When it comes to simulation, at the heart of every algorithm lie the developer's choices made about how to perform *numerical integration*. That is, how the problem of calculating the definite integral of a general function f between two limits $a < b$

$$\int_a^b f(x)dx \tag{1}$$

is solved. This problem is central in computational physics, the research field dedicated to developing and analysing numerical operations to solve systems of differential equations. In this context, the above operation is called is called *quadrature* to distinguish it from the integration process of analytically solving (i.e. formulating the mathematical equations which satisfy) the above equation.[1]

[1] Steven E Koonin, Dawn C Meredith, and William H Press. "Computational Physics: Fortran Version". In: *Physics Today* 44 (1991), p. 112

Many different methods have been devised to tackle the problem, each having different strength and weaknesses. The most important aspect of these numerical operations is that all these methods are *approximations* and thus affected by error. The understanding of how this error affects the found numerical solution identifies two distinguishing characteristics of each method: its *order*, its *stability*, and the *computational effort* it needs:

- the *order* of a method indicates how big the error of the method is with respect to the segment length $b - a$: higher order means the method produces smaller errors;
- the *stability* of the method refers to the property of the algorithm of magnifying (instability) or not (stability) the above error when it is repeatedly applied;
- the *computational effort* refers to the number of operations needed by the algorithm to calculate the result. Specially in a framework (like *henri* (see ??), where quadrature operations are performed in real-time at audio rate, this aspect plays an important role. In general, less *computational effort* means smaller *order* and therefore worse approximations; ; therefore, given a particular numerical problem, finding a good balance between these two aspects, approximation and effort, is a central task for the programmer / researcher.

In general we are dealing with *initial value problems* for ordinary differential equations: meaning that we are looking for $x(t)$ functions which are solutions to

$$\dot{x}(t) = f(x(t)) \tag{2}$$

given the value

$$x(t_0 = 0) = x_0 \tag{3}$$

for some initial time t_0. It is easy to see that this kind of problem reduces to a similar operation as in eq 1 as we need to integrate $f(x)$ in order to find $x(t)$. For instance, This kind of if we are given the momentum of a particle and its position as time t_0 and wish to know it position at some later time, we are dealing with this kind of problem. If the function $f(x(t))$ is a continuous function, $x(t)$ is also continuous and can therefore be expressed in terms of its derivatives $\dot{x}, \ddot{x}, \ldots$ using a *Taylor series* to expand it in the neighbourhood of $t = 0$:

$$x(t) = x_0 + \dot{x}\, t + \ddot{x}\, \frac{t^2}{2!} + \dddot{x} \cdot \frac{t^3}{3!} + \ldots \tag{4}$$

where the derivatives are evaluated at $t = 0$.

Specifically we are interested in the value of $x(t)$ at particular values of t that are integer multiples of some fixed step h:

$$x_n = x(t = nh) \quad f_n = f(x_n) \quad n = 0, \pm 1. \pm 2, \ldots \tag{5}$$

e.g., h could be, as in the case of *henri*, the time interval between two audio samples, $1/44100 = 2.26e^{-5}$ seconds for a $44100Hz$ sampling rate. The above expansion in eq 4 becomes at $nt = \pm 1$

$$x_{\pm 1} = x_0 \pm \dot{x}_0 h + \ddot{x}_0 \frac{h^2}{2} + O(h^3) \tag{6}$$

where $O(h^3)$ stands for the terms of order h^3 or higher. Assuming that x and its derivatives are all approximately of the same order of magnitude, as is the case in many physical systems, these higher order terms will get smaller and smaller for higher powers if h is chosen small enough.

From the previous equation, focusing only on the lower order terms we can easily derive the *forward and backward difference formulas*:

$$\dot{x}_0 \approx \frac{x_1 - x_0}{h} + O(h) \quad (7)$$

$$\dot{x}_0 \approx \frac{x_0 - x_{-1}}{h} + O(h) \quad (8)$$

Equation 7 thus readily leads to *Euler's method*, also called the *forward Euler method*, the simplest of all quadrature algorithms, which for any n and $n+1$ and using eq 2 becomes

$$\frac{x_{n+1} - x_n}{h} + O(h) = f_n \quad (9)$$

and therefore

$$x_{n+1} = x_n + f_n h + O(h^2) \quad (10)$$

which gives us a method for calculating the next step of the trajectory $x(t)$ given x_n. On the one hand, this method has a very low *computational effort* and thus is very attractive for time-critical applications in audio synthesis. However, on the other hand it is neither very accurate (the step's error is just of second *order* i.e., $O(h^2)$) nor it is very *stable*.

The *numerical stability* of a method is established by applying the method to the numerical solution of a simple differential equation[2]:

$$\dot{x} = \lambda x \quad (11)$$

which has the analytical solution

$$x = e^{\lambda t} x_0 \quad (12)$$

with λ a complex number. $x_0 = 1$ usually. If $Re\{\lambda\} < 0$ the solution is analytically stable as all possible trajectories remain bounded as time tends to infinity (see figure 1).

Applying the forward Euler method to the numerical solution of equation 11 thus using equation 10 we get the following iterative rule:

$$\begin{aligned} x_1 &= x_0 + \lambda h x_0 = (1 + \lambda h) x_0 \\ x_2 &= x_1 + \lambda h x_1 = (1 + \lambda h) x_1 = (1 + \lambda h)^2 x_0 \\ x_3 &= (1 + \lambda h)^3 x_0 \\ &\vdots \\ x_n &= (1 + \lambda h)^n x_0 \end{aligned} \quad (13)$$

[2] Abbas I Abdel Karim. "Criterion for the stability of numerical integration methods for the solution of systems of differential equations". In: *J. Res. NBS* 718 (1967)

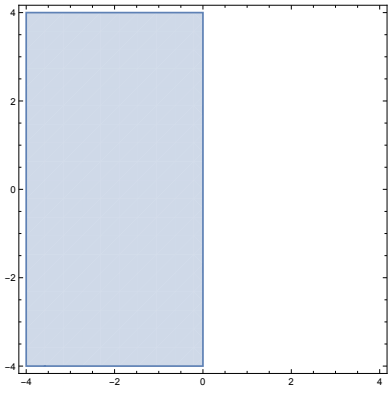

Figure 1: In blue the region in the complex plane of analytical stability of the solution of equation 11

Equation 13 describes a stable system for $n \to \infty$ if

$$|1 + \lambda h| < 1 \tag{14}$$

which is a disc of radius 1 in the complex plane of λh as depicted in figure 2. As we can see, the region of numerical stability of the method is very small and does not cover the whole region of stability the analytical solution has. That is, the forward Euler method does a poor job in approximating the analytical solution.

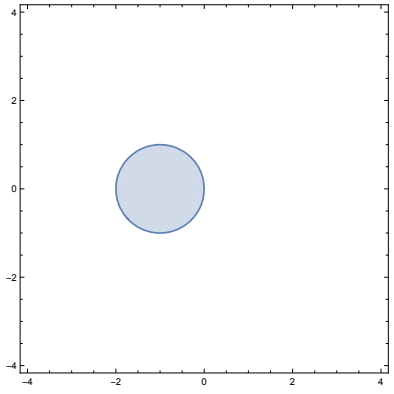

Figure 2: Region of numerical stability of the forward Euler method in the complex plane λh.

This can be demonstrated with an example. We can, for example, consider the equation 11 with $k = -2.3$ and $x_0 = 1$, which gives the stable analytical solution $x = e^{-2.3t}$. Applying the forward Euler method to this problem and choosing $h = 0.7$ we would be in the stability region as equation 13 indicates. As depicted in figure 3, after a short initial oscillating region, the method would be stable. Choosing instead $h = 1$ would mean being outside the stability region and would therefore be unstable. The method would produce oscillating solutions growing in amplitude, and is thus extremely sensitive to the right choice of the step h - which should be sufficiently small. This instability is particularly evident in oscillatory solutions to equation 11, i.e., when $Im\{k\} \neq 0$. These are of particular interest to us: in this case, even with a very small step size with respect to the frequency of the system, the method would always be unstable, and the energy of the system would grow exponentially.

Taking equation 8 instead would lead to to a different iterative method, known as *backward* or *implicit Euler*:

$$\frac{x_{n+1} - x_n}{h} + O(h) = f_{n+1} \tag{15}$$

and therefore

$$x_{n+1} = x_n + f_{n+1}h + O(h^2) \tag{16}$$

Even if this method seems very similar to the previous, it is exhibits substantial differences. The *numerical stability* analysis of this method, applying the previous process, would lead to:

$$\begin{aligned}
x_1 &= x_0 + \lambda h x_1 \Rightarrow x_1 = \frac{1}{(1+\lambda h)} x_0 \\
x_2 &= x_1 + \lambda h x_2 \Rightarrow x_2 = \frac{1}{(1+\lambda h)} x_1 = \frac{1}{(1+\lambda h)^2} x_0 \\
&\vdots \\
x_n &= \frac{1}{(1+\lambda h)^n} x_0
\end{aligned} \tag{17}$$

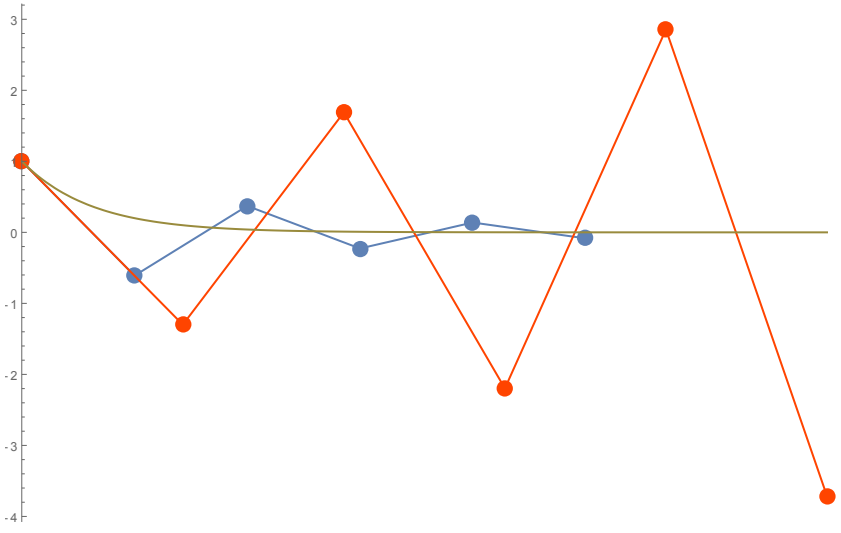

Figure 3: Plot of the solution to the differential equation $\dot{x} = -2.3x$: in green the exact solution $x = e^{-2.3t}$, in blue the solution computed with the forward Euler method and $h = 0.7$, in orange the solution computed with the forward Euler method and $h = 1$ which is unstable

which would be stable if

$$\frac{1}{|1+\lambda h|} < 1 \tag{18}$$

As shown in figure 4, the shape of the stability region of this method is very different to that in the former method. As can be seen, this method is good for approximating solutions for the stable region of the analytical solution. It produces a stable solution even where the analytical solution gives unstable (i.e. growing) solutions in the complex half plane $Re\{\lambda\} > 0$

Furthermore, the method, as with all other *implicit methods*, presents an ulterior difficulty. In fact, reformulating equation 16 taking into account that $f_n = f(x_n)$, we see

$$x_{n+1} = x_n + f(x_{n+1}) \tag{19}$$

that the term x_{n+1}, which we want to find, is on both sides of the equation: this is the fundamental characteristic of all implicit methods. As a consequence, one needs to solve an *algebraic equation* in the unknown x_{n+1}: this problem can be reformulated as to find the roots of the function $g(x_{n+1})$:

$$g(x_{n+1}) = x_{n+1} - x_n - f(x_{n+1}) = 0 \tag{20}$$

that is the points x_{n+1} for which this function is zero. This can in general be a very difficult problem to solve numerically as f could be any non-linear function. Usually this kind of problem is solved with iterative methods, such

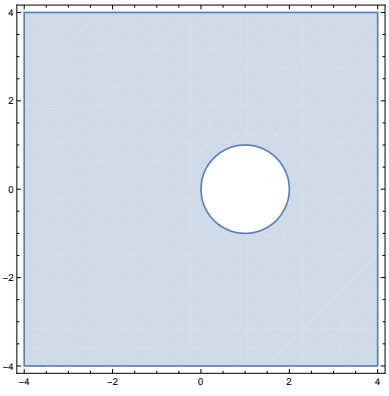

Figure 4: Region of numerical stability of the backward Euler method in the complex plane λh.

as the *Newton-Raphson method* which drastically increase the computational effort. However, neither the Euler methods nor an iterative method are suited for implementation in a software framework that needs to perform fast and stable (i.e., at audio rate) numerical integration.

Of course, the forward and backward Euler are the most simple and error prone numerical methods; still those methods show the basis on which all integration computational methods are constructed. The methods that tend to produce better results are constructed using two principal paths.

Linear multistep methods (also known as the {Adam-Bashford methods}) depart from a slightly different formulation, as in equation 4 to compute $x(t)$. From equation 2

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x(t))dt \tag{21}$$

that is, in discrete time steps:

$$x_{n+1} = x_n + \int_{n}^{n+1} f(t)dt \tag{22}$$

the derivation of these methods follows the idea to approximate better the value of the integral of the function f by taking into account its value at previous time steps and thus producing linear, quadratic, cubic, etc. polynomial approximations of f. This leads to a whole family of higher-order explicit or implicit methods. As an example, the explicit methods following the linear and cubic approximation of f would be respectively:

$$\begin{aligned} x_{n+1} &= x_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right) \\ x_{n+1} &= x_n + h\left(\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right) \end{aligned} \tag{23}$$

and the respective implicit methods would be:

$$\begin{aligned} x_{n+1} &= x_n + h\frac{1}{2}(f_n + f_{n-1}) \\ x_{n+1} &= x_n + h\left(\frac{5}{12}f_n + \frac{2}{3}f_{n-1} - \frac{1}{12}f_{n-2}\right) \end{aligned} \tag{24}$$

The *Runge-Kutta method* family comprises widely used numerical integration algorithms that use higher order expansions of the the Taylor series in equation 4 to better

approximate the integral of the function f. The so derived second order method algorithm would be:

$$\begin{aligned} k &= hf(x_n) \\ x_{n+1} &= x_n + hf(x_n + \frac{1}{2}k) \end{aligned} \tag{25}$$

and the widely used fourth order method:

$$\begin{aligned} k_1 &= hf(x_n) \\ k_2 &= hf(x_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + k_3) \\ x_{n+1} &= x_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{26}$$

These methods can be very accurate and exhibit better stability properties, but involve the computation of the value of the function f multiple times for each time step.

In *henri* a different kind of integration scheme is used: a *symplectic scheme*. This particular method can be used in the numerical integration of a special class of problems of the 1 equation, called *Hamiltonian systems*. These are systems of *coupled* differential equations and grounded on Newton's second law:

$$m\dot{v} = F(x) = -\frac{dU(x)}{dx} \tag{27}$$

$$\dot{x} = \frac{dv}{dt} \tag{28}$$

which describes how a mass under the influence of the force F, or a potential field U accelerates. To understand these methods, a small step backwards into theory is necessary.

Hamiltonian systems are dynamical systems which can be described with the Hamiltonian function, embodying Newton's second law of mechanics.[3] These are of utmost interest in physics: they are used to describe most systems found in nature from planetary system to the motion of an electron in an electromagnetic field. These equations depend on the characteristics of the Hamiltonian function H, related to position, velocity of the involved elements (masses), and time. The special interest in physics for this function derives from the fact that the Hamiltonian is for these systems the sum of the kinetic and potential energies T and U:[4]

$$H(q,p,t) = T(p) + U(q) \tag{29}$$

For instance, the Hamiltonian of the simple harmonic oscillator would be:

$$H = \frac{p^2}{2m} + \frac{1}{2}kx^2 \tag{30}$$

Thus, usually the Hamiltonian is the energy of the formulated system and for *closed systems*, given the conservation of energy, it is constant and time independent:

$$\frac{\partial H}{\partial t} = 0 \tag{31}$$

A principal characteristic of this function is that it describes the *evolution* of the state of the dynamical system, i.e., it describes how the coordinates q and p evolve in time via the so called *Hamilton equations*, a system of differential equations of the general form of equation 2:

$$\begin{aligned} \dot{p} &= -\frac{\partial H}{\partial q} \\ \dot{q} &= \frac{\partial H}{\partial p} \end{aligned} \tag{32}$$

Considering the space spanned by the coordinates (q,p), the *phase-space*, the integration of the former equations results in a so-called *flow* in this space. To any (continuous and differentiable) Hamiltonian corresponds a *flow* ϕ_t, which describes the time evolution of the system. Given any initial coordinate in the phase-space (q_0,p_0), this returns to the point (q,p) to which the system would evolve at any time t:

$$\phi_t : (q_0,p_0) \to (q(t),p(t)) \tag{33}$$

An important characteristic of this function is that, for Hamiltonian systems, it is a so-called *symplectic map* meaning that it is area preserving in the phase space. In other

[3] Herbert Goldstein, Charles Poole, and John Safko. *Classical mechanics*. Addison Wesley, 2002

[4] For the sake clarity and conciseness, I'm following a simplified mathematical treatment in this section trying to bring across the most important concepts qualitatively. Further I will use, as in most texts, the *generalised coordinates* notation for position and momenta, q and p respectively. Therefore in the next equation, I assume separable Hamiltonians (the potential U is not dependent of the momentum q).

words, given a section of the phase space, transforming this section with a symplectic map would translate it to different section in the phase-space, which could be different in form, but would have the same area (see figure 5).[5]

This quality of the Hamiltonian systems, which are the systems we are mostly dealing with in *rattle*, is essentially characterising this set of problems and is ultimately related to fundamental principles of physics as *Liouvilles' theorem* and the principle of energy conservation.

It seems therefore obvious to ask that the *symplecticity* property of the exact solutions of Hamiltonian systems should also be embodied and respected by the numerical integration methods.[6] That is, any numerical method Φ_h approximating the flow of the exact solution such that

$$(q_{n+1}, p_{n+1}) = \Phi_h(q_n, p_n) \tag{34}$$

given any point (q_n, p_n), should be a symplectic transformation. None of the methods described above, whether explicit and implicit, are symplectic independently from the order they could reach. Nor, could any of the above methods be guaranteed to respect fundamental characteristics of dynamical systems as the conservation of energy. This can be depicted on the basis of the Euler methods we introduced above. Recalling the explicit Euler method, we know it would tend to expand the energy of the system (solutions grow in energy) and the section of the phase space would grow in area. The implicit Euler method, meanwhile, would tend to reduce it (solutions would tend towards stability even if analytically they would not). This behaviour is depicted graphically in figure 6, considering the example phase space flow generated by the Hamiltonian system of the simple pendulum.[7]

The symplecticity request leads to the formulation of a new family of *symplectic methods* which guarantee conservation of energy and area when applied to the integration of a dynamical system. The first of these methods is the *symplectic Euler* method, which can be equivalently expressed in two ways:[8]

$$\begin{aligned} p_{n+1} &= p_n - h\frac{\partial H(q_n, p_{n+1})}{\partial q} \\ q_{n+1} &= q_n + h\frac{\partial H(q_n, p_{n+1})}{\partial p} \end{aligned} \tag{35}$$

[5] This descends from a 1899 Theorem by Poincarê, published in *Les Methodes Nouvelles de la Mecanique Celeste*.

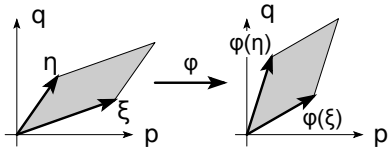

Figure 5: Simplecticity (area preservation) of the mapping Φ_t

[6]

[7] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Vol. 31. Springer Science & Business Media, 2006

[8] Rene de Vogelaere. "Methods of integration which preserve the contact transformation property of the Hamiltonian equations". In: *Department of Mathematics, University of Notre Dame, Report* 4 (1956), p. 30

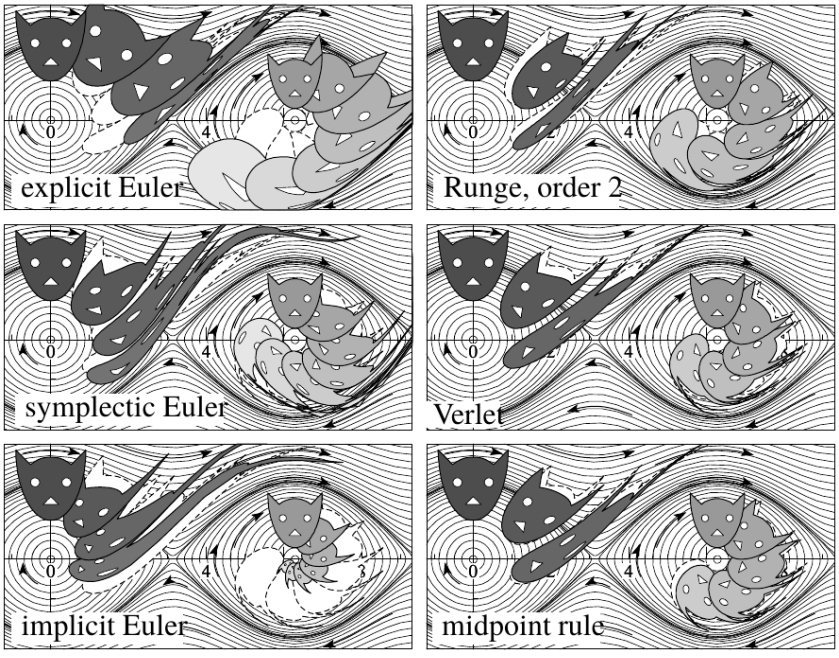

Figure 6: Area preservation behaviour of various numerical integration methods on the basis of the a phase space of the simple pendulum. Same initial areas (and values) are chosen

or

$$\begin{aligned} p_{n+1} &= p_n - h\frac{\partial H(q_{n+1}, p_n)}{\partial q} \\ q_{n+1} &= q_n + h\frac{\partial H(q_{n+1}, p_n)}{\partial p} \end{aligned} \tag{36}$$

which, recalling that:

$$-\frac{\partial H(q,p)}{\partial q} = -\frac{\partial U(q)}{\partial q} = f(q) \tag{37}$$

where $f(q)$ is the force acting on the mass and

$$\frac{\partial H(q,p)}{\partial p} = \frac{p}{m} = v \tag{38}$$

the former reduce to

$$\begin{aligned} p_{n+1} &= p_n + hf(q_n) \\ q_{n+1} &= q_n + h\frac{p_{n+1}}{m} \end{aligned} \tag{39}$$

and the equivalent:

$$\begin{aligned} q_{n+1} &= q_n + h\frac{p_n}{m} \\ p_{n+1} &= p_n + hf(q_{n+1}) \end{aligned} \tag{40}$$

That is, each of these methods uses an implicit method for the evolution of one state variable and the explicit

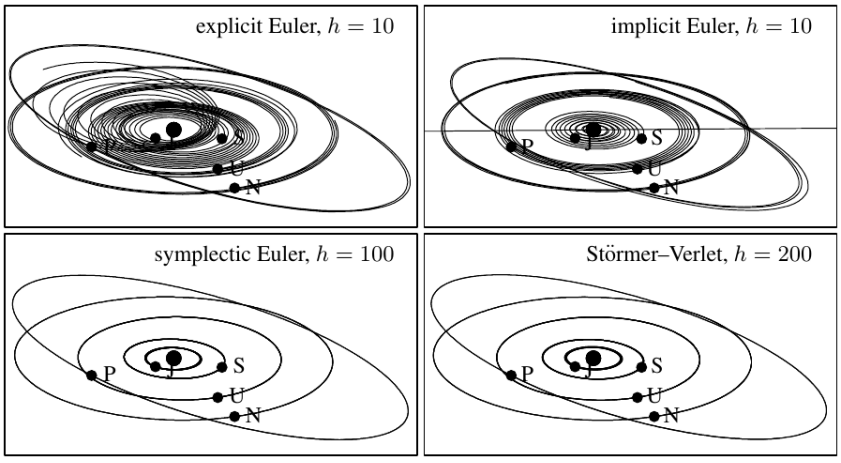

Figure 7: Solution to the outer solar system as computed with the explicit, implicit, and symplectic Euler and Strömer-Verlet methods. The graphic is taken from the book by E. Hairer: *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*

method for the other. The performance of these two methods, even if only of first order, is already clearly more stable as exemplified in in figure 7.

One of the most far-reaching consequences of the *symplecticity* of Hamiltonian systems is that a geometrical way of thinking about the numerical integration of such systems' evolution is made possible. In fact, these integration methods are usually also referred to as *geometric integrators*.

This geometric perspective is the basis of further development of those methods, given the following observations:

- *Composition*: Numerical methods can be composed in the same way functions can be composed. That is if Φ_h and Ψ_h are two different numerical methods of order r and s respectively for the same problem, their composition $\Phi_{\frac{h}{2}} \circ \Psi_{\frac{h}{2}}$ is also a method X_h for the same problem with order $r + s$.

- *Symmetry*: The exact flow of a dynamical system ϕ_t usually satisfies the relation $\phi_t^1 = \phi_t$: This property is in general not satisfied by the flow Φ_h of a numerical method. The *adjoint method* Φ_h^* is defined as equal to the inverse method with reversed time.

$$\Phi_h^* = \Phi_{-h}^{-1} \tag{41}$$

  and a method is called *symmetric* if is is equal to its adjoint $\Phi_h^* = \Phi_h$. Further, the adjoint of an adjoint method is the original method $(\Phi_h^*)^* = \Phi_h$ and the adjoint of a composition, if the composition of the single adjoint methods is reversed in order $(\Phi_h \circ \Psi_h)^* = \Psi_h^* \circ \Phi_h^*$. *Symmetry*

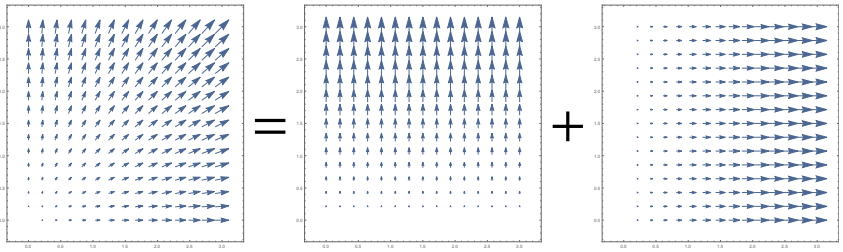

Figure 8: The splitting of a flow in two-dimensional phase space is expressed as the sum of two more simple flows

is an important quality of flows which is related to the reversibility of dynamical systems, a fundamental characteristic of all conservative systems and is therefore a quality that a numerical method should provide.

- *Splitting*: A flow in phase space, i.e., a vector field, can be split into the sum of two (or more) simple flows along one of the dimensions of the phase space. The total flow is then the composition of the two flows (see figure 8). For instance, the first symplectic Euler method Φ_h formulated in equation 40 could be split into two flows $\phi_h^{[1]}$ and $\phi_h^{[2]}$ respectively along the p and q dimensions:

$$\begin{aligned} & \phi_h^{[1]} \\ q_{n+1} &= q_n \\ p_{n+1} &= p_n + hf(q_n) \\ & \phi_h^{[2]} \\ q_{n+1} &= q_n + \frac{h}{m} p_n \\ p_{n+1} &= p_n \end{aligned}$$

so that

$$\Phi_h = \phi_h^{[1]} \circ \phi_h^{[2]} \tag{42}$$

Combining principles of *composition*, *symmetry*, and *splitting*, a general rule for the generation of symmetric symplectic methods of high order can be formulated.[9] As an example, we can see the Euler method in equation 40, split into two flows and compose with its adjoint, and simplified. We obtain:

$$\begin{aligned} \Phi_{\frac{h}{2}}^* \circ \Phi_{\frac{h}{2}} &= (\phi_{\frac{h}{2}}^{[1]} \circ \phi_{\frac{h}{2}}^{[2]})^* \circ (\phi_{\frac{h}{2}}^{[1]} \circ \phi_{\frac{h}{2}}^{[2]}) \\ &= \phi_{\frac{h}{2}}^{[2]} \circ \phi_{\frac{h}{2}}^{[1]} \circ \phi_{\frac{h}{2}}^{[1]} \circ \phi_{\frac{h}{2}}^{[2]} \\ &= \phi_{\frac{h}{2}}^{[2]} \circ \phi_{h}^{[1]} \circ \phi_{\frac{h}{2}}^{[2]} \end{aligned} \tag{43}$$

[9] Gilbert Strang. "On the construction and comparison of difference schemes". In: *SIAM Journal on Numerical Analysis* 5.3 (1968), pp. 506-517; Robert I McLachlan and G Reinout W Quispel. "Splitting methods". In: *Acta Numerica* 11 (2002), pp. 341-434

This is a symmetric method of the second order. The above equation may also be rewritten as:

$$\begin{aligned} q_{n+\frac{1}{2}} &= q_n + \frac{h}{2m} p_n \\ p_{n+1} &= p_n + h f(q_{n+\frac{1}{2}}) \\ q_{n+1} &= q_{n+\frac{1}{2}} + \frac{h}{2m} p_{n+1} \end{aligned} \tag{44}$$

This is also known as the *Strömer-Verlet method*.

By reapplying *composition* and *splitting* to the above equation 43, we can deduce higher-order symmetric integration schemes. Furthermore, these methods can be generalised and be applied to multi-dimensional dynamical systems where the flow of the system can be reformulated as a composition of simple flows along each dimension. For instance, a n dimensional dynamical system governed by the flow Φ_h:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n) \\ \dot{x}_2 &= f_2(x_1, x_2, \ldots, x_n) \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \ldots, x_n) \end{aligned}$$

can be reformulated as a splitting into n first-order flows

$$\Phi_h = \phi_h^1 \circ \phi_h^2 \circ \cdots \circ \phi_h^n$$

and therefore, using the adjoint, a second-order symmetric method would be:

$$\Phi_{\frac{h}{2}}^* \circ \Phi_{\frac{h}{2}} = \phi_{\frac{h}{2}}^n \circ \cdots \circ \phi_{\frac{h}{2}}^2 \circ \phi_h^1 \circ \phi_{\frac{h}{2}}^2 \circ \cdots \circ \phi_{\frac{h}{2}}^n \tag{45}$$

This is the integration method I used in the second formulation of *rattle* for integrating arbitrary multi-dimensional dynamical systems. To formulate a fourth-order symmetric and symplectic integration method of the above, one would simply use composition and write the method:

$$\Phi_{\frac{h}{4}}^* \circ \Phi_{\frac{h}{4}} \circ \Phi_{\frac{h}{4}}^* \circ \Phi_{\frac{h}{4}} \tag{46}$$

and so on for higher orders.