Scientific Computing: ODEs — Shooting

David A.W. Barton

2019/20

Numerical shooting is a simple method for solving ordinary differential equation boundary value problems. It relies on a suitable numerical integrator, along with a numerical root finder.

Periodically-forced ordinary differential equations

Consider the ordinary differential equation (ODE)

$$m\ddot{x} + c\dot{x} + kx = \Gamma \sin(\omega t). \tag{1}$$

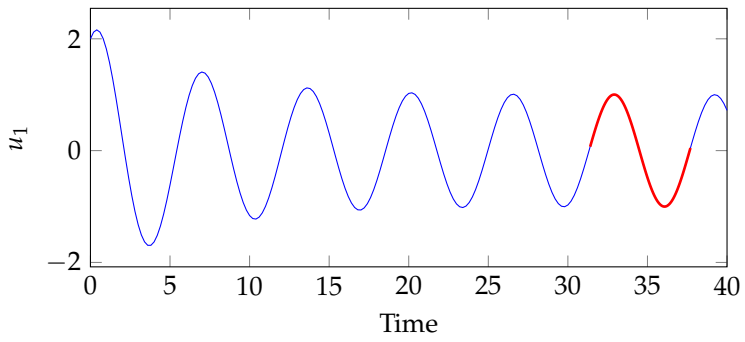
This is the equation of motion for a mass-spring-damper with harmonic excitation. For a given set of parameters, it has a periodic response (a periodic orbit/limit cycle oscillation[1]) where the period is given by[2]

$$T = \frac{2\pi}{\omega}. \tag{2}$$

To simulate (1), it must be rewritten in first-order (vector) form, that is

$$\begin{aligned} \dot{u}_1 &= u_2, \\ \dot{u}_2 &= \frac{1}{m}\left(\Gamma \sin(\omega t) - cu_2 - ku_1\right). \end{aligned} \tag{3}$$

This is achieved by setting $u_1 = x$ and $u_2 = \dot{x}$. Equation (3) is the standard form required for (almost) all ODE solvers.[3] Simulating (3) gives the time series shown in figure 1.



A numerical ODE solver produces the (time series) solution at prescribed value(s) of time; that is, it provides a (vector-valued) function[4] $\mathbf{F}$ such that

$$\mathbf{u}(\tau) = \begin{bmatrix} u_1(\tau) \\ u_2(\tau) \end{bmatrix} = \mathbf{F}(\mathbf{u}_0, \tau) \tag{4}$$

where $\mathbf{u}_0$ is a vector of initial conditions and τ is the prescribed time.

To find limit cycles, we must solve the periodic boundary value problem (BVP)

$$\mathbf{u}_0 - \mathbf{u}(T) = 0 \qquad \Rightarrow \qquad \mathbf{u}_0 - \mathbf{F}(\mathbf{u}_0, T) = 0. \tag{5}$$

[1] A *limit cycle* is simply an isolated periodic orbit.

[2] Since the period of the response is known, it is easier to apply numerical shooting to periodically-forced ODEs than general ODEs.

[3] Some specialist mechanical engineering solvers expect the differential equations to be in the form of (1), however, they are the exception to the rule and very rare outside of structural dynamics.

Figure 1: A simulation of (3). The blue curve shows transient behaviour converging onto a limit cycle. A single period of the (almost converged) limit cycle is shown in red.

[4] The function $\mathbf{F}$ is the numerical integrator, e.g., the `ode45` function in MATLAB, the `solve_ivp` or `odeint` functions in Python (Scipy), and the `solve` function in Julia (OrdinaryDiffEq.jl).

Equation (5) is a vector equation with the same number of dimensions as $\mathbf{u}_0$. For this particular example $\mathbf{u}_0$ is an unknown two-dimensional vector and T is a known scalar parameter[5] (given by (2)).

In order to solve (5) we define the function

$$\mathbf{G}(\mathbf{u}_0) = \mathbf{u}_0 - \mathbf{F}(\mathbf{u}_0, T). \tag{6}$$

For an arbitrary initial guess $\tilde{\mathbf{u}}_0$, the value of $\mathbf{G}(\tilde{\mathbf{u}}_0)$ will be non-zero (see Figure 2). However, if we are able to find a value for $\tilde{\mathbf{u}}_0$ such that $\mathbf{G}$ is close to zero[6] it can be used as the inital value for a numerical root finder such as a Newton iteration.

Hence, limit cycles of (3) can be found by passing (6) along with a suitable initial guess $\tilde{\mathbf{u}}_0$ to a numerical root finder such as `fsolve` in MATLAB or Python (Scipy) or `nlsolve` in Julia.

All of the above can be trivially generalised to arbitrary periodically-forced ODEs of any number of dimensions.

[5] In the general case T will also be unknown, as discussed in the next section.

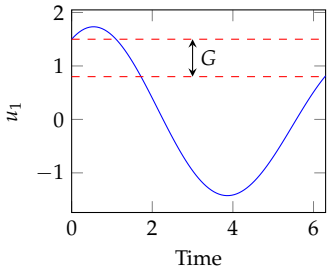


Figure 2: The result of an arbitrary initial guess $\tilde{\mathbf{u}}_0$.

[6] Close to zero does not mean 10^{-6}; a good root finder will often converge even when starting with a large value for $\mathbf{G}$.

Autonomous ordinary differential equations

The previous section focussed on periodically-forced ODEs where the time period of oscillation was giving by the period of the forcing. Limit cycles in autonomous ODEs[7] have the complication that there is no explicit time scale for the oscillations, hence the period is an additional unknown that must be determined as part of the nonlinear root finding step. Whenever solving systems of equations, the general rule is that the number of equations should match the number of unknowns; this implies that an extra equation must be added to the nonlinear root finding problem (5) in order to determine the period T.

Consider the van der Pol equation

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0 \tag{7}$$

for $\mu > 0$ there exists a limit cycle oscillation as shown in figure 3.

[7] Autonomous ODEs have no explicit time dependency in the equations.

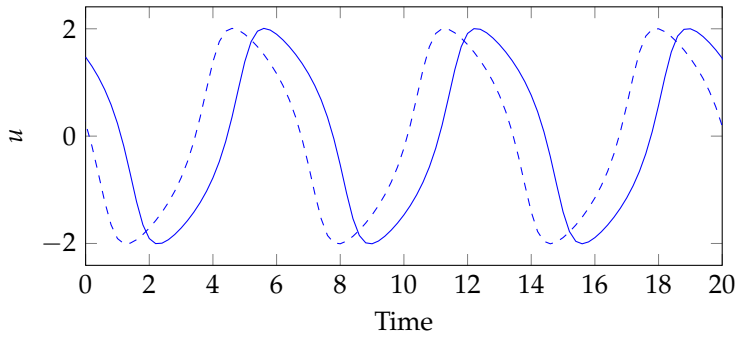


Figure 3: A simulation of (7) for $\mu = 1$. Both the solid blue curve and the dashed blue curve are valid solutions; since (7) has no explicit time dependency, solutions can be arbitrarily shifted in time.

In addition to the unknown period, the lack of explicit time dependency in (7) means that solutions can be arbitrarily shifted in time, leading to a family of possible initial values $\mathbf{u}_0$ for the limit cycle rather than a single isolated value as with periodically-forced ODEs.[8]

[8] The existence of a family of solutions tends to cause problems for numerical root finders since they assume that there exists an isolated root.

It turns out that both of these problems can be solved by including a *phase condition* ϕ into the definition of $\mathbf{G}$ in (5). A phase condition is simply a means for fixing the phase (i.e., the time shift) of the periodic orbit; this extra constraint is sufficient to also determine the period of oscillation.

The choice of phase condition is largely arbitrary. For example, for the van der Pol equation the phase condition $x(0) = 0$ can be used; the problem with fixing a state variable with this manner is that we cannot usually know a priori that the state variable will obtain that value anywhere in the cycle.[9] Despite this problem, choosing a phase condition in this way is often simple and convenient.

[9] For example, choosing the phase condition $x(0) = 5$ for the van der Pol with $\mu = 1$ will not work since $-2 \leq x \leq 2$ (approximately; see Figure 3.)

To implement a phase condition $x(0) = a$ for some constant a, we set

$$\phi(\mathbf{u}_0) = u_{0,1} - a \tag{8}$$

where $\mathbf{u}_0 = [x(0), \dot{x}(0)]$ and $u_{0,1}$ is the first component of the vector $\mathbf{u}_0$. The function $\mathbf{G}$ in (6) becomes (for autonomous ODEs)

$$\mathbf{G}(\mathbf{u}_0, T) = \begin{bmatrix} \mathbf{u}_0 - \mathbf{F}(\mathbf{u}_0, T) \\ \phi(u_0) \end{bmatrix}. \tag{9}$$

$\mathbf{G}$ is thus a $n+1$ dimensional vector equation (where n is the dimension of $\mathbf{u}_0$) with $n+1$ inputs. This equation put into a numerical root finder to solve for both $\mathbf{u}_0$ and T.

Since the phase condition specified above is not always appropriate, an alternative phase condition is to set

$$\frac{\mathrm{d}\, x}{\mathrm{d}\, t}(0) = 0 \tag{10}$$

this has the advantage that, since x is periodic, the limit cycle will satisfy this condition at some point. It is relatively easy to implement this phase condition when $\dot{x}$ is a state variable in its own right (like both examples here), though it is slightly tricker when this is not the case. For example, the A-B reaction equations[10]

[10] Uppal, Ray & Poore (1974).

$$\begin{aligned} \dot{u}_1 &= -u_1 + p_1(1-u_1)\exp(u_2) \\ \dot{u}_2 &= -u_2 + p_1 p_2 (1-u_1)\exp(u_2) - p_3 u_2 \end{aligned} \tag{11}$$

In this case $\dot{u}_1$ is not a state variable (u_1 and u_2 are the state variables) and so to implement $\frac{\mathrm{d}\, u_1}{\mathrm{d}\, t}(0) = 0$ we require the slightly more complicated phase condition

$$\phi(\mathbf{u}) = -u_1 + p_1(1-u_1)\exp(u_2). \tag{12}$$

More generally, the most widely used phase-condition is an integral phase condition.[11] However, this is quite difficult to implement for shooting methods.

[11] Lecture Notes on Numerical Analysis of Nonlinear Equations (in Numerical Continuation Methods for Dynamical Systems, Springer), E.J. Doedel, section 1.4.2.