Methods conserving first integrals

October 9, 2024

We can say that there are two classes of one-step numerical methods preserving first integrals (conservation laws): methods that require the knowledge of the first integral to preserve and rewrite the system of ODEs in a specific way to preserve it, methods that preserve a broad class of first integrals without having to know them or manipulate the equations.

We start with the second class of methods, focusing on polyomial invariants.

1 Polynomial invariants

1.1 Linear invariants

Theorem 1 (Conservation of linear first integrals)**.** *All explicit and implicit Runge–Kutta methods conserve linear invariants.*

Proof. We focus on the differential equation $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$, and suppose it admits the linear first integral $I(\mathbf{x}) = \mathbf{v}^T\mathbf{x}$, $\mathbf{v} \in \mathbb{R}^d$. In other words, we suppose that $\mathbf{v}^T f(\mathbf{x}) = 0$ for every $\mathbf{x} \in \mathbb{R}^d$.

Let us define the Runge–Kutta stages

$$\mathbf{k}_i = \mathbf{x}_0 + h\sum_{j=1}^{s} a_{ij} f(\mathbf{k}_j),$$

and the associated one-step update

$$\mathbf{x}_1 = \mathbf{x}_0 + h\sum_{i=1}^{s} b_i f(\mathbf{k}_i).$$

The proof can be concluded by the following derivation:

$$I(\mathbf{x}_1) = \mathbf{v}^T\mathbf{x}_0 + h\sum_{i=1}^{s} b_i \mathbf{v}^T f(\mathbf{k}_i) = \mathbf{v}^T\mathbf{x}_0 = I(\mathbf{x}_0).$$

□

Exercise 1. *Implement the Runge–Kutta method you prefer, and verify that when applied to the equations of the SIR model*

$$\begin{cases} \dot{S}(t) = -\beta I(t)S(t), \\ \dot{I}(t) = \beta I(t)S(t) - \gamma I(t), \\ \dot{R}(t) = \gamma I(t), \end{cases}$$

it preserves the linear invariant $H(S, I, R) = S + I + R$.

1.2 Quadratic invariants

We now focus on quadratic functions of the form $Q(\mathbf{x}) = \mathbf{x}^T C\mathbf{x}$ with $C \in \mathbb{R}^{d\times d}$ symmetric. This is a generic enough quadratic function since if we manage to conserve it, we will also be able to conserve something like $\tilde{Q}(\mathbf{x}) = \mathbf{x}^T C\mathbf{x} + \mathbf{v}^T\mathbf{x}$ by linearity and Theorem 1.

Remark 1. *The theorem and proof we are going to see now is very important, since this result will be useful in several other situations.*

Theorem 2 (Conservation of quadratic first integrals)**.** *Let us consider a Runge–Kutta method with tableau determined by the triple* (A, b, c)*. Let us define the matrices* $B = \mathrm{diag}(b)$ *and* $M = BA + A^T B - bb^T$*. If* $M = 0_{d\times d}$ *then the Runge–Kutta method under consideration conserves quadratic first integrals.*

Before moving to the proof, we explicitly write down the entry m_{ij} of the matrix $M \in \mathbb{R}^{s\times s}$:

$$m_{ij} = b_i a_{ij} + a_{ji} b_j - b_i b_j.$$

Proof. Let us focus on the differential equation $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$ and suppose it admits the first integral $Q(\mathbf{x}) = \mathbf{x}^T C\mathbf{x}$, i.e., that $\mathbf{x}^T C f(\mathbf{x}) = 0$ for every $\mathbf{x} \in \mathbb{R}^d$. The hidden stages of the Runge–Kutta method under consideration write

$$\mathbf{k}_i = \mathbf{x}_0 + h\sum_{j=1}^{s} a_{ij} f(\mathbf{k}_j), \tag{1}$$

leading to the update $\mathbf{x}_1 = \mathbf{x}_0 + h\sum_{i=1}^{s} b_i f(\mathbf{k}_i)$. We now expand $Q(\mathbf{x}_1)$ to get

$$Q(\mathbf{x}_1) = \mathbf{x}_1^T C\mathbf{x}_1 = Q(\mathbf{x}_0) + h^2 \sum_{i,j=1}^{s} b_i b_j f(\mathbf{k}_i)^T C f(\mathbf{k}_j) + 2h\sum_{i=1}^{s} b_i \mathbf{x}_0^T C f(\mathbf{k}_i). \tag{2}$$

We now notice that, from (1), we can express $\mathbf{x}_0$ in s different ways as

$$\mathbf{x}_0 = \mathbf{k}_i - h\sum_{j=1}^{s} a_{ij} f(\mathbf{k}_j).$$

We replace this i-dependent rewriting in the last term of (2), to get

$$\begin{aligned}
Q(\mathbf{x}_1) = \mathbf{x}_1^T C \mathbf{x}_1 &= Q(\mathbf{x}_0) + h^2 \sum_{i,j=1}^{s} b_i b_j f(\mathbf{k}_i)^T C f(\mathbf{k}_j) \\
&+ \underbrace{2h \sum_{i=1}^{s} b_i \mathbf{k}_i^T C f(\mathbf{k}_i)}_{=0} - 2h^2 \sum_{i,j=1}^{s} b_i a_{ij} f(\mathbf{k}_j)^T C f(\mathbf{k}_i) \\
&= Q(\mathbf{x}_0) + h^2 \sum_{i,j=1}^{s} b_i b_j f(\mathbf{k}_i)^T C f(\mathbf{k}_j) - h^2 \sum_{i,j=1}^{s} b_i a_{ij} f(\mathbf{k}_j)^T C f(\mathbf{k}_i) \\
&- h^2 \sum_{i,j=1}^{s} b_j a_{ji} f(\mathbf{k}_i)^T C f(\mathbf{k}_j) \\
&= Q(\mathbf{x}_0) - h^2 \sum_{i,j=1}^{s} m_{ij} f(\mathbf{k}_i)^T C f(\mathbf{k}_j) = Q(\mathbf{x}_0)
\end{aligned}$$

if $m_{ij} = 0$ for every $i, j \in \{1, ..., s\}$. □

The condition $M = 0$ also becomes necessary for irreducible Runge–Kutta methods, see [1] for details. **add details here to a reference, but do not say this in the lecture.**

Example 1. *An example of a Runge–Kutta method which preserves quadratic first integrals is the implicit mid-point method, which has tableau*

$$A = 1/2, \quad b = 1, \quad c = 1/2,$$

and writes

$$\mathbf{x}_1 = \mathbf{x}_0 + hf\left(\frac{\mathbf{x}_0 + \mathbf{x}_1}{2}\right).$$

In this case, in fact, one has

$$M = BA + A^T B - bb^T = 1/2 + 1/2 - 1 = 0.$$

The implicit mid-point is a particular case of a more general family of methods preserving quadratic first integral, as we argue in the following proposition.

Proposition 1. *All Gauss-Legendre collocation methods preserve quadratic first integrals.*

Proof. The proof does not aim to show that these methods satisfy the assumptions of Theorem 2, but we use the main properties of collocation methods.

Let $\mathbf{u}(t)$ be the polynomial approximation of the solution we build, and let us consider $q(t) := Q(\mathbf{u}(t)) = \mathbf{u}(t)^T C \mathbf{u}(t)$. The time derivative $\dot{q}(t)$ writes

$$\frac{d}{dt} q(t) = 2\dot{\mathbf{u}}(t)^T C \mathbf{u}(t).$$

We thus have

$$Q(\mathbf{x}_1) := Q(u(t+h)) = Q(\mathbf{x}_0) + \int_{t_0}^{t_0+h} \dot{q}(t)dt \tag{3}$$

$$= Q(\mathbf{x}_0) + 2h \int_{t_0}^{t_0+h} \dot{\mathbf{u}(t)}^T C\mathbf{u}(t)dt. \tag{4}$$

Recalling that Gaussian quadrature rules based on s quadrature nodes are exact for polynomials up to degree $2s-1$, and that $\mathbf{u}(t)$ is of degree s, we conclude that the integral in (4) can be exactly computuded by Gaussian quadrature. Thus, we get

$$\begin{aligned} Q(\mathbf{x}_1) - Q(\mathbf{x}_0) &= 2h \sum_{i=1}^{s} \dot{\mathbf{u}}(t_0 + c_i h)^T C\mathbf{u}(t_0 + c_i h) \\ &= 2h \sum_{i=1}^{s} f(\mathbf{u}(t_0 + c_i h))^T C\mathbf{u}(t_0 + c_i h) = 0, \end{aligned}$$

which allows us to conclude the proof. □

Example 2 (Free rigid body)**.** *An interesting example in this case is provided by the equations of motion of a free rigid body, which can be written as*

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & \mathbf{x}_3/I_3 & -\mathbf{x}_2/I_2 \\ -\mathbf{x}_3/I_3 & 0 & \mathbf{x}_1/I_1 \\ \mathbf{x}_2/I_2 & -\mathbf{x}_1/I_1 & 0 \end{bmatrix} \mathbf{x}(t),$$

and hence have the quadratic first integral $Q(\mathbf{x}) = \|\mathbf{x}\|_2^2$. They also conserve another first integral which is still quadratic: $\tilde{Q}(\mathbf{x}) = \mathbf{x}^T I^{-1}\mathbf{x}$ where $I = \mathrm{diag}(I_1, I_2, I_3) \in \mathbb{R}^{3\times 3}$.

Approximate the solution to this system by using the implicit mid-point method and verify that the two first integrals are both preserved. What does this imply?

PROBLEM: As can be found in [1, Theorem IV 3.3], for $n \geq 3$, there is no Runge–Kutta method that can conserve all the poylnomial invariants of degree n. We thus need to look for other families of methods that help us preserving them, but also non-polynomial first integrals.

2 Methods preserving generic first integrals

The methods we are going to see now belong two classes: projection methods and discrete gradient methods. Both of these families of methods rely on the knowledge of the first integral to be preserved, and the method is built in such a way that we can preserve the level sets of this target function.

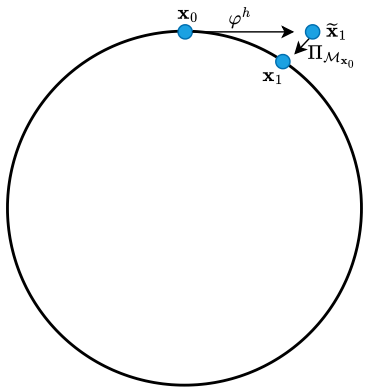

Figure 1: Illustration of one step of a projection method based on the one-step method $\varphi^h : \mathbb{R}^d \to \mathbb{R}^d$.

2.1 Projection methods

We can reformulate the problem of preserving the invariant function $I : \mathbb{R}^d \to \mathbb{R}$ into something much more generic. Conserving a first integral indeed means preserving its level sets. More precisely, let us suppose to start from the initial condition $\mathbf{x}_0$, then what we really want to do is ensure that the updated positions we get starting there remain on the (non-linear) manifold

$$\mathcal{M}_{\mathbf{x}_0} := \left\{ \mathbf{x} \in \mathbb{R}^d : \ I(\mathbf{x}) = I(\mathbf{x}_0) \right\} \subset \mathbb{R}^d,$$

which has dimension $d - 1$. We can thus interpret this desired result as a constraint over the numerical method.

After one step starting at $\mathbf{x}_0$ a generic one-step method will exit the manifold $\mathcal{M}_{\mathbf{x}_0}$, but we will not get "too far" from it. We can then project back the obtained point to the correct level set. This is the main idea behind projection methods, also illustrated in Figure 1. We now formalise the definition of these methods and provide an example for a simple conserved quantity.

Let us consider an arbitrary one-step method $\varphi^h : \mathbb{R}^d \to \mathbb{R}^d$ of order p, for example a Runge–Kutta method. We define a projection method based on φ^h as follows:

$$\widetilde{\mathbf{x}}_1 = \varphi^h(\mathbf{x}_0), \quad \mathbf{x}_1 := \Pi_{\mathcal{M}_{\mathbf{x}_0}}(\widetilde{\mathbf{x}}_1),$$

where

$$\Pi_{\mathcal{M}_{\mathbf{x}_0}}(\widetilde{\mathbf{x}}_1) := \arg \min_{\mathbf{y} \in \mathcal{M}_{\mathbf{x}_0}} \|\mathbf{y} - \widetilde{\mathbf{x}}_1\|_2 .$$

A practical way to compute these terms, is by defining

$$\mathbf{x}_1(\lambda) = \widetilde{\mathbf{x}}_1 + \lambda \nabla I(\widetilde{x}_1),$$

and looking for a $\lambda_* \in \mathbb{R}$ such that $\mathbf{x}_1(\lambda_*) \in \mathcal{M}_{\mathbf{x}_0}$, i.e., $I(\mathbf{x}_1(\lambda_*)) = I(\mathbf{x}_0)$. We also remark that if I is smooth enough, one has

$$I\left(\widetilde{\mathbf{x}}_1\right) = I\left(\phi^h(\mathbf{x}_0) + \mathcal{O}(h^{p+1})\right) = I\left(\phi^h(\mathbf{x}_0)\right) + \mathcal{O}\left(h^{p+1}\right) = I\left(\mathbf{x}_0\right) + \mathcal{O}\left(h^{p+1}\right),$$

and hence $\mathbf{x}_0$ tends to be a good initial guess for the iterative method we have to use to solve the (non-linear) algebraic equation to satisfy the constraints.

Example 3. *For the case $I(\mathbf{x}) = \|\mathbf{x}\|_2^2/2$, we have $\nabla I(\mathbf{x}) = \mathbf{x}$, and hence $\mathbf{x}_1(\lambda) = (1+\lambda)\varphi_h(\mathbf{x}_0)$. The preservation of the first integral is then equivalent to*

$$(1+\lambda)^2 \left\|\varphi_h(\mathbf{x}_0)\right\|_2^2 = \left\|\mathbf{x}_0\right\|_2^2,$$

which implies

$$\lambda = -1 \pm \frac{\|\mathbf{x}_0\|_2}{\|\varphi_h(\mathbf{x}_0)\|_2}.$$

Since when $h = 0$ we would like to recover $\lambda = 0$, we choose the plus sign, hence getting the projection method

$$\mathbf{x}_1 = \frac{\varphi_h(\mathbf{x}_0)}{\left\|\varphi_h(\mathbf{x}_0)\right\|_2} \left\|\mathbf{x}_0\right\|_2.$$

Exercise 2. *Think of how one could extend this reasoning to multiple first integrals, so to situations where one does not just want to preserve the level sets of I, but of a finite set of independent first integrals $I_1, ..., I_n : \mathbb{R}^d \to \mathbb{R}$.*

Typically projection methods are not so well seen in the community of geometric integration, because even though effective in preserving the conservation laws, they tend to break any other property the base method φ_h might have had (e.g., preserving linear invariants, being symplectic, or being volume preserving).

2.2 Discrete gradient methods

The last class of energy preserving methods we are going to see are discrete gradient methods. These are based on rewriting the vector field $f : \mathbb{R}^d \to \mathbb{R}^d$ in a convenient way, which is the skew-gradient formulation we have already mentioned. More explicitly, we rewrite them as

$$f(\mathbf{x}) = S(\mathbf{x})\nabla I(\mathbf{x}),\ \mathbb{R}^{d\times d} \ni S(\mathbf{x})^T = -S(\mathbf{x}),$$

where $I : \mathbb{R}^d \to \mathbb{R}$ is known to be a first integral of f. Differently from projection methods, discrete gradient methods have to be implicit. They are based on the notion of discrete gradient, which we now define.

Definition 1 (Discrete gradient)**.** *Let $I : \mathbb{R}^d \to \mathbb{R}$ be a scalar valued continuously differentiable function. A discrete gradient of I is a function $\overline{\nabla} I : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ such that*

1. $\lim_{\mathbf{y}\to\mathbf{x}} \overline{\nabla} I(\mathbf{x},\mathbf{y}) = \nabla I(\mathbf{x})$, *for all* $\mathbf{x}\in\mathbb{R}^d$,
2. $\overline{\nabla} I(\mathbf{x},\mathbf{y})^T(\mathbf{y}-\mathbf{x}) = I(\mathbf{y}) - I(\mathbf{x})$ *for all* $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$.

We now provide three examples of discrete gradients:

1. Average Vector Field (AVF) :

$$\overline{\nabla} I(\mathbf{x},\mathbf{y}) := \int_0^1 \nabla I\left((1-s)\mathbf{x}+s\mathbf{y}\right) ds,$$

2. Gonzalez:

$$\overline{\nabla} I(\mathbf{x},\mathbf{y}) := \nabla I\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) + \frac{I(\mathbf{y}) - I(\mathbf{x}) - (\mathbf{y}-\mathbf{x})^T \nabla I\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right)}{\|\mathbf{y}-\mathbf{x}\|_2}(\mathbf{y}-\mathbf{x}),$$

3. Itoh-Abe:

$$\overline{\nabla} I(\mathbf{x},\mathbf{y}) := \begin{bmatrix} \frac{I(y_1,x_2,\ldots,x_d)-I(\mathbf{x})}{y_1-x_1} \\ \frac{I(y_1,y_2,x_3,\ldots,x_d)-I(y_1,x_2,x_3,\ldots,x_d)}{y_2-x_2} \\ \vdots \\ \frac{I(\mathbf{y})-I(y_1,\ldots,y_{d-1},x_d)}{y_d-x_d} \end{bmatrix}.$$

We now prove that the AVF provides a discrete gradient, and we leave the proof for the others as an exercise. First, let us notice that

$$\begin{aligned} I(\mathbf{y}) - I(\mathbf{x}) &= \int_0^s \frac{d}{ds} I((1-s)\mathbf{x}+s\mathbf{y}) ds \\ &= \int_0^s \nabla I((1-s)\mathbf{x}+s\mathbf{y})^T(\mathbf{y}-\mathbf{x}) ds = \overline{\nabla} I(\mathbf{x},\mathbf{y})^T(\mathbf{y}-\mathbf{x}) \end{aligned}$$

as desired. Furthermore,

$$\lim_{\mathbf{y}\to\mathbf{x}} \overline{\nabla} I(\mathbf{x},\mathbf{y}) = \int_0^1 \lim_{\mathbf{y}\to\mathbf{x}} \nabla I((1-s)\mathbf{x}+s\mathbf{y}) ds = \nabla I(\mathbf{x}).$$

Once we choose a discrete gradient $\overline{\nabla} I$, we can define the associated discrete gradient method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\overline{S}(\mathbf{x}_n,\mathbf{x}_{n+1})\overline{\nabla} I(\mathbf{x}_n,\mathbf{x}_{n+1}), \tag{5}$$

where $\overline{S}:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}^{d\times d}$ is such that $\overline{S}(\mathbf{x},\mathbf{x}) = S(\mathbf{x})$ and for every $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$, one has $\overline{S}(\mathbf{x},\mathbf{y})^T = -\overline{S}(\mathbf{x},\mathbf{y})$.

Proposition 2. *The scheme in* (5) *conserves the first integral* $I:\mathbb{R}^d\to\mathbb{R}$ *of* $f:\mathbb{R}^d\to\mathbb{R}^d$.

Proof. To proof follows by direct calculation. First, by one of the properties of discrete gradients we see that

$$I(\mathbf{x}_{n+1}) - I(\mathbf{x}_n) = \overline{\nabla} I(\mathbf{x}_n, \mathbf{x}_{n+1})^T (\mathbf{x}_{n+1} - \mathbf{x}_n).$$

Replacing the expression for $\mathbf{x}_{n+1} - \mathbf{x}_n$ defined by (5), we can see that

$$\frac{I(\mathbf{x}_{n+1}) - I(\mathbf{x}_n)}{h} = \overline{\nabla} I(\mathbf{x}_n, \mathbf{x}_{n+1})^T \overline{S}(\mathbf{x}_n, \mathbf{x}_{n+1}) \overline{\nabla} I(\mathbf{x}_n, \mathbf{x}_{n+1}) = 0$$

by the skew-symmetry of $\overline{S}(\mathbf{x}_n, \mathbf{x}_{n+1})$. □

An interesting remark is that in case of polynomial first integrals, the AVF method is appealing since polynomial integrals can be exactly computed by quadratures. Furthermore, even when we have not so smooth situations, we could rely on the Itoh-Abe approach, since this is not relying on the gradient information at all! This is a choice used in non-smooth optimisation, since the proposed methodology can also be extended to dissipative systems of the form

$$\dot{\mathbf{x}}(t) = -\nabla V(\mathbf{x}(t))$$

for example. For this class of equations, (5) is slightly changed to

$$\mathbf{x}_{n+1} = \mathbf{x}_n - h \overline{\nabla} V(\mathbf{x}_n, \mathbf{x}_{n+1}),$$

since instead of conserving the function $V : \mathbb{R}^d \to \mathbb{R}$ we want to dissipate V. This is desirable in case we are interested in minimising V. Indeed, we see that

$$V(\mathbf{x}_{n+1}) - V(\mathbf{x}_n) = \overline{\nabla} V(\mathbf{x}_n, \mathbf{x}_{n+1})^T (\mathbf{x}_{n+1} - \mathbf{x}_n) = -h \left\| \overline{\nabla} V(\mathbf{x}_n, \mathbf{x}_{n+1}) \right\|_2^2 \leq 0.$$

References

[1] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006.