pydgq v0.1.0 user manual

Juha Jeronen

April 21, 2017

juha.jeronen@jyu.fi

Department of Mathematical Information Technology
University of Jyväskylä

1 Introduction

The role of this user manual is to explain the algorithms available in *pydgq*, which is a Cython-accelerated library for solving ODE systems in Python. Computational kernels for right-hand sides (RHSs) can be written in both Python and Cython, whichever is more appropriate for a specific use case. For code examples, see the *test/* subdirectory in the source code distribution.

A common target for many numerical integration methods for ordinary differential equations (ODEs) is the first-order initial value problem

$$\frac{\partial u}{\partial t} = f(\, u(t),\, t\,)\,, \tag{1}$$

$$u(0) = u_0\,. \tag{2}$$

In this document, we will summarize, and comment on, several commonly used methods for the numerical solution of problem (1)–(2), concentrating especially on nonlinear problems, where we make no assumption about the structure of f, beyond the minimal necessary continuity for each considered method.

1.1 Extension to ODE systems

Consider the extension of (1)–(2) into the initial value problem of a semilinear[1] system of ordinary differential equations

$$\mathbf{M}\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\, \mathbf{u}(t),\, t\,)\,, \tag{3}$$

$$\mathbf{u}(0) = \mathbf{u}_0\,, \tag{4}$$

where $\mathbf{M}$ is a matrix, and $\mathbf{u}$ and $\mathbf{f}$ are vector-valued. One must account for the fact that for each component equation i, the load component f_i now depends on *all* components of $\mathbf{u}$, making the treatment of (3)–(4) slightly different from that of the one degree of freedom model problem (1)–(2).

Formally, provided that $\mathbf{M}$ is invertible (which is usually true in physically motivated problems, and was shown above for our specific problem), we may multiply (3) from the left by the inverse matrix $\mathbf{M}^{-1}$, obtaining

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{M}^{-1}\mathbf{f}(\, \mathbf{u}(t),\, t\,)\,, \tag{5}$$

In practice, this is obviously not possible beyond very small systems, due to the prohibitive cost (in terms of both time and storage) of matrix inversion.

[1] In this context, a semilinear system is linear with respect to the time derivative $\partial\mathbf{u}/\partial t$, but the load $\mathbf{f}$ is allowed to be a nonlinear function of $\mathbf{u}$. See e.g.
http://wiki.math.toronto.edu/DispersiveWiki/index.php/Semilinear
https://www.ma.utexas.edu/mediawiki/index.php/Semilinear_equations

A simple way to proceed from here, in practical numerics, is to recognize that we have explicit access to $\mathbf{f}$ (given a candidate $\mathbf{u}$; in nonlinear problems fixed point iteration is often needed to obtain a computable approximation), we also have $\mathbf{M}$, and we need $\mathbf{M}^{-1}\mathbf{f}$. We define the effective load

$$\mathbf{g}(\,\mathbf{u}(t),\,t\,) = \mathbf{M}^{-1}\mathbf{f}(\,\mathbf{u}(t),\,t\,)\,, \tag{6}$$

with the help of which, equation (5) becomes

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{g}(\,\mathbf{u}(t),\,t\,)\,. \tag{7}$$

Thus, given $\mathbf{g}$, we may in principle integrate each component equation separately (unknown u_i with the ith component of the load, g_i), using methods designed for a single ODE. Note that as was already cautioned, we need *all* components of $\mathbf{u}$ for the evaluation of each load component g_i; this provides the connection between the equations in the ODE system (3).

Multiplying (6) from the left by $\mathbf{M}$, we have

$$\mathbf{M}\,\mathbf{g} = \mathbf{f}\,. \tag{8}$$

Thus, by solving the linear equation system (8) for the unknown vector $\mathbf{g}$, we obtain the numerical value of $\mathbf{M}^{-1}\mathbf{f}$ whenever it is needed. This turns $\mathbf{M}^{-1}(\dots)$ into a linear operator, which is much cheaper to evaluate than matrix inversion.

It should be recognized, though, that solving (8) may be expensive especially for nonlinear problems, where (8) may need to be solved several times for each iteration of the fixed point loop. (Equation (5) is typically converted into integral form; the RHS must be evaluated at the quadrature points when the integral is approximated numerically. During the discussions of fixed point methods and discontinuous Galerkin, below, we will see this in practice.)

2 Explicit methods

Explicit methods supported by the solver are as follows.

2.1 Explicit Runge–Kutta methods

Perhaps one of the best-known families of classical time integrators are the Runge–Kutta (RK) methods.[2] Some useful explicit RK methods have been collected below. Instead of collecting the coefficients into a Butcher tableau, the methods are here presented in an algorithmic form, ready for implementation into numerical codes.

Below, for $\mathbf{u}$ and t, the subscripts n and $n+1$ refer to timestep numbers; e.g. $\mathbf{u}_n \equiv \mathbf{u}(t = t_n)$. The subscripts for the $\mathbf{k}_j$ simply label the quantities.

We will list the methods in a form directly suitable for ODE systems with $\mathbf{M}$ a unit matrix. If this is not the case, observe that — by the form of the problem, equation (1) — the $\mathbf{k}_j = \mathbf{f}(\dots)$ represent approximations to the derivative. Hence, if the ODE system has a nontrivial mass matrix $\mathbf{M}$, evaluating $\mathbf{f}$ actually gives

$$\mathbf{M}\,\mathbf{k}_j = \mathbf{f}(\dots)$$

and $\mathbf{k}_j$ can be obtained by solving this linear equation system. This solution process must be repeated for each j (in this setting e.g. RK4 requires solving four linear equation systems per timestep). Earlier comments about tricks to speed up the solution process apply.

Note that explicit RK methods cannot be unconditionally stable.[3]

[2] For an introduction and a list, see e.g.
https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods
https://en.wikipedia.org/wiki/List_of_Runge%E2%80%93Kutta_methods
http://www.scholarpedia.org/article/Runge-Kutta_methods
Some of this material is summarized here.

[3] This is noted in e.g.
https://en.wikipedia.org/wiki/Stiff_equation#General_theory

RK1 The first-order explicit RK method is the forward Euler method:

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n,\ t_n)\ , \tag{9}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\, \mathbf{k}_1\ . \tag{10}$$

Due to its tendency for extreme numerical instability, this method is not practically useful, and we will not consider it further.

RK2 The second-order explicit RK method is given in parametric form by

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n,\ t_n)\ , \tag{11}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{u}_n + \beta\, \Delta t\, \mathbf{k}_1,\ t_n + \beta\, \Delta t)\ , \tag{12}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \left[(1 - \frac{1}{2\beta})\mathbf{k}_1 + \frac{1}{2\beta}\mathbf{k}_2 \right]\ . \tag{13}$$

Classical choices for β are $1/2$ (explicit midpoint method), $2/3$ (Ralston's method), and 1 (Heun's method, also known as the explicit trapezoid rule).

RK3 Kutta's third-order method is

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n,\ t_n)\ , \tag{14}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{u}_n + \frac{\Delta t}{2}\mathbf{k}_1,\ t_n + \frac{\Delta t}{2})\ , \tag{15}$$

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{u}_n - \Delta t\, \mathbf{k}_1 + 2\Delta t\, \mathbf{k}_2,\ t_n + \Delta t)\ , \tag{16}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3)\ . \tag{17}$$

RK4 The classical fourth-order Runge–Kutta (RK4), which is the most popular of the RK methods, is

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n,\ t_n)\ , \tag{18}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{u}_n + \frac{\Delta t}{2}\mathbf{k}_1,\ t_n + \frac{\Delta t}{2})\ , \tag{19}$$

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{u}_n + \frac{\Delta t}{2}\mathbf{k}_2,\ t_n + \frac{\Delta t}{2})\ , \tag{20}$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{u}_n + \Delta t\, \mathbf{k}_3,\ t_n + \Delta t)\ , \tag{21}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\ . \tag{22}$$

RK4 alternative ("3/8 rule") There is an alternative, less often used fourth-order explicit RK method, that was also proposed by Kutta:

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n,\ t_n)\ , \tag{23}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{u}_n + \frac{1}{3}\Delta t\, \mathbf{k}_1,\ t_n + \frac{1}{3}\Delta t)\ , \tag{24}$$

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{u}_n - \frac{1}{3}\Delta t\, \mathbf{k}_1 + \Delta t\, \mathbf{k}_2,\ t_n + \frac{2}{3}\Delta t)\ , \tag{25}$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{u}_n + \Delta t\, \mathbf{k}_1 - \Delta t\, \mathbf{k}_2 + \Delta t\, \mathbf{k}_3,\ t_n + \Delta t)\ , \tag{26}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{\Delta t}{8}(\mathbf{k}_1 + 3\mathbf{k}_2 + 3\mathbf{k}_3 + \mathbf{k}_4)\ . \tag{27}$$

2.2 Symplectic Euler (a.k.a. semi-implicit Euler)

This is a symplectic $O(\Delta t)$ method due to Niiranen[4], closely related to the classical explicit and implicit Euler methods. From the computational viewpoint, the method is explicit; hence the placement in this section.

For this method only, let us restrict our consideration to state vectors of the form

$$\mathbf{u} = (q_1,\, \dot{q}_1,\, q_2,\, \dot{q}_2,\, \ldots q_n,\, \dot{q}_n)\ ,$$

which result from second-order problems when they are reduced into first-order form. The subscripts for q_j and $\dot{q}_j$ refer to vector components. The timestep update is performed with the following sequence of operations:

$$\begin{aligned}
\mathbf{k} &\leftarrow f(\mathbf{u}_n,\, t_n) \\
\mathbf{u} &\leftarrow \mathbf{u}_n \\
u_{2j} &\leftarrow u_{2j} + \Delta t\, k_{2j}\ , \quad j = 1, 2, \ldots, n \\
u_{2j-1} &\leftarrow u_{2j-1} + \Delta t\, u_{2j}\ , \quad j = 1, 2, \ldots, n \\
\mathbf{u}_{n+1} &\leftarrow \mathbf{u}\ .
\end{aligned}$$

The subscripts for the non-bolded quantities refer to vector components. In other words, we first compute $\ddot{q}_j = k_{2j}$, and update $\dot{q}_j = u_{2j}$ using Forward Euler. Then, we use these updated values (which now correspond to the velocities at the end of the timestep) to update $q_j = u_{2j-1}$ by Backward Euler. Note that k_{2j-1}, containing $\dot{q}_j$ at the beginning of the timestep, are not needed.

3 Classical implicit methods

Classical implicit methods supported by the solver are as follows. Note that also dG and cG are implicit; these are treated in their own sections further below.

3.1 Implicit midpoint rule

In principle, the classical implicit midpoint rule (IMR) works as follows:

$$\mathbf{k} = \mathbf{f}(\mathbf{u}_{n+1/2},\, t_{n+1/2})\ , \tag{28}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\, \mathbf{k}\ , \tag{29}$$

where the half-timestep ("midpoint") value of the state vector is taken as the linear interpolant

$$\mathbf{u}_{n+1/2} \approx \frac{1}{2}(\mathbf{u}_n + \mathbf{u}_{n+1})\ . \tag{30}$$

The accuracy of the method is $O(\,(\Delta t)^2\,)$; furthermore, it is known that the error tends to oscillate around zero. The method is symplectic and approximately conserves energy (which are useful properties for problems in Hamiltonian mechanics). The method is not unconditionally stable; a problem-specific maximum timestep size exists above which numerical stability is lost.

Consider equation (30). As in all implicit methods, there is the practical issue that $\mathbf{u}_{n+1}$ is unknown (indeed, the whole aim of the method is to compute it). For linear problems, the standard answer is to set up a linear equation system, using (30) in (28), expanding $\mathbf{f}$ (for each specific problem), and then moving terms with $\mathbf{u}_{n+1}$ to the left-hand side and $\mathbf{u}_n$ (and knowns) to the right-hand side. The resulting linear equation system, when numerically solved, gives $\mathbf{u}_{n+1}$.

From a computational viewpoint, in the linear case, the need for a computable representation for $\mathbf{u}_{n+1}$ has thus been sidestepped by rewriting the problem into a standard form, where $\mathbf{u}_{n+1}$ remains unknown, but for which standard solvers are available. Obviously, in the general case of an arbitrary nonlinear problem, such an approach of reduction into a standard form is not possible (since each nonlinear problem class is different).

More relevant to our case is a general approach for arbitrary nonlinear problems. We replace $\mathbf{u}_{n+1}$ in (30) with some computable approximation $\mathbf{u}^*$:

$$\mathbf{u}_{n+1/2} \approx \frac{1}{2}(\mathbf{u}_n + \mathbf{u}^*)\ . \tag{31}$$

[4]Niiranen, Jouko: Fast and accurate symmetric Euler algorithm for electromechanical simulations. Proceedings of the Electrimacs'99, Sept. 14-16, 1999 Lisboa, Portugal, Vol. 1, pages 71 - 78.

The result is then refined by iteration. Given an initial guess for $\mathbf{u}^*$, we evaluate the approximate $\mathbf{u}_{n+1/2}$ from (31), insert it into (28), and evaluate (29), interpreting the left-hand side (the output) as the updated value for $\mathbf{u}^*$. Then we repeat, obtaining the corresponding updated $\mathbf{u}_{n+1/2}$ from (31), and so on. This iteration continues until $\mathbf{u}^* (\approx \mathbf{u}_{n+1})$ has converged, or a prescribed maximum number of iterations is reached.

The easiest choice for the initial guess for $\mathbf{u}^*$ is to use $\mathbf{u}_n$. The requirement of contractivity in the Banach fixed point theorem is the theoretical reason behind the upper limit for timestep size in this iterative version of IMR for nonlinear problems; contrast this with von Neumann stability analysis for the version specialized for linear problems. See section 3.3.

Algorithmic summary of IMR For each timestep n, do the following:

1. Initialize:
$$\begin{aligned} \mathbf{u}_{n+1}^{(0)} &\leftarrow \mathbf{u}_n \\ i &\leftarrow 0 \end{aligned}$$
2. Update:
$$\begin{aligned} \mathbf{u}_{n+1/2} &\leftarrow \frac{1}{2}(\mathbf{u}_n + \mathbf{u}_{n+1}^{(i)}) \\ \mathbf{k} &\leftarrow \mathbf{f}(\mathbf{u}_{n+1/2},\ t_{n+1/2}) \\ i &\leftarrow i+1 \\ \mathbf{u}_{n+1}^{(i)} &\leftarrow \mathbf{u}_n + \Delta t\, \mathbf{k} \end{aligned}$$
3. If $\mathbf{u}_{n+1}^{(i)}$ is "close" to $\mathbf{u}_{n+1}^{(i-1)}$, the method has converged and the solution is complete. Jump to step 6.
4. If $i = i_{\max}$, the method did not converge. No solution obtained; raise an exception.
5. Next iteration. Continue from step 2.
6. Set the solution obtained as
$$\mathbf{u}_{n+1} \leftarrow \mathbf{u}_{n+1}^{(i)}$$

Notes:

- The superscript in the parentheses denotes the iteration number.
- At step 3 and later, the variable i holds the current number of iterations taken.
- On failed exit, step 4, sometimes the closest obtained solution may be useful even though it did not converge (e.g. if the tolerance is set all the way down to floating point equality; a numerical limit cycle due to roundoff and cancellation may prevent exact convergence).
- In practice, it does not make sense to store the intermediate iterates $\mathbf{u}_{n+1}^{(i)}$ (unless for technical purposes such as convergence monitoring or debugging); only the latest and previous iterates are needed.

3.2 Backward Euler

Sometimes unconditional stability and high numerical dissipation may be desirable properties. The simplest method that fits this target is classical backward Euler (BE; also called implicit Euler). We may formulate it as

$$\mathbf{k} = \mathbf{f}(\mathbf{u}_{n+1},\ t_{n+1}) \tag{32}$$
$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\, \mathbf{k}\ , \tag{33}$$

The iterative approximation is simply

$$\mathbf{u}_{n+1} \approx \mathbf{u}^*\ , \tag{34}$$

and the iteration proceeds using the same procedure as above. The accuracy is $O(\Delta t)$.

If the fixed-point iteration process is used to implement BE for a general nonlinear problem, the initial guess for $\mathbf{u}^*$ (at each timestep) must be chosen such that it falls into the basin of attraction of the sought-after fixed point. This introduces a practical issue, because the contractivity of the iterative procedure (34), (32), (33) will likely fail above some critical timestep size for this iterative BE for nonlinear problems (where the initial guess becomes too inaccurate), even though the classical linear version of BE poses no limitation on timestep size (the classical von Neumann stability analysis result being a-stability).See section 3.3.

Algorithmic summary of BE For each timestep n, do the following:

1. Initialize:
$$\begin{aligned} \mathbf{u}_{n+1}^{(0)} &\leftarrow \mathbf{u}_n \\ i &\leftarrow 0 \end{aligned}$$
2. Update:
$$\begin{aligned} \mathbf{k} &\leftarrow f(\mathbf{u}_{n+1}^{(i)},\ t_{n+1}) \\ i &\leftarrow i+1 \\ \mathbf{u}_{n+1}^{(i)} &\leftarrow \mathbf{u}_n + \Delta t\, \mathbf{k} \end{aligned}$$
3. If $\mathbf{u}_{n+1}^{(i)}$ is "close" to $\mathbf{u}_{n+1}^{(i-1)}$, the method has converged and the solution is complete. Jump to step 6.
4. If $i = i_{\max}$, the method did not converge. No solution obtained; raise an exception.
5. Next iteration. Continue from step 2.
6. Set the solution obtained as
$$\mathbf{u}_{n+1} \leftarrow \mathbf{u}_{n+1}^{(i)}$$

The same algorithmic notes apply as to IMR.

3.3 Implicit methods and the Banach fixed point theorem

As an example, let us again consider the iterative version of IMR presented above. Mathematically, the iteration procedure (31), (28), (29) is based on contractive self-maps on metric spaces. One iteration of the procedure maps a metric space to itself. Recall that a metric space[5] is an ordered pair (X, d), where X is a set, and d is a metric, defining how to measure distance between any two points in X.

The Banach fixed point theorem states that on a metric space, if a self-map is contractive, then a fixed point exists for the map, and moreover, the fixed point is unique. In symbols, let (X, d) be a metric space. For a contractive self-map $T : X \to X$, there exists a unique point x^* such that $T(x^*) = x^*$. The mapping T is contractive if it satisfies the Lipschitz condition $d(\, T(x_1),\, T(x_2)\,) \le q\, d(x_1, x_2)$, where $q \in [0, 1)$ is the Lipschitz constant (of the mapping T), and $x_1, x_2 \in X$ are arbitrary.

Note that Lipschitz continuity is a fairly stringent requirement, and it may also depend on the domain of the map. For example, in one dimension, the map $x \mapsto x^2$ is Lipschitz on any finite interval, but on the whole of $\mathbb{R}$ it is not. Note also that this particular example is contractive only in the set $|x| \le 1$.

This topic is closely related to the Picard–Lindelöf theorem[6], which is concerned with the existence and uniqueness of solutions to the first-order initial value problem. Indeed, the Banach fixed point theorem is invoked in the proof of the Picard–Lindelöf theorem.

The iterative IMR procedure presented above is also essentially a discrete application of the Picard–Lindelöf idea of recasting the first-order initial value problem (1)–(2) as a fixed-point problem for an integral operator. Observe that equation (29) is a discrete approximation of the integral

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\, \mathbf{u}(t),\, t\,)\, \mathrm{d}t\,, \tag{35}$$

[5] https://en.wikipedia.org/wiki/Metric_space#Definition

[6] https://en.wikipedia.org/wiki/Picard%E2%80%93Lindel%C3%B6f_theorem

which obviously comes from the fundamental theorem of calculus, after using the standard form of the problem, equation (1), to represent the derivative. Equation (35) is the integral that is considered in the Picard–Lindelöf theorem. Note that the initial value $\mathbf{u}_n$ in (35) is effectively a constant, but the $\mathbf{u}(t)$ in the integrand may need to be evaluated anywhere in the interval $[t_n, t_{n+1}]$, depending on the details of the numerical method.

The same observation holds for the formula for $\mathbf{u}_{n+1}$ in the backward Euler method, given in equation (33). In the theoretical framework, as an implicit method, it belongs in the same category as IMR.

The observation also holds for the formulas for $\mathbf{w}_{n+1}$ in the explicit RK methods, namely equations (10), (13), (17), (22) and (27). Being explicit methods, however, the approximation to the integral is there constructed in such a way that no "future data" is needed. For explicit RK methods, although they too can be thought of as being based on rewriting the problem in the integral form (35), there is no need to invoke the fixed point theorem, since all needed quantities can be computed by an explicit sequence of operations.

To produce a practical numerical method for (35), two things must be specified. First, because the $\mathbf{u}(t)$ in the integrand is unknown, it must be modeled. A function (or algorithm) is specified to construct an approximate $\widetilde{\mathbf{u}}(t) \approx \mathbf{u}(t)$, given some discrete set of data, e.g. point values at a set of points in the interval $[t_n, t_{n+1}]$, or alternatively, coefficients for a Galerkin representation. Thus, at each timestep, we seek to solve the approximate problem

$$\mathbf{u}_{n+1} \approx \mathbf{u}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\, \widetilde{\mathbf{u}}(t),\, t\,)\, \mathrm{d}t\,. \tag{36}$$

In any reasonable model, at the beginning of the timestep, $\widetilde{\mathbf{u}}(t_n) = \mathbf{u}_n$. Of course, in an implicit method, also $\widetilde{\mathbf{u}}(t_{n+1}) = \mathbf{u}_{n+1}$.

In this view, the choice of the model $\widetilde{\mathbf{u}}(t)$ is the critical difference between explicit and implicit methods. In explicit methods, the model is constructed such that for any fixed $\tau \in (t_n, t_{n+1}]$, the function value $\widetilde{\mathbf{u}}(\tau)$ is based on information at $t < \tau$ (i.e. "old" information) only. Implicit methods lift this restriction, allowing (for any τ) the use of information from anywhere in $[t_n, t_{n+1}]$.

Secondly, an algorithm to evaluate the integral must be specified. A typical choice is a quadrature formula, approximating the integral as a sum of weighted values of the integrand at a pre-selected set of points.

Now, to bring (36) into a form where fixed point theorems are applicable, we define the self-map (on the space where the instantaneous field values $\mathbf{u}_{n+1}$ live)

$$\mathbf{u}_{n+1}^{(i+1)} = \mathbf{u}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\, \widetilde{\mathbf{u}}^{(i)}(t),\, t\,)\, \mathrm{d}t\,, \tag{37}$$

where the superscript in parentheses indexes the iterate sequence. Considering the class of methods which approximate the integral as a quadrature — indeed, such as IMR and BE — we further approximate

$$\mathbf{u}_{n+1}^{(i+1)} \approx \mathbf{u}_n + \Delta t \sum_{k=1}^{N} q_k\, \mathbf{f}(\, \widetilde{\mathbf{u}}^{(i)}(\tau_k),\, \tau_k\,)\,, \quad \tau_k \equiv t_n + p_k \Delta t\,, \tag{38}$$

where N is the number of quadrature points, q_k are their weights and $p_k \in [0,1]$ their positions on the standard unit interval.

In collocation methods (i.e. methods considering function values on a discrete set of points), the model iterate $\widetilde{\mathbf{u}}^{(i)}(t)$ depends on $\mathbf{u}_n$, $\mathbf{u}_{n+1}^{(i)}$, and possibly a set of in-between point values that must be determined internally. In the case of IMR and BE, only the endpoint values are used. For these methods, we choose $\widetilde{\mathbf{u}}^{(i)}(t)$ as the linear interpolant (expressed on the standard unit interval)

$$\widetilde{\mathbf{u}}^{(i)}(p) = (1-p)\, \mathbf{u}_n + p\, \mathbf{u}_{n+1}^{(i)}\,, \quad p \in [0,1]\,, \tag{39}$$

transforming the quadrature (38) into

$$\mathbf{u}_{n+1}^{(i+1)} = \mathbf{u}_n + \Delta t \sum_{k=1}^{N} q_k\, \mathbf{f}(\, (1-p_k)\, \mathbf{u}_n + p_k\, \mathbf{u}_{n+1}^{(i)}\,,\, \tau_k\,)\,, \quad \tau_k \equiv t_n + p_k \Delta t\,. \tag{40}$$

In IMR, the quadrature is simply the midpoint rule $N = 1$, $p_1 = 1/2$, $q_1 = 1$, leading to

$$\mathbf{u}_{n+1}^{(i+1)} = \mathbf{u}_n + \Delta t\, \mathbf{f}(\, \frac{1}{2}\mathbf{u}_n + \frac{1}{2}\mathbf{u}_{n+1}^{(i)}\,,\, t_n + \frac{1}{2}\Delta t\,)\,, \tag{41}$$

which matches (28), (29) and (31), as expected.

One the left-hand side of (40) we have only $i+1$, and on the right-hand side only i. Hence (40) can be explicitly iterated to produce a sequence of values $\mathbf{u}_{n+1}^{(i+1)}$, $i = 0, 1, 2, \ldots$; it is the mapping $T: \ \mathbf{u}_{n+1}^{(i)} \mapsto \mathbf{u}_{n+1}^{(i+1)}$ for which a fixed point is being sought. By the Banach fixed point theorem, if T is contractive (i.e. the mapped points $T(\mathbf{u}_{n+1}^{(i_1)})$ and $T(\mathbf{u}_{n+1}^{(i_2)})$ are closer together than the original points $\mathbf{u}_{n+1}^{(i_1)}$ and $\mathbf{u}_{n+1}^{(i_2)}$, for any choice of original points), it has a unique fixed point. Moreover, the fixed point is the solution $\mathbf{u}_{n+1}$.

Note that in (40), $\mathbf{f}$ and $\mathbf{u}_n$ can be considered fixed (given by the problem setup and the initial condition, respectively), whereas $\mathbf{u}_{n+1}^{(0)}$ (the initial guess) and Δt (the timestep size) are free.

As a final remark to equation (40), the technique of relaxation is sometimes offered as a convergence aid for fixed point iteration methods. For example, for IMR, we replace the update procedure (41) with this modified version:

$$\widehat{\mathbf{u}}_{n+1}^{(i+1)} = \mathbf{u}_n + \Delta t\, \mathbf{f}(\, \frac{1}{2}\mathbf{u}_n + \frac{1}{2}\mathbf{u}_{n+1}^{(i)}\,,\, t_n + \frac{1}{2}\Delta t\,)\,, \tag{42}$$

$$\mathbf{u}_{n+1}^{(i+1)} = (1-\alpha)\mathbf{u}_{n+1}^{(i)} + \alpha\widehat{\mathbf{u}}_{n+1}^{(i+1)}\,, \quad \alpha \in (0,1) \text{ given.} \tag{43}$$

The first equation is the same as (41); we have simply renamed the left-hand side. Subtracting $\mathbf{u}_{n+1}^{(i)}$ from (43), we see that

$$\mathbf{u}_{n+1}^{(i+1)} - \mathbf{u}_{n+1}^{(i)} = \alpha(\widehat{\mathbf{u}}_{n+1}^{(i+1)} - \mathbf{u}_{n+1}^{(i)})\,. \tag{44}$$

The application of the classical relaxation technique thus makes the distance between successive iterates smaller. Compared to the version without relaxation, this obviously has two effects: contractivity is improved, but on the other hand, convergence requires more iterations, because the steps are smaller. Despite the slower convergence, as is well known, this may be useful in cases where contractivity would otherwise fail. (The same remark applies to any iterative implicit method, replacing (42) with the appropriate update formula.)

IMR is often presented as a tool specifically for linear problems, trading away generality for power. Interestingly, this also leads to developing the method into a completely different direction — and even giving it a different theoretical origin, in finite differences. (Although (41) can be re-interpreted as a finite difference expression, equation (35) — or indeed the whole derivation of (41) — makes no use of finite differences!)

Specialization to linear problems enables additional formal manipulations to be performed, leading, as is well known, to linear equation systems and von Neumann stability analysis (which gives the classical upper limit on timestep size). These two versions of IMR do not have much in common, beside the core idea of seeking an implicit approximation at the midpoint — which is formally encoded into equations (28)–(30), common to both forms.

We note that the speed of convergence of the sequence of fixed point iterates x_n (unrelated to the use of n to index the timestep above) is characterized by the following equivalent expressions[7]:

$$d(x^*, x_n) \leq \frac{q^n}{1-q} d(x_1, x_0)\,, \tag{45}$$

$$d(x^*, x_{n+1}) \leq \frac{q}{1-q} d(x_{n+1}, x_n)\,, \tag{46}$$

$$d(x^*, x_{n+1}) \leq q\, d(x^*, x_n)\,. \tag{47}$$

Any value of $q \in [0, 1)$ satisfying these relations for a given contractive mapping T is a Lipschitz constant of T; the smallest possible value is the best Lipschitz constant of T.

Relations (45)–(47) suggest that the points x_n form a geometric sequence converging toward the limit point x^*. (This also suggests a linear convergence rate, as is well known for fixed point procedures; the geometric nature of the sequence, roughly speaking, adds a constant number of correct digits in each iteration.)

If the sequence is (at least approximately) geometric, q can be approximated in a simple manner as

$$q \approx \frac{d(x_{n+1}, x_n)}{d(x_n, x_{n-1})}\,. \tag{48}$$

The expression (48) should stay approximately constant throughout the iteration. One can then use (45) or (46) to estimate the remaining distance to the (unknown) fixed point, providing a computable *error indicator* that can be used for error control. (This is not an *error estimator*, since (48) makes no guarantees.) The test (48) also tells us when contractivity has failed (and hence, when the sequence x_n cannot be guaranteed to converge).

[7] https://en.wikipedia.org/wiki/Banach_fixed-point_theorem

Contractivity may be local. Typically, for fixed-point procedures arising in numerical integration of differential equations, contractivity occurs if the timestep Δt is "small enough" and/or the initial guess for $\mathbf{u}^*$ (iterate x_0) is "close enough" to the solution. In this case, the set X must be chosen appropriately such that T remains contractive within it; then X can be considered the "basin of attraction"[8] of the fixed point.

The implied comparison with dynamical systems (which typically have several basins of attraction) may however be a bit misleading, so the following must be emphasized. The Picard–Lindelöf theorem states that the solution of the first-order initial value problem $\partial u/\partial t = f(\,u(t),\,t\,)$, $u(t_0) = u_0$ exists *and is unique* on an interval $t \in [t_0 - \varepsilon,\, t_0 + \varepsilon]$ for some $\varepsilon > 0$, if the load function f is uniformly Lipschitz continuous in u (i.e. Lipschitz continuous with the same constant independent of t), and continuous in t. (Concerning the design of numerical methods, it may be of interest to note that in the proof, the Banach fixed point theorem is invoked with the initial iterate set to $u(t) = u_0$ for the whole interval.)

Obviously, different initial values $\mathbf{u}_n$ will typically lead to different solutions $\mathbf{u}_{n+1}$ (otherwise the problem would not be very interesting). What the Banach and PL theorems say is only that *for a given initial value* ($\mathbf{u}_n$), the choice of the initial iterate (x_0, initial guess for $\mathbf{u}^*$) within the set X does not matter; if contractivity holds, the unique solution $\mathbf{u}_{n+1}$ (corresponding to the given initial $\mathbf{u}_n$) will be found.

This is also the theoretical origin of the upper limit on timestep size for this iterative version of IMR. Contrast this with the version of IMR specialized for linear problems, where the upper limit arises via von Neumann stability analysis. There is no requirement for these maximum possible timestep sizes to be the same!

What typically happens outside the basin of attraction is that the contractivity of the iterative procedure is violated, and the sequence fails to converge (unless, by pure luck, one of the non-converging iterates happens to fall into the basin of attraction).

Thus we arrive at an important practical consideration: unless the iterative procedure (31), (28), (29) has a globally contractive self-map (for any given problem), the initial guess for $\mathbf{u}^*$ must be made to fall into the basin of attraction of the sought-after fixed point $\mathbf{u}_{n+1}$ of the procedure. This sets an upper limit on the timestep size for iterative IMR, and similarly for other implicit methods based on the application of the Banach fixed point theorem.

If we use an explicit integration method to generate the initial guess, this basin-of-attraction consideration is separate from the usual stability limit (with regard to timestep size) of that method[9], since that method will not be used for the actual integration. The stability limit only tells us whether integration *using that method* would remain stable. But here, the relevant question is entirely different: whether our iterative procedure, with the given initial guess (however inaccurate or even qualitatively wrong), will converge to the fixed point $\mathbf{u}_{n+1}$ or not. For nonlinear problems, this question is highly nontrivial.

Is there only one basin of attraction? By the Picard–Lindelöf theorem, if the timestep is "small enough", existence and uniqueness hold for the solution of the first-order initial value problem. Thus, it follows that, with Δt small enough, and with $\mathbf{u}_n$ and $\mathbf{f}$ kept fixed (i.e. under the constraint that we keep the problem fixed), there can be no competing basins of attraction leading to other solutions. However, this does assume that the approximations made when constructing the numerical integration method have not changed the qualitative behavior, which may not be the case.

Finally, what about global existence and uniqueness of the solution on $t \in [0, t_\mathrm{f}]$, where t_f is the end time of the simulation (t_f may be large)? The key observation here is that we may treat each timestep as a separate initial value problem. For example, on the interval $t \in [t_n, t_{n+1}]$ (of length Δt), we use the initial condition $\mathbf{u}(t_n) = \mathbf{u}_n$, and the duration of integration (for which we need existence and uniqueness of the solution, considering only this subproblem) is only Δt. Then, performing the integration (i.e. solving one timestep), we obtain the final value $\mathbf{u}(t_{n+1}) = \mathbf{u}_{n+1}$; this is the initial condition for the next timestep. Thus, even if the ε in the Picard–Lindelöf theorem sometimes turns out to be small, this does not pose a problem for the global existence and uniqueness of the solution.

[8] http://www.scholarpedia.org/article/Basin_of_attraction

[9] This is true regardless of how the stability limit was derived. (Keep in mind that the classical von Neumann stability analysis is not directly applicable to nonlinear problems.)

4 Discontinuous Galerkin

The family of discontinuous Galerkin (dG) methods seeks a weak solution of the initial value problem (1)–(2). We begin by multiplying (1) by a test function $w(t)$, and integrate in time from 0 to the simulation end time t_f:

$$\int_0^{t_\mathrm{f}} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \int_0^{t_\mathrm{f}} f(\,u(t),\,t\,)\,w \,\mathrm{d}t\,, \quad \forall w\,. \tag{49}$$

The quantifier is taken over admissible test functions w (in some technically appropriate sense).

Unlike in finite elements, which are typically used for second-order boundary value problems (or initial boundary value problems), integration by parts would not do us much good here, since (49) only needs the first derivative. In a pure Galerkin method, the same set of functions is used as both the basis functions of u, and as the test functions w. In such a setting, moving the derivative from u to w would not gain us anything.

Instead, looking at what this integration buys us, the idea is to relax the requirements on u (and w): we seek a solution in C^{-1}, i.e. allow u (and w) to have finite jumps (discontinuities) across the element boundaries. This complicates the method slightly, because we must account for "derivatives of finite discontinuities", i.e. Dirac deltas.

In order to develop the argument further, some background is first needed. The following five sections cover the necessary topics. In the first two, we will briefly review the basics. The rest are focused on issues getting successively closer to the problem at hand. We will continue with the problem itself in section 4.6.

4.1 Piecewise continuous functions and jumps

Let $v(t)$ be a left-continuous or right-continuous function with a jump at t_0,

$$v(t) = \begin{cases} v_-(t)\,, & t \le t_0 \text{ (if L.C.)}\,, \quad t < t_0 \text{ (if R.C.)}\,, \\ v_+(t)\,, & t > t_0 \text{ (if L.C.)}\,, \quad t \ge t_0 \text{ (if R.C.)}\,, \end{cases} \tag{50}$$

where $v_-(t)$ and $v_+(t)$ are continuous (i.e. at least C^0) functions.

According to the theory of distributions (generalized functions), the derivative of $v(t)$ can be written as

$$\frac{\partial v}{\partial t}(t) = \begin{cases} \dfrac{\partial v_-}{\partial t}(t)\,, & t < t_0\,, \\ \lim\limits_{\varepsilon \to 0^+} \left(v_+(t_0+\varepsilon) - v_-(t_0-\varepsilon)\right) \delta(t-t_0)\,, & t = t_0\,, \\ \dfrac{\partial v_+}{\partial t}(t)\,, & t > t_0\,, \end{cases} \tag{51}$$

where $\delta(\dots)$ is the Dirac delta distribution. Generalization of (50)–(51) to any finite number of discontinuities is obvious.

To shorten the notation, the limit expression in (51) is often abbreviated by defining the *jump operator* $[\dots]$, which maps functions to functions:

$$[F](t) := \lim_{\varepsilon \to 0^+} \left(F(t+\varepsilon) - F(t-\varepsilon)\right)\,. \tag{52}$$

Note that if F is continuous at a point t, then at that point $[F](t) = F(t)$; for continuous functions $[\dots]$ is the identity operator. In general, it is of course assumed that F is sufficiently mildly behaved so that the left and right limits exist at the point t, so that the definition (52) actually makes sense. (This of course is true for $v(t)$ defined above.)

Inserting (52) in (51), we have the (hopefully) more readable notation

$$\frac{\partial v}{\partial t}(t) = \begin{cases} \dfrac{\partial v_-}{\partial t}(t)\,, & t < t_0\,, \\ [v](t_0)\,\delta(t-t_0)\,, & t = t_0\,, \\ \dfrac{\partial v_+}{\partial t}(t)\,, & t > t_0\,, \end{cases}\,. \tag{53}$$

Observe that $[v](t_0)$ can be treated simply as a number, which is in principle explicitly obtainable from the definitions (52) and (50) (just take the right and left limits of $v(t)$ at t_0 and subtract them). Thus, as well as the jump operator $[\dots]$, we may speak of *the jump* $[v](t_0)$ — i.e. the result, when the jump operator is applied to a given function at a given point.

4.2 The Dirac delta distribution and the Heaviside step function

Formally, the Dirac delta is given by

$$\delta(\tau) = \begin{cases} +\infty\,, & \tau = 0\,, \\ 0\,, & \tau \neq 0\,, \end{cases} \tag{54}$$

with the constraint

$$\int_{-\infty}^{+\infty} \delta(\tau)\, \mathrm{d}\tau = 1\,, \tag{55}$$

which defines the singularity to be of "unit mass" (when the delta distribution is interpreted as the density function of an ideal point mass). The Dirac delta distribution satisfies the property

$$\int_{-\infty}^{+\infty} f(\tau)\, \delta(\tau)\, \mathrm{d}\tau = f(0)\,. \tag{56}$$

Importantly, the Dirac delta distribution is not square integrable ($\delta \notin L^2(\mathbb{R})$).

Strictly speaking, (56) is an abuse of notation, because no function (in the classical sense) satisfying (56) exists. What equation (56) actually means, must be defined in some mathematically rigorous fashion. One option is to define δ as a measure. When given a subset A of the real line $\mathbb{R}$, we define $\delta(A) = 1$ if $0 \in A$, and $\delta(A) = 0$ otherwise. Then we take the left-hand side of (56) to mean the integral of f against this measure, $\int_{-\infty}^{+\infty} f(\tau)\delta\{\mathrm{d}\tau\}$. Alternatively, one may use the theory of distributions, where distributions are integral functionals; then "$\delta(\tau)$" does not need to exist as an independent object outside the context of the integral (56).[10]

The Dirac delta can be understood as a limit of a sequence of functions. For example, consider the following sequence of piecewise constant functions (with two jumps):

$$d_n(\tau) := \begin{cases} 0\,, & \tau < -1/2n\,, \\ n\,, & -1/2n < \tau < +1/2n\,, \\ 0\,, & \tau > +1/2n\,, \end{cases} \tag{57}$$

where $n = 1, 2, \ldots$ Left- or right-continuity does not matter for this example, so we have simply left $d_n(\tau)$ undefined at the discontinuities. Geometrically, (57) describes a sequence of rectangles, which become narrower and higher as n increases. The width and height are chosen such that the area remains constant; each $d_n(\tau)$ satisfies the normalization (55). At $n \to \infty$, the limit of the sequence (57) is the Dirac delta.

One may also use[11] a sequence of zero-centered normal distributions

$$\rho_a(\tau) := \frac{1}{a\sqrt{\pi}} \exp(-x^2/a^2)\,,$$

as $a \to 0$.

A close relative of the Dirac delta distribution is the Heaviside step function:

$$H(t) = \begin{cases} 0\,, & t \leq t_0 \text{ (if L.C.)}\,, \quad t < t_0 \text{ (if R.C.)}\,, \\ 1\,, & t > t_0 \text{ (if L.C.)}\,, \quad t \geq t_0 \text{ (if R.C.)}\,, \end{cases} \tag{58}$$

These are related as follows. Observe that if $b < 0$, then

$$\int_{-\infty}^{b} \delta(\tau)\, \mathrm{d}\tau = 0\,, \quad b < 0\,, \tag{59}$$

because $\delta(\tau) = 0$ for all $\tau < 0$. On the other hand, for any $b \neq 0$, by (56) we have

$$1 = \int_{-\infty}^{\infty} \delta(\tau)\, \mathrm{d}\tau = \int_{-\infty}^{b} \delta(\tau)\, \mathrm{d}\tau + \int_{b}^{+\infty} \delta(\tau)\, \mathrm{d}\tau\,. \tag{60}$$

Consider the case $b > 0$. The second term vanishes (because $\delta(\tau) = 0$ for all $\tau > 0$). Only the first term remains, with the result

$$\int_{-\infty}^{b} \delta(\tau)\, \mathrm{d}\tau = 1\,, \quad b > 0\,. \tag{61}$$

[10] https://en.wikipedia.org/wiki/Dirac_delta_function#Definitions
[11] https://en.wikipedia.org/wiki/Dirac_delta_function

Comparing (59), (61) and (58), we see that at least for any $b \neq 0$, the Heaviside step function is the cumulative distribution function of the Dirac delta distribution:

$$\int_{-\infty}^{b} \delta(\tau)\,\mathrm{d}\tau = H(b)\ , \quad b \neq 0\ . \tag{62}$$

To cover the final case $b = 0$, as always when dealing with infinities, one must be careful. The singularity must be counted only once. One way to do this is to modify (60) to read

$$1 = \int_{-\infty}^{\infty} \delta(\tau)\,\mathrm{d}\tau = \lim_{\varepsilon\to 0^+} \left(\int_{-\infty}^{\varepsilon} \delta(\tau)\,\mathrm{d}\tau + \int_{\varepsilon}^{+\infty} \delta(\tau)\,\mathrm{d}\tau \right)\ , \tag{63}$$

which places the split unambiguously on one side of the origin (here the positive side). Then we evaluate the integrals and take the limit, in that order. The first term remains, while the second one vanishes. Considering that the result is similar in form to (62) (but now with $b = 0$), we conclude that $H(0) = 1$. Thus placing the split on the positive side of the origin, as in (63), corresponds to choosing H to be right-continuous.

If, on the other hand, the split is placed on the negative side of the origin,

$$1 = \int_{-\infty}^{\infty} \delta(\tau)\,\mathrm{d}\tau = \lim_{\varepsilon\to 0^+} \left(\int_{-\infty}^{-\varepsilon} \delta(\tau)\,\mathrm{d}\tau + \int_{-\varepsilon}^{+\infty} \delta(\tau)\,\mathrm{d}\tau \right)\ , \tag{64}$$

then the first term, which (as above) tentatively defines $H(0)$, is zero. Hence this choice corresponds to choosing H to be left-continuous.

We conclude that

$$\int_{-\infty}^{b} \delta(\tau)\,\mathrm{d}\tau = H(b) \tag{65}$$

for any b, with the case $b = 0$ requiring special interpretation as explained above. (If H is L.C., then also the LHS is zero; if H is R.C., then the LHS is 1.)

Finally, considering b as a variable and formally differentiating (65) with respect to b, it is seen (only formally!) that

$$\delta(b) = \frac{\partial H}{\partial b}(b)\ . \tag{66}$$

Rigorously showing (66) requires further development beyond the scope of this short review.[12]

4.3 The continuous part

It is useful to define the *continuous part* of $v(t)$, denoted here $\widehat{v}(t)$:

$$\widehat{v}(t) := v(t) - [v](t_0)\,H(t - t_0)\ , \tag{67}$$

where $H(\dots)$ is the Heaviside step function. The jump $[v](t_0)$ is simply a number obtainable from $v(t)$.

To justify the name, it must be shown that (67) is indeed continuous. The continuity follows from (50). Obviously, at any fixed $t \neq t_0$, (67) is continuous, because v is, and the Heaviside function is then constant in a small neighborhood of t.

Thus we only need to show continuity at the point $t = t_0$. The limit from the left is

$$\begin{aligned} \lim_{\varepsilon\to 0^+} \widehat{v}(t_0 - \varepsilon) &= v_-(t_0) - [v](t_0) \lim_{\varepsilon\to 0^+} H(-\varepsilon) \\ &= v_-(t_0)\ , \end{aligned} \tag{68}$$

where we have used the fact that $H(-t)$ is zero for all $t > 0$. In order to have continuity, (68) must match the limit from the right,

$$\begin{aligned} \lim_{\varepsilon\to 0^+} \widehat{v}(t_0 + \varepsilon) &= v_+(t_0) - [v](t_0) \lim_{\varepsilon\to 0^+} H(+\varepsilon) \\ &= v_+(t_0) - [v](t_0) \\ &= v_+(t_0) - (v_+(t_0) - v_-(t_0)) \\ &= v_-(t_0)\ , \end{aligned} \tag{69}$$

[12] Equation (66) holds at least in the distributional sense.
https://en.wikipedia.org/wiki/Heaviside_step_function
https://en.wikipedia.org/wiki/Distribution_%28mathematics%29#Differentiation

where we have used $H(t) = 1$ for all $t > 0$. Thus continuity holds also at t_0. Here it does not matter whether H is taken to be left- or right-continuous, because it is never evaluated at $\varepsilon = 0$; only the one-sided limits are needed.

Now we may reinterpret the definition of $\widehat{v}(t)$, equation (67), as decomposing $v(t)$ into a continuous function and a Heaviside step function times the jump:

$$v(t) = \widehat{v}(t) + [v](t_0)\, H(t - t_0) \,. \tag{70}$$

For this interpretation to make sense at $t = t_0$, the left- or right-continuity of H must be chosen to match that of v.

4.4 Differentiation and definite integration of piecewise continuous functions

The decomposition (70) allows us to rewrite (53) as

$$\frac{\partial v}{\partial t}(t) = \frac{\partial \widehat{v}}{\partial t}(t) + [v](t_0)\, \delta(t - t_0) \,, \tag{71}$$

using the relation between the Heaviside step function and the Dirac delta distribution. Equation (71) gives us the necessary tool to formally treat the integration of $v'(t)$ over an interval (a, b) containing the singularity at t_0 (i.e. $a < t_0 < b$). We have

$$\begin{aligned}
\int_a^b v'(t)\, \mathrm{d}t &= \int_a^b \widehat{v}'(t)\, \mathrm{d}t + \int_a^b [v](t_0)\, \delta(t - t_0)\, \mathrm{d}t \\
&= \int_a^b \widehat{v}'(t)\, \mathrm{d}t + [v](t_0) \int_a^b \delta(t - t_0)\, \mathrm{d}t \\
&= \int_a^b \widehat{v}'(t)\, \mathrm{d}t + [v](t_0) \\
&= \widehat{v}(b) - \widehat{v}(a) + [v](t_0) \,.
\end{aligned}$$

The last integral on the second line evaluates to 1, because we are considering the case where $t_0 \in (a, b)$.

Finally, consider terms of the same form as the first term in the weak form, equation (49), and let both u and w have a jump at $t_0 \in (a, b)$. (This choice is motivated by the fact that we aim to construct a classical Galerkin method, where the basis and test functions are taken to be the same.) By (70) and (71), we have

$$\begin{aligned}
\int_a^b u'(t) w(t)\, \mathrm{d}t &= \int_a^b \Big(\widehat{u}'(t) + [u](t_0)\, \delta(t - t_0)\Big)\Big(\widehat{w}(t) + [w](t_0)\, H(t - t_0)\Big)\, \mathrm{d}t \\
&= \int_a^b \widehat{u}'(t)\widehat{w}(t)\, \mathrm{d}t \\
&+ [w](t_0) \int_a^b \widehat{u}'(t) H(t - t_0)\, \mathrm{d}t \\
&+ [u](t_0) \int_a^b \widehat{w}(t)\, \delta(t - t_0)\, \mathrm{d}t \\
&+ [u](t_0)[w](t_0) \int_a^b H(t - t_0)\delta(t - t_0)\, \mathrm{d}t \\
&= \int_a^b \widehat{u}'(t)\widehat{w}(t)\, \mathrm{d}t + [w](t_0) \int_{t_0}^b \widehat{u}'(t)\, \mathrm{d}t + [u](t_0)\widehat{w}(t_0) + [u](t_0)[w](t_0) H(0) \\
&= \int_a^b \widehat{u}'(t)\widehat{w}(t)\, \mathrm{d}t + [w](t_0)(\, \widehat{u}(b) - \widehat{u}(t_0)\,) + [u](t_0)(\, \widehat{w}(t_0) + [w](t_0) H(0)\,) \\
&= \int_a^b \widehat{u}'(t)\widehat{w}(t)\, \mathrm{d}t + [w](t_0)(\, \widehat{u}(b) - \widehat{u}(t_0)\,) + [u](t_0)\, w(t_0) \,,
\end{aligned} \tag{72}$$

where in the last step we have used (70). Here it does not matter whether u and w are left- or right-continuous.

4.5 Functions with several jumps

Finally, let us consider how to treat functions with several discontinuities. Let $u(t)$ be a left-continuous function with $N \geq 2$ jumps, located at points t_k, $k = 1, 2, \ldots, N$:

$$u(t) = \widehat{u}(t) + \sum_{k=1}^{N} [u](t_k)\, H(t - t_k) \,. \tag{73}$$

Because $H(\tau) = 1$ for all $\tau > 0$, we may write for given κ and $t > t_\kappa$ that

$$u_\kappa(t) = \widehat{u}(t) + c_\kappa + \sum_{k=\kappa+1}^{N} [u](t_k)\, H(t - t_k)\ , \quad 1 \le \kappa \le N,\ t > t_\kappa\ . \tag{74}$$

where

$$c_\kappa := \sum_{k=1}^{\kappa} [u](t_k)\ . \tag{75}$$

Following the convention from many programming languages, if $\kappa = N$, we ignore the sum in (74) because then its start index is greater than its end index, and similarly for the case $\kappa = 0$ in (75).

If we further restrict our consideration to the interval $t \in (t_\kappa, t_{\kappa+2})$, then

$$u_\kappa(t) = \widehat{u}(t) + c_\kappa + [u](t_{\kappa+1})\, H(t - t_{\kappa+1})\ , \quad 1 \le \kappa \le N,\ t \in (t_\kappa, t_{\kappa+2})\ , \tag{76}$$

locally returning to a form with only one discontinuity, allowing us to use the equations already developed. (For the purposes of (76), we use the convention that for any $k > N$, $t_k = +\infty$.)

To obtain a version of (72) for the present case, it is convenient to split the interval of integration so that each subinterval contains up to one discontinuity, allowing us to apply (72) separately to each, and for this, use the form (76) to represent u (and w).

Considering that the aim is to develop a timestepping method on an interval $t \in [a, b]$, we would like to choose the limits of integration as $I_k = (t_k, t_{k+1})$ for $k = 0, 1, \ldots, N$, setting $t_0 = a$ and $t_{N+1} = b$. However, to avoid ambiguities in handling the delta distribution, we must actually use $I_k \equiv \lim_{\varepsilon \to 0^+} I_k^\varepsilon$, where $I_k^\varepsilon = (t_k - \varepsilon, t_{k+1} - \varepsilon)$.

Keep in mind that we plan to treat our solution as left-continuous. To keep things in line with the intuitive picture, we require $\varepsilon < \frac{1}{2} \min\{t_{k+1} - t_k \ :\ k = 0, 1, \ldots, N\}$, guaranteeing that each interval covers only one discontinuity. The epsilon in the lower limit makes the local discontinuity fall strictly inside each I_k^ε, and the one in the upper limit retains the lengths and the non-overlapping property of the intervals.

Thus, we have

$$\int_a^b u'(t) w(t)\ \mathrm{d}t = \lim_{\varepsilon \to 0^+} \sum_{k=0}^{N} \int_{t_k - \varepsilon}^{t_{k+1} - \varepsilon} u'(t) w(t)\ \mathrm{d}t\ . \tag{77}$$

It will however be simpler in practice to consider the case with just one timestep, sidestepping the need for (77), and also avoiding the need to keep track of the constants c_κ for the representation (76). This is possible due to two reasons.

First, we observe that each jump connects only adjacent intervals. Due to the form of our problem (1)–(2), we may split it into a sequence of subproblems on I_k, with the final value $u(t_{k+1})$ from the current timestep fed into the next one as the initial condition u_0.

Secondly, without loss of generality, in the Galerkin setup to follow later, we may choose the global basis functions (and in a pure Galerkin method, hence also the global test functions) to have support only on one element each.[13]

If we have a function, continuous across element boundaries, having support on several elements, that we would like to use in the basis, then, because this is a discontinuous method, we may always split it into parts restricted to individual elements (by multiplying with a suitable indicator function), and use those parts instead of using the original function. (If the parts are used separately, this allows more freedom than the original continuous function, since the pieces may then have different values for their Galerkin coefficients. If used as a sum (forcing the same coefficient for each part), this reduces to using the original continuous function.)

From these two properties we conclude that the method can be designed to work locally, and hence it is sufficient to develop an algorithm to treat a single interval (t_k, t_{k+1}) at a time.

This completes the background.

[13] Note that this differs from the finite element method, where (considering classical conforming variants of the method, with classical basis functions such as the Lagrange interpolation polynomials) the global basis functions are typically attached to nodes, and have support on the patch of elements surrounding the node. There the global basis functions are pieced together in a matching way from local basis functions on each element belonging to the patch. Here, because the method is constructed to handle discontinuities of the basis across element boundaries, there is no need to perform this patching.

4.6 Developing the weak form

Now we have the tools required for the problem at hand. Let us split the domain $[0, t_\mathrm{f}]$ into N intervals $I_k = [t_k, t_{k+1}]$, with $t_1 = 0$ and $t_{N+1} = t_\mathrm{f}$. Let $u(t) = u_0$ for all $t < 0$ (to make the expression $u(-\varepsilon)$ valid).

Let u be left-continuous and w right-continuous. This "complementarity" of the continuity types is needed so that the test at I_k will "see" the final value of u on I_{k-1}. At the start of the kth timestep, $u(t_k)$ is the final value from the previous timestep, while $w(t_k)$ is the starting value of the test for the current timestep.

Keep in mind that the weak form (49) must hold separately for each choice of w. Thus, each of the test functions must be defined on the whole interval $[0, t_\mathrm{f}]$. In practice, we will choose them to be zero in most of the domain, nonzero on one element, and having up to two jumps (one at each endpoint of its support).

Denote $u(t_k) = u_0$, the initial condition for this timestep. Let ε be arbitrary, with $0 < \varepsilon < \frac{1}{2}\min\{t_{k+1} - t_k : k = 0, 1, \ldots, N\}$. Consider a right-continuous test function w having support only on $t \in [t_k - \varepsilon, t_{k+1} - \varepsilon)$, and having a jump at t_k. Equation (49), i.e.,

$$\int_0^{t_\mathrm{f}} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \int_0^{t_\mathrm{f}} f(\,u(t),\,t\,)\, w \,\mathrm{d}t\,, \quad \forall w\,,$$

for such w becomes

$$\lim_{\varepsilon\to 0^+} \int_{t_k-\varepsilon}^{t_{k+1}-\varepsilon} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \lim_{\varepsilon\to 0^+} \int_{t_k-\varepsilon}^{t_{k+1}-\varepsilon} f(\,u(t),\,t\,)\, w \,\mathrm{d}t\,, \tag{78}$$

because elsewhere $w = 0$. As was explained above, we must use limits in order to place the singularity in $u'(t)$ strictly inside the domain of integration. This is also needed to count only the singularity at t_k. Because u is L.C. and we approach t_{k+1} from the left, *within this timestep* $u'(t)$ has no singularity at t_{k+1} (in the sense of the limit in (78)).

On the right-hand side of (78), by the problem definition (1)–(2), the load function f involves no time differentiations of u or w. These functions are C^{-1} continuous. Thus, if f itself is at least C^{-1} continuous in u and t, then the integrand will not be worse than C^{-1}, and hence does not need any special handling. This allows us to immediately drop the limit from the right-hand side:

$$\lim_{\varepsilon\to 0^+} \int_{t_k-\varepsilon}^{t_{k+1}-\varepsilon} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \int_{t_k}^{t_{k+1}} f(\,u(t),\,t\,)\, w \,\mathrm{d}t\,.$$

The left-hand side now has exactly one singularity, which is strictly inside the domain of integration, so we may apply (72), obtaining

$$\lim_{\varepsilon\to 0^+} \left\{ \int_{t_k-\varepsilon}^{t_{k+1}-\varepsilon} \frac{\partial \widehat{u}}{\partial t} \widehat{w} \,\mathrm{d}t + [w](t_k)(\,\widehat{u}(t_{k+1}-\varepsilon) - \widehat{u}(t_k)\,) + [u](t_k)\, w(t_k) \right\} = \int_{t_k}^{t_{k+1}} f(\,u(t),\,t\,)\, w \,\mathrm{d}t\,.$$

Evaluating the limit on the left-hand side yields, obviously,

$$\int_{t_k}^{t_{k+1}} \frac{\partial \widehat{u}}{\partial t} \widehat{w} \,\mathrm{d}t + [w](t_k)(\,\widehat{u}(t_{k+1}) - \widehat{u}(t_k)\,) + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\,u(t),\,t\,)\, w \,\mathrm{d}t\,. \tag{79}$$

Note that w, which was defined to have support on $t \in [t_k - \varepsilon, t_{k+1} - \varepsilon)$, because $\varepsilon \to 0$, has now converged into having support on $t \in [t_k, t_{k+1})$. (We could have denoted the original by w_ε and its limit by w, but the presentation may look clearer without the extra subscript.)

Equation (79) still contains a mix of u, $\widehat{u}$, w and $\widehat{w}$. We make the following observations:

$$u(t_k) = u_0 \qquad \text{(initial condition for this timestep)}\,, \tag{80}$$

$$\widehat{u}(t_k) = u(t_k) \qquad (u \text{ L.C.})\,, \tag{81}$$

$$\widehat{u}(t) = u(t) - [u](t_k)\,, \quad t \in (t_k, t_{k+1}] \qquad \text{(eq. (67), } u \text{ L.C.)}\,, \tag{82}$$

$$\widehat{w}(t) = w(t) - [w](t_k)\,, \quad t \in [t_k, t_{k+1}) \qquad \text{(eq. (67), } w \text{ R.C.)}\,. \tag{83}$$

It also holds that

$$[w](t_k) = w(t_k) \qquad \text{(because } \lim_{\epsilon\to 0^+} w(t_k - \epsilon) = 0)\,, \tag{84}$$

which may be useful in implementing numerics (since it allows us to process each element locally).

The expression $\widehat{u}(t_{k+1}) - \widehat{u}(t_k)$ may be treated using one-sided limits of u from inside the timestep:

$$\begin{aligned}\widehat{u}(t_{k+1}) - \widehat{u}(t_k) &= (u(t_{k+1}) - [u](t_k)) - u(t_k) \\ &= u(t_{k+1}) - (u(t_k) + [u](t_k)) \\ &= \lim_{\varepsilon\to 0^+} u(t_{k+1} - \varepsilon) - \lim_{\varepsilon\to 0^+} u(t_k + \varepsilon)\ , \end{aligned} \tag{85}$$

where in the last form we have used the left-continuity of u (hence $u(t_{k+1})$ agrees with the limit from the left). This holds very generally; in the context of multiple timesteps, if there is any offset between u and $\widehat{u}$ due to the history of jumps at earlier timesteps, in (85) this offset is cancelled by the subtraction; what matters is only how much the continuous part $\widehat{u}$ — or indeed u itself — changes across the timestep. (This can be made rigorous using (73).)

Rewriting (79), i.e.

$$\int_{t_k}^{t_{k+1}} \frac{\partial \widehat{u}}{\partial t}\widehat{w}\ \mathrm{d}t + [w](t_k)(\ \widehat{u}(t_{k+1}) - \widehat{u}(t_k)\ ) + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\ u(t),\ t\ )\, w\ \mathrm{d}t\ ,$$

with the help of (82), (83) and (85) to eliminate $\widehat{u}$ and $\widehat{w}$, we have

$$\int_{t_k}^{t_{k+1}} \frac{\partial u(t)}{\partial t}\big(w(t) - [w](t_k)\big)\ \mathrm{d}t + [w](t_k)\big(\lim_{\varepsilon\to 0^+} u(t_{k+1} - \varepsilon) - \lim_{\varepsilon\to 0^+} u(t_k + \varepsilon)\big) + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\ u(t),\ t\ )\, w\ \mathrm{d}t\ . \tag{86}$$

When manipulating the equations, it is important to keep in mind that u is L.C., whereas w is R.C. Here we have used the fact that $\partial\widehat{u}/\partial t = \partial u/\partial t$ on $t \in (t_k, t_{k+1}]$, because the jump is a constant. (Hence, when doing numerics, we must be careful in the first term not to evaluate $\partial u/\partial t$ at t_k when it is written in this way!)

Noting further that $[w](t_k)$ is a constant, and u is continuous on $t \in (t_k, t_{k+1}]$, we may evaluate the second term in the first integral, obtaining

$$\begin{aligned}\int_{t_k}^{t_{k+1}} \frac{\partial u(t)}{\partial t} w(t)\ \mathrm{d}t - [w](t_k)\big(\ u(t_{k+1}) - \lim_{\varepsilon\to 0^+} u(t_k + \varepsilon)\ \big) \\ + [w](t_k)\big(\ u(t_{k+1}) - \lim_{\varepsilon\to 0^+} u(t_k + \varepsilon)\ \big) + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\ u(t),\ t\ )\, w\ \mathrm{d}t\ .\end{aligned}$$

Now the terms involving $[w](t_k)$ cancel, yielding the perhaps familiar-looking weak form for piecewise continuous u and w:

$$\int_{t_k}^{t_{k+1}} \frac{\partial u(t)}{\partial t} w(t)\ \mathrm{d}t + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\ u(t),\ t\ )\, w\ \mathrm{d}t\ . \tag{87}$$

To recap: when deriving (87), we have taken the solution u as left-continuous and the test function w as right-continuous, with jumps occurring at timestep boundaries.

So far we have treated the weak initial value problem, first in the context of general piecewise continuous functions having finite discontinuities at timestep boundaries, and then with test functions having their support restricted to $[t_k, t_{k+1})$. Now we are in a position to introduce the Galerkin component of the method.

First, let $\phi_n(t)$, $n = 1, 2, \ldots$ be a set of (at least C^0) continuous functions defined on the closed interval $[t_k, t_{k+1}]$. Let us define the half-open intervals $L_k = (t_k, t_{k+1}]$ and $R_k = [t_k, t_{k+1})$ (mnemonic: the name denotes the open side). Let $\chi_A(t)$ denote the indicator function of a set A:

$$\chi_A(t) = \begin{cases} 1\ , & t \in A\ , \\ 0\ , & t \notin A\ . \end{cases} \tag{88}$$

We define

$$\varphi_n(t) := \chi_{L_k}(t)\phi_n(t)\ , \tag{89}$$

$$\psi_n(t) := \chi_{R_k}(t)\phi_n(t)\ , \tag{90}$$

making φ_n (respectively ψ_n) suitable as a basis for left-continuous (resp. right-continuous) functions. (The only thing the multiplication by χ does is that it removes the inappropriate endpoint from the support.) In the Galerkin spirit, in all other respects the bases are the same.

We now represent u as a Galerkin series,

$$u(t) := \sum_{m=1}^{\infty} u_m \varphi_m(t) \ , \tag{91}$$

where u_m are the Galerkin coefficients, and $\varphi_m(t)$ are the global basis functions. We choose the test functions w_i as

$$w_i(t) := \psi_i(t) \ , \quad i = 1, 2, \ldots \tag{92}$$

i.e. otherwise the same as the basis for u, but with right-continuity.

Differing from continuous Galerkin methods (such as the finite element method), because now there is no requirement of continuity across the timestep boundaries, we may actually take the local basis functions — of course appropriately positioned and scaled on the t axis separately for each element, as the time integration proceeds — as the global basis functions. (In practice, we will of course scale each timestep onto a reference element on $[0, 1]$.)

Inserting (91)–(92) into (87) obtains

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \frac{\partial}{\partial t} \left(\sum_{m=1}^{\infty} u_m \varphi_m(t) \right) \psi_i(t) \, \mathrm{d}t + \left(\left(\sum_{m=1}^{\infty} u_m \lim_{\varepsilon \to 0^+} \varphi_m(t_k + \varepsilon) \right) - u_0 \right) \psi_i(t_k) \\ = \int_{t_k}^{t_{k+1}} f\big(\sum_{n=1}^{\infty} u_m \varphi_m(t), \, t \, \big) \, \psi_i(t) \, \mathrm{d}t \ , \quad \forall i = 1, 2, \ldots \ , \end{aligned} \tag{93}$$

where u_0 is the initial condition for this timestep.

Because there are no discontinuities remaining in the first term, we may rearrange it as

$$\begin{aligned} \int_{t_k}^{t_{k+1}} \frac{\partial}{\partial t} \left(\sum_{m=1}^{\infty} u_m \varphi_m(t) \right) \psi_i \, \mathrm{d}t &= \int_{t_k}^{t_{k+1}} \psi_i \sum_{m=1}^{\infty} u_m \frac{\partial \varphi_m}{\partial t}(t) \, \mathrm{d}t \\ &= \sum_{m=1}^{\infty} u_m \int_{t_k}^{t_{k+1}} \psi_i \, \frac{\partial \varphi_m}{\partial t}(t) \, \mathrm{d}t \ , \end{aligned} \tag{94}$$

first using the fact that every continuous function on $[0, 1]$ (which is our reference element) is a uniform limit of polynomials, so that we may exchange differentiation and infinite summation[14], and then using Tonelli's and Fubini's theorems to justify the exchange of integration and infinite summation[15].

The final step of the theoretical derivation of the dG procedure is to apply (94) to (93). In practical numerics, we then formally truncate the Galerkin series (choosing a discrete subspace of the function space), obtaining

$$\begin{aligned} \sum_{m=1}^{M} u_m \int_{t_k}^{t_{k+1}} \psi_i \, \frac{\partial \varphi_m}{\partial t}(t) \, \mathrm{d}t + \left(\left(\sum_{m=1}^{M} u_m \lim_{\varepsilon \to 0^+} \varphi_m(t_k + \varepsilon) \right) - u_0 \right) \psi_i(t_k) \\ = \int_{t_k}^{t_{k+1}} f\big(\sum_{m=1}^{M} u_m \varphi_m(t), \, t \, \big) \, \psi_i(t) \, \mathrm{d}t \ , \quad \forall i = 1, 2, \ldots, M \ , \end{aligned} \tag{95}$$

where M is the total number of basis functions ϕ_n in the discrete basis.

Transferring terms and recognizing that this can be formally written as a linear equation system, we have

$$\mathbf{A}\mathbf{u} = \mathbf{b}(\mathbf{u}) \ , \tag{96}$$

where $\mathbf{u}$ is an M-element vector consisting of the coefficients u_m, and

$$A_{im} = \int_{t_k}^{t_{k+1}} \psi_i(t) \, \frac{\partial \varphi_m}{\partial t}(t) \, \mathrm{d}t + \psi_i(t_k) \lim_{\varepsilon \to 0^+} \varphi_m(t_k + \varepsilon) \ , \tag{97}$$

$$b_i(\mathbf{u}) = \int_{t_k}^{t_{k+1}} f\big(\sum_{m=1}^{M} u_m \varphi_m(t), \, t \, \big) \, \psi_i(t) \, \mathrm{d}t + \psi_i(t_k) u_0 \ . \tag{98}$$

[14] http://math.stackexchange.com/questions/147869/interchanging-the-order-of-differentiation-and-summation

[15] http://math.stackexchange.com/questions/83721/when-can-a-sum-and-integral-be-interchanged

Treating the one degree of freedom model problem (1)–(2) has lead to an equation system with M equations, because there are M basis functions modeling the behavior of this degree of freedom across a single timestep.

Equations (96)–(98), in principle, allow us to solve for the Galerkin coefficients u_m for this timestep. Together with relation (91) (similarly truncated), we obtain the behavior of $u(t)$ for all $t \in (t_k, t_{k+1}]$. Especially, evaluating $u(t_{k+1})$ from (91) then gives us the initial condition for the next timestep.

The limit expression in (97) is easier to evaluate than it looks; it is simply $\phi_m(t_k)$.

Note that in general, for nonlinear problems, $\mathbf{b}$ depends on $\mathbf{u}$; we have thus presented the method in this form. Fixed point iteration (for the Galerkin coefficients $\mathbf{u}$) is thus needed to evaluate the load vector $\mathbf{b}$. If f is linear, its linear dependence on $\mathbf{u}$ would of course be extracted and transferred into $\mathbf{A}$ (but here we concentrate on the general nonlinear case).

The final step in practical numerics (for this single degree of freedom problem) is to rewrite the integrals in (97)–(98) using quadratures on the reference element $[0, 1]$. The standard tools from finite elements methods can be used for this.

4.7 Extension to ODE systems

The general treatment from section 1.1 applies also here. In this section, we work out the details specifically for dG.

The weak form of the problem, equation (49), does not require many changes. Understanding the testing of a vector-valued quantity by a vector-valued test function as projection into the direction of the test function, we write

$$\int_0^{t_\mathrm{f}} \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{w} \, \mathrm{d}t = \int_0^{t_\mathrm{f}} \big(\, \mathbf{M}^{-1} \mathbf{f}(\, \mathbf{u}(t), \, t \,) \, \big) \cdot \mathbf{w} \, \mathrm{d}t \, , \quad \forall \mathbf{w} \, , \tag{99}$$

where $\mathbf{w}$ is a vector-valued test function, and the dot denotes the usual Euclidean inner product.

Again, by defining the effective load

$$\mathbf{g}(\, \mathbf{u}(t), \, t \,) = \mathbf{M}^{-1} \mathbf{f}(\, \mathbf{u}(t), \, t \,) \, , \tag{100}$$

we have

$$\int_0^{t_\mathrm{f}} \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{w} \, \mathrm{d}t = \int_0^{t_\mathrm{f}} \big(\, \mathbf{g}(\, \mathbf{u}(t), \, t \,) \, \big) \cdot \mathbf{w} \, \mathrm{d}t \, , \quad \forall \mathbf{w} \, . \tag{101}$$

Let us look at the contribution of a single component of the test, w_j:

$$\int_0^{t_\mathrm{f}} \frac{\partial u_j}{\partial t} w_j \, \mathrm{d}t = \int_0^{t_\mathrm{f}} \big(\, g_j(\, u_1(t), u_2(t), \ldots, \, t \,) \, \big) w_j \, \mathrm{d}t \, , \quad \forall w_j \, . \tag{102}$$

Obviously, the previous machinery applies directly to each of the equations (102), producing equations of the form (87). (No assumptions were made about f beyond C^{-1} continuity.)

After summing the results over j, and finally using (100) to replace $\mathbf{g}$ with its definition, we have the result

$$\int_{t_k}^{t_{k+1}} \frac{\partial \mathbf{u}(t)}{\partial t} \cdot \mathbf{w}(t) \, \mathrm{d}t + [\mathbf{u}](t_k) \cdot \mathbf{w}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{M}^{-1} \mathbf{f}(\, \mathbf{u}(t), \, t \,) \cdot \mathbf{w} \, \mathrm{d}t \, , \quad \forall \mathbf{w}. \tag{103}$$

As before, from here we may proceed by treating $\mathbf{M}^{-1}\mathbf{f}$ as an effective load, and solving the linear equation system $\mathbf{M}\,\mathbf{g} = \mathbf{f}$ for $\mathbf{g}$ whenever its numerical value is needed.

Because the degrees of freedom have been decoupled on the left-hand side, it is convenient to test them individually, taking the test functions $\mathbf{w}$ as w_{ji}, $j = 1, 2, \ldots, N$ (indexing the degrees of freedom), $i = 1, 2, \ldots, M$ (indexing the set of test functions in a timestep): $w_{1i}(t) = (\, w_i(t), \, 0, \, 0, \ldots, \, 0 \,)$, $w_{2i}(t) = (\, 0, \, w_i(t), \, 0, \, 0, \ldots, \, 0 \,)$, $\ldots$, $w_{Ni}(t) = (\, 0, \, 0, \, 0, \ldots, \, 0, \, w_i(t) \,)$, where $w_i(t)$, $i = 1, 2, \ldots, M$, are the test functions $\psi_i(t)$ for a scalar quantity on the timestep. This produces NM equations in total (as indeed expected, there being N space degrees of freedom, and M equations per space degree of freedom for one timestep).

Note that each evaluation of $\mathbf{g}$ — i.e. each quadrature point for the integral on the right-hand side — will require the solution of a (possibly large) linear equation system. Furthermore, because $\mathbf{u}(t)$ is unknown at any point $t > t_k$, for nonlinear problems this expensive solution of multiple linear equation systems (for $\mathbf{g}$) must occur for each iteration inside the fixed point iteration loop. Thus, for a general semilinear system of ODEs, dG (at least when implemented the way described here) can be expensive.

For linear problems, of course, it is possible to avoid the need for fixed point iteration in implicit integration methods, making dG much cheaper.

Finally, if the space discretization is such as to make $\mathbf{M}$ the unit matrix, then $\mathbf{g} = \mathbf{f}$ and (103) becomes somewhat cheaper to solve. One particularly simple implementation for nonlinear problems of this form is to just use (96)–(98) (in principle) independently for each space degree of freedom, again with fixed point iteration to make it possible to explicitly evaluate the right-hand side. The only required modification is that the last available iterate from all space degrees of freedom is needed to evaluate f (each space degree of freedom $u_j(t)$ has its own Galerkin coefficients in time, $(u_j)_m$, replacing the original f with $f(\, u_1(t),\, u_2(t),\, \ldots,\, t\,) = f(\, \sum_{m=1}^{M} (u_1)_m \varphi_m(t),\, \sum_{m=1}^{M} (u_2)_m \varphi_m(t), \ldots,\, t\,)$.

4.8 Choosing the basis

As the basis functions $\phi_n(t)$, polymomials are a typical choice. One often speaks of the dG(q) method, where q is the polynomial order of the basis.

4.8.1 dG(0)

(dG(0) is not supported by the solver; this section is only included for information.)

The basis of zeroth-order discontinuous Galerkin consists of just the constant 1 on each element. Looking at equation (87), i.e.

$$\int_{t_k}^{t_{k+1}} \frac{\partial u(t)}{\partial t} w(t)\, \mathrm{d}t + [u](t_k)\, w(t_k) = \int_{t_k}^{t_{k+1}} f(\, u(t),\, t\,)\, w\, \mathrm{d}t\ ,$$

we see that if $w \equiv 1$, we have

$$\int_{t_k}^{t_{k+1}} \frac{\partial u(t)}{\partial t}\, \mathrm{d}t + [u](t_k) = \int_{t_k}^{t_{k+1}} f(\, u(t),\, t\,)\, \mathrm{d}t\ .$$

Furthermore, if the basis of u is piecewise constant, then inside the timestep $\partial u/\partial t \to 0$ (recall that the first term concerns only values inside the current timestep), and we are left with

$$[u](t_k) = \int_{t_k}^{t_{k+1}} f(\, u(t),\, t\,)\, \mathrm{d}t\ ,$$

or in other words,

$$u_{k+1} - u_k = \int_{t_k}^{t_{k+1}} f(\, u(t),\, t\,)\, \mathrm{d}t\ , \tag{104}$$

where we have denoted $u(t_{k+1}) = u_{k+1}$ and $u(t_k) = u_k$, and used the fact that u is piecewise constant and left-continuous. (Hence $u(t_k)$ is the "old" value. The value immediately after the jump is $\lim_{\varepsilon \to 0^+} u(t_k + \varepsilon) = u(t_{k+1})$, which also justifies the label u_{k+1} for the "new" value at timestep k.) Equation (104) coincides with the integral formula (35), which is simply the integral version of the original problem statement (1).

Strictly speaking, in the dG(0) basis, $u(t)$ is constant on $(t_k, t_{k+1}]$, so actually the match with classical methods is not perfect. It is possible to rigorously obtain backward Euler from (104) by sampling the integrand only at t_{k+1}, but e.g. IMR and forward Euler model $u(t)$ (inside the RHS integral) differently.

4.8.2 dG(1)

The next simplest choice of basis are the linear polynomials on $[0, 1]$,

$$p_1(x) = 1 - x\ ,$$
$$p_2(x) = x\ ,$$

representing linear interpolation on the element.

4.8.3 dG(q)

One convenient choice for arbitrary q, considered the modern approach to polynomial-based Galerkin methods, are the *hierarchical basis functions* (a.k.a. *Lobatto basis functions*), derived from the *Legendre polynomials*.[16] The discontinous Galerkin implementation in the solver uses this form of dG(q).

The Legendre polynomials themselves are polynomials on $[-1,1]$ (note the interval!) that are orthogonal in the L^2 sense:

$$\int_{-1}^{1} P_m(x)P_n(x)\,\mathrm{d}x = \frac{2}{2n+1}\delta_{mn}\ , \tag{105}$$

where δ_{mn} is the Kronecker delta.

The Legendre polynomials are given by the recurrence relation

$$P_0(x) = 1\ , \tag{106}$$
$$P_1(x) = x\ , \tag{107}$$
$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)\ , \tag{108}$$

where (108) is known as *Bonnet's recursion formula*. Among possible direct representations are

$$\begin{aligned} P_n(x) &= \frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}^2(x-1)^{n-k}(x+1)^k \\ &= \sum_{k=0}^{n}\binom{n}{k}\binom{-n-1}{k}\left(\frac{1-x}{2}\right)^k \\ &= 2^n\sum_{k=0}^{n}x^k\binom{n}{k}\binom{\frac{1}{2}(n+k-1)}{n}\ . \end{aligned} \tag{109}$$

The polynomials have been normalized such that $P_n(1) = 1$ (this is the usual normalization).

A shift onto $[0,1]$ is possible by defining $\widetilde{P}_n(x) = P_n(2x-1)$. For the shifted version,

$$\int_{-1}^{1} \widetilde{P}_m(x)\widetilde{P}_n(x)\,\mathrm{d}x = \frac{1}{2n+1}\delta_{mn}\ . \tag{110}$$

4.8.4 The hierarchical basis

We will use zero-based indexing to match the notation used in the solver code. The first basis function is N_0, and matrix element indexing starts at A_{00} (for any matrix $\mathbf{A}$).

On the reference element $[-1,1]$, the hierarchical basis is defined as

$$N_0(x) = \frac{1}{2}(1-x)\ , \tag{111}$$
$$N_1(x) = \frac{1}{2}(1+x)\ , \tag{112}$$
$$N_j(x) = \psi_j(x)\ , \quad j = 2,3,\ldots \tag{113}$$

where $x \in [-1,1]$, the bubble functions are defined as

$$\psi_j(x) := \sqrt{\frac{1}{2}(2j-1)}\int_{-1}^{x} P_{j-1}(t)\,\mathrm{d}t\ , \tag{114}$$

and $P_n(t)$ are the Legendre polynomials using the standard normalization (i.e. $P_n(1) = 1$ for all n). (Note that although $j \geq 2$ in (113), formula (114) requires only $j \geq 1$ to be valid. We will use this later.)

The integral in (114) can be evaluated analytically. For any $n \geq 1$, the following recursion formula holds:

$$(2n+1)P_n(x) = \frac{\partial}{\partial x}\left(\,P_{n+1}(x) - P_{n-1}(x)\,\right)\ . \tag{115}$$

[16] https://en.wikipedia.org/wiki/Legendre_polynomials

By integrating both sides of (115) from -1 to x (and relabeling the dummy variable as t), we obtain

$$(2n+1)\int_{-1}^{x} P_n(t)\,\mathrm{d}t = P_{n+1}(x) - P_{n-1}(x) - \big(\,P_{n+1}(-1) - P_{n-1}(-1)\,\big)\ .$$

It holds that $P_{n+1}(\pm 1) = P_{n-1}(\pm 1)$ for any $n \geq 1$. The parenthetical expression vanishes, and we have

$$(2n+1)\int_{-1}^{x} P_n(t)\,\mathrm{d}t = P_{n+1}(x) - P_{n-1}(x)\ . \tag{116}$$

Choosing $n = j-1$,

$$\big(\,2(j-1)+1\,\big)\int_{-1}^{x} P_{j-1}(t)\,\mathrm{d}t = P_j(x) - P_{j-2}(x)\ ,$$

i.e.

$$(2j-1)\int_{-1}^{x} P_{j-1}(t)\,\mathrm{d}t = P_j(x) - P_{j-2}(x)\ ,$$

and rearranging, we have

$$\int_{-1}^{x} P_{j-1}(t)\,\mathrm{d}t = \frac{1}{(2j-1)}\,\big(\,P_j(x) - P_{j-2}(x)\,\big)\ .$$

The definition (114) thus becomes

$$\psi_j(x) = \sqrt{\frac{1}{2}(2j-1)}\int_{-1}^{x} P_{j-1}(t)\,\mathrm{d}t = \sqrt{\frac{1}{2}(2j-1)}\frac{1}{(2j-1)}\,\big(\,P_j(x) - P_{j-2}(x)\,\big)\ .$$

Simplifying, we have the result

$$\psi_j(x) = \frac{1}{\sqrt{2(2j-1)}}\,\big(\,P_j(x) - P_{j-2}(x)\,\big)\ ,\quad j \geq 2\ . \tag{117}$$

Equation (117) shows that the functions $\psi_j(x)$ for $j \geq 2$ are bubbles; the parenthetical expression vanishes both at -1 and at $+1$ for any $j \geq 2$.

For the special case $j = 1$ (which is not part of the basis), directly from (114) by using $P_0(x) \equiv 1$ we have

$$\psi_1(x) = \frac{1}{\sqrt{2}}(x+1)\ . \tag{118}$$

We will also need the derivative of ψ_j, which we obtain from (114) (by differentiation of a definite integral by its upper limit) as

$$\psi_j'(x) = \left(\sqrt{\frac{1}{2}(2j-1)}\ \right) P_{j-1}(x)\ . \tag{119}$$

The integral of ψ_j is

$$\int_{-1}^{x} \psi_j(t)\,\mathrm{d}t = \frac{1}{\sqrt{2(2j-1)}}\int_{-1}^{x}\big(\,P_j(t) - P_{j-2}(t)\,\big)\,\mathrm{d}t\ ,\quad j \geq 2\ . \tag{120}$$

It follows from (114) that

$$\int_{-1}^{x} P_j(t)\,\mathrm{d}t = \frac{1}{\sqrt{\frac{1}{2}(2j+1)}}\psi_{j+1}(x)\ ,$$

and hence also

$$\int_{-1}^{x} P_{j-2}(t)\,\mathrm{d}t = \frac{1}{\sqrt{\frac{1}{2}(2j-3)}}\psi_{j-1}(x)$$

if we allow for the definition of "ψ_1". Thus we may rewrite (120) as

$$\int_{-1}^{x} \psi_j(t)\,\mathrm{d}t = \frac{1}{\sqrt{2(2j-1)}}\left(\frac{1}{\sqrt{\frac{1}{2}(2j+1)}}\psi_{j+1}(x) - \frac{1}{\sqrt{\frac{1}{2}(2j-3)}}\psi_{j-1}(x)\right)\ . \tag{121}$$

Especially, since for $j \geq 2$ the ψ_j are bubbles, at $x = 1$ the expression (121) vanishes for all $j \geq 3$. For $j = 2$, we note that by (118), we have $\psi_1(1) = \sqrt{2}$ (in the last term above), and hence

$$\int_{-1}^{1} \psi_2(t)\,\mathrm{d}t = -\frac{1}{\sqrt{2 \cdot 3}} \frac{1}{\sqrt{\frac{1}{2}}} \sqrt{2} = -\sqrt{\frac{2}{3}}\,.$$

We will need this for a special case at the end.

Finally, we recall the L^2-orthogonality of the Legendre polynomials on $(-1, 1)$:

$$\int_{-1}^{1} P_m\, P_n\,\mathrm{d}t = \frac{2}{2n+1}\,\delta_{mn}\,, \tag{122}$$

where δ_{mn} is the Kronecker delta and $m, n \geq 0$.

Alternative explicit representation

As was noted above, Legendre polynomials $P_n(x)$ follow Bonnet's recurrence formula, equation (108), repeated here for convenience:

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)\,. \tag{123}$$

Considering the ψ_j, by (117), for any $j \geq 2$ it holds that

$$\sqrt{2(2j-1)}\psi_j(x) = P_j(x) - P_{j-2}(x)\,.$$

From (123), we observe that if we set $j = n + 1$ (whence $n = j - 1$), we have

$$jP_j = (2(j-1)+1)P_{j-1} - (j-1)P_{j-2} = (2j-1)xP_{j-1} - (j-1)P_{j-2}\,.$$

By subtracting jP_{j-2} from both sides,

$$j \cdot (\,P_j - P_{j-2}\,) = (2j-1)xP_{j-1} - (2j-1)P_{j-2} = (2j-1)(\,xP_{j-1} - P_{j-2}\,)\,.$$

This gives us an alternative explicit representation for $\psi_j(x)$:

$$\psi_j(x) = \frac{\sqrt{2j-1}}{j\sqrt{2}}(\,xP_{j-1}(x) - P_{j-2}(x)\,)\,. \tag{124}$$

Numerically, neither one seems better a priori; as j increases, both (117) and (124) have cancellation problems at some parts of the domain.

Numerical evaluation of the basis functions

For up to $n = 24$ or so, standard double precision floating point (with 53-bit mantissa) is sufficient. This was tested in two ways.

The first test used numpy.polynomial.legendre.leg2poly(), numpy.polynomial.Polynomial, and the integ() method of Polynomial objects, evaluating the definition (114) directly. Note that (114) must not be integrated numerically; rounding errors accumulate, and thus for high degrees, when integrated numerically, the result will miss zero at the right endpoint. Much better is to observe that ψ_j is a polynomial, treating the integration as an exact operation on the coefficients.

In the second test, similar results were obtained by Legendre series evaluation (np.polynomial.legendre) of the explicit solution (117). The problem in this form is cancellation; whether one uses (117) or (124), there are some parts of the domain $[-1, 1]$ where one must subtract numbers that are almost equal in order to obtain the value of ψ_j. Since the Legendre polynomials P_n are oscillatory for $n \geq 2$, the higher j is, the more problematic regions appear. Above $n = 30$, accuracy is completely lost (as one can observe from the computed values of the P_n near $x = 1$).

In either case, the problem is that the coefficients of the Legendre polynomials P_n grow very quickly as the degree n increases, and that the P_n are oscillatory — making ψ_j difficult to evaluate, even if one were to use (114) with analytically computed P_{j-1}, exact arithmetic for the coefficients, and analytical integration for polynomials.

For high degrees, an easy practical approach is to use SymPy's mpmath module[17], which provides an arbitrary-precision floating point library. The library includes sympy.mpmath.legendre for evaluating Legendre polynomials. The arbitrary precision (and even for 53-bit results, higher internal working precision) makes it possible to evaluate (117) directly without worrying about cancellation. This was tested up to $q = 300$ and the results seemed very accurate. The downside is that it is very slow, because it is running in pure software. Making a precomputed table of course helps in practice; this is implemented in precalc.py.

4.8.5 Useful matrices for Galerkin applications

Because the coefficients of the Legendre polynomials P_n grow very quickly as the degree n increases, for reasons of numerical stability it is preferable to use a semi-analytical method, where the integrals in the Galerkin matrices are treated analytically.

In this section, we give some useful matrices for the treatment of up to second-order problems in weak form. This is slightly more general than we need (our ODE system needs only the matrix C_{ji}), but is provided for the sake of completeness.

Let us restrict our consideration to $i, j \geq 2$. The first two rows and columns will be handled later.

A typical stiffness matrix is (j = row, i = column)

$$\begin{aligned} K_{ji} &:= \int_{-1}^{1} \psi_i' \, \psi_j' \, \mathrm{d}t \\ &= \int_{-1}^{1} \left(\sqrt{\frac{1}{2}(2i-1)} P_{i-1} \sqrt{\frac{1}{2}(2j-1)} P_{j-1} \right) \mathrm{d}t \\ &= \sqrt{\frac{1}{2}(2i-1)} \sqrt{\frac{1}{2}(2j-1)} \int_{-1}^{1} P_{i-1} \, P_{j-1} \, \mathrm{d}t \\ &= \sqrt{\frac{1}{2}(2i-1)} \sqrt{\frac{1}{2}(2j-1)} \frac{2}{2j-1} \delta_{ij} \\ &= \frac{1}{2}(2j-1) \frac{2}{2j-1} \delta_{ij} \\ &= \delta_{ij} \, . \end{aligned} \tag{125}$$

This is of course the motivation for using integrals of Legendre polynomials as the basis functions ψ_j, and also for the particular choice of the scaling factor in (114).

A typical damping or gyroscopic matrix is

$$C_{ji} = \int_{-1}^{1} \psi_i' \, \psi_j \, \mathrm{d}t \, . \tag{126}$$

Note that this is also seen as the mass matrix for the first-order initial value problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= f(\, u(t), \, t \,) \, , \\ u(0) &= u_0 \, . \end{aligned}$$

By (117) and (119), the matrix element (126) becomes

$$C_{ji} = \sqrt{\frac{1}{2}(2i-1)} \frac{1}{\sqrt{2(2j-1)}} \int_{-1}^{1} P_{i-1} \left(\, P_j - P_{j-2} \, \right) \, \mathrm{d}t \, , \tag{127}$$

i.e.

$$C_{ji} = \sqrt{\frac{2i-1}{2j-1}} \int_{-1}^{1} \left(\, P_{i-1} \, P_j \; - \; P_{i-1} \, P_{j-2} \, \right) \, \mathrm{d}t \, . \tag{128}$$

[17] http://docs.sympy.org/0.7.1/modules/mpmath/index.html

By applying the L^2-orthogonality relation (122) and performing algebraic manipulation, we have

$$\begin{aligned}
C_{ji} &= 2\sqrt{\frac{2i-1}{2j-1}}\left(\frac{1}{2j+1}\,\delta_{i-1,\,j} \;-\; \frac{1}{2j-3}\,\delta_{i-1,\,j-2}\right) \\
&= 2\sqrt{\frac{2i-1}{2j-1}}\left(\frac{1}{2j+1}\,\delta_{i-1,\,j} \;-\; \frac{1}{2j-3}\,\delta_{i+1,\,j}\right) \\
&= 2\sqrt{\frac{2i-1}{2j-1}}\,\frac{1}{2j+1}\,\delta_{i-1,\,j} - 2\sqrt{\frac{2i-1}{2j-1}}\,\frac{1}{2j-3}\,\delta_{i+1,\,j} \\
&= 2\sqrt{\frac{2(j+1)-1}{2j-1}}\,\frac{1}{2j+1}\,\delta_{i-1,\,j} - 2\sqrt{\frac{2(j-1)-1}{2j-1}}\,\frac{1}{2j-3}\,\delta_{i+1,\,j} \\
&= 2\sqrt{\frac{2j+1}{2j-1}}\,\frac{1}{2j+1}\,\delta_{i-1,\,j} - 2\sqrt{\frac{2j-3}{2j-1}}\,\frac{1}{2j-3}\,\delta_{i+1,\,j} \\
&= 2\sqrt{\frac{1}{(2j-1)(2j+1)}}\,\delta_{i-1,\,j} - 2\sqrt{\frac{1}{(2j-1)(2j-3)}}\,\delta_{i+1,\,j}\,. \qquad (129)
\end{aligned}$$

Note the bidiagonal structure and zero main diagonal.

How about the symmetry properties? Consider

$$\begin{aligned}
C_{ij} &= 2\sqrt{\frac{1}{(2i-1)(2i+1)}}\,\delta_{j-1,\,i} - 2\sqrt{\frac{1}{(2i-1)(2i-3)}}\,\delta_{j+1,\,i} \\
&= 2\sqrt{\frac{1}{(2(j-1)-1)(2(j-1)+1)}}\,\delta_{j-1,\,i} - 2\sqrt{\frac{1}{(2(j+1)-1)(2(j+1)-3)}}\,\delta_{j+1,\,i} \\
&= 2\sqrt{\frac{1}{(2j-3)(2j-1)}}\,\delta_{j-1,\,i} - 2\sqrt{\frac{1}{(2j+1)(2j-1)}}\,\delta_{j+1,\,i} \\
&= 2\sqrt{\frac{1}{(2j-3)(2j-1)}}\,\delta_{j,\,i+1} - 2\sqrt{\frac{1}{(2j+1)(2j-1)}}\,\delta_{j,\,i-1} \\
&= 2\sqrt{\frac{1}{(2j-3)(2j-1)}}\,\delta_{i+1,\,j} - 2\sqrt{\frac{1}{(2j+1)(2j-1)}}\,\delta_{i-1,\,j} \\
&= -2\sqrt{\frac{1}{(2j+1)(2j-1)}}\,\delta_{i-1,\,j} + 2\sqrt{\frac{1}{(2j-3)(2j-1)}}\,\delta_{i+1,\,j} \\
&= -2\sqrt{\frac{1}{(2j-1)(2j+1)}}\,\delta_{i-1,\,j} + 2\sqrt{\frac{1}{(2j-1)(2j-3)}}\,\delta_{i+1,\,j} \\
&= -C_{ji}\,.
\end{aligned}$$

We see that C_{ji} is antisymmetric (skew-symmetric).

A typical mass matrix is

$$
\begin{aligned}
M_{ji} &:= \int_{-1}^{1} \psi_i \, \psi_j \, \mathrm{d}t \\
&= \int_{-1}^{1} \left[\frac{1}{\sqrt{2(2i-1)}} \left(P_i - P_{i-2} \right) \frac{1}{\sqrt{2(2j-1)}} \left(P_j - P_{j-2} \right) \right] \mathrm{d}t \\
&= \frac{1}{\sqrt{2(2i-1)}} \frac{1}{\sqrt{2(2j-1)}} \int_{-1}^{1} \left(P_i \, P_j \; - \; P_i \, P_{j-2} \; - \; P_{i-2} \, P_j \; + \; P_{i-2} \, P_{j-2} \right) \mathrm{d}t \\
&= 2 \frac{1}{\sqrt{2(2i-1)}} \frac{1}{\sqrt{2(2j-1)}} \left(\frac{1}{2j+1} \, \delta_{ij} - \frac{1}{2j-3} \, \delta_{i,j-2} - \frac{1}{2j+1} \, \delta_{i-2,j} + \frac{1}{2j-3} \, \delta_{i-2,j-2} \right) \\
&= \frac{1}{\sqrt{2i-1}} \frac{1}{\sqrt{2j-1}} \left(\frac{1}{2j+1} \, \delta_{ij} - \frac{1}{2j-3} \, \delta_{i,j-2} - \frac{1}{2j+1} \, \delta_{i-2,j} + \frac{1}{2j-3} \, \delta_{i-2,j-2} \right) \\
&= \frac{1}{2j-1} \left(\frac{1}{2j+1} + \frac{1}{2j-3} \right) \delta_{ij} - \frac{1}{\sqrt{2(j-2)-1}} \frac{1}{\sqrt{2j-1}} \frac{1}{2j-3} \, \delta_{i+2,j} - \frac{1}{\sqrt{2(j+2)-1}} \frac{1}{\sqrt{2j-1}} \frac{1}{2j+1} \, \delta_{i-2,j} \\
&= \frac{1}{2j-1} \left(\frac{1}{2j+1} + \frac{1}{2j-3} \right) \delta_{ij} - \frac{1}{\sqrt{2j-5}} \frac{1}{(2j-3)} \frac{1}{\sqrt{2j-1}} \, \delta_{i+2,j} - \frac{1}{\sqrt{2j-1}} \frac{1}{(2j+1)} \frac{1}{\sqrt{2j+3}} \, \delta_{i-2,j} \, . \qquad (130)
\end{aligned}
$$

We have used $\delta_{i-k,j-k} = \delta_{ij}$ for any k, and $\delta_{i,j-2} = \delta_{i+2,j}$. Note the tridiagonal structure.

Note the symmetry $M_{ij} = M_{ji}$. We do not need to check this from the final result, because the definition (first line) is symmetric with respect to the exchange of i and j.

Equations (125), (129) and (130) allow explicit analytical treatment of the matrices **K**, **C** and **M** for matrix elements at $i, j \geq 2$. To complete the treatment, we must consider the first two rows and columns.

The first two rows and columns

First, observe that by definition, $P_0(x) = 1$ and $P_1(x) = x$. The functions $N_0(x)$ and $N_1(x)$ are actually their linear combinations:

$$N_0(x) = \frac{1}{2}(1 - x) = \frac{1}{2}(P_0 - P_1) \, , \qquad (131)$$

$$N_1(x) = \frac{1}{2}(1 + x) = \frac{1}{2}(P_0 + P_1) \, , \qquad (132)$$

so if needed, we have the option of utilizing the properties of Legendre polynomials also in these cases.

We will first treat the block $i, j = \{0, 1\}$ (upper left corner), and then consider the regions $i \geq 2$, $j = \{0, 1\}$ (first two rows) and $i = \{0, 1\}$, $j \geq 2$ (first two columns).

Upper left corner Let $i, j = \{0, 1\}$.

For the stiffness matrix, we have

$$
\begin{aligned}
K_{ji} &= \int_{-1}^{1} N_i' \, N_j' \, \mathrm{d}t \\
&= 2 \left(\frac{1}{2}(-1)^{i+1} \, \frac{1}{2}(-1)^{j+1} \right) \\
&= \frac{1}{2}(-1)^{i+1} \, (-1)^{j+1} \, ,
\end{aligned}
$$

i.e.

$$K_{ji} = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix} \, .$$

For the damping/gyroscopic matrix,

$$\begin{aligned} C_{ji} &= \int_{-1}^{1} N_i' \, N_j \, \mathrm{d}t \\ &= \int_{-1}^{1} \frac{1}{2}(-1)^{i+1} \, \frac{1}{2} \left(1 + (-1)^{j+1} t \right) \, \mathrm{d}t \\ &= \frac{1}{4} \left((-1)^{i+1} \int_{-1}^{1} 1 \, \mathrm{d}t + (-1)^{i+j+2} \int_{-1}^{1} t \, \mathrm{d}t \right) \\ &= \frac{1}{4} \left((-1)^{i+1} \cdot 2 \right) \\ &= \frac{1}{2}(-1)^{i+1} \, , \end{aligned}$$

i.e.

$$C_{ji} = \begin{bmatrix} -1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix} .$$

For the mass matrix,

$$\begin{aligned} M_{ji} &= \int_{-1}^{1} N_i \, N_j \, \mathrm{d}t \\ &= \int_{-1}^{1} \frac{1}{2} \left(P_0 + (-1)^{i+1} P_1 \right) \frac{1}{2} \left(P_0 + (-1)^{j+1} P_1 \right) \, \mathrm{d}t \\ &= \frac{1}{4} \int_{-1}^{1} \left(P_0 + (-1)^{i+1} P_1 \right) \left(P_0 + (-1)^{j+1} P_1 \right) \, \mathrm{d}t \\ &= \frac{1}{4} \left(2 + (-1)^{i+j} \cdot \frac{2}{3} \right) \\ &= \frac{1}{2} + \frac{1}{6}(-1)^{i+j} \, , \end{aligned}$$

i.e.

$$M_{ji} = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} .$$

We have used the L^2-orthogonality of the Legendre polynomials, equation (122), to discard the terms involving $P_0 \, P_1$, and to evaluate the terms involving P_0^2 and P_1^2.

First two rows Let $j = \{0, 1\}$ and $i \geq 2$.

For the stiffness matrix, we have

$$\begin{aligned} K_{ji} &= \int_{-1}^{1} N_i' \, N_j' \, \mathrm{d}t \\ &= \int_{-1}^{1} \psi_i' \cdot \frac{1}{2}(-1)^{j+1} \, \mathrm{d}t \\ &= \frac{1}{2}(-1)^{j+1} \int_{-1}^{1} \psi_i' \, \mathrm{d}t \\ &= \frac{1}{2}(-1)^{j+1} \left(\psi_i(1) - \psi_i(-1) \right) \\ &= 0 \, , \end{aligned}$$

because the functions ψ_i, $i \geq 2$, are bubbles. Because the first line is symmetric with respect to exchanging i and j, the same result holds also for the first two columns.

For the damping/gyroscopic matrix,

$$\begin{aligned}
C_{ji} &= \int_{-1}^{1} N_i' \, N_j \, \mathrm{d}t \\
&= \int_{-1}^{1} \psi_i' \cdot \frac{1}{2} \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= \int_{-1}^{1} \left(\sqrt{\frac{1}{2}(2i-1)} \right) P_{i-1} \cdot \frac{1}{2} \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= \frac{1}{2} \sqrt{\frac{1}{2}(2i-1)} \int_{-1}^{1} P_{i-1} \cdot \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \, ,
\end{aligned}$$

where we have used (119) to represent ψ_i'.

Due to the L^2-orthogonality, the only nonzero terms are obtained for $i = 2$:

$$\begin{aligned}
C_{j2} &= \frac{1}{2} \sqrt{\frac{3}{2}} \int_{-1}^{1} P_1 \cdot \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= (-1)^{j+1} \frac{1}{2} \sqrt{\frac{3}{2}} \int_{-1}^{1} P_1 \, P_1 \, \mathrm{d}t \\
&= (-1)^{j+1} \frac{1}{2} \sqrt{\frac{3}{2}} \frac{2}{3} \\
&= (-1)^{j+1} \frac{1}{\sqrt{6}} \, ,
\end{aligned}$$

where $j = \{0, 1\}$.

For the mass matrix,

$$\begin{aligned}
M_{ji} &= \int_{-1}^{1} N_i \, N_j \, \mathrm{d}t \\
&= \int_{-1}^{1} \psi_i \cdot \frac{1}{2} \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= \int_{-1}^{1} \frac{1}{\sqrt{2(2i-1)}} \left(P_i(x) - P_{i-2}(x) \right) \cdot \frac{1}{2} \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= \frac{1}{2\sqrt{2(2i-1)}} \int_{-1}^{1} \left(P_i(x) - P_{i-2}(x) \right) \cdot \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \, ,
\end{aligned}$$

where we have used (117) to represent ψ_i. Keep in mind that $i \geq 2$. Only the term with P_{i-2} contributes; nonzero terms appear for $i = 2$ and $i = 3$. We have

$$\begin{aligned}
M_{ji} &= -\frac{1}{2\sqrt{2(2i-1)}} \int_{-1}^{1} P_{i-2}(x) \cdot \left(P_0 + (-1)^{j+1} P_1 \right) \mathrm{d}t \\
&= -\frac{1}{2\sqrt{2(2i-1)}} \left(2\delta_{i-2,0} + (-1)^{j+1} \frac{2}{3} \delta_{i-2,1} \right) \\
&= -\left(2 \frac{1}{2\sqrt{2(2i-1)}} \delta_{i-2,0} + (-1)^{j+1} \frac{1}{2\sqrt{2(2i-1)}} \frac{2}{3} \delta_{i-2,1} \right) \\
&= -\left(\frac{1}{\sqrt{2}} \delta_{i-2,0} + (-1)^{j+1} \frac{1}{3\sqrt{10}} \delta_{i-2,1} \right) \\
&= -\left(\frac{1}{\sqrt{2}} \delta_{i,2} + (-1)^{j+1} \frac{1}{3\sqrt{10}} \delta_{i,3} \right) \, .
\end{aligned}$$

This holds for $j = \{0, 1\}$, $i = \{2, 3\}$.

Again, the definition of M_{ji} is symmetric with respect to exchanging i and j, so the same result holds also for the first two columns (swapping the roles of i and j).

First two columns As was observed, $M_{ji} = M_{ij}$ and $K_{ji} = K_{ij}$. The only case that has not been treated yet is C_{ji} for $j \geq 2$, $i = \{0, 1\}$.

$$\begin{aligned} C_{ji} &= \int_{-1}^{1} N_i' N_j \,\mathrm{d}t \\ &= \int_{-1}^{1} \frac{1}{2}(-1)^{i+1} \cdot \psi_j \,\mathrm{d}t \\ &= \frac{1}{2}(-1)^{i+1} \int_{-1}^{1} \psi_j \,\mathrm{d}t \,. \end{aligned}$$

As was noted at the beginning, for $j \geq 3$ the integral is zero. For $j = 2$, it is $-\sqrt{2/3}$, and thus we have the final result

$$C_{ji} = \frac{1}{\sqrt{6}}(-1)^i \delta_{j,2} \,.$$

This is valid for $j \geq 2$, $i = \{0, 1\}$. We note that this is antisymmetric (skew-symmetric) to the result for the first two rows, obtained further above; hence C_{ji} is antisymmetric for any $i, j \geq 0$.

5 Continuous Galerkin

A simpler, more classical cousin of dG(q) is obtained if we seek the solution in the class of continuous functions. This is interesting mainly for comparison; in practical use, the accuracy seems to be much worse than with dG(q).

This leads to continuous Galerkin methods ("cG", not to be confused with CG, the conjugate gradient method for linear equation systems). If we allow the sets of basis and test functions to be different, we can also form "cPG" i.e. continuous Petrov–Galerkin methods (not to be confused with PCG, the preconditioned conjugate gradient method).

Obviously, for a given polynomial degree q of the basis, some accuracy will be lost in comparison to dG(q). This is because of the additional requirement of continuity across timestep boundaries, which restricts the freedom of the Galerkin procedure in determining the optimal values for the coefficients. Instead of locally L^2-orthogonalizing the residual across each timestep, independent of the other timesteps, the residual is L^2-orthogonalized under the restriction that the Galerkin representation of the solution remains continuous across timestep boundaries. Hence, because there is less freedom to perform the fit, it is likely that the accuracy of the result will suffer.

Let us consider the integration of (1)–(2) over a single timestep using the Galerkin approach. As before, we multiply (1) by a test function w and integrate from $t_k - \varepsilon$ to $t_{k+1} - \varepsilon$:

$$\lim_{\varepsilon \to 0^+} \int_{t_k - \varepsilon}^{t_{k+1} - \varepsilon} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \lim_{\varepsilon \to 0^+} \int_{t_k - \varepsilon}^{t_{k+1} - \varepsilon} f(\,u(t),\, t\,)\, w \,\mathrm{d}t \quad \forall w \,, \tag{133}$$

$$u(t_k) = u_0 \,. \tag{134}$$

Note that u_0 denotes the initial condition for the current timestep. We require $u \in C^0(0, t_\mathrm{f})$ and $w \in C^{-1}(0, t_\mathrm{f})$. There are at most finite discontinuities in the integrands of (133). Hence, no jump terms will be generated, and we may immediately get rid of the limits, obtaining

$$\int_{t_k}^{t_{k+1}} \frac{\partial u}{\partial t} w \,\mathrm{d}t = \int_{t_k}^{t_{k+1}} f(\,u(t),\, t\,)\, w \,\mathrm{d}t \quad \forall w \,, \tag{135}$$

Then, in the usual Petrov–Galerkin manner, we represent u as a Galerkin series,

$$u(t) := \sum_{m=1}^{\infty} u_m \varphi_m(t) \,, \tag{136}$$

where u_m are the Galerkin coefficients, and $\varphi_m(t)$ are the global basis functions. We choose the test functions w_i as

$$w_i(t) := \psi_i(t) \,, \quad i = 1, 2, \ldots \tag{137}$$

allowing them to be different from the basis functions. We require $\varphi_m \in C^0(0, t_\mathrm{f})$ and $\psi_i \in C^{-1}(0, t_\mathrm{f})$. As before, typically the functions will be defined piecewise across $(0, t_\mathrm{f})$, with any discontinuities in φ_m' and ψ_i occurring on the element boundaries.

After truncation at M degrees of freedom, we obtain

$$\mathbf{A}\mathbf{u} = \mathbf{b}(\mathbf{u}) \,, \tag{138}$$

where $\mathbf{u}$ is an M-element vector consisting of the coefficients u_m, and

$$A_{im} = \int_{t_k}^{t_{k+1}} \psi_i(t) \, \frac{\partial \varphi_m}{\partial t}(t) \, \mathrm{d}t \,, \tag{139}$$

$$b_i(\mathbf{u}) = \int_{t_k}^{t_{k+1}} f\big(\sum_{m=1}^{M} u_m \varphi_m(t), \, t \, \big) \, \psi_i(t) \, \mathrm{d}t \,. \tag{140}$$

Considering what it describes, the equation system (138) by itself is rank-deficient. It becomes uniquely solvable when the initial condition (134) is applied. This may directly prescribe one of the u_m (thus eliminating one row and column), or just modify all of the rows slightly, depending on the basis used.

5.1 cP(0)G(1)

(cP(0)G(1) is not supported by the solver; this section is only included for information.)

Choosing to test (135) with the constant $w \equiv 1$, we obtain

$$\int_{t_k}^{t_{k+1}} \frac{\partial u}{\partial t} \, \mathrm{d}t = \int_{t_k}^{t_{k+1}} f\big(\, u(t), \, t \, \big) \, \mathrm{d}t \,,$$

from which, immediately,

$$u(t_{k+1}) - u(t_k) = \int_{t_k}^{t_{k+1}} f\big(\, u(t), \, t \, \big) \, \mathrm{d}t \,,$$

and by rearranging,

$$u(t_{k+1}) = u(t_k) + \int_{t_k}^{t_{k+1}} f\big(\, u(t), \, t \, \big) \, \mathrm{d}t \,. \tag{141}$$

Equation (141) coincides with (35), which (as was noted in section 3.3) reduces to different classical methods depending on the choices of the model for u and the quadrature rule to approximate the integral.

In the Petrov–Galerkin context, the model is (136), the details depending on the choice of basis. For example, we can choose a piecewise linear basis

$$\varphi_1(p) = 1 - p \,,$$
$$\varphi_2(p) = p \,,$$

where $p \in [0, 1]$ is the scaled time (on the reference element $[0, 1]$). This is the cP(0)G(1) method, i.e. continuous Petrov–Galerkin with (piecewise) constant test and piecewise linear basis. For this basis, $u_1 = u_0$ (by (136) and (134)) and only u_2 is unknown. Hence after applying the initial condition, the single equation (141) is enough to close the system.

If we now approximate the integral using the midpoint rule, we again obtain IMR. Approximating by the "endpoint rule" gives BE, and the "startpoint rule" gives FE.