Solving IVPs, BVPs

Numerical Integration, Collocation, Shape Functions

Philipp T. Tempel

Laboratoire des Sciences du Numérique de Nantes (LS2N)

January 21, 2021

Outline and Objectives

- Recap of solving...
  - initial value problems (IVPs)
  - boundary value problems (BVPs)
- Recap "conventionally" obtaining solutions to IVPs
- Introduce collocation method for solving IVPs and BVPs
- Create link between numerical integrator schemes and collocation method

Section 1

Fundamentals

Differential Equations

A *differential equation* of the general form

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

specifies the relation between an *unknown differentiable function* $y(x)$ and its derivative $\frac{\mathrm{d}y}{\mathrm{d}x}$.
We term above equation an *ordinary differential equation* because it involves derivatives with respect to a single independent variable, here x.
This is in contrast to a *partial differential equation*, which involves several independent variables e.g.,

$$\frac{\partial u(t,x)}{\partial t} = \frac{\partial u(t,x)}{\partial x}, \qquad \text{or} \qquad \frac{\partial^2 u}{\partial x^2} + a(x,y)\frac{\partial^2 u}{\partial y^2} = b(x,y).$$

For ODEs, time is often the independent variable, which is suggested by the letter t.

Solution of a Differential Equation

A *solution* to the ODE is a *differential function* $\tilde{y}(t)$ that satisfies the differential equation

$$\frac{\mathrm{d}\tilde{y}(t)}{\mathrm{d}t} = f(t, \tilde{y}(t)),$$

on some interval of interest i.e., for $t \in [t_0, t_f]$.
To determine a *unique solution*, it is necessary to specify the value of the function at some point t_0. This creates an *initial value problem* combining:

a differential equation	$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)$,
an initial value	$y(t_0) = y_0$,
an interval	$t \in [t_0, t_f = t_0 + T]$.

System of Differential Equations

For completeness, let us introduce the general form of a *system of d differential equations*

$$\frac{\mathrm{d}y_1}{\mathrm{d}t} = f_1(t, y_1, y_2, \ldots, y_d)\,,$$
$$\frac{\mathrm{d}y_2}{\mathrm{d}t} = f_2(t, y_1, y_2, \ldots, y_d)\,,$$
$$\vdots$$
$$\frac{\mathrm{d}y_d}{\mathrm{d}t} = f_d(t, y_1, y_2, \ldots, y_d)\,.$$

for vector notation $\boldsymbol{y}(t)\colon \mathbb{R} \to \mathbb{R}^d$ and $\boldsymbol{f}(t, \boldsymbol{y})\colon \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$:

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1(t), & \ldots, & y_d(t) \end{bmatrix}^\top, \qquad \boldsymbol{f}(t, \boldsymbol{y}(t)) = \begin{bmatrix} f_1(t, \boldsymbol{y}(t)), & \ldots, & f_d(t, \boldsymbol{y}(t)) \end{bmatrix}^\top.$$

The same notation goes for the initial value problem i.e.,

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = f(t, \boldsymbol{y})\,, \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0\,, \qquad t \in [t_0, t_f]\,.$$

Section 2

Initial Value Problems

Analytical Solution

We can solve a wide variety of differential equations of the form

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y)\,, \qquad\qquad y, f : \mathbb{R} \to \mathbb{R}$$

as follows:

$$\frac{\mathrm{d}y}{f(y)} = \mathrm{d}t$$

then

$$\int \frac{\mathrm{d}y}{f(y)} = G(y) = t + C\,,$$

where C is an arbitrary constant. It may be difficult to evaluate the integral even when an antiderivative $G(y)$ can be written down because $G(y) = t + C$ is generally nonlinear.

Example

Analytical Solution

For the sake of an example, let us solve the logistic model

$$\frac{\mathrm{d}n}{\mathrm{d}t} = r\,n\,(1-n)$$

by decomposition into partial fractions:

$$\begin{aligned} r\;\mathrm{d}t &= \frac{\mathrm{d}n}{n\,(1-n)} \\ r\,t + C &= \int \frac{\mathrm{d}n}{n\,(1-n)} = \int \frac{1}{n} + \frac{1}{1-n}\;\mathrm{d}n \\ &= \ln n - \ln(1-n)\,. \end{aligned}$$

Taking the exponential of both sides yields, with $c = \exp C$,

$$\frac{n}{1-n} = c\,\exp(r\,t) \qquad \Rightarrow \qquad n(t) = \frac{c\,\exp(r\,t)}{1 + c\,\exp(r\,t)}\,.$$

Example

Analytical Solution

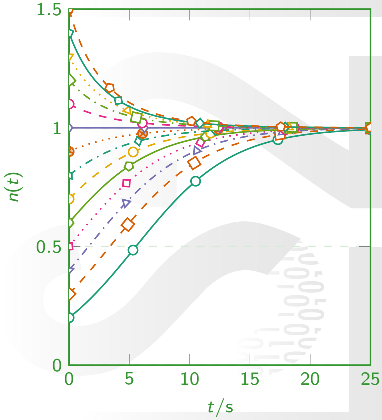
If $n(0) = n_0$, then $c = \frac{n_0}{1-n_0}$ and the solution

$$n(t) = \frac{c\ \exp(r\,t)}{1 + c\ \exp(r\,t)}\,.$$

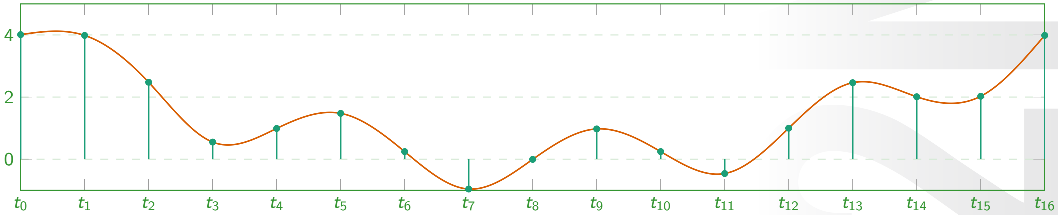
can be written as

$$n(t) = \frac{n_0\ \exp(r\,t)}{(1 - n_0) + n_0\ \exp(r\,t)}\,.$$

However, the set of ODEs with an analytical solution is limited and in general, there *exists no* analytical solution. So, we need to find another way to be able to handle any ODE we may encounter.



Numerical Solution



Let us replace the interval $\Upsilon = [t_0, t_0 + T]$ by a *finite number N of discrete times* $t_n = t_0 + n\,h$, $n = 0, 1, \ldots, N$ where $h = T/N$ is the *step size*. Similarly, the *continuous solution* $y(t)$ on Υ will be replaced with the *numerical solution* $y_n \approx y(t_n)$, $n = 0, 1, \ldots, N$[1].
Procedures to generate each y_n for $n > 0$[2] are *time-stepping procedures* or *numerical integrator schemes* and come in two flavors: *one-step* methods and *multi-step* methods.

[1] Think of y_n being snapshots of the system state at the discrete times and the sequence $\{y_0, y_1, \ldots, y_N\}$ as a movie.

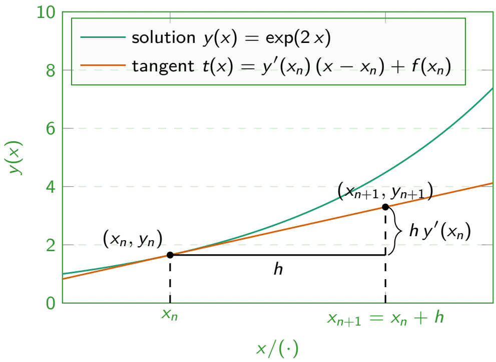
[2] y_0 is the *given initial condition*

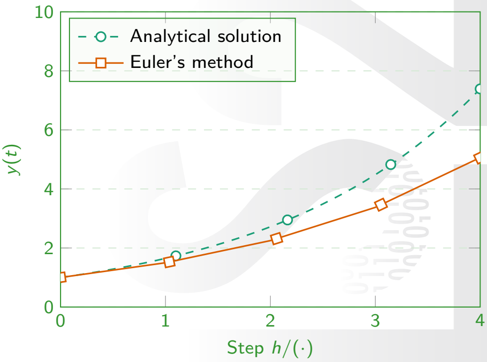
Euler's Method

Numerical Solution

The simplest and oldest method, *Euler's method*, is based on the rectangle rule for approximation of an integral:

$$y_{n+1} = y_n + h\, f(t_n, y_n)\,.$$





Trapezoidal Rule

Numerical Solution

A better approximation of the rectangle rule is the *trapezoidal rule*.

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right).$$

However, the trapezoidal rule defines y_{n+1} *implicitly* as a function of y_n and y_{n+1}. In other words, we must solve the typically *nonlinear system of algebraic equations*

$$0 = \boldsymbol{e}(y) := -y_{n+1} + y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right).$$

Implicit methods are *more expensive to evaluate* than explict methods such as Euler's method. However, for h *small enough*, the implicit function theorem assues us there is a *unique solution* and we will find it. For larger h, uniqueness and existence are not guaranteed.

Runge-Kutta Methods

Numerical Solution

Runge-Kutta methods are a *class of methods* judiciously using information on the slope of the ODE at *more than one intermediate* points in the interval $[t_n, t_n + h]$ to *extrapolate* the solution:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i \, k_i \, ,$$

with

$$k_i = f\big(t_n + c_i \, h, y_n + \sum_{j=1}^{i-1} a_{ij} \, k_j\big) \, .$$

Runge-Kutta Methods

Numerical Solution

The classical *second-order accurate Runge-Kutta* method (RK2) reads

$$
\begin{aligned}
k_1 &= h\, f(t_n, y_n)\,, \\
k_2 &= h\, f(t_n + h, y_n + k_1)\,, \\
y_{n+1} &= y_n + \frac{1}{2}\,(k_1 + k_2)\,.
\end{aligned}
$$

To determine y_{n+1}, we need to evaluate $f(t, y)$ *multiple times* at intermediate steps $t_n + c_i\, h$, which can be expensive.

Adams Methods

Numerical Solution

We can rewrite the IVP

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)\,,$$

with the fundamental theorem of calculus to read

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y)\,\mathrm{d}t\,.$$

Adams Methods are based on replacing the integrand with a *polynomial* that *interpolates* $f(t, y)$ at selected points (t_j, y_j). The k-th order *Adams-Bashforth* method is *explicit* (uses current point (t_n, y_n) and $k-1$ historical points); k-th order *Adams-Moulton* method is *implicit* (uses future point (t_{n+1}, y_{n+1}) and $k-1$ historical points).

Adams Methods: Derivation

Numerical Solution

In k-th order Adams-Bashforth, we set

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p_{k-1}(t)\, \mathrm{d}t\,,$$

where $p_{k-1}(t)$ *interpolates* $f(t,y)$ at (t_{n-j}, y_{n-j}), $j = 0, 1, \ldots, k-1$.

Order	Interpolant	AB Interpolation Points
1st	constant	(t_n, f_n)
2nd	linear	$(t_n, f_n), (t_{n-1}, f_{n-1})$
3rd	quadratic	$(t_n, f_n), (t_{n-1}, f_{n-1}), (t_{n-2}, f_{n-2})$
4th	cubic	$(t_n, f_n), (t_{n-1}, f_{n-1}), (t_{n-2}, f_{n-2}), (t_{n-3}, f_{n-3})$
5th	quartic	$(t_n, f_n), (t_{n-1}, f_{n-1}), (t_{n-2}, f_{n-2}), (t_{n-3}, f_{n-3}), (t_{n-4}, f_{n-4})$

Adams Methods: Derivation

Numerical Solution

If $k = 1$, then one-point Newton-Cotes rule is applied and we get *Euler's method*:

$$y_{n+1} = y_n + h_n\, f(t_n, y_n)\,, \qquad h_n = t_{n+1} - t_n\,.$$

If $k = 2$, the *second-order AB method*, we set p_{k-1} as the linear interpolant of (t_{n-1}, f_{n-1}) and (t_n, f_n) yielding

$$p_{k-1}(t) = f_{n-1} + \frac{f_n - f_{n-1}}{h_{n-1}}\,(t - t_{n-1})\,,$$

from which we obtain

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\,\mathrm{d}t \approx \int_{t_n}^{t_{n+1}} f_{n-1} + \frac{f_n - f_{n-1}}{h_{n-1}}\,(t - t_{n-1})\,\mathrm{d}t = \frac{h_n}{2}\left(\frac{h_n + 2\,h_{n-1}}{h_{n-1}}\,f_n - \frac{h_n}{h_{n-1}}\,f_{n-1}\right).$$

With h constant i.e., $h_n = h_{n-1} = h$, then

$$y_{n+1} = y_n + \frac{h}{2}\,(3\,f_n - f_{n-1})\,.$$

Numerical Integration

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y) \quad \Leftrightarrow \quad y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(\tau, y)\, \mathrm{d}\tau\,.$$

1. The integrand $f(\cdot)$ may be known only at certain points, such as obtained by sampling. Some embedded systems and other computer applications may need numerical integration for this reason.
2. A formula for the integrand may be known, but it may be difficult or impossible to find an antiderivative that is an elementary function. An example of such an integrand is $f(x) = \exp(-x^2)$, the antiderivative of which (the error function, times a constant) cannot be written in elementary form.
3. It may be possible to find an antiderivative symbolically, but it may be easier to compute a numerical approximation than to compute the antiderivative. That may be the case if the antiderivative is given as an infinite series or product, or if its evaluation requires a special function that is not available.

Gauss-Legendre Quadrature

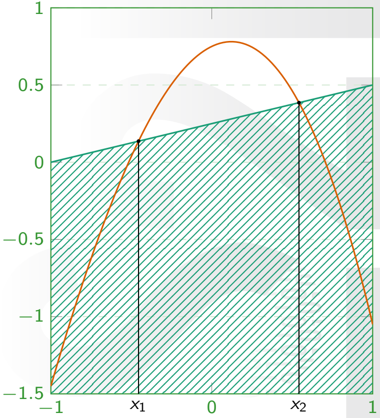
Numerical Integration

The *Gauss-Legendre quadrature rule* is an approximation of the *definite integral* of a function $f(x)$

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i\, f(x_i)\,,$$

which is exact for polynomials of degree $d = 2\,n - 1$. For any integrand over an interval of $[a, b]$, it reads

$$\int_{a}^{b} f(x)\,\mathrm{d}x = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}\,\xi + \frac{a+b}{2}\right)\,\mathrm{d}\xi\,.$$



To approximate the integral, we only need the *weights* w_i and the value of the functions at select *collocation points* x_i.

Gauss-Legendre Quadrature

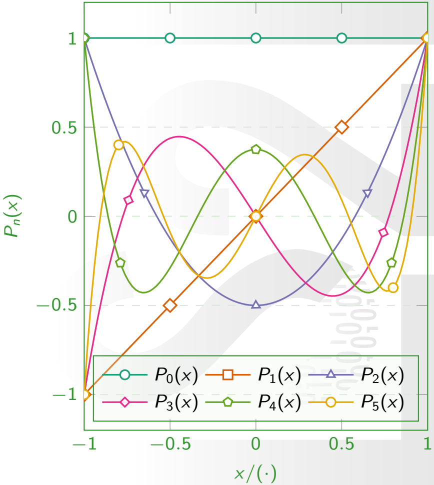
Numerical Integration

The quadrature weights can be obtained with help of the associated orthogonal *Legrende polynomials*

$$(n+1)\,P_n(x) = (2\,n+1)\,x\,P_{n-1}(x) - n\,P_{n-2}(x)\,,$$

with $P_0 = 1,\ P_1 = x$.
The *collocation point* x_i is the i-th root of P_n and the *weights* are given by the formula

$$w_i = \frac{2}{(1-x_i^2)\,[P_n'(x_i)]^2}\,.$$



Recap: How to Solve IVPs

Analytical Solution

$$\int \frac{\mathrm{d}y}{f(y)} = t + C\,,$$

Numerical Solution

Runge-Kutta:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i\, k_i\,,$$

$$k_i = f\Big(t_n + c_i\, h, y_n + \sum_{j=1}^{i-1} a_{ij}\, k_j\Big)\,.$$

Adams-Bashforth:

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p_{k-1}(t, f(t))\ \mathrm{d}t\,,$$

Numerical Integration

$$\int_{-1}^{1} f(x)\ \mathrm{d}x \approx \sum_{i=1}^{n} w_i\, f(x_i)\,,$$

Section 3

Collocation

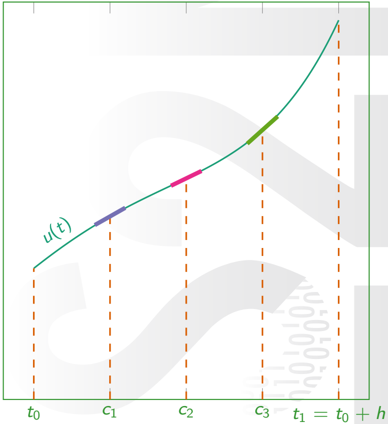
Derivation

Collocation

Let us construct a *one-step method* of given order of accuracy for the *first time step interval* $[t_0, t_0 + h]$. Let $0 \leq c_1 < c_2 < \cdots < c_s \leq 1$ be *distinct nodes* on the unit interval. The *collocation polynomial* $u(t) \in \mathbb{R}^n$ is a polynomial of degree s satisfying

$$u(t_0) = y_0$$
$$u'(t_0 + c_i h) = f(u(t_0 + c_i h)) \qquad i = 1, \ldots, s,$$

and the numerical solution of the *collocation method* over the interval $[t_0, t_0 + h]$ is given by $y_1 = u(t_0 + h)$.



We *construct a polynomial* that passes through y_0 and *agrees with the ODE* at s nodes on $[t_0, t_0 + h]$.

Derivation

Collocation

Let F_i, $i = 1, \dots, s$, be the values of the (as of yet undetermined) *interpolating polynomial* at the nodes

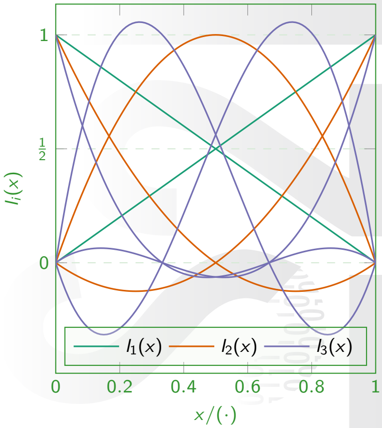
$$F_i := u'(t_0 + c_i\, h)\,.$$

We use *Lagrange's interpolation formula* to define the polynomial $u'(t)$ passing through these points

$$u'(t) = \sum_{i=1}^{s} F_i\, l_i\left(\frac{t - t_0}{h}\right), \qquad l_i(x) = \prod_{\substack{j=1\\ j\neq i}}^{s} \frac{x - c_j}{c_i - c_j}\,.$$

Integrating over the intervals $[0, c_i]$ gives

$$u(t_0 + c_i\, h) = y_0 + h \sum_{j=1}^{s} F_j \int_0^{c_i} l_j(x)\,\mathrm{d}x\,.$$



Derivation

Collocation

Denoting

$$a_{ij} := \int_0^{c_i} l_j(x)\,\mathrm{d}x\,, \qquad b_i := \int_0^1 l_i(x)\,\mathrm{d}x\,, \qquad i,j = 1,\dots,s\,,$$

and substituting in the collocation conditions gives

$$F_i = f(y_0 + h\sum_{j=1}^{s} a_{ij}\,F_j) \qquad i = 1,\dots,s\,.$$

Similarly integrating over $[0,1]$ yields

$$y_1 := y(t_0+h) = y_0 + h\sum_{i=1}^{s} F_i \int_0^1 l_j(t)\,\mathrm{d}t = y_0 + h\sum_{i=1}^{s} b_i\,F_i =: y_1\,.$$

This has the same form as the *Runge-Kutta* scheme, showing that the collocation scheme is *identical to a one-step implicit Runge-Kutta scheme*.

A Simple Example

Collocation

For illustration, let us solve the IVP on the interval $t \in [0, 1]$

$$y' = 3\,t^2\,, \qquad\qquad y(0) = 1\,.$$

The exact solution is

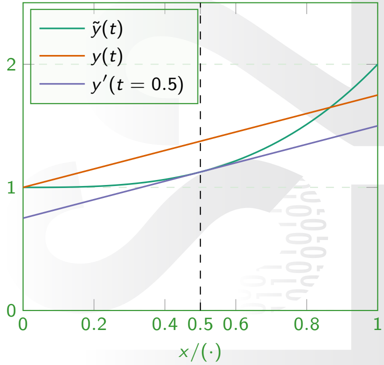
$$\tilde{y}(t) = 1 + t^3\,,$$

which we want to approximate with the first-degree polynomial

$$y(t) = a_0 + a_1\,t\,.$$

Since $y(0) = 1$, $a_0 = 1$, substituting gives $a_1 = 3\,t^2$. Requiring the collocation satisfied at $t = 0.5$ gives $a_1 = 0.75$ yields

$$y(t) = 1 + 0.75\,t\,.$$



A More Detailed Example

Collocation

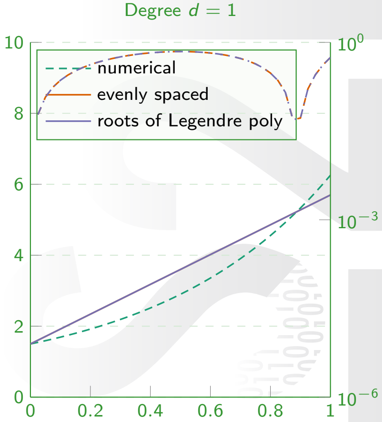
Let our IVP be given

$$y' = 1.75 \exp(1.75\, t)\,, \qquad y(0) = 1.5\,.$$

Our *collocation polynomial* shall be

$$u(t) = a_0 + a_1\, t + a_2\, t^2 + \cdots + a_d\, t^d\,.$$

d	c_i	a_0	a_1	a_2	a_3	a_4	a_5
1	lin	1.50	4.20	—	—	—	—
	LP	1.50	4.20	—	—	—	—
2	lin	1.50	0.65	3.73	—	—	—
	LP	1.50	0.91	3.83	—	—	—
3	lin	1.50	2.04	0.53	2.18	—	—
	LP	1.50	1.91	0.62	2.23	—	—
4	lin	1.50	1.70	1.80	0.29	0.95	—
	LP	1.50	1.73	1.73	0.32	0.97	—
5	lin	1.50	1.76	1.48	1.06	0.13	0.33
	LP	1.50	1.75	1.50	1.03	0.13	0.34



Generic First-Order ODE Collocation

Collocation

Assume we want to find the solution for

$$y' = f(t, y)\,, \qquad y(0) = y_0\,.$$

on the interval $t \in [0, 1]$ with the collocation polynomial

$$u(t) = \sum_{i=0}^{d} a_i\, t^i = \left[1, t, \dots, t^d\right]\, \alpha = \tau^\top \alpha\,.$$

In addition, we have

$$u'(t) = \left[0, 1, t, \dots, d\, t^{d-1}\right]\, \alpha = {\tau'}^\top \alpha\,,$$

Collocation method requires satisfying

$$u(0) = y(0) = y_0\,,$$
$$u'(t_0 + c_i) = f(t_0 + c_i, u(t_0 + c_i))\,,$$

at all *inner collocation points* $0 \le c_1 < \dots < c_i < \dots < c_d \le 1$. Substituting $u' = {\tau'}^\top \alpha$ yields

$$\begin{bmatrix} \tau^\top(t_0) \\ {\tau'}^\top(t_0 + c_1) \\ \vdots \\ {\tau'}^\top(t_0 + c_d) \end{bmatrix} \alpha = \begin{bmatrix} y_0 \\ f(t_0 + c_1, u(t_0 + c_1)) \\ \vdots \\ f(t_0 + c_d, u(t_0 + c_d)) \end{bmatrix}$$

which are $1 + d$ equations for the $d + 1$ unknowns of $u(t)$, respectively of α.

How to Use the Collocation Method

Collocation

To use the collocation method, a few facts have to be considered

- collocation function must satisfy the initial value
- collocation points must be well-chosen

polynomial	roots of shifted Legendre polynomial
splines	knots of Greville abscissae

- Choose between global or piecewise collocation
  - Piecewise reduces degree of local polynomial
  - Continuity of collocation function between intervals must be satisfied

To solve the ODE

$$y' = f(t, y)\,, \qquad y(t_0) = y_0\,,$$

remember that the *collocation function* $u(t)$ must satisfy

$$u(t_0) = y_0\,.$$

$$u'(t_0 + c_i\,h) = f(t_0 + c_i\,h, u(t_0 + c_i\,h))\,,$$

at all *inner collocation points* $t_0 + c_i\,h$. The resulting system of (non)linear equations can be solved with Newton-Raphson, Levenberg-Marquardt, etc.

Collocation Method vs. Numerical Integrators

Collocation Method

1. Requires more preparative work
2. Continuous solution of the IVP even between integration points
3. Interpolates the solution between t_n and t_{n+1}
4. Readily applicable to higher-order ODE
5. In principle applicable to any ODE/IVP
6. Transforms differential equation(s) into algebraic equation(s) (Can allow to define Jacobian in analytical form)
7. Comes in global and piecewise collocation (depending on collocation function)

Numerical Integrators

1. Only needs the ODE/IVP
2. Discretizes solution snap shots at integration points
3. Extrapolates solution from t_n to t_{n+1}
4. Needs state-reduction into first-order ODE
5. Handling of stiff ODEs is tricky

Section 4

Boundary Value Problems

Definition

Boundary Value Problems

A *boundary value problem* is a differential equation on an interval $[a, b]$ with *constraints on the interval boundary*:

$$\frac{\partial y}{\partial t} = f(t, y)\,,$$
$$y(a) = y_0\,,$$
$$y(b) = y_1\,.$$

The constraints may also be defined in terms of the derivatives i.e.,

$$y(t_0) = y_0\,,$$
$$y'(t_1) = y'_1\,.$$

Finite Difference Solution

Boundary Value Problems

Let us consider the *linear BVP* describing the steady state concentration profile $C(x)$ in the *reaction-diffusion* problem on the unit-interval $0 \leq x \leq 1$

$$\begin{aligned} \frac{\mathrm{d}^2 C}{\mathrm{d}x^2} - C &= 0\,, \\ C(x=0) &= 0\,, \\ \frac{\mathrm{d}\,C(x=1)}{\mathrm{d}x} &= 0\,. \end{aligned}$$

Its analytical solution is

$$C(x) = \frac{\exp(2-x) + \exp(x)}{1 + \exp(2)}$$

which we want to obatin using finite differences on $[0, 1]$.

Finite Difference Solution

Boundary Value Problems

We first *partition* the domain $[0,1]$ into *n equi-distant* sub-domains of length h i.e., $n\,h = 1$, and denote x_i the node at the interval end points.

Using *finite difference approximation*, the differential operator's discrete form reads

$$\frac{\mathrm{d}\,C_i}{\mathrm{d}x} = \frac{C_{i+1} - 2\,C_i + C_{i-1}}{h^2}\,, \qquad \frac{\mathrm{d}\,C_{n+1}}{\mathrm{d}x} = \frac{C_{n+2} - C_n}{2\,h}\,.$$

The boundary conditions at $x = 0$ and $x = 1$ give

$$C_1 = 1\,, \qquad C_{n+2} - C_n = 0\,.$$

Which allows us to write the BVP in matrix form and solve for $\boldsymbol{c}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -(2+h^2) & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -(2+h^2) & 1 & \dots & 0 & 0 & 0 \\ & & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & -(2+h^2) & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 & 1 \end{bmatrix} \boldsymbol{c} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}\,.$$

BVP as a Set of IVPs

Boundary Value Problems

Rewrite the BVP as a *set of IVP* with unknown initial slope s i.e.,

$$\frac{\partial^2 y}{\partial t^2} = f(t, y)\,,$$
$$y(a) = y_0\,,$$
$$y'(a) = s\,,$$

where s is to be found such that the residual of the IVP solution $\tilde{y}$ at $t = b$ vanishes

$$\boldsymbol{e}(s) := \tilde{y}(b, s) - y_1 = 0\,.$$

This approach is called a *shooting method* as we are shooting for the solution with varying initial conditions.
The *basic difficulty* with shooting is that a perfectly nice BVP can require the integration of IVPs that are *unstable*. That is, the *solution of a BVP* can be *insensitive* to changes in boundary values, yet the *solutions of the IVPs* of shooting are *sensitive* to changes in initial values.

The Correct Approach to Solving BVP

Boundary Value Problems

Most BVP solvers such as MATLAB's `bvp4c` , however, implement a *collocation method* for the solution of BVPs of the form

$$y' = f(t, y)\,, \qquad t \in [a, b]\,,$$

subject to *nonlinear, two-point boundary conditions*

$$g(y(a), y(b)) = 0\,.$$

The *solution approximation* $u(t)$ is a *continuous function* that is a *cubic polynomial* on each subinterval $[t_i, t_{i+1}]$ of the mesh $a = t_0 < t_1 < \cdots < t_n = b$. It *satisfies the boundary conditions*

$$g(u(a), u(b)) = 0\,,$$

and *satisfies the ODE* at *both ends* and the *mid-point* of each subinterval

$$\begin{aligned} u'(t_i) &= f(t_i, u(t_i))\,, \\ u'((t_i + t_{i+1})/2) &= f((t_i + t_{i+1})/2, u((t_i + t_{i+1})/2))\,, \\ u'(t_{i+1}) &= f(t_{i+1}, u(t_{i+1}))\,. \end{aligned}$$

Manual Collocation for BVP

Boundary Value Problems

To solve on an interval $[a, b]$ the BVP

$$y' = f(t, y)\,,$$
$$g(y(a), y(b)) = 0\,,$$

the *collocation function* $u(t)$ must satisfy

$$u'(t_0 + c_i\,h) = f(t_0 + c_i\,h, u(t_0 + c_i\,h))\,,$$

at all *inner collocation points* $t_0 + c_i\,h$ and must satisfy the boundary constraints function

$$g(u(a), u(b)) = 0\,.$$

The resulting system of (non)linear equations can be solved with Newton-Raphson, Levenberg-Marquardt, etc.

Today I Learned

- There is another technique to solve IVPs and BVPs namely the *collocation method* and a *collocation function* $u(t)$
- Using collocation is already being done in solving BVPs with commercially available software (MATLAB, scipy, MAPLE, Mathematica, ...)
- Collocation is very similar to modal decomposition for linear IVPs/BVPs
- Collocation can help turn my PDE into an ODE

EOF