Solving differential-algebraic systems of equations (DAEs)

AM 205

Jovana Andrejevic, Catherine Ding

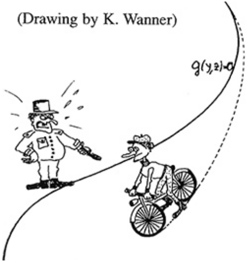
November 4, 2020

Table of Contents

Applications of DAEs

DAEs arise in the mathematical modeling of a wide variety of problems from engineering and science, such as

- multibody problem
- flexible mechanics
- electrical circuit design
- optimal control
- incompressible fluids
- molecular dynamics
- chemical kinetics (quasi steady state, partial equilibrium approximations, chemical process control).



Differential-algebraic systems (DAEs)

In general, we can write any system of differential equations in implicit form as

$$F(t, x, x') = 0$$

where x and x' may be vectors. For a system of ordinary differential equations, the matrix $\partial F/\partial x'$ is not singular. A differential-algebraic system arises when $\partial F/\partial x'$ is singular.

Another way to think about this is that some equations in F are purely algebraic; they contain no derivative terms with respect to t, so some rows of $\partial F/\partial x'$ are zero, producing a singular matrix.

Differential-algebraic systems (DAEs)

One important class of DAEs are those written in semi-explicit form:

$$y' = f(t, y, z)$$
$$0 = g(t, y, z)$$

where y are the differential variables, and z are algebraic variables. Decoupling y and z has nicer implications for numerical integration. The DAE

$$y_1' + y_2' + 2y_2 = 0$$
$$y_1 + y_2 - t^2 = 0$$

is not in semi-explicit form, but can be converted through variable substitution. We'll restrict our discussion today to semi-explicit DAEs.

Differential-algebraic systems (DAEs)

Setting

$$
\begin{aligned}
y &= y_1 + y_2 \quad \text{(differential variable)} \\
z &= y_2 \qquad\qquad \text{(algebraic variable)}
\end{aligned}
$$

we may obtain

$$
\begin{aligned}
y' &= -2z \\
0 &= y - t^2
\end{aligned}
$$

This is indeed in the form of

$$
\begin{aligned}
y' &= f(t, y, z) \\
0 &= g(t, y, z)
\end{aligned}
$$

.

Differential-algebraic systems (DAEs)

In some cases, DAEs arise naturally as limits of singularly perturbed ODEs:

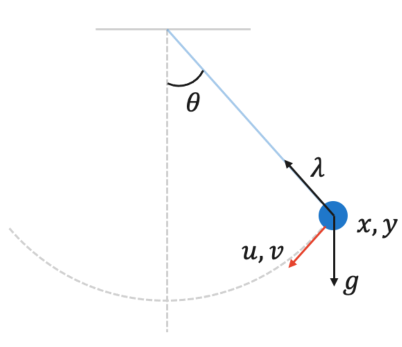
$$y' = f(t, y, z)$$
$$\epsilon z' = g(t, y, z)$$

where ϵ is small. The limit of $\epsilon \to 0$ results in a DAE.

Since z will change rapidly for small ϵ, our integration scheme must resolve widely disparate time scales - a stiff problem. Since the underlying problem is stiff, we'll see that it's a good idea to consider implicit methods for integrating DAEs as well.

Example: The simple pendulum

For a pendulum of unit mass and length, the system of equations which describes its evolution in Cartesian coordinates is



$$
\begin{aligned}
x' &= u, \\
y' &= v, \\
u' &= -\lambda x, \\
v' &= -\lambda y - g, \\
0 &= x^2 + y^2 - 1,
\end{aligned}
$$

where x, y are the position coordinates of the pendulum, u, v the velocities, λ the tension per unit length, and $g = 9.80665$ the acceleration due to gravity.

Hidden constraints

- This system has 5 unknowns to solve for: x, y, u, v, λ.
- It appears to have 4 independent degrees of freedom, due to the single constraint.
- However, there are in fact only 2 independent degrees of freedom in this problem, due to hidden constraints that must be satisfied.
- How do we obtain the hidden constraints?

Differentiation index

- Differentiate the algebraic constraint, $0 = x^2 + y^2 - 1$, repeatedly with respect to t:

$$\begin{aligned} \text{once: } & 0 = 2xx' + 2yy' \to 0 = xu + yv \\ \text{twice: } & 0 = u^2 + v^2 - \lambda(x^2 + y^2) - gy \\ \text{3 times: } & \lambda' = -3gv \end{aligned}$$

- Reveals new constraint equations, for a total of 3 constraints that govern our consistent initial conditions.
- 3 differentiations were needed to obtain a pure ODE system - this is known as the differentiation index of the DAE, and is a measure of how close the DAE system is to its corresponding ODE.

Differentiation index

- Any intermediate equation is a valid substitute for our algebraic constraint:

  index 3: $0 = x^2 + y^2 - 1$ (*length constraint*)

  index 2: $0 = xu + yv$ *(tangential motion)*

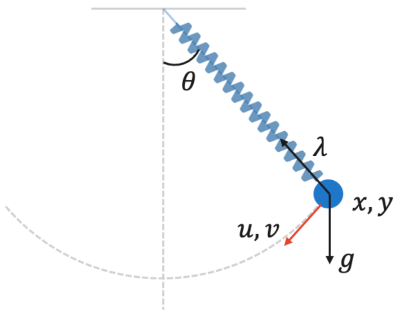
  index 1: $0 = u^2 + v^2 - \lambda(x^2 + y^2) - gy$ *(centripetal accel.)*

  index 0: $\lambda' = -3gv$ *(ODE)*

- However, we forego guaranteeing one constraint by choosing another; the total length would be susceptible to numerical drift, for example, if we use the index 2 formulation, but we would guarantee tangential motion.

Differentiation index

The index 3 pendulum DAE can in fact be regarded as a reduced form of a singularly perturbed index 1 DAE in which the rod is replaced by a stiff spring of spring constant $k = \epsilon^{-1}$:



$$
\begin{aligned}
x' &= u, \\
y' &= v, \\
u' &= -\lambda x, \\
v' &= -\lambda y - g, \\
\epsilon\lambda &= 1 - \frac{1}{\sqrt{x^2 + y^2}}.
\end{aligned}
$$

Here, λ is the spring force per unit length. In the limit $\epsilon \to 0$, the last equation can be rearranged into our length constraint.

Summary - Pros and Cons of DAEs

+ It's typically advantageous to work with the DAE directly, provided we have consistent initial conditions
− Initialization can pose a challenge - have to satisfy hidden constraints
+ DAEs allow us to explicitly enforce constraints
+ A reduced form higher index DAE is often simpler to solve than singularly perturbed ODE/lower index DAE that is stiff/has fast oscillations.

Numerical integration of DAEs

- Due to constraints that must be satisfied at each time, explicit methods are typically not as well-suited for solving DAEs in general.
- Instead, we use implicit methods to discretize the differential part of DAEs, and solve for the algebraic variables simultaneously.
- We will look at one particular family of implicit methods: Backward differentiation formulae.

BDF Methods

The backward differentiation formulae (BDF)[1] are a family of implicit multi-step methods which discretize $y' = f(t, y)$ as:

$$y_{k+1} + \sum_{s=0}^{p-1} \alpha_{k-s} y_{k-s} = \beta h f(t_{k+1}, y_{k+1}),$$

where p is the order of the method.

The first few formulae:

$$p = 1 : y_{k+1} - y_k = hf(t_{k+1}, y_{k+1}) \textit{ (backward Euler)}$$

$$p = 2 : y_{k+1} - \frac{4}{3} y_k + \frac{1}{3} y_{k-1} = \frac{2}{3} hf(t_{k+1}, y_{k+1})$$

$$p = 3 : y_{k+1} - \frac{18}{11} y_k + \frac{9}{11} y_{k-1} - \frac{2}{11} y_{k-2} = \frac{6}{11} hf(t_{k+1}, y_{k+1})$$

[1] See: wikipedia.org/wiki/Backward_differentiation_formula

BDF Methods

Consider the DAE system

$$\begin{aligned} y' &= f(t, y, z) \\ 0 &= g(t, y, z) \end{aligned}$$

We discretize the differential equations and leave the algebraic equations, resulting in a root-finding problem at each step:

$$\begin{aligned} y_{k+1} + \sum_{s=0}^{p+1} \alpha_{k-s} y_{k-s} - \beta h f_{k+1} &= 0 \\ g_{k+1} &= 0 \end{aligned}$$

with y_{k+1}, z_{k+1} as our unknowns to solve for.

Newton's method

In general, our discretized system of equations will be nonlinear in the unknowns, so the equations cannot be rearranged for y_{k+1}, z_{k+1} explicitly. Instead, our root-finding problem can be solved via Newton's method.

Newton's method iteratively finds the roots x^* of a function $r(x)$ so that $r(x^*) = 0$ by iterating

$$x_{i+1} = x_i - J^{-1}(x_i)r(x_i)$$

from an initial guess x_0 until convergence, where J is the Jacobian of r. As a matrix equation to solve:

$$J(x_i)\Delta x_i = -r(x_i),$$

where $x_{i+1} = x_i + \Delta x_i$. It's natural to think of r as our residual, measuring the distance from 0.

Newton's Method

Consider a DAE system with m differential and n algebraic degrees of freedom, given by:

$$\begin{aligned} y' &= f(t, q) = f(t, y, z) \\ 0 &= g(t, q) = g(t, y, z) \end{aligned}$$

where $q = (y, z)$, and $f, y \in \mathbb{R}^m$, $g, z \in \mathbb{R}^n$. Our root-finding problem is

$$r_{k+1} = \begin{bmatrix} y_{k+1} + \sum_{s=0}^{p+1} \alpha_{k-s} y_{k-s} - \beta h f_{k+1} \\ g_{k+1} \end{bmatrix} = 0$$

with Jacobian

$$J_{k+1} = \nabla_q r|_{k+1} = \begin{bmatrix} \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \end{bmatrix}_{k+1} = \begin{bmatrix} I_{m\times m} - \beta h \frac{\partial f}{\partial y} & -\beta h \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix}_{k+1}$$

Newton's Method

Decompose into two contributions:

$$
\begin{aligned}
J_{k+1} &= A + BJ_{k+1}^{dae} \\
&= \underbrace{\left[\begin{array}{c|c} I_{m\times m} & \begin{matrix} 0 \\ \vdots \end{matrix} \\ \hline 0 \quad \cdots & 0_{n\times n} \end{array}\right]}_{A} + \underbrace{\left[\begin{array}{c|c} -\beta h I_{m\times m} & \begin{matrix} 0 \\ \vdots \end{matrix} \\ \hline 0 \qquad \cdots & I_{n\times n} \end{array}\right]}_{B} J_{k+1}^{dae}
\end{aligned}
$$

where

$$
J_{k+1}^{dae} = \left[\begin{array}{c|c} \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \hline \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{array}\right]_{k+1}
$$

is the purely problem-specific part of the Jacobian, while J_{k+1} will depend on the details of the integrator.

Newton's Method

For the simple pendulum:

$$f(t,q) = \begin{cases} u \\ v \\ -\lambda x \\ -\lambda y - g_{acc.} \end{cases}$$

$$g(t,q) = \begin{cases} x^2 + y^2 - 1 \end{cases}$$

where $q = x, y, u, v, \lambda$, and

$$J^{dae} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\lambda & 0 & 0 & 0 & -x \\ 0 & -\lambda & 0 & 0 & -y \\ 2x & 2y & 0 & 0 & 0 \end{bmatrix},$$

Newton's Method

The complete Jacobian is $J_{k+1} = A + BJ^{dae}_{k+1}$ with

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} -\beta h & 0 & 0 & 0 & 0 \\ 0 & -\beta h & 0 & 0 & 0 \\ 0 & 0 & -\beta h & 0 & 0 \\ 0 & 0 & 0 & -\beta h & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$J^{dae}_{k+1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\lambda_{k+1} & 0 & 0 & 0 & -x_{k+1} \\ 0 & -\lambda_{k+1} & 0 & 0 & -y_{k+1} \\ 2x_{k+1} & 2y_{k+1} & 0 & 0 & 0 \end{bmatrix}$$

Newton's iterations for DAE integration

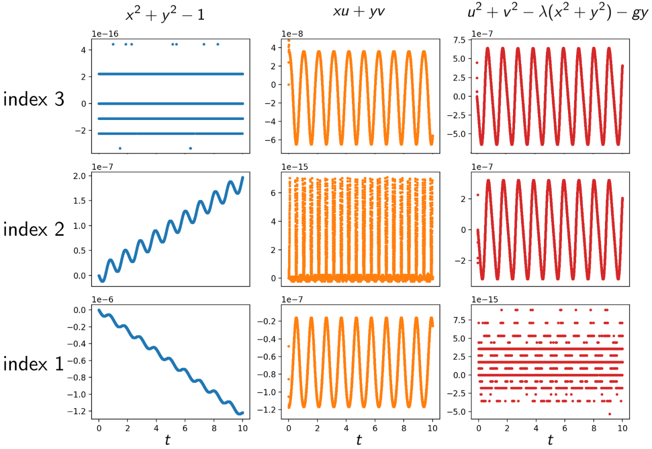
To solve for q_{k+1} at integration step $t = t_{k+1}$:
Initialize guess $q = q_k$
Compute the residual $r(t, q)$
$i = 0$
while $||r||_2 > tol$ and $i < maxiter$ **do**
    $J = A + BJ^{dae}(t, q)$
    Solve $J\Delta q = -r$
    $q \leftarrow q + \Delta q$
    Compute the new residual $r(t, q)$
    $i \leftarrow i + 1$
end while
Solution $q_{k+1} = q$

Exercise 1

- The provided framework for a DAE solver class sets up a third-order BDF method with first and second order start-up steps.
- Your task is to implement the Newton's method routine performed at each integration step.
- Then, test your solver on the provided simple pendulum problem.

Index comparison

We can look at how well all constraints are maintained when varying the index of the simple pendulum example:



Notice drift in constraints that are not explicitly enforced.

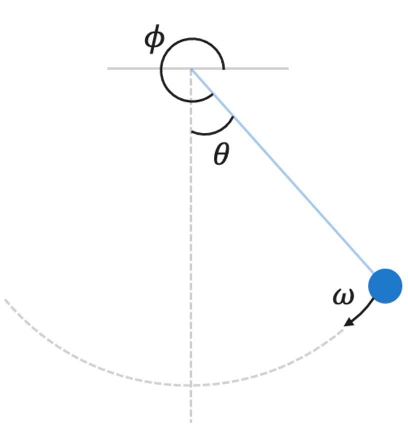
Practical considerations

- Newton's method initialization: We can improve our initial guess to Newton's iterations by constructing a Lagrange interpolant of our prior solutions (featured as optional extension).

- Convergence: Error convergence for DAEs can be complicated, though there are analytical results for index ≤ 2 and special index 3 examples. Strict tolerance on Newton iterations is critical to achieving expected BDF convergence (See references for more detail on this topic).

Convergence

There is a two-variable ODE system - the state-space form - of the pendulum which can be solved explicitly:



$$\phi' = \omega,$$
$$\omega' = -g \cos \phi,$$

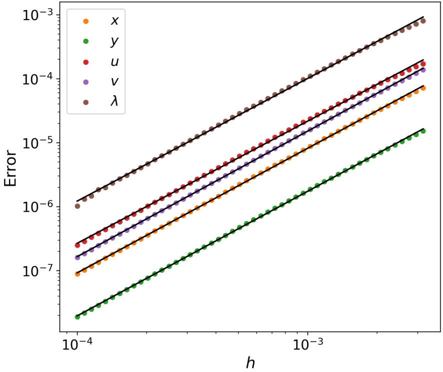
and maps as

$$x = \cos \phi,$$
$$y = \sin \phi,$$
$$u = -\omega \sin \phi,$$
$$v = \omega \cos \phi,$$
$$\lambda = \omega^2 - g \sin \phi.$$

We can use an adaptive step method with strict tolerance (using odeint) as a reference solution.

Convergence

Our error turns out to be $O(h^2)$:



In general, the convergence of algebraic variables can differ from differential variables, and be lower order.

Practical considerations

- Step size adaptivity: Start-up steps for fixed step integrators incur larger errors; in practice, we can solve DAEs with an adaptive step method, and take small initial steps to avoid the larger error penalty.

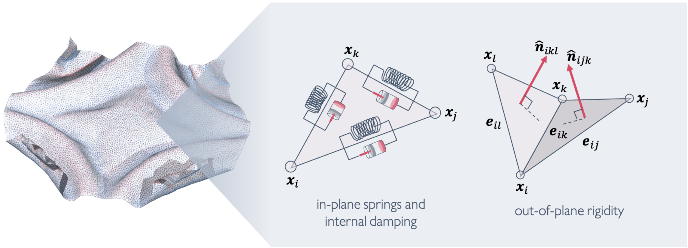
- Conditioning: The Jacobian

$$J_{k+1} = \begin{bmatrix} I_{m\times m} - \beta h \frac{\partial f}{\partial y} & -\beta h \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix}_{k+1}$$

  can become ill-conditioned for very small h particularly for higher index DAEs, when $\partial g/\partial z = 0$. May call for better approaches to solve $J\Delta q = -r$ (e.g. regularization, preconditioned conjugate gradient methods).

Example: Crumpling a thin sheet

- A thin sheet may be modeled as a network of masses and springs:



The equations of motion for a node i of mass m are given by

$$\dot{x}_i = v_i$$
$$m\dot{v}_i = F_i,$$

where the net force F_i includes contributions from stretching, damping, bending, self-avoidance, and external forces.

Example: Crumpling a thin sheet

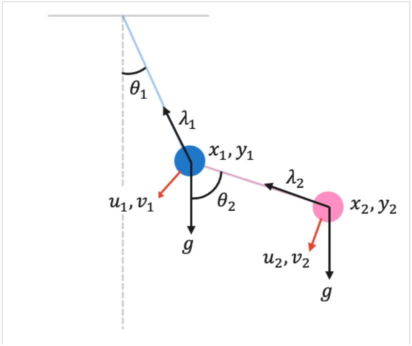
- Typically, these equations form an ODE system that we can solve, e.g. using the classic RK4 method.
- However, when acceleration is small, the equations of motion can be approximated as the DAE

$$\dot{x}_i = v_i$$
$$F_i = 0$$

- Solved using a 3rd order adaptive step BDF method (implicit)
- Turns out to be very efficient for this problem - can take large integration steps
- Integrator automatically detects when DAE formulation is appropriate, and switches to ODE formulation otherwise

The double pendulum

Once we have a general solver in place, we can easily swap out the DAE system. As an extension, consider the double pendulum:



$$
\begin{aligned}
x_1' &= u_1, \\
y_1' &= v_1, \\
u_1' &= -\lambda_1 x_1 - \lambda_2(x_1 - x_2), \\
v_1' &= -\lambda_1 y_1 - \lambda_2(y_1 - y_2) - g, \\
x_2' &= u_2, \\
y_2' &= v_2, \\
u_2' &= -\lambda_2(x_2 - x_1), \\
v_2' &= -\lambda_2(y_2 - y_1) - g, \\
0 &= x_1^2 + y_1^2 - 1, \\
0 &= (x_2 - x_1)^2 + (y_2 - y_1)^2 - 1.
\end{aligned}
$$

Exercise 2

- Implement the DAE system for the double pendulum, and derive and implement its Jacobian.
- Use your DAE solver from Exercise 1 to integrate the double pendulum, and visualize its chaotic motion!

References

1. Campbell, Stephen L., Vu Hoang Linh, and Linda R. Petzold. "Differential-algebraic equations." Scholarpedia 3.8 (2008): 2849.
2. Ascher, Uri M., and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Vol. 61. Siam, 1998.
3. Eich-Soellner, Edda, and Claus Führer. *Numerical Methods in Multibody Dynamics*. Vol. 45. Stuttgart: Teubner, 1998.
4. Hairer, Ernst. and Wanner, Gerhard. *Solving Ordinary Differential Equations II*. Springer, 1996.