delta t$ in Eq. (4.3b). It is these compensation terms that will prove to be essential to prevent error accumulation.

Next we state our main result with respect to stable numerical integration of mechanical systems with equality state-space constraints. We restrict ourselves to the numerical algorithms discussed in Section 3. Observe that the application of explicit one-step or multi-step methods decouples the underlying ODE into two discrete equations on each individual time interval. Hence they only interact in the set of discrete points and can be solved separately. For example, with the Forward–Euler integration method the set of discrete equations now reads:

$$x_{n+1} = x_n + \Delta t(y_n + X_n C_n^T \mu_{Xn}^d)\,, \tag{4.4a}$$

$$y_{n+1} = y_n + \Delta t(M_n^{-1}(B_n + F_{cn}) + Y_n C_n^T \lambda_{Yn}^d)\,, \tag{4.4b}$$

Furthermore, from Eq. (4.3) and Eq. (4.4) it follows that reducing the time-step does not lead to numerically stiff equations, in contrast to Baumgarte's stabilization technique (see Section 2). These observations are used in the following:

THEOREM 4.2 (stable numerical integration of constrained mechanical systems) [20]. *The system given by Theorem 4.1(ii) can be solved in a stable manner with*

– *Forward–Euler*,
– *Runge–Kutta*-2, *and*
– *Runge–Kutta*-4 *if*:
(*i*) *Assumptions A, B', C, and D hold,*
(*ii*) *we define the discrete generalized Lagrange multipliers as in Eq. (4.3) on the interval* $[t_n, t_{n+1}]$, *and*
(*iii*) *the errors in the initial conditions are such that* $P_0 = O(\Delta t)$ *and* $C_0 y_0 - d_0 = O(\Delta t)$.
Furthermore, if the discrete generalized Lagrange multipliers are solved with sufficient accuracy, after k steps

$$P(x_{n+1}, t_{n+1}) = \text{'0'} + O((\Delta t)^{k+1}), \quad \textit{and}$$

$$C_{n+1} y_{n+1} - d_{n+1} = \text{'0'} + O((\Delta t)^{k+1}),$$

with k the order of the applied numerical method.
The proof can be found in [20].

In Section 2 we saw that simulation of mechanical systems with position constraints can lead to unstable behaviour. By Theorem 4.2, simulation of constrained mechanical systems can be performed in a stable manner also in the case where the constraints are originally on position level. No stability problem occurs, and, also important, use of the ODE-solvers in Theorem 4.2 results in numerical methods that are also robust with respect to errors in the initial conditions.

In [20] also explicit multistep methods are shown to be useful.

The order of the applied numerical algorithm as predicted by the theory was also verified by simulation studies. The results can be found in the next section.

5. Stable numerical integration of mechanical systems subject to equality state-space constraints: simulation

In this section the simulation results are presented. In [20] a number of test cases are discussed. All test cases involve the simulation of a mechanical system with several constraints superimposed on this mechanical system. In this paper, due to space limitations, we will illustrate the theory with respect to stable numerical integration with the aid of a simple mechanical system: a single planar pendulum that consists of a point-mass m swinging on an inelastic rod of length l (see Fig. 1). A DAE description of this system can be obtained as follows. First we model the point-mass as a free-falling body. Let (x_1, x_2) denote the Cartesian coordinates of the point-mass. The equations of motion read

$$m\ddot{x}_1 = 0, \tag{5.1a}$$

$$m\ddot{x}_2 = -mg. \tag{5.1b}$$

Here g represents the gravity vector. The superimposed constraint equation now reads:

$$0 = x_1^2 + x_2^2 - 1. \tag{5.2}$$

This equation expresses that the motion of the point-mass is to be restricted to the unit circle, i.e. the length of the rod equals 1. In order to apply the theory one must have an equivalent first-order description. For this we introduce the notation: $\dot{y}_1 = x_1$, and $\dot{y}_2 = x_2$. With this notation, differentiation of the constraint equation results in:

$$0 = 2x_1 y_1 + 2x_2 y_2 \,. \tag{5.3}$$

Hence the constraint Jacobian matrix C equals $(2x_1\ 2x_2)$. Application of Theorem 4.1, with the control set to zero, and with $X = I$ and $Y = M^{-1}$, now yields:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + C^T \mu \,, \tag{5.4a}$$

$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} 0 \\ -mg \end{pmatrix} + C^T \lambda \,, \tag{5.4b}$$

and the constraint equations as in Eq. (5.2) and Eq. (5.3). Our claim of stable numerical integration by starting with the appropriate continuous formulation is demonstrated with the aid of these equations.

In the simulation studies described here the following combinations of Lagrange multipliers are used (with simplified notation):

(A) $\lambda = -(CM^{-1}C^T)^{-1}(CM^{-1}(B + F_c) + \dot{C}y - \dot{d})$, $\mu = 0$; the classical approach to simulation, and

(B) λ as in Eq. (4.3b) and μ as in Eq. (4.3a); the combination that makes full use of the theory here developed.

For the simulation studies use was made of the NAG library [25]. As ODE-solver a Runge–Kutta–Merson method, also known as Runge–Kutta–Fehlberg, was used. Inversion of the matrix $CM^{-1}C^T$ was performed in a direct manner. Unless stated otherwise all integration time-steps where chosen as $\Delta t = 1/100$ and the integration interval as [0, 1].

First we discuss the case where all initial conditions are in agreement with the constraint equations. Second, we introduce errors in the position constraint while the velocity constraint is still initially satisfied. Third, we treat the case where both the position and the velocity constraint are not initially satisfied.

We start with the case where all initial conditions are in agreement with the constraint equations. At time $t = 0$ we take $(x_1, x_2) = (1, 0)$ and $(y_1, y_2) = (0, 0)$. First we take combination (A), corresponding to the classical approach to simulation (but without additional stabilization as discussed in Section 2). The constraint violations are depicted in Fig. 3. The constraints on position level as well as on velocity level are satisfied with error level E-7. The oscillatory behaviour of the constraint violations is due to the oscillatory behaviour of the pendulum itself.

If we now compare the error level obtained from a simulation by the classical way (Fig. 3) and the error level obtained by using the theory here developed (Fig. 4) we can deduce the following. As far as the error in the velocity constraint is concerned: use of our formulation reduced the error. In Fig. 3 the velocity constraint violation varies between −0.4E-7 and 0.4E-7, whereas in Fig. 4 this violation varies between −0.2E-8 and 0.4E-8. Hence a factor 10 is attained. A more dramatic reduction is obtained for the position constraint violation. In

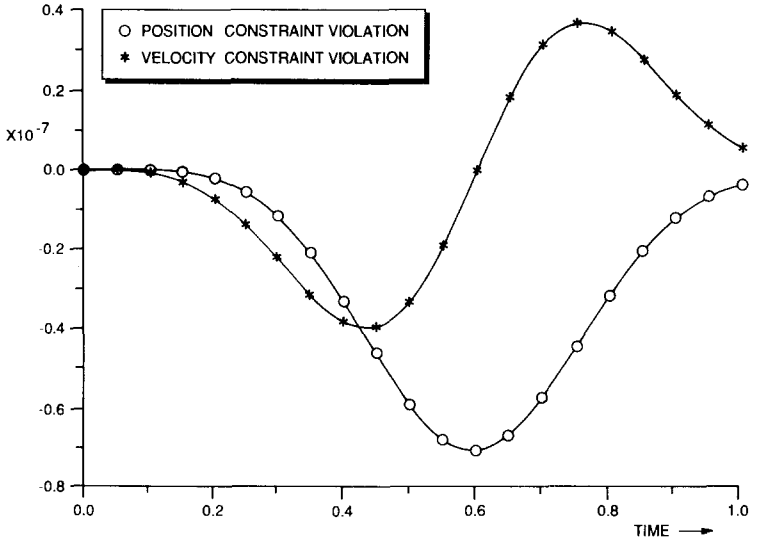

Fig. 3. Constraint violations when simulation is performed in the classical manner. The initial conditions satisfy the constraint equations.

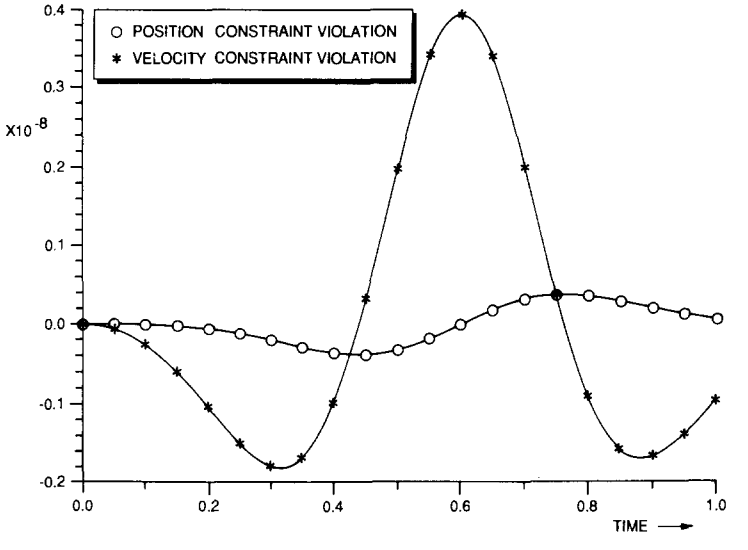

Fig. 4. Constraint violations when simulation is performed with the full model: both Lagrange multipliers and both compensation terms are used. The initial conditions satisfy the constraint equations.

Fig. 3 the position constraint violation varies between 0 and −0.7E-7. Whereas the situation depicted in Fig. 4 shows that the position constraint violation varies between −0.4E-9 and 0.4E-9. That is, we gained two orders with respect to the violation of the position constraint.

Second, we discuss the case where the initial conditions are not correct. We start with $(x_1, x_2) = (2, 0)$ and $(y_1, y_2) = (0, 0)$. So, the position constraint is not initially satisfied. This amount of violation may not be very likely in real world applications where initial conditions are obtained from sensor information. Here it serves to show the difference between

simulations performed in the classical manner and simulations performed with the full model as developed in this paper. In Fig. 5 the behaviour of the point-mass is depicted (dashed line) when simulation is performed with combination (A), i.e. the classical manner. No additional stabilization technique (see Section 2) is applied. Instead of being constrained to the unit-circle the point-mass is, and remains, on a circle with radius two: the error due to the initial conditions is not reduced. So correct initial conditions are essential in the classical approach to simulation as can also be seen from Eq. (3.5) and Observation 3.3. If we perform simulations with the full model as developed in this paper (combination (B)) we obtain the behaviour depicted in Fig. 5 as a solid line. Note that initial position is: $(x_1, x_2) = (2, 0)$. Apart from this initial error the point-mass is constrained to the unit circle and behaves as a pendulum. The accompanying constraint violations (apart from the first three set-points) are depicted in Fig. 6. Even though the position constraint is represented by a nonlinear equation, within the first integration step the initial error is greatly reduced. Apart from the first three time-steps the same error functions are obtained as those from a simulation with correct initial conditions, see Fig. 4.

We conclude this section with a test case in which the initial conditions violate the position constraint as well as the velocity constraint. As starting conditions we take: $(x_1, x_2) = (1, 1/100)$ and $(y_1, y_2) = (1, 1/10)$. Note that this implies that the violation with respect to the desired radius (1) is 1/10000, so the position constraint violation is indeed small. This test case was also used in Section 2 (Fig. 2) to show that a stability problem indeed exists when simulations are performed in the classical manner. With combination (A), we obtain the dynamics behaviour depicted as the dashed line in Fig. 2. Compared to the desired dynamics behaviour (the solid line) we conclude that not only the point-mass is not on the unit circle, but also that the position constraint violation increases with time. This test case shows why, when the classical solution method is applied, considerable effort is put in the

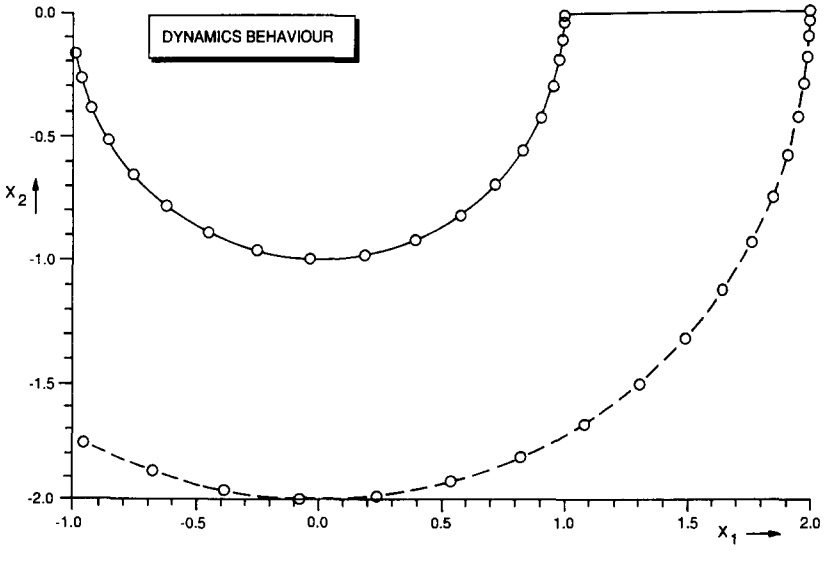

Fig. 5. The dynamics behaviour obtained from a simulation performed in the classical manner (dashed line) versus the dynamics behaviour obtained with the use of the model developed in this paper (solid line). The initial position of the point-mass equals (2, 0) and is not compatible with the constraint equation. The present model follows the correct constraint.

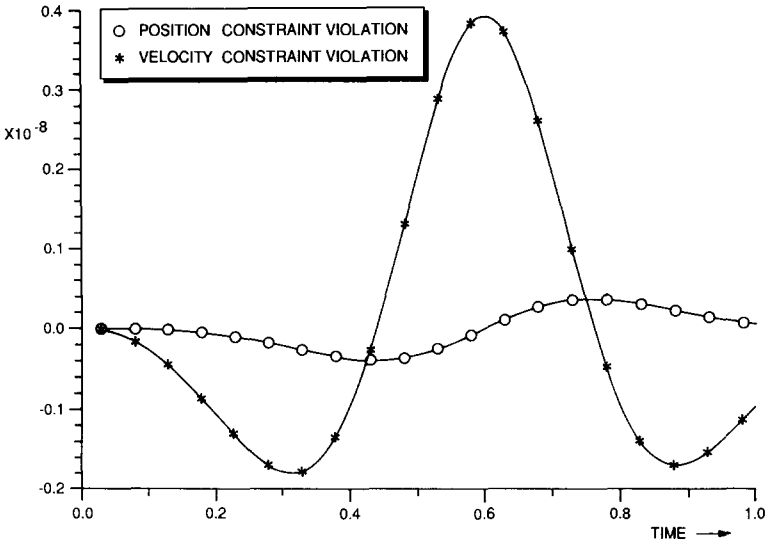

Fig. 6. Constraint violations when simulation is performed with the full model and with incompatible initial conditions (see Fig. 5). The constraint violations shown exhibit the same characteristics as those depicted in Fig. 4. In this figure it cannot be seen that we started with incorrect initial conditions.

search for stabilization methods. If one applies the theory developed here the improvement is clear. The dynamics behaviour is plotted as a solid line in Fig. 2, leaving out the first discrete set points. As can easily be seen, we now obtain the desired behaviour. The error functions are not plotted but are similar to those given in Fig. 6.

Table 1. Verification of the order of the numerical algorithm as predicted by the theory. The values within the rectangular boxes are close to the predicted value of 32.

Position constraint violation

Integration Step: 0.0625			0.03125		0.015625
T	Violation	./.	Violation	./.	Violation
0.000	0.0000E + 00	–	0.0000E + 00	–	0.0000E + 00
0.250	−0.1107E − 05	28.6	−0.3864E − 07	30.7	−0.1258E − 08
0.500	−0.4243E − 05	37.1	−0.1143E − 06	35.4	−0.3231E − 08
0.750	−0.2459E − 05	25.1	0.9800E − 07	29.4	0.3329E − 08
1.000	0.5951E − 06	31.7	0.1872E − 07	32.8	0.5716E − 09
1.250	−0.2402E − 08	5.2	−0.4639E − 09	22.7	−0.2047E − 10
1.500	−0.2182E − 05	30.2	−0.7227E − 07	31.5	−0.2297E − 08
1.750	−0.3002E − 05	47.2	−0.6356E − 07	42.9	−0.1483E − 08
2.000	0.2685E − 05	28.5	0.9417E − 07	30.9	0.3048E − 08

Velocity constraint violation

Integration Step: 0.0625			0.03125		0.015625
T	Violation	./.	Violation	./.	Violation
0.000	0.0000E + 00	–	0.0000E + 00	–	0.0000E + 00
0.250	−0.1357E − 04	30.7	−0.4426E − 06	31.7	−0.1397E − 07
0.500	0.4853E − 05	11.7	0.4155E − 06	24.5	0.1695E − 07
0.750	0.1598E − 04	59.6	0.2682E − 06	57.6	0.4659E − 08
1.000	−0.9525E − 05	32.6	−0.2919E − 06	32.7	−0.8928E − 08
1.250	−0.4239E − 06	16.3	−0.2605E − 07	26.3	−0.9900E − 09
1.500	−0.1903E − 04	33.3	−0.5714E − 06	33.3	−0.1719E − 07
1.750	0.2552E − 04	25.6	0.9943E − 06	29.8	0.3334E − 07
2.000	−0.1876E − 05	7.7	−0.2445E − 06	23.8	−0.1025E − 07

The simulations have been extended to over 100 periods, and no error amplification took place. We would like to emphasize that compared to the classical simulations, no additional computational effort is necessary to obtain these improvements.

The case where one starts with correct initial conditions was also used to verify the order of the constraint violation, as predicted by Theorem 4.2. Runge–Kutta–Merson uses a 4th order Runge–Kutta integration method. Hence by Theorem 4.2 we expect to see a factor 32, being 2^5, when we reduce the time-step Δt by a factor two and take the quotient of the constraint violations at the same discrete time-point. The results are depicted in Table 1. The factors within the rectangular boxes are clearly in agreement with the theory here developed.

In [20] also a pair of double pendulums, which allow for the interpretation of a pair of two-link rigid manipulators, was simulated. Several nonlinear constraints were superimposed upon the system. In [26] the theory here developed was applied to simulate a constrained (6-DOF) manipulator with gearbox flexibility. In each of these cases the simulation results were in agreement with the theory: no error accumulation took place and the correct dynamical behaviour was obtained.

6. Concluding remarks

We showed that the problems with respect to stable numerical integration of constrained dynamical systems originate from the transformation of a set of continuous equations to a set of discrete equations. The stability problem does not originate from the additional constraint equations themselves. A remedy was stated for the stability problem. This remedy involved solving the combination of differential and algebraic equations only after they have been discretized. It was shown that the resulting set of discrete equations differs from the set of equations obtained if the classical approach to constrained dynamical systems is applied.

Use of the discrete equations here derived, in combination with standard (explicit) ODE-solvers, such as Forward–Euler, Runge–Kutta and multistep methods, showed that stable numerical integration can be attained. Features of the method are that no additional iteration process is necessary, the computational effort is not increased, non-linear rheonomic constraints can be used, and decreasing the time-step does not yield numerically stiff equations. Furthermore, for linear constraints, constraint violations can be reduced up to machine accuracy, while for non-linear constraints the constraint violation is related to the characteristics of the ODE-solver that is used. The theory was extended to the case of mechanical systems with holonomic constraints. Also in this case no additional stabilization is necessary in contrast to the classical approach. In addition, the derived numerical method is robust with respect to errors in the initial conditions and is stable with respect to errors made during the integration process. Simulation studies were performed based on the theory developed here. All simulation results were in agreement with the theoretical results.

Acknowledgement

The author would like to thank Dr. R.J.P. Groothuizen and Prof. A.E.P. Veldman for their constructive remarks during the course of this work.

References

1. A.A. ten Dam and H. de Jong, Development environment for digital manipulator simulators. NLR TP 89030 (1989). National Aerospace Laboratory NLR, Amsterdam, The Netherlands.
2. L. Petzold, Differential/algebraic equations are not ODEs. *SIAM J. Sci. Stat. Comp.* 3 (1982) 312–321.
3. C.W. Gear and L.R. Petzold, ODE methods for the solution of differential/algebraic systems. *SIAM Journals of Numerical Analysis* 21(4) (1984) 716–728.
4. C. Fuehrer and B. Leimkuhler, Formulation and numerical solution of the equations of constrained mechanical motion. DFVLR Forschungsbericht 89-08 1989. Institut fuer Dynamic der Flugsysteme, Oberpfaffenhofen, Germany.
5. L. Petzold, A description of DASSL: a differential/algebraic system solver. In: *Proc. 10th IMACS World Congress*, August 8–13, Montreal (1982).
6. C.W. Hirt and F.H. Harlow, A general corrective procedure for the numerical solution of initial-value problems. *Journal Computational Physics* 2 (1967) 114–119.
7. A.A. ten Dam, Modelling dynamical systems with equality state-space constraints. NLR TP 89418 (1989). National Aerospace Laboratory NLR, Amsterdam, The Netherlands.
8. J. Baumgarte, Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanical Engineering* 1 (1972) 1–16.
9. E. Bayo, J. Garcia de Jalon and M.A. Serna, A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer Methods in Applied Mechanics and Engineering* 71 (1988) 183–195.
10. R.A. Wehage and E.J. Haug, Generalized coordinate partitioning for dimensional reduction in analysis of constrained dynamic systems. *Journal of Mechanical Design* 104 (1982) 247–255.
11. D. Sciacovelli, Flexible spacecraft: computer-oriented control analysis. In: *Systems and Control Encyclopedia*. Pergamon Press (1985) pp. 1649–1665.
12. C.W. Gear, G.K. Gupta and B.J. Leimkuhler, Automatic integration of the Euler–Lagrange equations with constraints. *Journal of Computation and Applied Mathematics* 12 & 13 (1985) 77–90.
13. W.A. Wolovitch, *Robotics: Basic Analysis and Design*. Holt, Rinehart and Winston, Inc. (1987).
14. J. Baumgarte, A new method of stabilization for holonomic constraints. *Journal of Applied Mechanics* 50 (1983) 869–870.
15. C.O. Chang and P.E. Nikravesh, An adaptive constraint violation stabilization method for dynamic analysis of mechanical systems. *Journal of Mechanisms, Transmissions, and Automation in Design* 107 (1985) 488–492.
16. S.E. Mattsson, On modelling and differential/algebraic systems. *Simulation* 52(1) (1989) 24–32.
17. K.E. Brenan and B.E. Engquist, Backward differential approximations of nonlinear differential/algebraic systems. *Mathematics of Computational Analysis* 51 (1988) 659–676.
18. M. Roche, Implicit Runge–Kutta methods for differential/algebraic equations. *Siam Journal of Numerical Analysis* 26(4) (1989) 963–975.
19. K.E. Brenan and L.R. Petzold, The numerical solution of higher index differential/algebraic equations by implicit methods. *Siam Journal of Numerical Analysis* 26(4) (1989) 997–1005.
20. A.A. ten Dam, Numerical solution of dynamical systems with equality state-space constraints. NLR TP 89149, National Aerospace Laboratory NLR, Amsterdam, The Netherlands.
21. A.E.P. Veldman, Missing boundary conditions? discretise first, substitute next, combine later. *SIAM J. Scient. Stat. Comp.* 11 (1990) 82–91.
22. A.E.P. Veldman, Viscous-inviscid interaction, partitioned dynamical systems and i(n)te(g)ration. NLR TP 89232 L, National Aerospace Laboratory NLR, Amsterdam, The Netherlands.
23. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. Springer Verlag (1979).
24. K.E. Brenan, S.L. Campbell and L.R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. New York: Elsevier Science Publishing Co., Inc. (1989) 210 pp.
25. *NAG FORTRAN Mini Manual*, Mark 12. Numerical Algorithms Group, March 1987.
26. M.J.H. Couwenberg and A.A. ten Dam, Dynamics motion simulation of the HERMES robotarm subject to constraints, NLR TP 90282, National Aerospace Laboratory NLR, Amsterdam, The Netherlands.