Numerical methods for initial and boundary value problems in Mechanics and Control

Sato Martín de Almagro, Rodrigo

March 18, 2018

1 Initial value problems (IVPs) for ODEs

An IVP for an ordinary differential equation (ODE) system is as follows:

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(x_0) = y_0 \end{cases} \tag{1}$$

where $y' = \frac{\mathrm{d}y}{\mathrm{d}x}$, $y_0, y(x) \in \mathbb{R}^n$ and we are asked to find such a y for $x \in [x_0, x_1]$, with $x_1 > x_0$.

We will disregard the vast topic of conditions for existence and uniqueness of solutions of such an IVP [7]. We will instead assume that those conditions are satisfied and so it is possible to solve the problem uniquely.

In general, the vector field f can be quite complicated and finding an analytic solution may be impossible. In order to obtain quantitative data for such problems we turn to numerical methods to approximate their solution.

1.1 Numerical integrators for IVP

In this course we will be primarily interested in mechanical systems, so let us use t instead of x for the independent (evolution or time) variable and $\dot{y}$ for derivation w.r.t. t.

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \\ y(x_0) = y_0 \end{cases} \tag{2}$$

If we use the Lagrangian or Hamiltonian equations of motion of a mechanical system as ODE system for an IVP, then $y = [q, v]^T \in TQ$ (velocity phase space or tangent bundle of Q) or $y = [q, p]^T \in T^*Q$ (phase space or cotangent bundle of Q) respectively.

The numerical methods for an IVP that we are going to discuss try to provide an approximation y_1 to the actual value of the solution $y(t_0 + h)$ for small h by using suitable combinations of intermediate values, Y_i, known as (internal) stages. If we want to find $y(T)$, with $T \gg t_0 + h$, we only need to reapply the numerical method with the last computed point as initial value as many times as necessary until we reach the desired point. This can be done at a constant or at a variable-step rate, but we will be focusing on the former.

A single step of a typical constant-step method with s stages takes the form:

$$t_1 = t_0 + h, \qquad T_i = t_0 + hc_i h, \tag{3a}$$

$$y_1 = y_0 + h\sum_{j=1}^{s} b_j f\left(T_j, Y_j\right), \quad Y_i = y_0 + h\sum_{j=1}^{s} a_{ij} f\left(T_j, Y_j\right), \tag{3b}$$

where the coefficients a_{ij}, b_j, c_i define the method completely. These are usually arranged in a *Butcher tableau*:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Methods where $c_i = \sum_{j=1}^{s} a_{ij}$ are called Runge-Kutta methods. Some of the simplest examples are:

- The forward / explicit Euler method:

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{aligned} t_1 &= t_0 + h \\ y_1 &= y_0 + hf\left(t_0, y_0\right) \end{aligned}$$

- The backward / implicit Euler method:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{aligned} t_1 &= t_0 + h \\ y_1 &= y_0 + hf\left(t_1, y_1\right) \end{aligned}$$

- 1-stage Gauss method (a.k.a. the implicit midpoint rule):

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array} \qquad \begin{aligned} t_1 &= t_0 + h \\ T_1 &= t_0 + \frac{h}{2} \\ y_1 &= y_0 + hf\left(T_1, Y_1\right) \\ Y_1 &= y_0 + \frac{h}{2} f\left(T_1, Y_1\right) \end{aligned}$$

  which can be summarized as $y_1 = y_0 + hf\left(t_0 + \frac{h}{2}, \frac{1}{2}\left(y_1 + y_0\right)\right)$.

- 2-stage Lobatto method (a.k.a. the trapezoidal rule):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \qquad \begin{aligned} t_1 &= t_0 + h \\ y_1 &= y_0 + \frac{h}{2}\left[f\left(t_0, y_0\right) + f\left(t_1, y_1\right)\right] \end{aligned}$$

  where we have used that $(T_1, Y_1) = (t_0, y_0), (T_2, Y_2) = (t_1, y_1)$.

A method is said to be of (local) order p if:

$$y_1 - y(t_0 + h) = \mathcal{O}\left(h^{p+1}\right) \quad \text{as } h \to 0 \tag{4}$$

The first two methods listed above are of order 1 and the latter two are of order 2.

Do note that most if not all of the numerical methods that come already implemented in MATLAB or Python's SciPy package do not fall in this category. The former uses adaptive (variable step-size) RK methods such as the Dormand-Prince algorithm for its current `ode45` implementation, whereas the latter favours the use of linear multi-step (LM) and backward differentiation formula (BDF) methods that differ from those we have treated here.

1.2 Partitioned methods and mechanical systems

There is a more general class of RK-type methods called *partitioned methods*, where if the system admits a natural partition of the form:

$$\begin{cases} \dot{y}(t) = f(t, y(t), z(t)), \\ \dot{z}(t) = g(t, y(t), z(t)), \\ y(x_0) = y_0, \\ z(x_0) = z_0, \end{cases} \tag{5}$$

then we can apply different numerical rules for each part:

$$t_1 = t_0 + h, \qquad T_i = t_0 + hc_i h, \tag{6a}$$

$$y_1 = y_0 + h\sum_{j=1}^{s} b_j f\left(T_j, Y_j, Z_j\right), \quad Y_i = y_0 + h\sum_{j=1}^{s} a_{ij} f\left(T_j, Y_j, Z_j\right), \tag{6b}$$

$$z_1 = z_0 + h\sum_{j=1}^{s} \hat{b}_j g\left(T_j, Y_j, Z_j\right), \quad Z_i = z_0 + h\sum_{j=1}^{s} \hat{a}_{ij} g\left(T_j, Y_j, Z_j\right), \tag{6c}$$

so long as $c_i = \hat{c}_i$.

This happens to be the case in Mechanics and Geometric Control, in particular in the Hamiltonian setting. In the case of time-independent Mechanics the Hamiltonian equations of motion are naturally partitioned as:

$$\begin{cases} \dot{q}(t) = \frac{\partial H}{\partial p}(q, p), \\ \dot{p}(t) = -\frac{\partial H}{\partial q}(q, p), \end{cases} \tag{7}$$

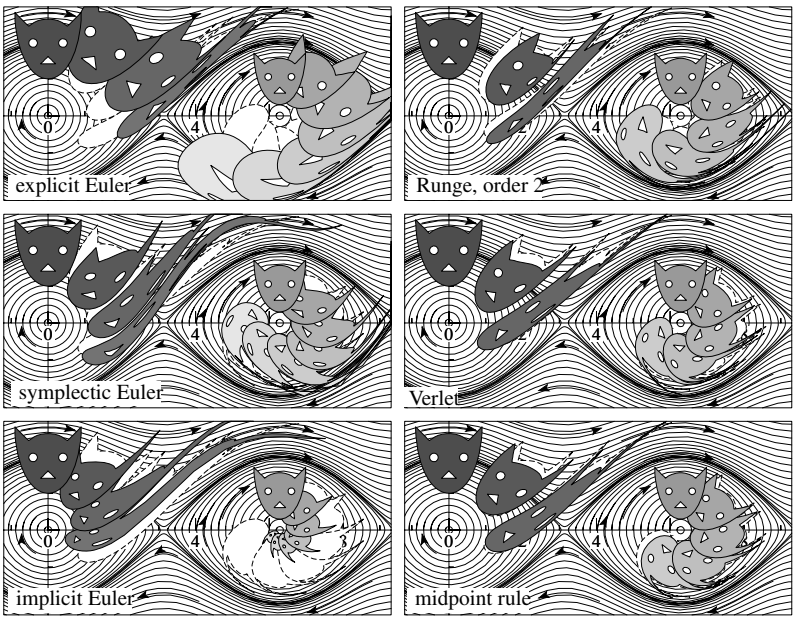

Figure 1: Phase space of the pendulum and showcase of volume (area) preservation properties of different integrators. Dashed lines show the exact evolution. First order methods on the left, second order methods on the right. Symplectic Euler, Verlet and midpoint rule are symplectic. From [6].

where $H : T^*Q \to \mathbb{R}$ is the corresponding Hamiltonian function of the system. In a formal course on Classical Mechanics we are shown that these equations preserve the *symplectic* structure of T^*Q, which is tantamount to volume preservation in phase space. A particular subset of partitioned RK integrators, unsurprisingly called symplectic partitioned RK (SPRK) methods, manage to discretely preserve the symplectic structure. For a partitioned RK method to be symplectic, the following relations must hold [6],[13]:

$$\begin{aligned} b_i \hat{a}_{ij} + \hat{b}_j a_{ji} = b_i \hat{b}_j, \quad &\text{for } i, j = 1, ..., s, \\ b_i = \hat{b}_i, \quad &\text{for } i = 1, ..., s. \end{aligned}$$

Do note that it is not necessary for a method to be partitioned in order to be symplectic as in fact all of Gauss' methods satisfy the symplecticity condition.

A SPRK method for Hamilton's equations takes the form [6],[13]:

$$q_1 = q_0 + h \sum_{j=1}^{s} b_j D_2 H\left(Q_j, P_j\right), \quad p_1 = p_0 - h \sum_{j=1}^{s} \hat{b}_j D_1 H\left(Q_j, P_j\right), \tag{8a}$$

$$Q_i = q_0 + h \sum_{j=1}^{s} a_{ij} D_2 H\left(Q_j, P_j\right), \quad P_i = p_0 - h \sum_{j=1}^{s} \hat{a}_{ij} D_1 H\left(Q_j, P_j\right), \tag{8b}$$

where the coefficients satisfy the symplecticity condition and where we have used the notation D_i to denote differentiation w.r.t. the i-th argument of a function.

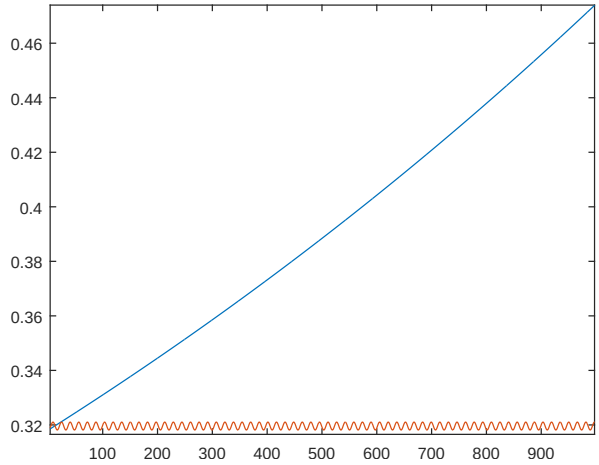

Figure 2: Energy preservation for the harmonic oscillator. Runge method of order 2 in blue (non-symplectic), trapezoidal rule in red (symplectic).

In general, symplectic methods do not preserve the energy (Hamiltonian) of the system exactly, but they present very good long-term behaviour. Typically they show oscillatory behaviour close to the exact energy of the system with no drift, so the integrator does not dissipate nor gain energy. In fact, it can be shown that every symplectic integrator is in fact preserving exactly a certain modified Hamiltonian which is close to the original one, which explains these good results.

1.3 Variational integrators

Every method obtained by applying the discrete variational principle is symplectic. The resulting numerical method obtained by deriving the discrete Euler-Lagrange equations (DEL) is directly related to the quadrature rule chosen to discretize the action and obtain the corresponding discrete Lagrangian.

As it turns out, SPRK methods can be derived automatically this way. In order to obtain partitioned methods we consider the Hamilton-Pontryagin action:

$$\mathcal{J}_{\mathcal{HP}}(q,v,p) = \int_0^h \left[L(q(t),v(t)) + \langle p(t), \dot{q}(t) - v(t)\rangle\right] \mathrm{d}t \tag{9}$$

and proceed to discretize this action. For this, one must choose a method that will be both applied as quadrature rule and as direct discretization of the dynamic constrain $\dot{q}(t) = v(t)$. If we chose an RK method with coefficients a_{ij}, b_j the resulting discretized action takes the form:

$$\begin{aligned}(\mathcal{J}_{\mathcal{HP}})_d = \sum_{k=0}^{N-1}\sum_{i=1}^{s} hb_i \Bigg[L\left(Q_k^i, V_k^i\right) + \left\langle P_k^i, \frac{Q_k^i - q_k}{h} - \sum_{j=1}^{s} a_{ij} V_k^j \right\rangle \\ + \left\langle p_{k+1}, \frac{q_{k+1}-q_k}{h} - \sum_{j=1}^{s} b_j V_k^j \right\rangle \Bigg]\end{aligned} \tag{10}$$

The resulting discrete Euler-Lagrange equations are [10]:

$$q_{k+1} = q_k + h\sum_{j=1}^{s} b_j V_k^j, \quad p_{k+1} = p_k + h\sum_{i=1}^{s} \hat{b}_j W_k^j, \tag{11a}$$

$$Q_k^i = q_k + h\sum_{j=1}^{s} a_{ij} V_k^j, \qquad P_k^i = p_k + h\sum_{j=1}^{s} \hat{a}_{ij} W_k^j, \tag{11b}$$

$$W_k^i = D_1 L(Q_k^i, V_k^i), \qquad P_k^i = D_2 L(Q_k^i, V_k^i), \tag{11c}$$

where the $\hat{a}_{ij}, \hat{b}_j$ coefficients are automatically symplectically conjugated with the original method we chose. Whenever we can move back and forth between the Lagrangian and Hamiltonian formulation of these systems, then (8) and (11) are equivalent.

In general the resulting equations differ from a naive application of a numerical method in TQ.

1.4 Forced mechanical systems

Forced mechanical systems are not variational and do not preserve the symplectic structure unless the forcing can be derived from a potential. Still, using symplectic / variational integrators for these seems to be a sensible choice, as numerical results show that they still closely match the behaviour of the continuous system without the addition of spurious energy dissipation or gain. Furthermore if the forcing vanishes, the system becomes effectively free and once more symplectic, so the argument in favour of symplectic integrators grows stronger.

In order to include forcing in our numerical description of mechanical systems we may use the discrete version of the Lagrange-D'Alembert principle.

The forced PRK methods that arise from the principle take the form:

$$q_{k+1} = q_k + h\sum_{j=1}^{s} b_j V_k^j, \qquad p_{k+1} = p_k + h\sum_{i=1}^{s} \hat{b}_j W_k^j, \tag{12a}$$

$$Q_k^i = q_k + h\sum_{j=1}^{s} a_{ij} V_k^j, \qquad P_k^i = p_k + h\sum_{j=1}^{s} \hat{a}_{ij} W_k^j, \tag{12b}$$

$$W_k^i = D_1 L(Q_k^i, V_k^i) + F(Q_k^i, V_k^i), \qquad P_k^i = D_2 L(Q_k^i, V_k^i), \tag{12c}$$

which is not a substantial change in the equations of a SPRK. This allows us to introduce control forces in the dynamics of a controlled mechanical system as $F(Q_k^i, V_k^i, U_k^i)$.

2 Boundary value problems (BVP) for ODEs

A Dirichlet BVP for a second order differential equation (SODE) has the typical form:

$$\begin{cases} \ddot{y}(t) = f(t, y(t), \dot{y}(t)), \\ y(t_0) = y_0, \\ y(t_1) = y_1. \end{cases} \tag{13}$$

This sort of problems can be extended to higher order differential equations or systems of equations of two or more dimensions.

Again, for this course we will disregard the topic of existence and uniqueness of solutions of such a BVP, which is far more challenging than the IVP counterpart [7]. We will instead assume that at least conditions for existence are met so it is possible to solve the problem. Global uniqueness may not be warranted, and indeed for many constrained optimal control problems there may be a multiplicity of solutions belonging to different homotopy classes.

2.1 Shooting method

Valid only for ODEs (i.e. it does not generalize to PDEs), it essentially involves transforming the BVP into an IVP where part of the initial values are taken as variable.

The concept behind it is simple and its name is already quite graphic. Imagine you have a cannon at some fixed position y_0 and you need to shoot a target sitting at y_1 which is within shooting range. In order to hit it you can aim your cannon, which amounts to changing the initial slope of the trajectory of your cannonball. In order to hit the target you shoot with some angle and evaluate how far away from the target you hit in order to correct your angle and try again. The process is to be repeated until you are satisfied with how close to the target position you hit.

The method can then be summarized as follows [4],[9]. Take your Dirichlet BVP and transform it into the IVP:

$$\begin{cases} \ddot{y}(t) = f(t, y(t), \dot{y}(t)), \\ y(t_0) = y_0, \\ \dot{y}(t_0) = \alpha. \end{cases} \tag{14}$$

Let us denote the solution of this problem as $y(t; \alpha)$ and define the function:

$$G(\alpha) = y(t_1; \alpha) - y_1. \tag{15}$$

Then, the solution of the original BVP amounts to finding α^* such that $G(\alpha^*) = 0$. Thus the process of solving the BVP reduces to solving a nonlinear equation.

Numerically one can solve the IVP for a fixed value α_0. This then serves as a seed for a Newton method:

$$\alpha_1 = \alpha_0 - \big(G'(\alpha_0)\big)^{-1} G(\alpha_0), \tag{16}$$

or similar root-finding algorithm. If the seed (or initial guess) is good enough, i.e. sufficiently close to an actual solution value, the iteration should converge.

If we use the Newton method we need to evaluate $G'(\alpha) = \frac{\partial y}{\partial \alpha}(t_1;\alpha)$. Differentiating the original SODE we have:

$$\frac{\partial \ddot{y}}{\partial \alpha} = \frac{\partial f}{\partial y}\frac{\partial y}{\partial \alpha} + \frac{\partial f}{\partial \dot{y}}\frac{\partial \dot{y}}{\partial \alpha} \tag{17}$$

Using the notation $z(t;\alpha) = \frac{\partial y}{\partial \alpha}(t;\alpha)$, then $G'(\alpha) = z(t_1;\alpha)$ can be found by solving the IVP:

$$\begin{cases} \ddot{z} = \frac{\partial f}{\partial y} z + \frac{\partial f}{\partial \dot{y}} \dot{z}, \\ z(t_0;\alpha) = 0, \\ \dot{z}(t_0;\alpha) = 1. \end{cases} \tag{18}$$

For its application with variational integrators, it seems far more sensible to apply the shooting method to the discrete equations instead of applying it to the continuous equations. If we take (11) with $p_k = D_2L(q_k, v_k)$ and set $v_0 = \alpha$, we can consider all variables as dependent of α and differentiate all the equations w.r.t. it. The discrete version of the G function will now be:

$$G_d(\alpha) = q_N(\alpha) - \hat{q}_N. \tag{19}$$

where $\hat{q}_N$ is the boundary value we want to reach.

Renaming the derivatives as $\left(\frac{\partial q_k}{\partial \alpha}, \frac{\partial v_k}{\partial \alpha}, \frac{\partial Q_k^i}{\partial \alpha}, \frac{\partial V_k^i}{\partial \alpha}\right) = \left(r_k, s_k, R_k^i, S_k^i\right)$, the resulting system that needs to be solved is:

$$r_{k+1} = r_k + h\sum_{j=1}^{s} b_j S_k^j \tag{20a}$$

$$R_k^i = r_k + h\sum_{j=1}^{s} a_{ij} S_k^j \tag{20b}$$

$$\begin{aligned} D_{21}L_{k+1}r_{k+1} + D_{22}L_{k+1}s_{k+1} &= D_{21}L_k r_k + D_{22}L_k s_k \\ &\quad + h\sum_{i=1}^{s} \hat{b}_j \left[D_{11}L_k^j R_k^j + D_{12}L_k^j S_k^j\right] \end{aligned} \tag{20c}$$

$$\begin{aligned} D_{21}L_k^i R_k^i + D_{22}L_k^i S_k^i &= D_{21}L_k r_k + D_{22}L_k s_k \\ &\quad + h\sum_{i=1}^{s} \hat{a}_{ij} \left[D_{11}L_k^j R_k^j + D_{12}L_k^j S_k^j\right] \end{aligned} \tag{20d}$$

with initial conditions $r_0 = 0, s_0 = 1$, and the value we are interested in is $r_N = G_d'(\alpha)$.

2.2 Multiple shooting method

As is, the shooting method can suffer from stability problems, especially when applied to highly non-linear or stiff problems. A variation of the method, called multiple shooting, has the potential to perform much better. As a nice little extra, it paves the way for the parallelization of the problem to take full advantage of current CPU and GPU trends.

The idea behind this method is also simple. Instead of performing a single shooting from the initial value and reach the final value, we can divide the complete interval into subintervals and simultaneously shoot from the starting point of each interval. Reparametrizing each so that they all run from $\tau = 0$ to $\tau = 1$, we effectively transform the many problems into a new single shooting problem of higher dimensions. The continuity conditions that must be imposed in order to solve the problem act as new boundary conditions.

We will not get into details here but the interested reader can check [4], [8].

2.3 Simultaneous solution methods

Simultaneous solution methods generalize to PDEs, in contrast with the shooting method. By these we mean a very wide variety of solution methods such as the finite differences method (FDM), the finite element method (FEM), or the boundary element method (BEM) where the algorithm reduces the BVP to the solution of a nonlinear system of equations involving the entire evolution, hence the name. In particular throughout this course we will use a variation of the FDM.

In the regular FDM one approximates a differential equation by directly discretising the differential operators that appear in it. Typically we would have [14]:

$$\dot{y}(t) = \frac{y(t+h) - y(t-h)}{2h} + \mathcal{O}\left(h^2\right), \tag{21a}$$

$$\ddot{y}(t) = \frac{y(t+h) - 2y(t) + y(t-h)}{h^2} + \mathcal{O}\left(h^2\right), \tag{21b}$$

Using the notation $t_i = t_0 + ih$ and $y(t_i) = y_i$, the BVP reduces to finding the solution of a system of equations of the form:

$$-y_{i+1} + 2y_i - y_{i-1} + h^2 f\left(t_i, y_i, \frac{y_{i+1} - y_i}{2h}\right) = 0, \quad i = 0, ..., N-1 \tag{22}$$

with the corresponding fixed boundary values y_0, y_N.

Instead of this, we will discretize our Hamilton-Pontryagin action, derive the corresponding discrete equations from the variational principle and try to solve them simultaneously with the corresponding boundary conditions. The solution of the resulting system of equations can be very demanding for a computer, and once again, as in the shooting method, the success of the algorithm will depend on how good our initial seed is.

The discrete Hamilton-Pontryagin action for a force-controlled Lagrangian sys-

tem is:

$$(\mathcal{J}_{\mathcal{HP}})_d = \sum_{k=0}^{N-1} \sum_{i=1}^{s} hb_i \Bigg[C\left(Q_k^i, V_k^i, U_k^i\right) \tag{23}$$

$$+ \left\langle \Lambda_k^i, \frac{Q_k^i - q_k}{h} - \sum_{j=1}^{s} a_{ij} V_k^j \right\rangle + \left\langle \lambda_{k+1}, \frac{q_{k+1} - q_k}{h} - \sum_{j=1}^{s} b_j V_k^j \right\rangle$$

$$+ \left\langle \mathrm{M}_k^i, \frac{D_2L(Q_k^i, V_k^i) - D_2L(q_k, v_k)}{h} - \sum_{j=1}^{s} \left(b_j - \frac{b_j a_{ji}}{b_i}\right) \left[D_1L(Q_k^j, V_k^j) + F(Q_k^j, V_k^j, U_k^j)\right] \right\rangle$$

$$+ \left\langle \mu_{k+1}, \frac{D_2L(q_{k+1}, v_{k+1}) - D_2L(q_k, v_k)}{h} - \sum_{j=1}^{s} b_j \left[D_1L(Q_k^j, V_k^j) + F(Q_k^j, V_k^j, U_k^j)\right] \right\rangle \Bigg]$$

where we have already substituted $\hat{a}_{ij}$ using the symplecticity condition. The resulting discrete Euler-Lagrange equations are left for the interested student to derive.

References

[1] R. Abraham and J.E. Marsden. *Foundations of Mechanics*. Addison Wesley, second edition, 1987.

[2] Sergio Blanes and Fernando Casas. *A concise introduction to geometric numerical integration*. Monographs and Research Notes in Mathematics. CRC Press, Boca Raton, FL, 2016.

[3] Cédric M. Campos, Sina Ober-Blöbaum, and Emmanuel Trélat. High order variational integrators in the optimal control of mechanical systems. *Discrete Contin. Dyn. Syst.*, 35(9):4193–4223, 2015.

[4] Moody T. Chu. Lecture notes - MA780: Numerical Analysis II. `http://www4.ncsu.edu/~mtchu/Teaching/Lectures/MA530/ma780.html`.

[5] Leonardo Colombo. *Geometric and numerical methods for optimal control of mechanical systems*. PhD thesis, Universidad Autónoma de Madrid, Madrid, 2014.

[6] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2010.

[7] P. Hartman. *Ordinary differential equations*, volume 38 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

[8] M. Kiehl. Parallel multiple shooting for the solution of initial value problems. *Parallel Computing*, 20(3):275 – 295, 1994.

[9] Taras I. Lakoba. Lecture notes - Math 337: Numerical Differential Equations. http://www.cems.uvm.edu/~tlakoba/math337/index.html.

[10] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numer.*, 10:357–514, 2001.

[11] John H. Mathews and Kurtis D. Fink. *Numerical Methods Using MATLAB.* Simon & Schuster, 3rd edition, 1998.

[12] Sina Ober-Blöbaum, Oliver Junge, and Jerrold E. Marsden. Discrete mechanics and optimal control: an analysis. *ESAIM Control Optim. Calc. Var.*, 17(2):322–352, 2011.

[13] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian problems*, volume 7 of *Applied Mathematics and Mathematical Computation.* Chapman & Hall, London, 1994.

[14] Eric W. Weisstein. Finite difference. http://mathworld.wolfram.com/FiniteDifference.html.