Rough notes about the exponential of an SVF

Some notes on the numerical computation of the exponential of a stationary velocity field.

UCL, TIG, for internal use, November 24, 2015, s.ferraris@ucl.ac.uk

Contents

1 Definition of SVF

Any vector-valued function from a subset Ω of $\mathbb{R}^D$ on $\mathbb{R}^D$ is called here *vector field*. For each $\mathbf{x} = (x_1, ..., x_D) \in \Omega$ a vector field $\mathbf{v}$ can be expressed in components as

$$\mathbf{v}(\mathbf{x}) = (v_1(\mathbf{x}), ..., v_D(\mathbf{x}) = \sum_{d=1}^{D} v_d(\mathbf{x})\mathbf{e}_d = \sum_{d=1}^{D} v_d(\mathbf{x}) \frac{d}{dx_d} \Big|_{\mathbf{x}} ,$$

(where the last choice is preferable when we intend to underline the action over the set $\mathbf{C}^\infty(\Omega)$ as directional derivative, see for example [10], [14]). Each component v_j is a function defined from Ω to $\mathbb{R}$ and its features characterise the vector field: if each of them is differentiable (or smooth) then $\mathbf{v}$ is a *smooth vector field*.
A *diffeomorphism* over an open compact subset Ω of $\mathbf{R}^D$ is a continuous and invertible function from Ω to itself that is also differentiable with differentiable inverse. The set of diffeomorphisms forms a group with the operation of composition, indicated with $\mathrm{Diff}(\Omega)$.
A *one-parameter subgroup of diffeomorphisms* of Ω, is a map

$$\begin{aligned} g : \mathbb{R} \times \Omega &\longrightarrow \Omega \\ (t, \mathbf{x}) &\longmapsto g(t, \mathbf{x}) = g^t \mathbf{x} \end{aligned}$$

that satisfies the following conditions:

1. g is differentiable for both of the variables.
2. $g^t : \Omega \to \Omega$, called flow of Ω is a diffeomorphism for each $t \in \mathbb{R}$.
3. The set $\{g^t \mid t \in \mathbb{R}\}$ form a commutative group with the operation of composition inherited by the action of the group $(\mathbb{R}, +)$ over Ω, and therefore satisfies $g^t \circ g^s = g^{t+s}$ and $g^0 = \mathrm{Id}$.

The variable t is called time-parameter or simply time. Some one parameter subgroup may have a time parameter defined only on a compact subset of $\mathbb{R}$. To simplify the notation we will consider the general case, since required restriction of the particular case are straightforward and not restrictive.

Given a one parameter subgroup g, it defines uniquely a diffeomorphism of Ω, when its time parameter has fixed value. If we chose this fixed value to 1 and we denote this diffeomorphism with ϕ_1, it follows $g^0 = \mathrm{Id}$ and $g^1 = \phi_1$ (see [1]), therefore the action of $t \in \mathbb{R}$ over Ω defined by g can be see as a parametrisation of a collection of diffeomorphisms that ranges continuously from Id to ϕ_1.
If we equate a smooth vector field with the infinitesimal variation $\delta = \frac{d}{dt}\big|_{t=0}$ of a one parameter subgroup of diffeomorphism g, we obtain a system of stationary ordinary differential equations where the unknown is the one parameter subgroup g:

$$\frac{d}{dt}\Big|_{t=0} g^t \mathbf{x} = \mathbf{v}(\mathbf{x}) \ . \tag{1}$$

That is equivalent to the system

$$\begin{cases} \delta g^t x_1 = v_1(x_1, x_2, \ldots, x_D) \\ \delta g^t x_2 = v_1(x_1, x_2, \ldots, x_D) \\ \vdots \\ \delta g^t x_D = v_1(x_1, x_2, \ldots, x_D) \ . \end{cases} \tag{2}$$

Using the Newton notation for the derivative and following the vectorised notation for ordinary differential equations (introduced by Peano, 1890 [15], pag 203) the previous system can be written as $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x})$.

If the infinitesimal variation is considered away from the origin of the real line, it can be proved that

$$\frac{d}{dt}\Big|_{t=\tau} g^t\mathbf{x} = \mathbf{v}(g^\tau\mathbf{x}) \ ; \tag{3}$$

for the particular case $\tau = 1$ and $\phi_1 = g^1$, the last equation become

$$\frac{d}{dt}\Big|_{t=1} g^t\mathbf{x} = \mathbf{v}(\phi_1(\mathbf{x})) \ .$$

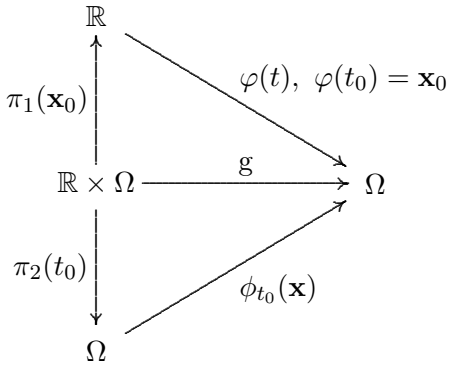
Given a one parameter subgroup g is it possible to fix a point of Ω and consider g only as a function of the time parameter. On the other side, fixing the time parameter, g can be considered as a function of the points in Ω. In the first case, for a fixed point $\mathbf{x}_0$, we have

$$\begin{aligned} g\mathbf{x}_0 : \mathbb{R} &\longrightarrow \Omega \\ t &\longmapsto g^t\mathbf{x}_0 =: \varphi(t) \ , \end{aligned}$$

where the function φ is called *integral curve* of g passing through $\mathbf{x}_0$. It represents the trajectory of the chosen point $\mathbf{x}_0$ under the forces defined by $\mathbf{v}$ as its tangent vector, at any moment in time. In the second case, for a fixed t_0, we have

$$\begin{aligned} g^{t_0} : \Omega &\longrightarrow \Omega \\ \mathbf{x} &\longmapsto g^{t_0}\mathbf{x} =: \phi_{t_0}(\mathbf{x}) \ . \end{aligned}$$

where ϕ_{t_0} is called *flow* of g at the time t_0. It represents the position of any of the points $\mathbf{x} \in \Omega$ after the fixed time t_0 has gone, starting at 0.
In the following commutative diagram, the function π_i is the projection over the i-th factor of the Cartesian product:



Using the definition of one-parameter subgroup, it is easy to prove that for a given t_0, the flow ϕ is one-to-one over Ω and that the integral curves of g, passing through the points $\mathbf{x}_0$ and $\mathbf{x}_1$ are coincident or disjoint.
If the integral curve passing through $\mathbf{x}_0$ is the unknown of the problem defined by a vector field $\mathbf{v}$, then the problem (3) can be conveniently reformulated as a *Cauchy problem*:

$$\begin{cases} \frac{d\varphi(t)}{dt} = \mathbf{v}(\varphi(t)) \\ \varphi(t_0) = \mathbf{x}_0 \ , \end{cases} \tag{4}$$

where the initial condition $\varphi(t_0) = \mathbf{x}_0$ specifies the through point of the integral curve and the value of the time parameter at that point of the curve.
If the unknown of the problem is the flow of g at a given time t_0, then the problem (3) can be conveniently reformulated as a *flow problem*

$$\frac{d\phi_{t_0}}{dt} = \mathbf{v}(\phi_{t_0}) \ , \tag{5}$$

where the dependence of ϕ over the time is defined via the underpinning one-parameter subgroup g ($\frac{d}{dt}\big|_{t=t_0} g^t(\mathbf{x}) = \frac{d\phi_{t_0}(\mathbf{x})}{dt}$), and the time step t_0, that represents the initial condition for the flow problem, is indicated in subscript.

Example The voltage-controlled oscillator neuron model (VCON) is a non-linear 2D ODE that is used here as example to show the difference between a Cauchy problem and a Flow problem:

$$\begin{cases} \frac{dx(t)}{dt} = \alpha(y + \tau_x) \\ \frac{dy(t)}{dt} = \alpha(-\sigma y + I + w\cos(x) + \tau_y) \end{cases}$$

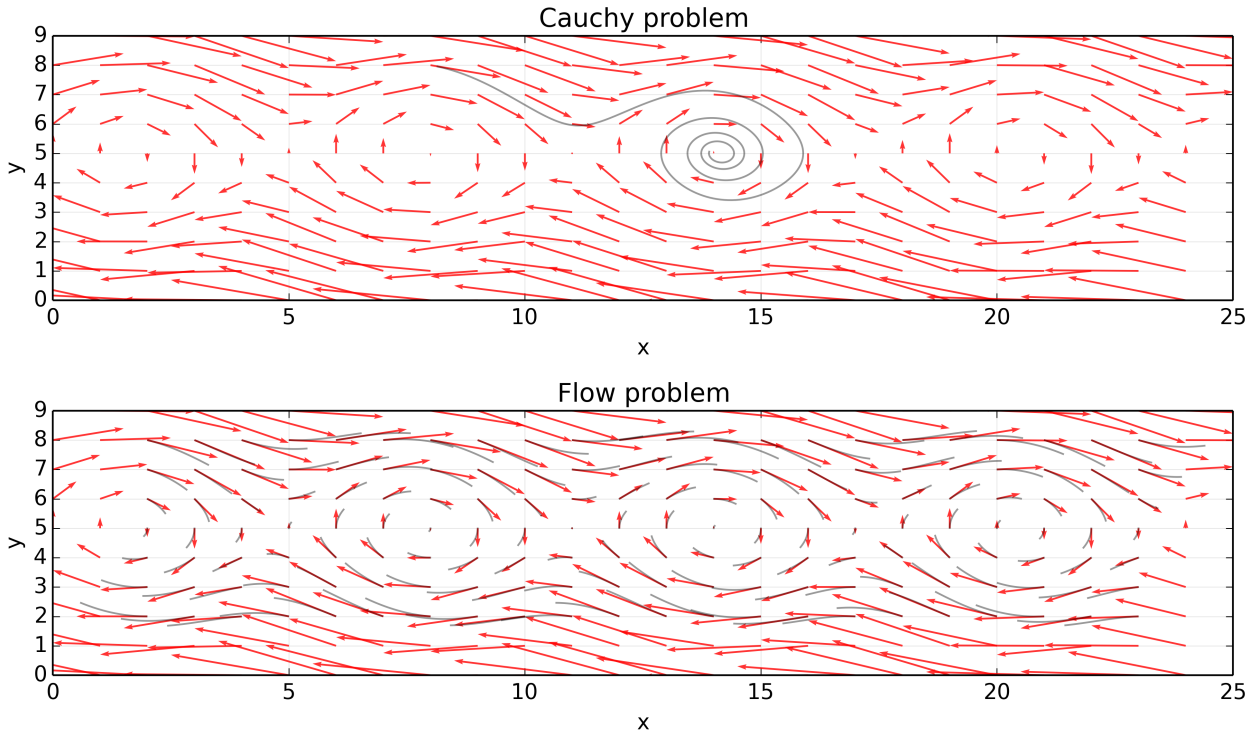


Figure 1: Integration of a voltage-controlled oscillator neuron model (VCON) of parameters $\sigma = 0.2$, $I = 0.99$, $w = 1.4$, $\alpha = 0.5$, and $\tau = (\tau_x, \tau_y) = (-5, 0)$. Above the numerical solution of a Caucy problem with initial condition $\phi(0) = (8, 8)$. Below the numerical solution of a Flow problem with initial condition $t_0 = 1$. Integral curve and flow is in gray for both cases, having the same direction of the tangent arrows.

The vector field is given by $\mathbf{v}(x, y) = (\alpha(y + t_1), \alpha(-\sigma y + I + w\cos(x) + \tau_2))$. If interested in the integral curve $\varphi(t)$, such that $\dot{\phi}(t) = \mathbf{v}(\varphi(t))$ passing through the point $\mathbf{x}_0 = (8, 8)$ at time $t_0 = 0$, then

we can reformulate it as a Cauchy problem:

$$\begin{cases} \frac{d\varphi(t)_x}{dt} = \alpha(\varphi(t)_y + \tau_x) \\ \frac{d\varphi(t)_y}{dt} = \alpha(-\sigma\varphi(t)_y + I + w\cos(\varphi(t)_x) + \tau_y) \\ \varphi(0) = (8,8)\ . \end{cases} \tag{6}$$

If interested in computing the diffeomorphism that the flow induces on the space after 1 unit of time has elapsed, then the problem can be reformulated as a flow problem:

$$\begin{cases} \frac{d\phi_{1,x}}{dt} = \alpha(\phi_{1,y} + \tau_x) \\ \frac{d\phi_{1,y}}{dt} = \alpha(-\sigma\phi_y + I + w\cos(\phi_{1,x}) + \tau_y)\ . \end{cases} \tag{7}$$

Numerical solutions can be visualized in figure 1.

We observe that Cauchy problem and the integral problem are two times the same problem (3), when different input data are provided. In fact, solving the flow problem at a given point of Ω for any initial condition t_0 is equivalent to solve the Cauchy problem. Vice versa, solving the Cauchy problem at a fixed time t_0, for any point in $\mathbf{x}$ in Ω is equivalent to solve the flow problem. This also shows the importance of the abstract concept of one parameter subgroup, that generalise both the Cauchy and the Flow problem for any initial condition, both temporal and spatial.
It is worth to notice that both equation 4 and 5 have an equivalent integral formulation:

$$\varphi(t) - \mathbf{x}_0 = \int_{t_0}^{t} \mathbf{v}(\varphi(\tau))d\tau \qquad\qquad \phi_t(\mathbf{x}) - \phi_{t_0}(\mathbf{x}) = \int_{t_0}^{t} \mathbf{v}(\phi_\tau(\mathbf{x}))d\tau \tag{8}$$

Finally, the vector field $\mathbf{v}$ as appears in the equation 3 is called *stationary velocity field* (SVF) and it defines the Cauchy problem (4) - when the aim is to find the integral curve for a given point - and the flow problem (5) - when the aim is to find the position of all of the points in Ω after a given time t_0. Moreover we observe that the solution to a Cauchy problem is a curve, while the solution to a flow problem is a diffeomorphisms over Ω.

1.1 Representing diffeomorphisms as vector field, Eulerian and Lagrangian coordinates

Let g be a one-parameter subgroup of diffeomorphisms over Ω, compact subset of $\mathbb{R}^D$. Its flow at time 1 is given by $\phi_1 = g^1$ and it can be represented as a vector field. In fact it become a function that at each point $\mathbf{x}$ in the domain Ω associates a unique new vector $\phi_1(\mathbf{x})$. This field defined by ϕ_1 can be represented in two reference frame: if the motion of a point $\mathbf{x}$ is considered respect to the absolute coordinate frame then its coordinates are called *Eulerian*. If the motion is considered respect to the point $\mathbf{x}$ itself then the coordinates are called *Lagrangian* (see figure 2). Passing from the Eulerian coordinates to the Lagrangian coordinates requires the subtraction of the identity for each point:

$$\hat{\phi}_1(\mathbf{x}) = \phi_1(\mathbf{x}) - \mathbf{x}\ .$$

The symbol $\hat{}$ will be utilised to avoid ambiguity between the two coordinate systems.
For computational reasons, it is preferable to take advantage of the Lagrangian coordinates when it is possible. In this case shorter vector are stored at every point of the discretised domain and therefore it is less prone to numerical error.

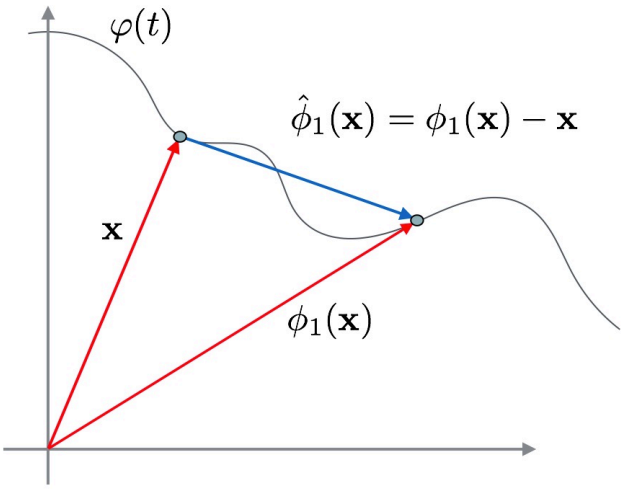


Figure 2: Comparison between Eulerian and Lagrangian coordinate system for the flow ϕ_1. The gray line represents the integral curve through $\mathbf{x}$ of the one-parameter subgroup. The red arrow reprensents the motion of the point $\mathbf{x}$, at time 0 and at time 1 in Eulerian coordinates. The blue arrow represents the same motion in Lagrangian coordinates.

1.2 When SVF depends on time: TVVF

If the components v_i depends also on the time parameter $t \in T$, then $\mathbf{v}$ is called here time varying vector field (TVVF) as well as non-stationary or non-autonomous. A TVVF can be written as

$$\mathbf{v}(t, x) = \sum_{i=1}^{d} v_i(t, x) \frac{\partial}{\partial x_i}$$

where v_i belongs to $\mathcal{C}^\infty(T \times \Omega)$. Also TVVF defines Cauchy problem and flow problem (non-stationary or non-autonomous). They can be rewritten as

$$\begin{cases} \frac{d\varphi(t)}{dt} = \mathbf{v}(t, \varphi(t)) \\ \varphi(t_0) = \mathbf{x}_0 \ , \end{cases} \qquad \frac{d\phi_{t_0}(\mathbf{x})}{dt} = \mathbf{v}(t, \phi_{t_0}(\mathbf{x})) \ , \tag{9}$$

respectively. Their integral formulation is given by:

$$\varphi(t) - \mathbf{x}_0 = \int_{t_0}^{t} \mathbf{v}(\tau, \varphi(\tau)) d\tau \qquad \phi_t(\mathbf{x}) - \phi_{t_0}(\mathbf{x}) = \int_{t_0}^{t} \mathbf{v}(\tau, \phi_\tau(\mathbf{x})) d\tau \tag{10}$$

1.3 Discretisation step: some concepts and notations

Digital images are discretised data-set representing fixed-time sections of the continuous reality (see rough notes on image registration). This definition alone is enough to justify the use of a continuous model to manipulate their transformations, therefore the passage from continuous to discrete does not happen only

in the phase of image acquisition, but it is crucial in the application of the continuous models developed. A vector field $\mathbf{v}$ defined over Ω can be discretised defining a D-dimensional (rectangular) grid within its domain (see meshpoint model or midpoint model [13]) indicated with $\Delta\Omega$. The discretisation of $\mathbf{v}$ is the restriction of $\mathbf{v}$ at the chosen discretised domain. The data structure to store discretised vector field is the standard 5-dimensional matrix (see nibabel) where the fourth value that represents the time is always zero. The operation of composition between continuous vector fields is performed in the discrete case with a resampling, and it is based on the chosen algorithm for the interpolation of the input discrete function (can be *nearest-neighborhood*, *linear* or *cubic*).

1.4 2D linear stationary ODE

It is possible to describe the behaviour of a system of ODE when the SVF that defines the problem is linear. The model is given by:

$$\begin{cases} \delta g^t x = ax + by + t_x \\ \delta g^t y = cx + dy + t_y \end{cases} \tag{11}$$

and it is uniquely defined by the matrix in homogeneous coordinates

$$A = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the matrix $R = [[a, b], [c, d]]$ is the *rotational part*, and the vector $T = [t_x, t_y]^T$ is the translational part. The matrix A has only one fixed point, given by the solution of $A\mathbf{x}^T = \mathbf{x}^T$, and the behaviour of the vector field around this point is provided by the values of the eigenvalue. The characteristic polynomial equation of R is given by

$$\lambda^2 - \mathrm{tr}(R)\lambda + \det(R) = 0$$

and according to its solutions, the eigenvalues λ_1 and λ_2, we can have 4 different vector field configurations (see [6] and [8]):

$$\lambda_{1,2} = \frac{\mathrm{tr}(A)}{2} \pm \sqrt{\left(\frac{\mathrm{tr}(A)}{2}\right)^2 - \det(A)} \ .$$

1. λ_1 and λ_2 are both real and have the same sign. If they are both positive, the fixed point is a parabolic source (integral curves goes from the critical point toward the external), otherwise a parabolic sink (integral curves goes to the critical point).

2. λ_1 and λ_2 are both real and have opposite signs, than the fixed point is a saddle.

3. λ_1 and λ_2 are complex conjugates, then the fixed point can be spiral (real part different from zero) or a center (real part equals to zero) - these are the SVF generated by an element of $\mathrm{SE}(D)$.

1.5 Exponential map for the Linear Case

Given a non-stationary linear Cauchy problem

$$\begin{cases} \dot{\varphi}(t) = \mathbf{v}(t, \varphi(t)) = A\varphi(t) + \mathbf{f}(t) \\ \varphi(t_0) = \mathbf{x}_0 \ , \end{cases} \tag{12}$$

where A is a time-independent matrix of constant and $\mathbf{f}(t)$ contain the terms directly dependent on time. The solution is given by

$$\begin{aligned} \mathbf{x}(t) &= \exp(A(t-t_0))\mathbf{x}_0 + \int_{t_0}^{t} \exp(A(t-\tau))\mathbf{f}(\tau)d\tau \\ &= \exp(A(t-t_0))\mathbf{x}_0 + \exp(At)\int_{t_0}^{t} \exp(-A\tau)\mathbf{f}(\tau)d\tau \ . \end{aligned}$$

When dealing with SVF the $\mathbf{f}(t)$ is zero, therefore the solution to the linear Cauchy problem is given by:

$$\varphi(t) = \exp(A(t-t_0))\mathbf{x}_0 \ .$$

Where the exponential of a matrix is defined as:

$$\exp(At) = \sum_{n=0}^{\infty} \frac{(At)^n}{n!} \ .$$

From Cayley-Hamilton theorem (any square matrix satisfies its own characteristic equation) it follows that the exponential of a matrix can be expressed as a polynomial of degree $D-1$ where D is the dimension of the matrix:

$$\exp(At) = \sum_{n=0}^{D-1} \alpha_n A^n \ .$$

1.6 2D Random generated SVF

At the moment, in the following experiments the SVF are generated in two ways:

1. *SE*(2)*-generated SVF*: field generated using an element of SE(2): the ground truth is know and the integral curves are as simple as circles.
2. *Gaussian generated SVF*: generated using a Gaussian filter over a random vector field. There is no ground truth in this case.

2 Involving Lie group theory: the log-Euclidean framework

The need for algebraic manipulation of stationary velocity fields and the diffeomorphisms generated by them as their one parameter subgroup at the time-point 1, introduces two algebraic structures where these elements live in.

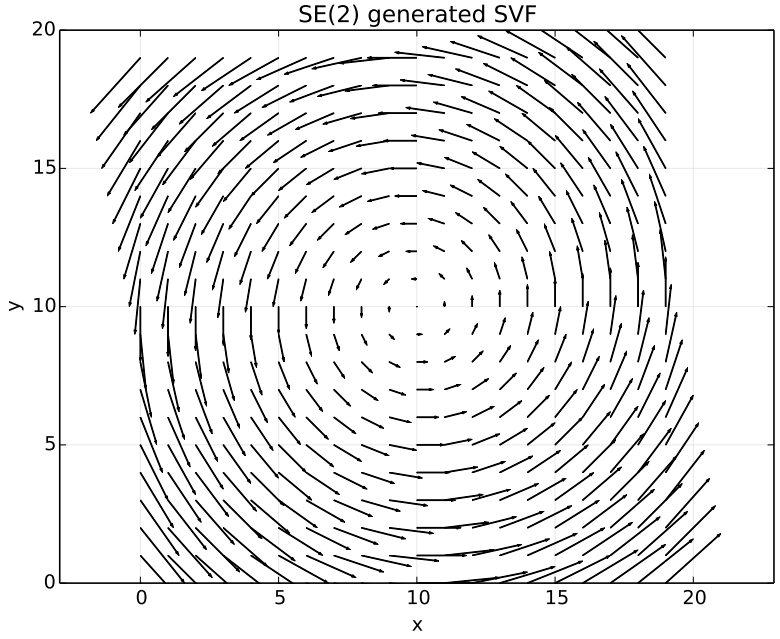


Figure 3: Example of an SE(2)-generated SVF.

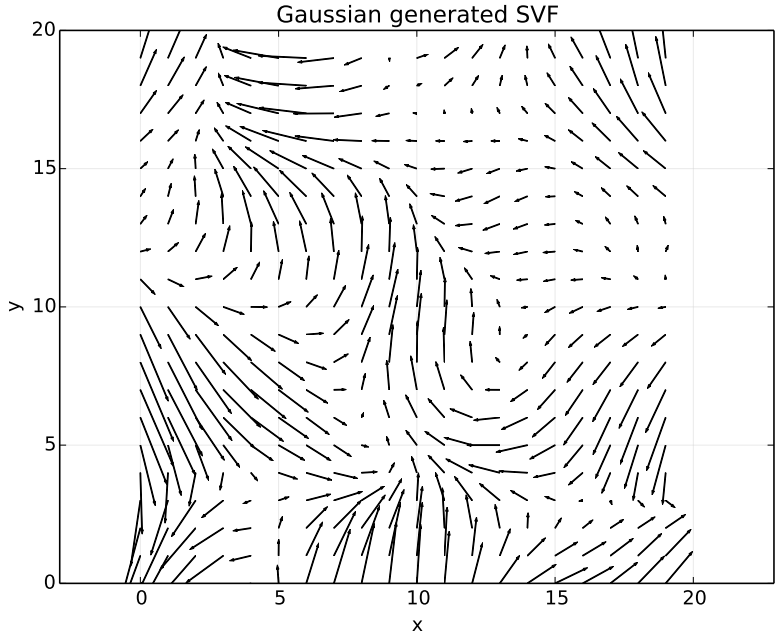


Figure 4: Example of a Gaussian generated SVF.

2.1 Algebraic structures for SVF and diffeomorphisms

The set of SVF defines a vector space, indicated with $\mathrm{Vect}(\Omega)$ with the canonical operations of addition and scalar multiplication. The set of diffeomorphisms over Ω, indicated with $\mathrm{Diff}(\Omega)$ defines a group with the operation of composition.

It has been proved that there is at least one diffeomorphism in $\mathrm{Diff}(\Omega)$, as close as we like to the identity, that can not be expressed as a one parameter subgroup, for any possible value of the time parameter (see[12]). Trusting the fact that useful diffeomorphisms to model image deformations are the one embedded in the one parameter subgroup, we restrict the group of diffeomorphism $\mathrm{Diff}(\Omega)$ to these elements, indicating them with $\mathrm{Diff}_g(\Omega)$.

With this restriction is is possible to consider a correspondence between an SVF $\mathbf{v} \in \mathrm{Vect}(\Omega)$ and a diffeomorphisms $g^1 = p \in \mathrm{Diff}_g(\Omega)$ that maps $\mathbf{v}$ in p. This is called Lie exponential, and it is defined as

$$\begin{aligned} \exp : \mathrm{Vect}(\Omega) &\longrightarrow \mathrm{Diff}_g(\Omega) \\ \mathbf{v} &\longmapsto \exp(\mathbf{v}) = g^1 \end{aligned}$$

where g^1 is the one parameter subgroup that is generated by $\mathbf{v}$ at the time 1, or the solution of the flow problem (5).

The relationship became more interesting if we consider that the tangent space of $\mathrm{Diff}(\Omega)$ is the infinite dimensional vector space of differentiable vector field over Ω, the previously defined $\mathrm{Vect}(\Omega)$ [9]. Therefore the exponential map is a map from the linear tangent space to the space of diffeomorphisms (only the one embedded in a one parameter subgroup) and the vector space of SVF is a linearization of these diffeomorphisms.
Another results from Lie group theory ensure the local biiectivity between $\mathrm{Diff}_g(\Omega)$ and $\mathrm{Vect}(\Omega)$ around the origin and the identity element.

The biiection is given by the Lie exponential and by its local inverse, the Lie logarithm.
The flow problem, given in equation (5), with the initial condition $t_0 = 1$ can be reformulated as

$$p = \exp(\mathbf{v}) \ , \tag{13}$$

where exp is the Lie exponential, while the inverse problem (when defined!) can be reformulated as

$$\mathbf{v} = \log(p) \ . \tag{14}$$

The diffeomorphisms p that arise form the previous equations deserve two ad hoc names: it will be called *deformation* when considered in Eulerian coordinates and *displacement* when considered in Lagrangian coordinates.

2.2 locally-defined operations

Diffeomorphisms can be composed and vector field can be summed or multiplied by scalar element, but thanks to the local isomorphisms it is possible to have operations in $\mathrm{Diff}_1(\Omega)$ that reflect the sum and the scalar product in the tangent plane, and an operation that reflect the composition in $\mathrm{Vect}(\Omega)$:

$$\begin{cases} p_1 \odot p_2 = \exp(\log(p_1) + \log(p_2)) \\ \lambda \circledast p = \exp(\lambda \log(p)) = p^\lambda \end{cases} \qquad \mathbf{v}_1 \oplus \mathbf{v}_2 = \log(\exp(\mathbf{v}_1) \circ \exp(\mathbf{v}_2)) \ .$$

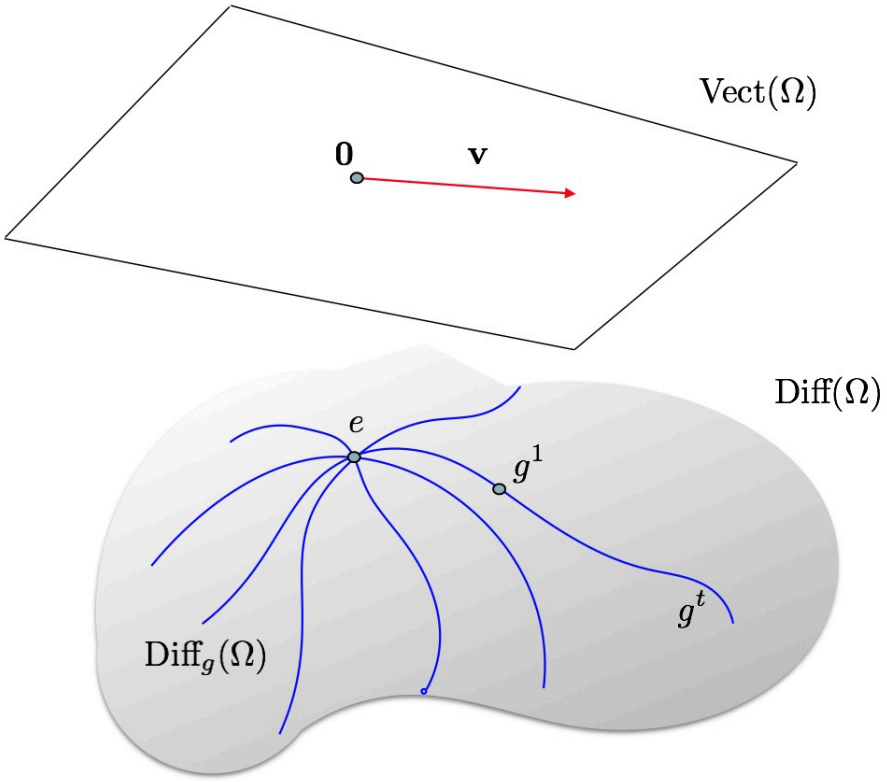


Figure 5: Schematic representation of Diff(Ω) Lie gorup of diffeomorphisms (gray shape), $\mathrm{Diff}_g(\Omega)$ subset of diffeomorphisms embedded in a one parameter subgroup (blue lines on the gray shape) and Vect(Ω) as its tangent vector space, linear approximation of Diff(Ω) at the identity.

I call the operation $\odot$ *exp-sum*, $\circledast$ *exp-multiplication* and $\oplus$ *log-composition*. Quick investigation of their behaviour respect to the inverse and neutral elements should be enough to justify the choice of the names and symbols.

With the inherited structure of vector space in an open neighbour of the identity element e, it is possible to define a metric (as for example $\mathrm{dist}(\varphi_1, \varphi_2) = ||\log(\varphi_1) - \log(\varphi_2)||$), norm, and therefore compute statistics on the group of diffeomorphisms embedded in the one-parameter subgroup. In this framework, numerical method for the computation of the exponential and the logarithm appears to be of fundamental importance and the next sections are devoted to their investigations.

3 Inegrators

The exponential of an SVF $\mathbf{v}$ is a displacement p, that at each point of the grid associates the vector defined by the one parameter subgroup defined by $\mathbf{v}$ at the time point 1, subtracted by the identity (the term displacement is used in literature to indicate any vector field in Lagrangian coordinate, in these

notes, to avoid confusion, I will use displacement of $\mathbf{v}$ only for the result of the exponential of an SVF):

$$p(\mathbf{x}) = g^1(\mathbf{x}) - \mathbf{x} \qquad \forall \mathbf{x} \in \Omega$$

An *flow-integrator* is a displacement ψ, defined over a discretization of Ω that approximate $p = \phi_1$, solution of the flow problem (5):

$$||\phi_1(x) - \psi(x)|| < \epsilon \qquad \forall \mathbf{x} \in \Delta\Omega \ . \tag{15}$$

A *curve-integrator* is a curve $\gamma(t)$ defined over a discrete set of points $\{t_n\}_{n=0}^N$ that approximate ϕ solution of the Cauchy problem (4):

$$||\phi(t_n) - \gamma(t_n)|| < \epsilon \qquad \forall n = 0, 1, ..., N \ . \tag{16}$$

Flow integrators can be numerical or geometrical. Numerical integrators originates from a numerical algorithm, while geometrical flow integrators originate from a numerical algorithm aimed to preserve some of the geometrical property of the flow (to be read: [11]).

3.1 Numerical inegrators from Taylor series: Euler, Heun and Runge Kutta

Some consideration on the Taylor expansion lead to the following well known numerical curve-integrator, that can be easily reformulated in term of flow-integrator to solve $\frac{d\phi_1(\mathbf{x})}{dt} = \mathbf{v}(\phi_1(\mathbf{x}))$. The integrators proposed start at the identity flow $\phi_0 = g^0$ and approximate ϕ_1 in N steps, having divided the time-parameter in $N+1$ equal parts $t_k = k/N$, $k = 0, ..., N$. Each step of the integrator ψ_k is an approximation of ϕ_{t_k}, and therefore the error at each step is accumulated from all of the previous steps (see [4]).

Euler Method

$$\begin{cases} \psi_0 & = 0 \\ \psi_{k+1} & = \psi_k + h\mathbf{v} \circ \psi_k \end{cases} \tag{17}$$

Where the integrator composed with the SVF $\mathbf{v}$ is always a discrete displacement (Lagrangian coordinate) while the field not composed with the SVF can be a deformation (Eulerian coordinates) -summing or subtracting the identity at both members. For computational purposes we will always deal with displacement.

Midpoint Method

$$\begin{cases} \psi_0 & = 0 \\ \psi_{k+1} & = \psi_k + h\mathbf{v} \circ \left(\psi_k + \frac{h}{2}\mathbf{v} \circ \psi_k\right) \end{cases} \tag{18}$$

Euler Modified Method

$$\begin{cases} \psi_0 & = 0 \\ \psi_{k+1} & = \psi_k + \frac{h}{2}\left(\mathbf{v} \circ \psi_k + \mathbf{v} \circ (\psi_k + h\mathbf{v} \circ \psi_k)\right) \end{cases} \tag{19}$$

Heun Method

$$\begin{cases} \psi_0 & = 0 \\ \psi_{k+1} & = \psi_k + \frac{h}{4}\big(\mathbf{v} \circ \psi_k + 3\mathbf{v} \circ (\psi_k + \frac{2h}{3}\mathbf{v} \circ \psi_k)\big) \end{cases} \tag{20}$$

Heun Modified Method

$$\begin{cases} \psi_0 & = 0 \\ \psi_{k+1} & = \psi_k + \frac{h}{4}\Big[\mathbf{v} \circ \psi_k + 3\mathbf{v} \circ \big(\psi_k + \frac{2h}{3}\mathbf{v} \circ (\psi_k + \frac{h}{3}\mathbf{v} \circ \psi_k)\big)\Big] \end{cases} \tag{21}$$

Runge Kutta Method

$$\begin{cases} \psi_0 & = 0 \\ R_k^{(1)} & = \mathbf{v} \circ \psi_k \\ R_k^{(2)} & = \mathbf{v} \circ (\psi_k + \frac{1}{2}R_k^{(1)}) \\ R_k^{(3)} & = \mathbf{v} \circ (\psi_k + \frac{1}{2}R_k^{(2)}) \\ R_k^{(4)} & = \mathbf{v} \circ (\psi_k + R_k^{(3)}) \\ \psi_{k+1} & = \psi_k + \frac{h}{6}\Big[R_k^{(1)} + 2R_k^{(2)} + 2R_k^{(3)}R_k^{(4)}\Big] \end{cases} \tag{22}$$

3.2 Numerical integrators based on the Scaling and Squaring

One of the most utilized numerical scheme for the numerical computation of equation $\frac{d\phi_1}{dt} = \mathbf{v}(\phi_1)$ is the he scaling and squaring algorithm [2, 3]. As a phase flow method [16], it exploits the one parameter subgroup property of the flow g^s and the fact that, when $\mathbf{v}$ is small the approximation $\exp(\mathbf{u}) \simeq 1 + \mathbf{v}$ holds:

$$\phi_1 = \exp(\mathbf{v}) = \big(\exp(\mathbf{v}/2^N)\big)^{2^N} \simeq \big(1 + \mathbf{v}/2^N\big)^{2^N} \ . \tag{23}$$

Computational Burdeen of this method is given only by the performance of the N compositions.

In the polyaffine scaling and squaring (or fitted scaling and squaring), instead of approximating the exponential of $\tilde{\mathbf{v}} = 2^{-N}\mathbf{v}$ with $1 + \tilde{\mathbf{v}}$, we consider to exponentiate $\tilde{\mathbf{v}}$ at each voxel independently, with a numerical method. According to the nomenclature here proposed, it is a flow-integrator for the first part, and a curve-integrator for the second part.
At the point $\mathbf{x}$ the vector $\tilde{\mathbf{v}}(\mathbf{x})$ can be written as the sum of an affine transformation and a translation $R\mathbf{x} + T$: if $\mathbf{x}$ is written as column vector in homogeneous coordinate, $(\mathbf{x}, 1)^T$ and its derivative respect to the one parameter subgroup is written as $\frac{\mathrm{d}\phi_1(\mathbf{x},1)^T}{\mathrm{d}t} = (\dot{\mathbf{x}}, 0)^T$, then the equation (5) that is linearised in (11) become

$$\begin{bmatrix} \dot{\mathbf{x}} \\ 0 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

The rotational matrix can be computed as the linear part of the transformation using the jacobian of the SVF at the point $\mathbf{x}$, while the translational part is the identity (the position of $\mathbf{x}$ itself).

Once the scaling step is performed, the computation of the approximation of $\tilde{\mathbf{v}} = 2^{-N}\mathbf{v}$ can be done in various way. Exploiting a curve-integrator means compute the approximation of $\tilde{\mathbf{v}}$ at each point of the discretised domain (as done for the polyaffine scaling and squaring).

It is possible to use instead a flow-integrator to approximate $\tilde{\mathbf{v}}$, avoiding in this way the voxelwise computation. For example if the flow-integrator is an Euler method we have:

$$\frac{d\phi_1}{dt} = \mathbf{v}(\phi_1) \qquad \phi_1 = \exp(\mathbf{v}) = (\exp(\mathbf{v}/2^N))^{2^N} = (\tilde{\mathbf{v}})^{2^N} \ .$$

If $\tilde{\psi}_1$ is the integrator of $\tilde{\mathbf{v}}$, then the its solution is given by:

$$\begin{cases} \tilde{\psi}_0 & = 0 \\ \tilde{\psi}_{k+1} & = \tilde{\psi}_k + h\tilde{\mathbf{v}} \circ \tilde{\psi}_k \ , \end{cases} \tag{24}$$

and therefore

$$\phi_1 = (\tilde{\psi}_1)^{2^N} \ .$$

This last equation is well defined if we consider the fact that both SVF and diffeomorphism can be expressed in a discretised form as vector field in Lagrangian coordinates (polyaffine Euler is not yet implemented).

3.3 A method to automatic select the step-size

Within both the scaling and squaring and the polyaffine scaling and squaring method it is possible to have an automatic selection of the step size. This is based on the number of squared scaling necessary so that the length of the biggest vector of the vector field has size less than the size of a voxel (TODO see code!)

3.4 Integration using Scipy

The framework Scipy provides some libraries to compute curve-integrators according to some methods. Given an SVF $\mathbf{v}$ we can compute the integral curve at the voxel $\mathbf{x}$ from 0 to 1; if we then subtract the position of the voxel to the resulting integral curve at time 1, we have obtained the displacement field at the point $\mathbf{x}$.

This method is a curve-integrator that exploit the scipy libraries. If compared to the flow methods is much slower, but it provides a more accurate result for small deformation.

3.5 Even more numerical inegrators: implicit and adaptative

TODO: Methods based on the Taylor approximation are all explicit and take into account only one fixed step size. Try also implicit methods (Adams Bashford) and adaptative (step size varies at each step according to some criteria).

3.6 Another numerical integrator: series method combined with accelerating convergence series

TODO

4 A farm of examples

5 Comparisons and Results

Some problems arises when we need to compare different numerical methods. First of all the choice of the number of steps has an impact in the choice of the method, and it is not obvious that increasing the number of steps always leads to a better solution. The following subsection illustrates this issue in a simpler example.

5.1 Examples about the stability of a numerical method

A problem can be well posed from an analysis point of view but unfruitful from a numerical point of view. In this case a problem is said to be *ill-conditioned*.
We examine the case of a Cauchy problem defined by a 1-d TVVF (easily generalizable to multiple dimensions):

Definition 5.1 *Given a TVVF $v(t,x)$ defined over $T \times \Omega \subseteq \mathbb{R} \times \mathbb{R}$, the Cauchy problem that it defines with an initial condition*

$$\begin{cases} \dot{\varphi}(t) = v(t, \varphi(t)) \\ \varphi(t_0) = x_0 \ , \end{cases}$$

it said to be well-posed*, or stable, if the solution to the perturbed problem associated to the original one has a solution close enough to the solution of the original. Formally, if*

1. *Exists only one solution $\varphi(t)$ ($v(t,x)$ has Lipschitz condition in the x variable).*
2. *For all $\epsilon > 0$ exists a constant $k(\epsilon) > 0$ such that for all ϵ_0 and $\delta(t)$, $|\epsilon_0| < \epsilon$ and δ is continuous on its domain T and $|\delta(x)| < \epsilon$ for all $t \in T$, than the problem*

$$\begin{cases} \dot{\varphi}(t) = v(t, \varphi(t)) + \delta(t) \\ \varphi(t_0) = x_0 + \epsilon_0 \ , \end{cases}$$

called perturbed problem*, has a unique solution $\tilde{\varphi(t)}$ that satisfies*

$$|\varphi(t) - \tilde{\varphi}(t)| < k(\epsilon)\epsilon \ .$$

We observe that ϵ is a bound for $\delta(t)$ and ϵ_0 that defines the perturbed problem. The change in the perturbation δ has to be commensurate to the perturbation on the initial condition with respect to a value that influences the difference between the exact solution and the solution of the perturbed problem. Sufficient condition for a Cauchy problem to be well-posed is v to be continuous and Lipschitz [4].

Definition 5.2 *A well posed Cauchy problem defined by a TVVF $v(t,x)$ and an initial condition, is said to be* ill-conditinoned *if $k(t)$ is bigger than the maximum value of the difference of the solution and the solution of the perturbed problem:*

$$k(\epsilon) > \max_{t \in T} |\varphi(t) - \tilde{\varphi}(t)| \ .$$

Example Considering a TVVF, dependent on the parameter α, $v(t,x) = \alpha x - (\alpha+1)\exp(-t)$, the associated Cauchy problem for a given initial condition is given by:

$$\begin{cases} \frac{d\varphi(t)}{dt} = \alpha\varphi(t) - (\alpha+1)\exp(-t) \\ \varphi(0) = 1 \ . \end{cases}$$

It is easy to verify that the analytical solution is given by $\varphi(t) = \exp(-t)$. Adding a perturbation on its initial condition we obtain

$$\begin{cases} \frac{d\varphi(t)}{dt} = \alpha\varphi(t) - (\alpha+1)\exp(-t) \\ \varphi(0) = 1 + \epsilon_0 \end{cases}$$

Its solution is given by $\tilde{\varphi}(t) = \exp(-t) + \epsilon_0 \exp(\alpha t)$ and therefore

$$|\varphi(t) - \tilde{\varphi}(t)| = |\epsilon_0 \exp(\alpha t)| \ .$$

For high values of α the problem is ill-conditioned even for small ϵ_0. Numerically this leads to the fact that a method as the Euler, does not converges increasing the number of steps for α above a threshold, as can be seen in figures 6, 7 and 8.

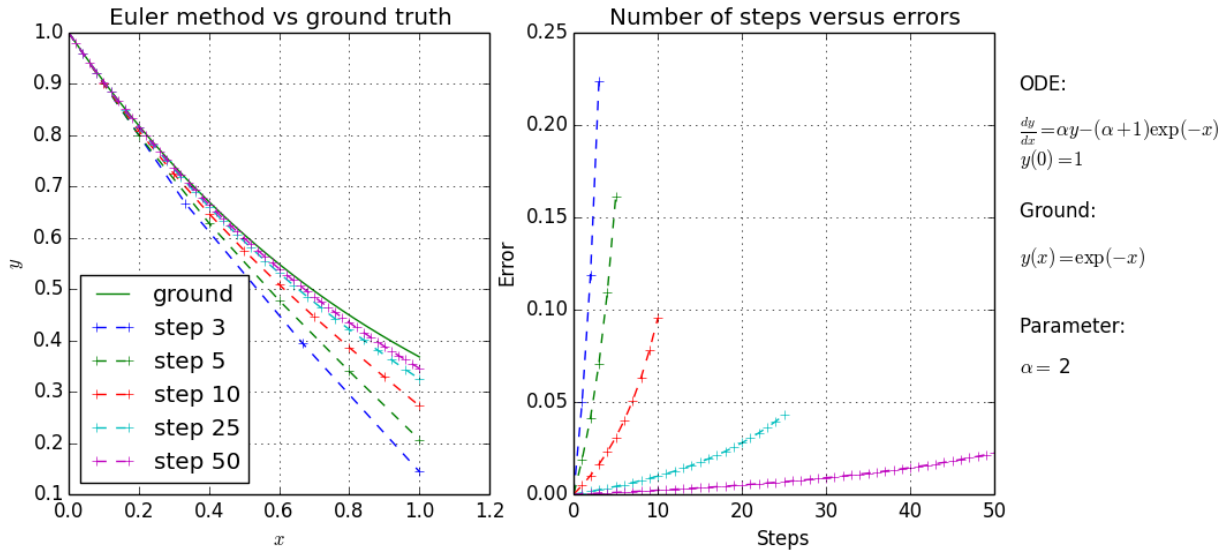


Figure 6: Euler method to approximate the solution of the Cauchy problem defined by $\frac{d\varphi(t)}{dt} = \alpha\varphi(t) - (\alpha+1)\exp(-t)$, passing throug $\varphi(0) = 1$, at the time point 1. In this case $\alpha = 2$: increasing the number of steps we have a reduction in the errors. The graph on the right shows both the final error and the stepwise errors for each choiche of the number of steps.

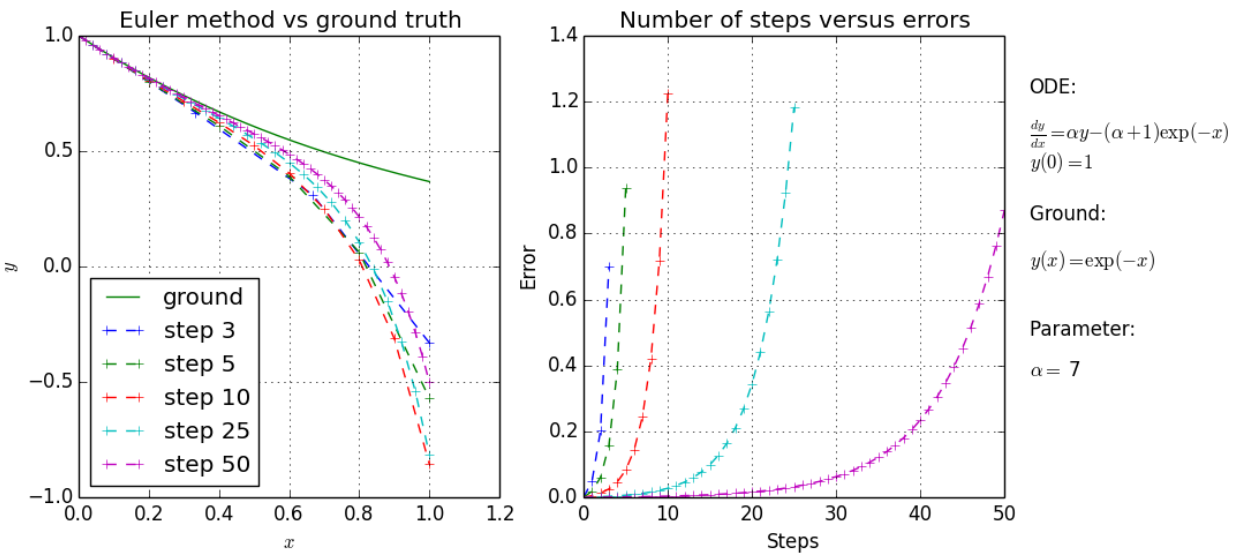


Figure 7: Euler method to approximate the solution of the Cauchy problem defined by $\frac{d\varphi(t)}{dt} = \alpha\varphi(t) - (\alpha+1)\exp(-t)$, passing throug $\varphi(0) = 1$, at the time point 1. In this case $\alpha = 7$: an increase in the number of steps does not imply a reduction in the errors. With 25 steps we have a bigger error than with 10 steps.

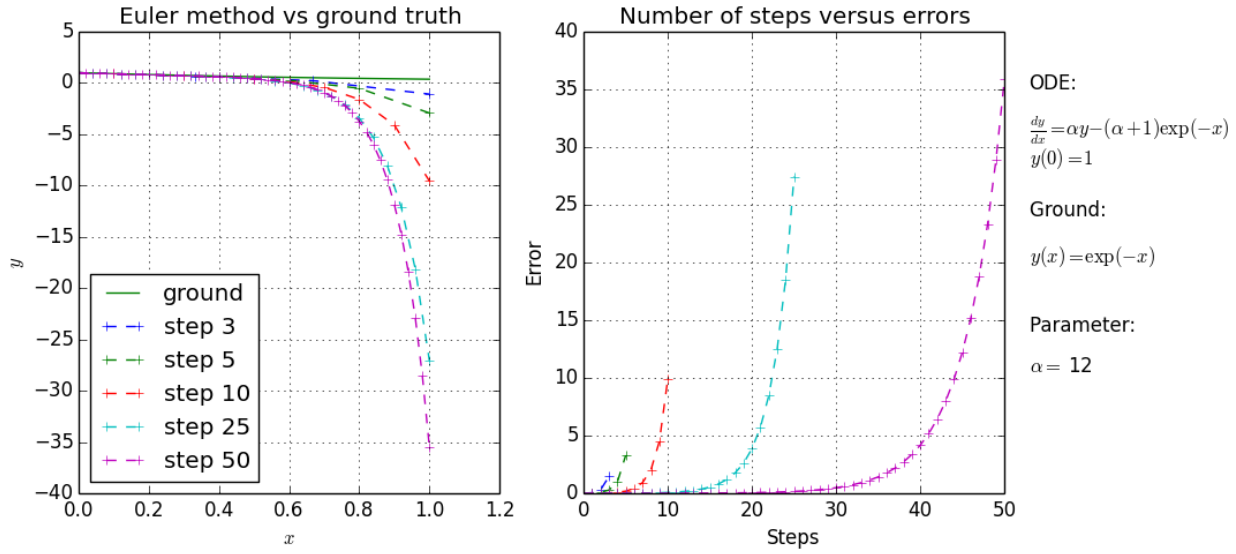


Figure 8: Euler method to approximate the solution of the Cauchy problem defined by $\frac{d\varphi(t)}{dt} = \alpha\varphi(t) - (\alpha+1)\exp(-t)$, passing throug $\varphi(0) = 1$, at the time point 1. In this case $\alpha = 12$: an increase in the number of steps never imply a reduction in the errors. A 3 steps method provides the best result.

The same concepts of well-posedness and ill-conditioned for the flow problem defined by the generic TVVF or SVF is not provided yet (TODO!). Here we consider only the numerical results of a wide enough

class of examples and we consider them as statistically significant (with a visual statistical analysis) to be able to extend the results to the generic case. - not very scientific but this is what we have so far -

Results at the moment are based on what we see with some numerical tests.

5.2 $se(2)$-generated SVF results

A first comparison between the methods for the SE(2)-generated vector field with the same number of steps for each method (10) results in the figure 9.

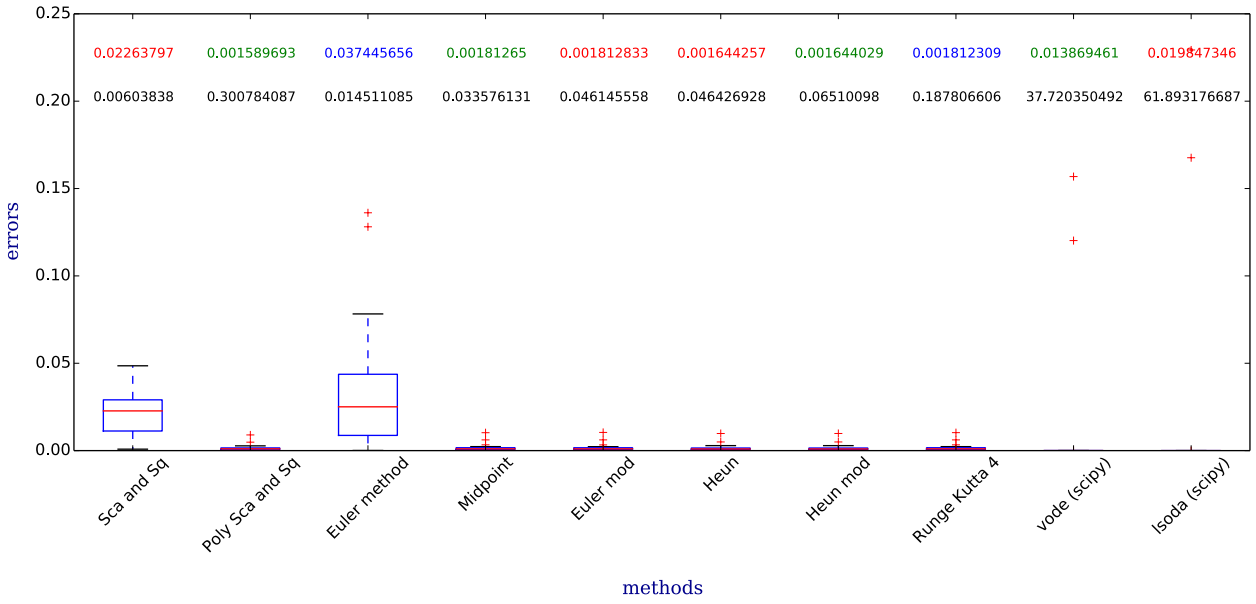


Figure 9: A data-set of 20 $se(2)$-generated SVF are sampled wth the rotation angle θ between $-\pi/4$ and $\pi/4$ excluding an interval centerd on the zero of radius 0.01. First row of coloured values in the foreground of the boxplot are teh mean of the resulting errors, values in black in the secon row are the mean of the computational time in seconds.

5.3 Number of iterations V.S. errors, some empirical considerations

If we consider the flow-integrator methods scaling and squaring, polyaffine scaling and squaring, Euler, midpoint, Euler modified, Heun, Heun modified and Runge Kutta 4 for a given SE(2)-generated SVF, where the ground truth of the exponential is available, we have the relation between the number of steps of each method and the error given in figure 10. As expected, the polyaffine scaling and squaring reach the solution in 1 step (the ODE is linear). Scaling and squaring is faster than Euler, but any other methods gets better than them (remarkably the RK4 behave really well for the linear case; Heun modified is comparable to RK4 after 2 steps).

The vertical dashed line represent the number of iteration automatically computed by the method presented in section 3.3. This shows that the automatic computation can be improved with a +2, or

settings the threshold length equals to the size of 1/4 of a voxel.

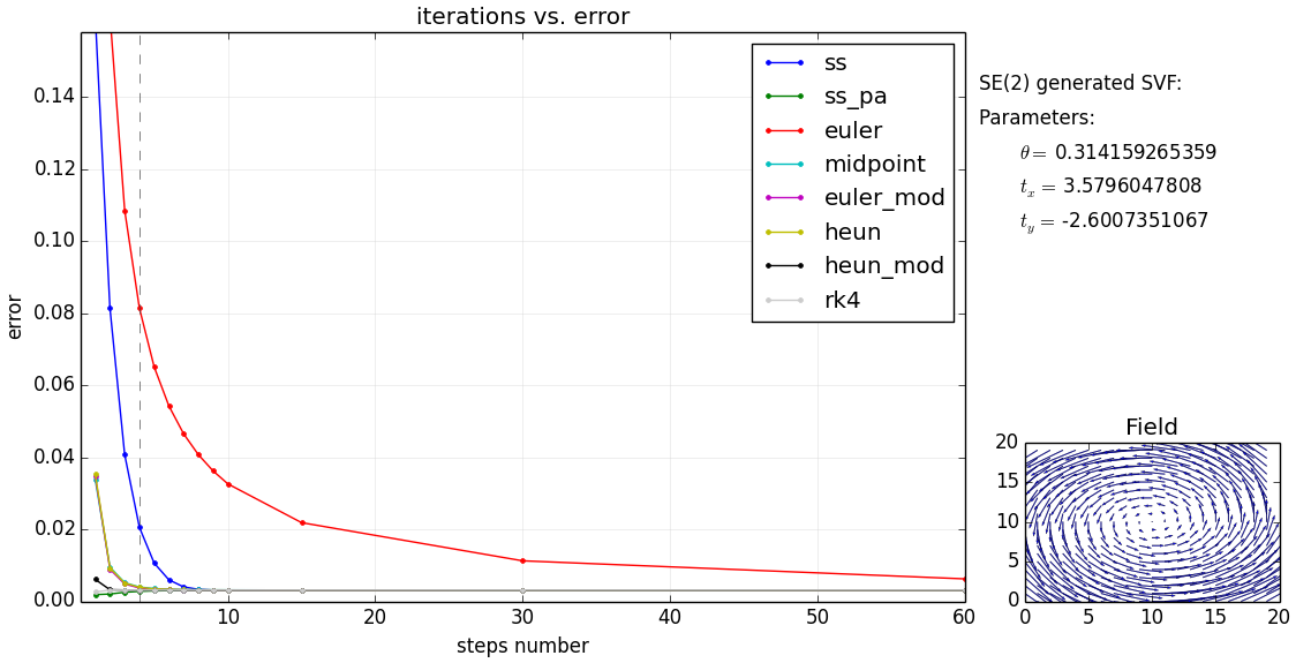


Figure 10: .

For the non-linear case, (Gauss-generated SVF) we do not have a ground truth. According to the results obtained in the linear case we selected RK4 as a *fake ground truth*. Result are shown in figure 11. Repeatedly sampling shows consistency in the results for the chosen parameters.

Heun modified is based on a similar concept than the RK4, so no surprise if it is the one with smallest error. The polyaffine scaling and squaring behave well with this fake ground truth.

These 2 experiments where useful to have some information to chose an ideal number of steps for each method:

1. Number of steps manually tuned for the linear case for each method: $[7, 1, 40, 10, 10, 10, 10, 1]$ (same order as in the legend).

2. Number of steps manually tuned for the non linear case for each method: $[8, 8, 40, 10, 10, 10, 10, 8]$.

With this choice of parameters we can repeat the experiments for a bigger data samples for both the linear and the non linear cases. Results are shown in figures 12 and 13.

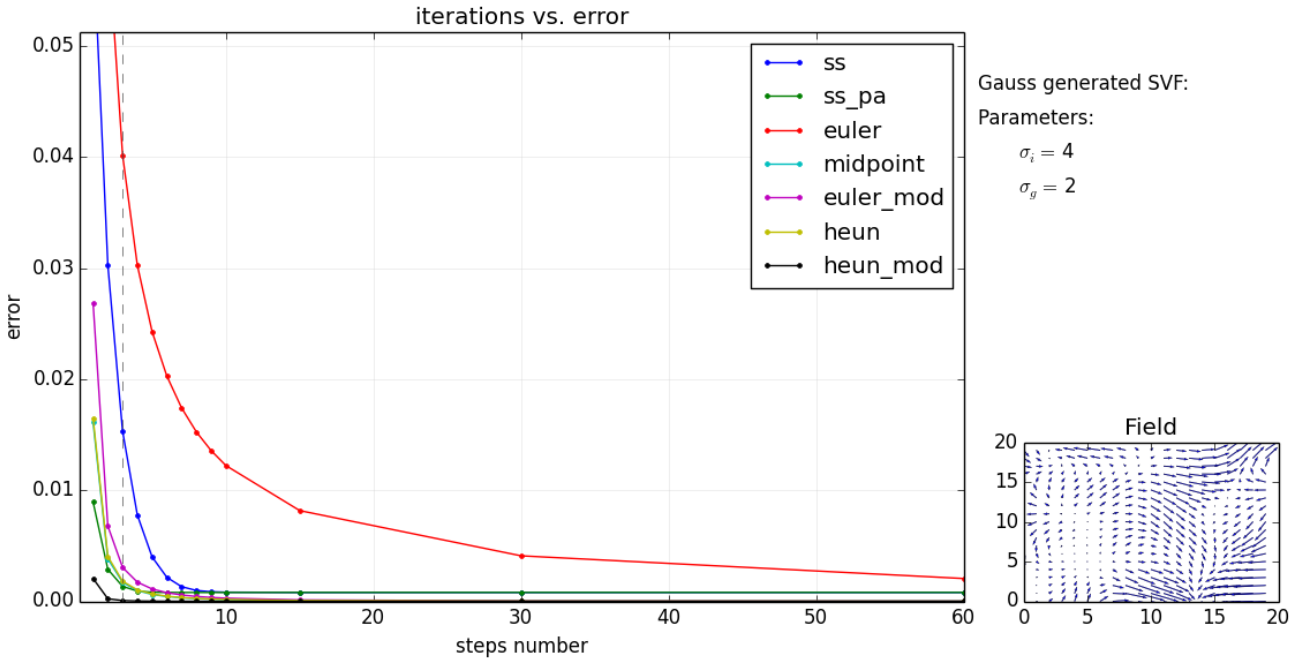


Figure 11: .

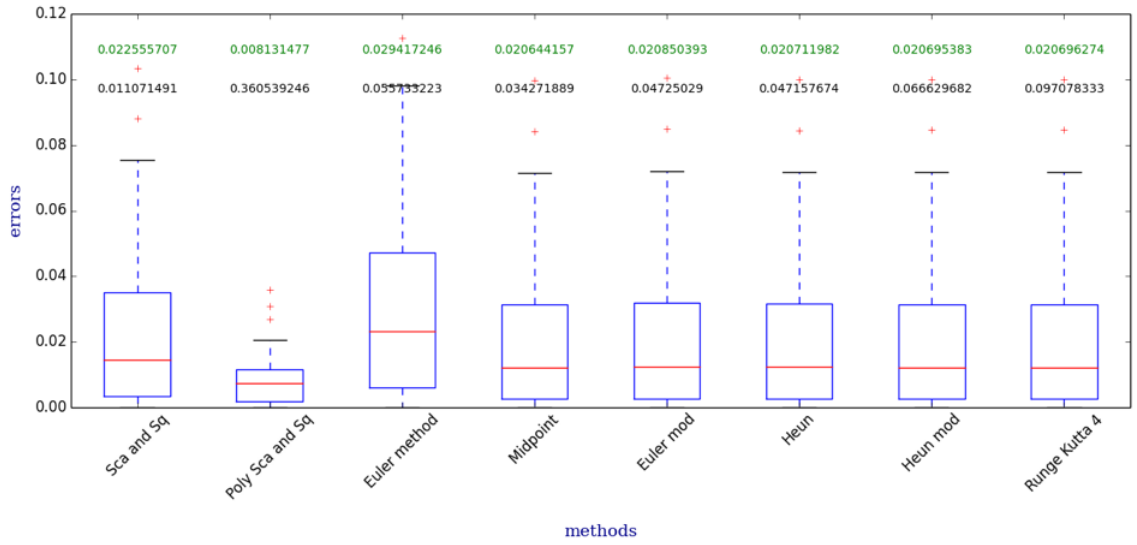


Figure 12: .

TODO list

1. Measure the steps error to see the stability of each method.

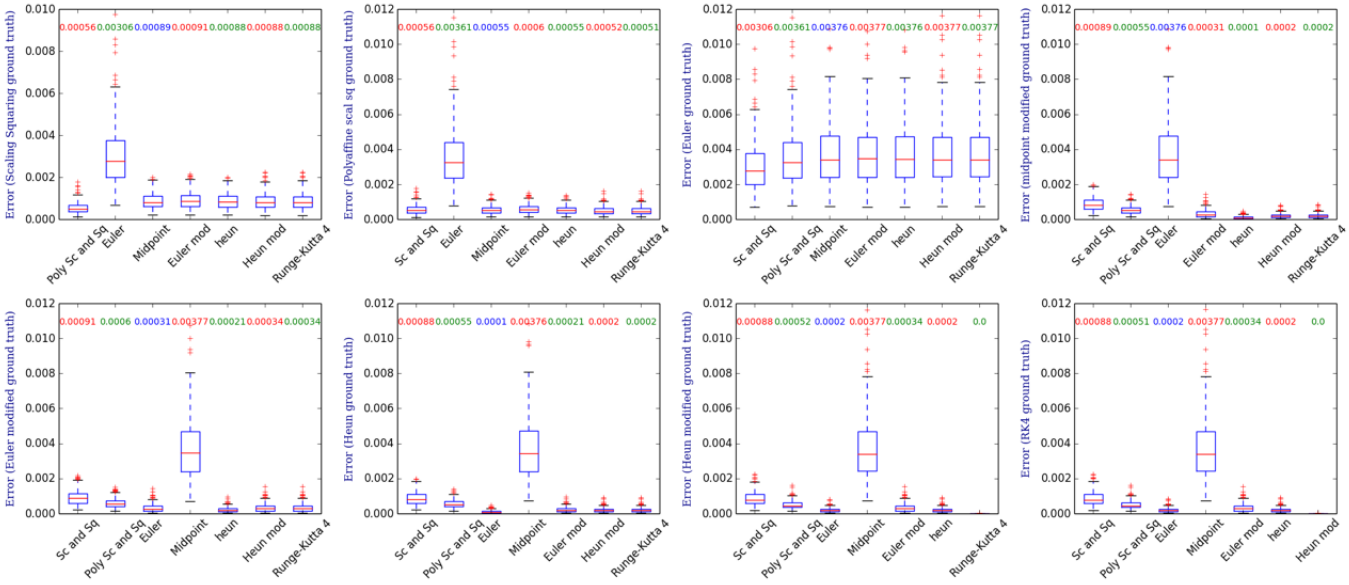


Figure 13: .

2. Validate numerical methods for the computation of the exp measuring the inverse consistency.
3. Build "ground truth" from SVF generated with ADNII database with up sampling integrate with Euler scheme and down sampling afterwards. Compare these to the one computed with other methods in the original space.
4. Begin a collection of examples of stationary ode, where the closed analytical form is available. So we have ground truth in less straightforward situation than SE(2).
5. Extend the closed form integrator to all of the possible linear case (not only SE(2))
6. series method and accelerating convergence series.
7. Look for a ground truth for small-curled vector fields.
8. Explore the geometrical integrators.
9. discuss about the shifted exponential function for the integration scheme: [7] pag. 2 and [5].
10. Explore other scaling and squaring flow-integrators.

TODO list code

1. Test exp code, each numerical method, once there is some ground example.
2. Integrate algorithm for the computation of the Logarithm (ISS or Bossa or some other things will come).

3. Speed, computational complexity: refactor the code so to have better operations for the multiplication and scalar multiplication that are correctly inherited in the new object. (requires some more tests and refactoring of the sfv methods class!)

References

[1] Vladimir Arnold. Ordinary differential equations. translated from the russian by roger cooke. second printing of the 1992 edition. universitext, 2006.

[2] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A log-euclidean framework for statistics on diffeomorphisms. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*, pages 924–931. Springer, 2006.

[3] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. Statistics on diffeomorphisms in a Log-Euclidean framework. In *1st MICCAI Workshop on Mathematical Foundations of Computational Anatomy: Geometrical, Statistical and Registration Methods for Modeling Biological Shape Variability*, pages 14–15, 2006.

[4] Richard L Burden and J Douglas Faires. Numerical analysis. *CENGAGE Learning Custom Publishing; 10th Revised edition edition (31 Jan. 2015)*.

[5] Nicholas J Higham and Lin Lijing. Matrix functions: A short course. 2013.

[6] Morris W Hirsch, Stephen Smale, and Robert L Devaney. *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2012.

[7] Marlis Hochbruck, Christian Lubich, and Hubert Selhofer. Exponential integrators for large systems of differential equations. *SIAM Journal on Scientific Computing*, 19(5):1552–1574, 1998.

[8] Frank C Hoppensteadt. *Analysis and simulation of chaotic systems*, volume 94. Springer Science & Business Media, 2008.

[9] Boris Khesin and Robert Wendt. The geometry of infinite-dimensional groups. volume 51. Springer Science & Business Media, 2008.

[10] Jeffrey M Lee. *Manifolds and differential geometry*, volume 107. American Mathematical Society Providence, 2009.

[11] Robert I McLachlan and GRW Quispel. *Six lectures on the geometric integration of ODEs*. La Trobe University, School of Mathematical and Statistical Sciences, 1998.

[12] John Milnor. Remarks on infinite-dimensional lie groups. In *Relativity, groups and topology. 2*. 1984.

[13] Jan Modersitzki. *Numerical methods for image registration*. Oxford university press, 2003.

[14] Shigeyuki Morita. *Geometry of differential forms*, volume 201. American Mathematical Soc., 2001.

[15] Giuseppe Peano. Démonstration de l'intégrabilité des équations différentielles ordinaires. *Mathematische Annalen*, 37(2):182–228, 1890.

[16] Lexing Ying and Emmanuel J Candès. The phase flow method. *Journal of Computational Physics*, 220(1):184–215, 2006.