Raymond J. Spiteri

A Numerical Algorithm for the Solution of DAIs - Differential systems subject to Algebraic Inequality constraints

This paper describes a numerical algorithm for the solution of a Differential system subject to Algebraic Inequality constraints (DAI). The algorithm consists of a dynamic integration of the equations of motion coupled with invocation of a control mechanism. After describing various applications and solution approaches to DAIs, we concentrate on one approach based on the principle of Least Constraint. *This is achieved by minimization of a barrier function defined using buffer zones near the constraint boundaries. Determination of the buffer zones is done dynamically, corresponding to a local planning strategy. The control is chosen in a way resembling interior-point methods in optimization. This ensures that the solution trajectory is 'centred' in some sense within the feasible region.*

1. Introduction

Recently, much study has been directed towards the theory and numerical solution of differential systems with algebraic equality constraints (DAEs) for both initial and boundary value problems; see, e.g., [2,4,6]. In contrast, relatively little is known about general theory or numerical methods for differential systems subject to algebraic *inequality* constraints (DAIs). We consider initial value problems for DAIs of the form

$$\dot{\mathbf{q}} = \mathbf{f}(t,\mathbf{q}) + B(t,\mathbf{q})\mathbf{u} \tag{1a}$$

$$\mathbf{q}(0) = \mathbf{q}_0 \tag{1b}$$

$$\mathbf{c}(t,\mathbf{q}) \geq \mathbf{0} \tag{1c}$$

$$|\mathbf{u}| \leq \mathbf{u}_{max}, \tag{1d}$$

where $\mathbf{q}(t) \in \mathbb{R}^m$, $\mathbf{u}(t) \in \mathbb{R}^{m_{cntrl}}, \mathbf{c}(t,\mathbf{q}(t)) \in \mathbb{R}^{m_{cnstr}}$ for each t on an interval $0 \leq t \leq b$ [1]. The number of inequalities is in general different from the number of algebraic variables: usually $m_{cnstr} \geq m_{cntrl}$, but otherwise these quantities are unrelated. We suggest that the sparsity of the literature on DAIs is not a result of lack of practical applications which give rise to such problems. Rather, we attribute this more to considerations such as that DAIs, compared to DAEs, require a less homogeneous treatment. We note that the choice of algebraic variables $\mathbf{u}(t)$ to satisfy the inequality constraints in (1) is nonunique in general. Despite being similar in appearance, inequalities (1d) are treated differently from inequalities (1c). The variables $\mathbf{u}$ in (1d) are 'independent' parameters of the problem to be determined by the algorithm, while the relations (1c) are quantities that must be preserved through this choice of $\mathbf{u}$. We assume that the *feasibile region* $D(t)$, consisting of all trajectories $\mathbf{q}(t)$ satisfying (1), is a connected set with a nonempty interior in $\mathbb{R}^m$. We note that the interpretations of $\mathbf{u}$ as Lagrange multiplier and B as $(\frac{\partial B}{\partial \mathbf{q}})^T$ are not valid as they often are in the DAE case.

2. Examples

Example 1. *Consider a robot arm moving under gravity, as depicted in Figure 1. Suppose that the 'hand' (x_2, y_2) is required to follow a prescribed path, such as a parabola in the Cartesian plane. This leads to an equality constraint and, when combined with the ODEs describing the dynamics, a DAE. Now if the hand is only required to stay in a certain region, say in between two parabolas in the Cartesian plane, then this leads to a DAI of the form (1). (A similar example results when requiring a vehicle to stay on a stretch of road, rather than prescribing precisely what path on the road it should follow.)*

Example 2. *The model is based on the Dynamite multiple mobile robot testbed developed by the Dynamo project in our laboratory [3]. Figure 2 shows the actual testbed and a schematic. The equations of motion of the robot may be written as*

$$\dot{x} = v\cos\theta; \qquad \dot{y} = v\sin\theta; \qquad \dot{\theta} = v\frac{\tan\psi}{L} := v\alpha; \qquad \dot{v} = a - \gamma v, \tag{2}$$

[1] The notation $\mathbf{v} \geq \mathbf{0}$ is understood to hold componentwise.

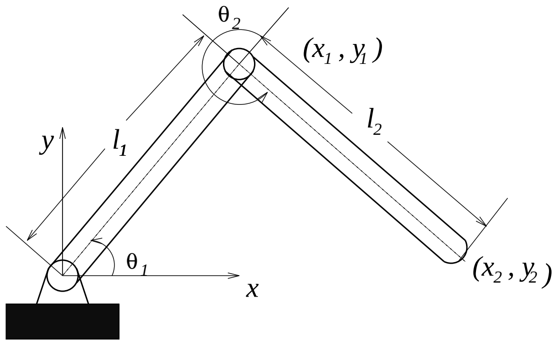
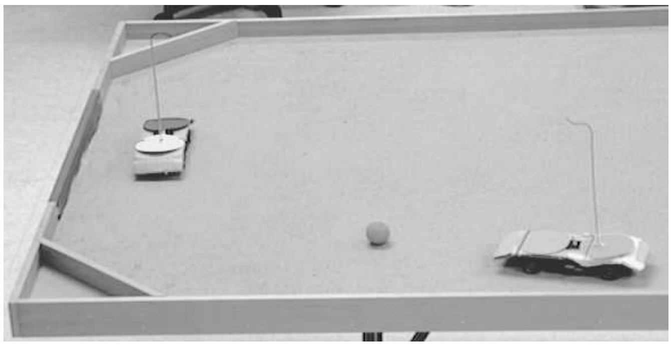


Figure 1: Robot arm



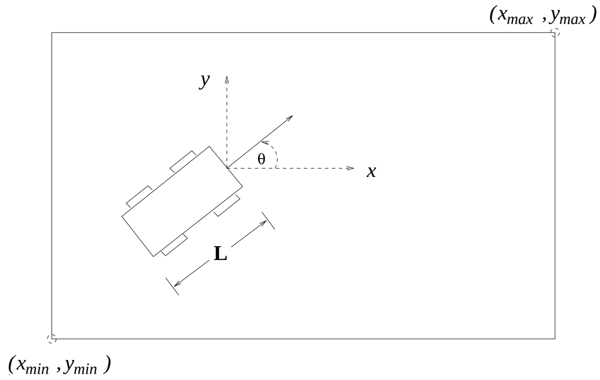


Figure 2: Dynamite mobile robots playing soccer and schematic

where $\gamma \geq 0$ *is a damping parameter* ($\gamma = 0$ *is the no-friction limit)* [2]*. Note the use of* α *as a control variable instead of the steering angle* ψ *in order to cast (2) in the form (1a). Thus system (2) is of the form (1a), with* $\mathbf{q} = (x, y, \theta, v)^T$*. The inequality constraints (1c) are taken to be*

$$\mathbf{c} = (x - x_{min}, x_{max} - x, y - y_{min}, y_{max} - y, v_{max} - v, v_{max} + v)^T \tag{3}$$

with an additional (driving) constraint to be added. The bounds on the controls are $|a| \leq 150\,\mathrm{cm\,s^{-2}}$; $|\alpha| \leq 1/25\,\mathrm{cm^{-1}}$*, representing maximum allowed throttle and reciprocal of minimum turning radius respectively.*

In both examples above, it is clear that the domain of feasible solutions (trajectories) has a nonempty interior. If the given problem has equality constraints specified in addition to (1c), then we treat them as defining an invariant manifold (see, e.g., [1,5] and references therein). There are many other applications involving ODEs and inequalities, the solutions of which do not quite fit the form we pursue here. These include ODEs with inequality invariants, where the *unique* solution of an ODE problem also satisfies some inequalities (e.g., non-negativity of mass).

3. Solution approaches

1. Explicit approaches: The simplest approach is to explicitly (and often arbitrarily) specify a trajectory and determine the forces needed to execute it, as in an inverse dynamics problem.

2. Implicit approaches: The most (mathematically) elegant approach is to specify a goal, possibly determining a unique solution, and solve the subsequent motion planning problem (see [7] for a survey).

3. In-between approaches: This type of approach is obtained using *programmed constraints*. These are artificial, user-defined constraints used as an executable specification language for the desired behavior of a robot (e.g. [8,9]). This approach allows for *local*, initial-value-like time integration strategies. Because of this it is more computationally tractable and more flexible than implicit approaches, while at the same time being more declarative than explicit approaches.

[2]For a more precise treatment, the ψ-dynamics and the rotational dynamics of the car would also be modeled.

Work done on DAIs has mostly concentrated on the active-set strategy, namely to make *active* constraints which are about to be violated. In an *active-set* algorithm, the set of active constraints is maintained as a function of time, introducing (equality) constraint forces by constructing B from the transpose of the active rows of $\frac{\partial \mathbf{c}}{\partial \mathbf{q}}$. In this case, $\mathbf{u}$ are Lagrange multipliers. When the presence of constraint forces is required, this may be the only way to proceed. But the common phenomenon of constraints frequently hopping in and out of the active set complicates matters. Moreover, when B does not have a full column rank, (like for the mobile robot example when $v = 0$.) this approach may not be possible. The Least Constraint approach [8] attempts to find a trajectory inside the feasibility region, away if possible from the boundary. This latter approach is more natural for robotics. We present an algorithm for solving DAIs of the form (1) based on the Least Constraint approach coupled with *local planning*, since it is intuitively clear that a lack of planning will inevitably lead to reachability problems.

4. Required Properties of a DAI Solver

- *Small computational expense* in determining the control.
- *Robustness* eg. to varying or difficult geometries.

Completeness is also a desirable property as well, but only in the sense of being able to *choose* a solution from the many that exist. A more complete discussion of this can be found in [10].

5. A DAI Solver for Robot Control

We assume that system (1) can be defined in such a way that maintaining $\mathbf{u}$ at some prescribed function $\mathbf{u}_{nat}$ leads to a 'natural' or otherwise desirable behaviour. For instance, in the mobile robot example, different values of the damping parameter γ lead to different interpretations of this normalization. The basic strategy is to proceed with natural motion until a constraint is deemed to be 'violated'. By this we mean that a previously defined buffer zone has been penetrated. The control function $\mathbf{u}(t)$ is taken to be piecewise constant on each open interval (t_{n-1}, t_n) with $\mathbf{u}_{nat}$ as the default. Here is the overall algorithm for advancing one time step, from t_{n-1} to t_n. A more detailed discussion of the various modules can be found in [10].

Starting with feasible $(t_{n-1}, \mathbf{q}_{n-1})$; buffer zone boundary $\mathbf{c}_{safe}$ and sensitivity parameter U_{frac} previously defined.

1. Predict safety of proceeding with $\mathbf{u} = \mathbf{u}_{nat}$.
2. If *weak approximate safety* then
   (a) Proceed with $\mathbf{u} = \mathbf{u}_{nat}$.
   (b) If resulting $(t_n, \mathbf{q}_n)$ feasible then accept $\mathbf{q}_n$, advance n and exit to step 5; else reduce U_{frac} and continue.
3. Take a control step starting from $(t_{n-1}, \mathbf{q}_{n-1})$.
4. If no feasible control then stuck; else
   (a) For each $u_j = u_{max,j}$ reduce $U_{frac,j}$
   (b) If $\mathbf{u} = \mathbf{u}_{nat}$, increase U_{frac},
   (c) Accept $\mathbf{q}_n$, advance n and exit to step 5
5. If $\mathbf{u}_{n-1} = \mathbf{u}_{nat}$ then update $\mathbf{c}_{safe} = \mathbf{c}(t_{n-1}, \mathbf{q}_{n-1})$.

6. A Numerical Simulation

A driving constraint $c_7 = r^2 - (x - x_c)^2 - (y - y_c)^2$ is used, representing a circle (spotlight) with centre at (x_c, y_c) and radius r. We use a parabolic spotlight path passing through the initial position of the robot $(x(0), y(0)) = (5.27, 50)$, the point $(x_{min}, y(0))$, and the corner of the table (x_{max}, y_{max}). The path is parameterized as $x_c = v_w t + x(0); y_c = ax^2 + bx + c$, where v_w is the x-component of the spotlight velocity and a, b, c are constants uniquely determined from the other criteria previously described. In Table 1, c_{freq} is the number of steps between constraint samplings, u_{frac} satisfies $U_{frac}(0) = u_{frac} I$, $\%\mathbf{u}$ is the percentage of steps in which a control was computed, $\%\mathbf{u}_{max}$ is the fraction of

v_w	r	γ	c_{freq}	u_{frac}	$\%\mathbf{u}$	$\%\mathbf{u}_{max}$ (a)	(α)	CPU
1	10	0	2	10^{-2}	23.25	0.0	39.02	26.25
1	10	$\frac{1}{6}$	2	10^{-2}	49.21	0.0	0.04	29.84
1	1	0	2	10^{-2}	46.43	0.0	2.18	18.04
1	1	$\frac{1}{6}$	1	10^{-2}	99.99	0.0	10.42	74.37
5	5	0	2	0.5	47.01	0.0	11.09	5.36
5	5	$\frac{1}{6}$	1	0.5	99.85	0.07	40.95	35.11

Table 1: Simulation results: Spotlight Problem

control steps which used the maximum value allowed for the control, and CPU represents the amount of processor time required to complete the simulation on an SGI Indy with an R4600 chip. The final time for the motion is given by $t_f = (x_{max} - x_{min})/v_w$. For this example, it was possible to produce various qualitatively different behaviours. Some solutions look as if they could have been produced by an active-set method: the robot slides along the 'trailing' part of the boundary. In other cases, most notably for high $\%\mathbf{u}$, the robot stays very close to the centre. It is clear that generally the system does not have enough controls to satisfy the $r = 0$ case (nor was the algorithm designed to handle such a case) for finite energy, nonimpulsive control inputs. In the presence of discretization errors coupled with imperfect constraint prediction and measurement, the radius for which this system will be unable to track the spotlight will be nonzero in practice. Hence it is possible to expect problems with sufficiently small r.

7. Conclusions

We have described a class of problems known as DAIs and outlined an algorithm designed primarily to solve programmed-constraint applications in robot programming. Our simulation results to date indicate that the method is efficient, both in deciding when to use a control as well as then performing the ensuing calculation. For the near future, we plan to concentrate on testing the algorithm with the real mobile robots in our laboratory, further improving the efficiency and robustness of the method, and attempting to solve non-robotic-programming applications.

Acknowledgements

The results presented in this paper are based on joint work with Uri M. Ascher *and* Dinesh K. Pai, *Department of Computer Science, University of British Columbia, Vancouver, Canada.*

8. References

1 U. Ascher, H. Chin and S. Reich: Stabilization of DAEs and invariant manifolds. Numer. Math. **67** (1994), 131–149.
2 U. Ascher and Raymond J. Spiteri: Collocation software for boundary-value differential- algebraic equations. SIAM Journal on Scientific Computing **15**(4) (July 1994) 938–952.
3 R. Barman, S. Kingdon, J. J. Little, A. K. Mackworth, D. K. Pai, M. Sahota, H. Wilkinson, and Y. Zhang: Dynamo: real-time experiments with multiple mobile robots. Proceedings of the IEEE Intelligent Vehicles Conference, Tokyo, Japan, (July 1993), 261–266.
4 K. Brenan, S. Campbell and L.R. Petzold: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. North-Holland (1989).
5 H. Chin: Stabilization Methods for Simulations of Constrained Multibody Dynamics. Ph.D. Thesis, University of British Columbia, Vancouver, Canada (1995).
6 E. Hairer, S.P. Nørsett and G. Wanner: Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer Verlag (1991).
7 J. C. Latombe: Robot Motion Planning. Kluwer, 1991.
8 Dinesh K. Pai: Least constraint: A framework for the control of complex mechanical systems. Proceedings of the American Control Conference, (1991) 1615–1621.
9 Dinesh K. Pai: Robot programming and constraints. First Workshop on Principles and Practice of Constraint Programming, Newport, RI (April 1993) 250–257.
10 Raymond J. Spiteri, Uri M. Ascher and Dinesh K. Pai: Numerical solution of differential systems with algebraic inequalities arising in robot programming. Refereed Proceedings IEEE Robotics & Automation, Nagoya, Japan (1995) 2373–2380.

Addresses: Raymond J. Spiteri, Institute of Applied Mathematics, University of British Columbia, Vancouver BC, Canada V6T 1Z2. Email: spiteri@math.ubc.ca.