Direct collocation method for differential-algebraic equations (DAEs)

Vincent Chan

The goal here is to solve differential-algebraic equations (DAEs) via the collocation method and somehow validate the solution. With initial conditions and paramaters given, the specific DAE considered here is

$$\begin{aligned} m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} &= -f\frac{x}{l} \\ m\frac{\mathrm{d}^2y}{\mathrm{d}t^2} &= mg - f\frac{y}{l} \\ 0 &= x^2 + y^2 - l^2 \ , \end{aligned}$$

and is equivalent to the ODE

$$m\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{mg}{l}\sin\theta = 0 \ .$$

The DAE can also be decomposed into a DAE of four 1^{st}-order ODEs plus an algebraic constraint:

$$\begin{aligned} \dot{x} &= u \\ m\dot{u} &= -f\frac{x}{l} \\ \dot{y} &= v \\ m\dot{v} &= mg - f\frac{y}{l} \\ 0 &= x^2 + y^2 - l^2 \ , \end{aligned}$$

again with the initial conditions and parameters given.

1 Intro: deriving the derivative matrix

Consider the expansion of an arbitrary (but smooth) function onto some orthogonal basis:

$$f(x) = \sum_{n=0}^{\infty} a_n T_n(x) \ , \tag{1}$$

for $x \in [a, b]$. Here, I actually have in mind for $\{T_n(x)\}$ the Chebyshev polynomials and $[a, b] = [-1, 1]$, but any set of orthogonal polynomials will serve as a basis.

In any of these orthogonal set of polynomials, $T_n(x)$ is a n^{th}-order polynomial. As such, for well behaved functions, it is sufficient to consider only the first N terms, for some natural number N:

$$f(x) = \sum_{n=0}^{N} a_n T_n(x) \ , \tag{2}$$

where the equality is kept because we've assumed any difference to be negligible.

Instead of $\{T_n(x)\}$ as the basis set, it is also possible to view $\{a_n\}$ as the basis set. If we further consider f at some $N + 1$ points, then we can view the expansion as a linear transformation of one set of basis $\{a_n\}$ to another $\{f_n\}$:

$$f(x_n) = \sum_{m=0}^{N} T_m(x_n) a_m \ , \tag{3}$$

or

$$|f\rangle = T(x)\,|a\rangle \ . \tag{4}$$

Now, take the derivative of both sides with respect to x:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}|f\rangle &= \frac{\mathrm{d}}{\mathrm{d}x}\left(T(x)\,|a\rangle\right) && (5)\\ &= T'(x)\,|a\rangle && (6)\\ &= T'(x)T^{-1}(x)\,|f\rangle && (7)\\ &= D\,|f\rangle \ , && (8) \end{aligned}$$

where we have now defined our derivative matrix $D(x) := T'(x)T^{-1}(x)$, a $(N+1)\times(N+1)$ matrix.

The only issue left is choosing the optimal set of points $\{x_n\}$ (note: these points are called **collocation points**.). Without going into details (because I don't know the details), there are two supposed best choices for the collocation points:

1. Gauss points, which are the zeros of the highest order polynomial, $T_N(x)$, and
2. Gauss-Legendre points, where are the zeros of the derivative of the highest order polynomial, $T'_N(x)$ plus the boundary points (which varies depending on the choice of othogonal polynomials).

For the purpose solving differential equations on a compact space (so that the problem of dealing with infinitly far away boundaries can be ignored), the Gauss-Legendre set of collocation points is the obvious choice. **Further, the matrix D has been worked out and can be found in, for example, the appendix of Boyd's book.**

2 Ordinary Differential Equations

To see how to utilize the derivative matrix D, it is simplest to give an example. So consider

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x) \ , \tag{9}$$

with initial condition $y(x_0) = y_0$.

Then we simply consider the truncated expansion and rewrite the problem as

$$D\,|y\rangle = |f\rangle \ , \tag{10}$$

with IC $|y\rangle_{(x_0)} = y_0$. At this point, should you try to solve for $|y\rangle$ as $|y\rangle = D^{-1}\,|f\rangle$, you'll come across the problem that D is singular. Of course, that is expected since there exists an infinite number of functions y whose derivative is f. So, we must find some way of incorporating the initial condition into the linear system of equations.

The solution is to simply substitute the ODE for $|y\rangle\,(x_0)$ with the initial condition. This gives us

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ d_{10} & d_{11} & \cdots & d_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N0} & d_{N1} & \cdots & d_{NN} \end{pmatrix} \begin{pmatrix} y(x_0) \\ y(x_1) \\ \vdots \\ y(x_N) \end{pmatrix} = \begin{pmatrix} y_0 \\ f(x_1) \\ \vdots \\ f(x_N) \end{pmatrix} \ , \tag{11}$$

which can be inverted without problem. And so, the ODE is solved.

For nonlinear problems of the type

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y) \ , \tag{12}$$

with initial condition $y(x_0) = y_0$, it is best (my subjected view) to reconsider the truncated version as a system of nonlinear equations:

$$F(x,y) = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ d_{10} & d_{11} & \cdots & d_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N0} & d_{N1} & \cdots & d_{NN} \end{pmatrix} \begin{pmatrix} y(x_0) \\ y(x_1) \\ \vdots \\ y(x_N) \end{pmatrix} - \begin{pmatrix} y_0 \\ f(x_1, y_1) \\ \vdots \\ f(x_N, y_N) \end{pmatrix} = \mathbf{0} \ , \tag{13}$$

which while cannot be solved directly but whose iterated solution may hopefully converge via the Newton-Raphson method or one of its simplified variants.

3 Differential Algebraic Equations

Differential algebraic equations (DAEs), which are ODEs with algebraic constraints, are really no different from the ODEs seen above. They all get converted to a system of (non)linear equations. However, the main issue here is that when applying the initial conditions or boundary conditions, it is necessary to make sure that important information is not accidentally deleted in the process.

4 Example: pendulum of fixed length

Consider the basic pendulum problem but with the y-direction points down to the ground (left-handed system).

In the polar coordinate, along with the substitution of the constraint, the result ODE is

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}\theta + \frac{mg}{l}\sin\theta = 0 \ , \tag{14}$$

with initial condition $\theta(t_0) = \theta_0$. In the small angle approximation $\sin\theta \approx \theta$, we get simple harmonic motion.

But since we want to solve it as a DAE, we consider the system in the (x, y)-coordinate system (so that sine and cosine may be replaced as x/l and y/l). The resulting system of ODEs and algebraic constraints, or DAE, is

$$\begin{aligned} m\frac{\mathrm{d}^2}{\mathrm{d}t^2}x + fx/l &= 0 && (15)\\ m\frac{\mathrm{d}^2}{\mathrm{d}t^2}y + fy/l - mg &= 0 && (16)\\ \frac{1}{2}\left(x^2 + y^2 - l^2\right) &= 0 \ , && (17) \end{aligned}$$

with initial conditions given for $x, y, \dot{x}$, and $\dot{y}$, and where f is the magnitude of the tension force. The independ variable is time t, and the dependent variables are x, y, and f.

Notice that here, we must choose 4 ODEs / algebraic equations to replace with the initial conditions. Assuming the initial conditions were chosen properly, then the initial tension force f is also known. As such, we may safely replace 3 of the equations without losing information. Unfortunately, we have 4 initial conditions to set and I just don't know how to do that here.

Instead, I choose to rewrite each of the 2^{nd}-order ODEs as two 1^{st}-order ODEs. Then, the system becomes

$$\begin{aligned} \dot{x} - u &= 0 && (18)\\ ml\dot{u} + fx &= 0 && (19)\\ \dot{y} - v &= 0 && (20)\\ ml\dot{v} - mlg + fy &= 0 && (21)\\ \frac{1}{2}\left(x^2 + y^2 - l^2\right) &= 0 \ , && (22) \end{aligned}$$

and the initial conditions for x, u, y, v, and f can be safely incorporated without problems.

Letting $X = diag(x)$, $U = diag(u)$, and so on, as well as $1 = 1_{N+1}$ and $0 = 0_{N+1}$, the

Jacobian of the system is almost

$$J = \begin{pmatrix} D & -1 & 0 & 0 & 0 \\ F & mlD & 0 & 0 & X \\ 0 & 0 & D & -1 & 0 \\ 0 & 0 & F & mlD & Y \\ X & 0 & Y & 0 & 0 \end{pmatrix}, \tag{23}$$

but with the appropriate substitution corresponding to the substitutions of the ODEs / constraints with the initial condition constraints.

4.1 Code

There are 5 files in total: *cheb.m*, *solve.m*, *func.m*, *jacobian.m*, and *pendulum.m*.

The first two files, *cheb.m* and *solve.m*, should not be touched. *cheb.m* contains a function "cheb" which outputs the derivative matrix D and collocation points x. *solve.m* contains a function that solves a system of nonlinear equations using the Newton–Raphson method.

The 3$^{\text{rd}}$, *func.m*, consists of a vector function containing the left-hand side of Equations (18-22), for each collocation point, and with initial conditions included. The 4$^{\text{th}}$, *jacobian.m*, consists of the Jacobian for the just-mentioned vector function.

The 5$^{\text{th}}$, *pendulum.m*, is the main program, and is separated into two main parts. The first part consist of setting up the pendulum problem given by the DAE, see Equations (18-22). The second part solves the DAE and consist primarily of while loop. The main thing to note (since some programmers have note seen this) is that I have created the data structures *param*, *op*, *ic*, and *sol* to group certain elements together for clarity. As likely expected, each iteration of the while-loop is one "timestep," each of which contains N collocation points.