UCSB, Physics 129L, Computational Physics
Lecture notes, Week 9

Zihang Wang (UCSB), zihangwang@ucsb.edu

March 9, 2025

Contents

1 Optimization **1**
1.1 Optimal transport 2
1.1.1 Deterministic transport, Monge problem 2
1.1.2 Non-deterministic (probabilistic) transport, Kantorovich problem 4

2 Deterministic and Stochastic Gradient Descent **5**

3 Numerical solvers for partial differential equations **7**
3.1 Explicit forward Euler methods 8
3.2 Explicit Runge-Kutta Methods 8
3.3 First Order Runge-Kutta Method 9
3.4 Second Order Runge-Kutta Method (Midpoint) 10
3.5 4th Order Runge-Kutta Method 10
3.6 Stiff equation 11

4 Symplectic Integrators **12**
4.1 Non-Symplectic Methods 13
4.2 Symplectic Euler method 14
4.3 leap-frog method (Verlet) 14

1 Optimization

Optimization is the process of finding a set of solutions $f(\mathbf{x})$ that minimizes a given **target functional** F. The functional has the following **action** form,

$$F[f(\mathbf{x})] = \int L[f(\mathbf{x})]dx, \tag{1}$$

where $L[f(\mathbf{x})]$ is the Lagrangian density.

1.1 Optimal transport

The essential idea behind **optimal transport** is to find the most efficient way to move one distribution of mass (or probability) to another while minimizing a given cost.

1.1.1 Deterministic transport, Monge problem

Given two probability distributions, **source distribution** (known) $\mu(\mathbf{x}, \boldsymbol{\theta_\mu})$ and a **target distribution** $\nu(\mathbf{y}, \boldsymbol{\theta_\nu})$, defined on spaces $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, respectively, we look for a **transport map** $T : X \to Y$ that pushforwards μ onto ν,

$$T[\mu] = \nu, \quad TX = Y, \tag{2}$$

where it implicitly involves the action of T on every individual point in the support of μ.

The **transport cost** associated with the transport plan is defined as,

$$C(T) =_X c[x, T(\mathbf{x})] d\mu(\mathbf{x}) =_X c[\mathbf{x}, T(\mathbf{x})] \mu(\mathbf{x}) d\mathbf{x}, \tag{3}$$

where $c(\mathbf{x}, T(\mathbf{x}))$ is a **cost function** (e.g., squared Euclidean distance $c(\mathbf{x}, T(\mathbf{x})) = \|\mathbf{x} - T(\mathbf{x})\|^2$ if $\mathbf{x}, \mathbf{y}$ are defined on the same domain $X = Y$). If $X \neq Y$, one must define a cost measure. We should note that the transport plan T is a function of hyperparameters $\boldsymbol{\theta_\mu}, \boldsymbol{\theta_\nu}$.

The **Monge problem** is to find T^* and the hyperparameter $\boldsymbol{\theta_\nu^*}$ that minimizes the transport cost $C(T)$. The map is one-to-one.

In particular, **Wasserstein-2 distance** is computed as,

$$W_2(T) =_X \|\mathbf{x} - T(\mathbf{x})\|^2 \, d\mu(\mathbf{x}). \tag{4}$$

The small variation t in the transport map δS leads to $T \to T + t\delta S$,

$$\delta_t W_2(T)|_{t=0} = 2_X[\mathbf{x} - T(\mathbf{x})] \cdot S(\mathbf{x}) \, d\mu(\mathbf{x}), \tag{5}$$

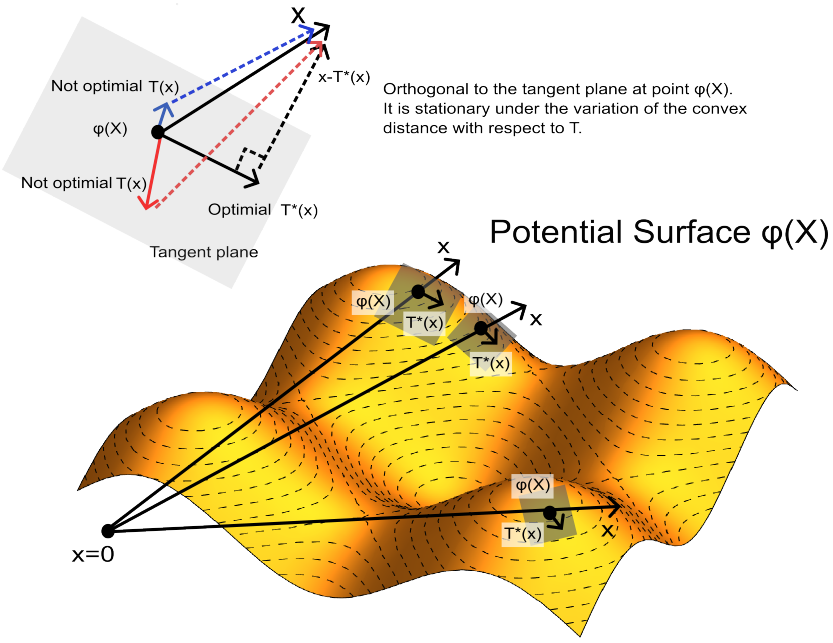
lead to the following orthogonality condition,

$$0 = [\mathbf{x} - T(\mathbf{x})] \cdot S(\mathbf{x}), \tag{6}$$

for any small deviation in the transport map $S(\mathbf{x})$ in **the tangent plane** and it can be understood as the tangent plane projection of the vector $\mathbf{x}$. The transport map $T(\mathbf{x})$ lives in the tangent plane and it defines the **shortest distance** between the tangent plane to the vector point $\mathbf{x}$ with magnitude $\|\mathbf{x} - T(\mathbf{x})\|^2$. This is exactly the term in Wasserstein-2 distance $W_2(T)$ we are trying to minimize. This concept is illustrated on the insert in the following figure. In addition, at each point $\mathbf{x}$, the tangent plane define a general surface $\varphi(\mathbf{x})$ where its gradient is the transport map,

$$\partial_{\mathbf{x}} \varphi(\mathbf{x}) = T(\mathbf{x}). \tag{7}$$

This is also illustrate in the figure below. For demonstration purpose, the vector $\mathbf{x}$ extends out of the plane, but it should be exactly touching the plane, and the projection is defined by the vector that connects to the origin.



The map should satisfy the following pushforward condition: The distribution $T(\mu)$ should have the same probability density function as ν,

$$\int_X \phi(T(\mathbf{x}))d\mu(\mathbf{x}) = \int_Y \phi(\mathbf{y})d\nu(\mathbf{y}), \quad \text{for all test functions } \phi. \tag{8}$$

Since we know, $T(\mathbf{x}) = \mathbf{y}$, and the Jacobian matrix, $|d\mathbf{y}/d\mathbf{x}| = |\det(\partial_{\mathbf{x}} T)|$

$$\begin{aligned} \int_X \phi(T(\mathbf{x}))\mu(\mathbf{x})d\mathbf{x} &= \int_Y \phi(\mathbf{y})\nu(\mathbf{y})d\mathbf{y} \\ \int_X \phi(T(\mathbf{x}))\mu(\mathbf{x})d\mathbf{x} &= \int_Y \phi(\mathbf{y})\nu(\mathbf{y})|\det(\partial_{\mathbf{x}} T)|d\mathbf{x}, \end{aligned} \tag{9}$$

since $\mathbf{y} = T(\mathbf{x})$, we have,

$$\nu[T(\mathbf{x})] \cdot |\det(\partial_{\mathbf{x}} T)| = \mu(\mathbf{x}), \tag{10}$$

if we assume there exist a **convex potential** such that the i-th component of the potential is given by $\partial_{\mathbf{x}}\phi = T(\mathbf{x})$, we have,

$$\nu[\partial_{\mathbf{x}}\phi] \cdot |\det(\partial_{\mathbf{x}}\partial_{\mathbf{x}}\phi)| = \mu(\mathbf{x}). \tag{11}$$

We note the **Hessian** $H(\phi) = \partial_{\mathbf{x}}\partial_{\mathbf{x}}\phi$, instead of the Laplacian (the trace of Hessian). The above equation is a type of **Monge–Ampère Equation**, a **nonlinear elliptic partial differential equation**.

1.1.2 Non-deterministic (probabilistic) transport, Kantorovich problem

For discrete distributions, we do not have the transport map in the Monge formulation. Suppose

$$\mu = \sum_{i=1}^{n} w_i \, \delta_{x_i}, \quad \nu = \sum_{j=1}^{m} v_j \, \delta_{y_j}, \tag{12}$$

where x_i and y_j are sample points with weights w_i and v_j respectively.

The **Kantorovich formulation of the optimal transport problem** is:

$$\min_{\pi} \sum_{i=1}^{n} \sum_{j=1}^{m} \pi_{ij} \, c(x_i, y_j), \tag{13}$$

where π_{ij} and $c(x_i, y_j)$ are the **transport plan** that bridges two distributions μ, ν, and, **transport cost** between x_i and x_j, respectively. Different from the tranport map T we discussed previously, transport plan does not uniquely map

$$\sum_{j=1}^{m} \pi_{ij} = w_i, \quad \sum_{i=1}^{n} \pi_{ij} = v_j, \quad \pi_{ij} \geq 0. \tag{14}$$

The mass conservation enforces that,

$$\sum_{i=1}^{n} w_i = \sum_{j=1}^{m} v_j. \tag{15}$$

Let's say π^* is the **optimal coupling**, The resulting transport plan is the optimal transport plan,

$$\pi^* = \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \\ \pi_{31} & \pi_{32} & \pi_{33} \end{bmatrix}. \tag{16}$$

We can approximate a deterministic transport map by the **barycentric projection**:

$$T(x_i) = \frac{1}{w_i} \sum_{j=1}^{m} \pi_{ij}^* \, y_j. \tag{17}$$

The corresponding transport cost is,

$$\sum_{i=1}^{n} w_i \, \|T(x_i) - x_i\|^2. \tag{18}$$

This optimal transport map brings $x \to y$.

Let's consider an example. We are given two discrete distributions,

$$\mu = [0.5, 0.3, 0.2], \quad \nu = [0.4, 0.3, 0.3], \tag{19}$$

and we want to find the optimal transport plan $\pi = [\pi_{ij}]$ that minimizes the transportation cost while satisfying the mass conservation constraints.

Let's assume the transportation cost between each pair (i, j) is defined by the following cost matrix,

$$c = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}, \tag{20}$$

where the cost $c(i, j)$ is given by $|i-j|$. The total transportation cost associated with a given transport plan is given by the sum,

$$C(\pi) = \sum_{i,j} \pi_{ij} \cdot c(i, j). \tag{21}$$

The mass conservation constraints for the source distribution μ are,

$$\pi_{11} + \pi_{12} + \pi_{13} = 0.5, \tag{22}$$

$$\pi_{21} + \pi_{22} + \pi_{23} = 0.3, \tag{23}$$

$$\pi_{31} + \pi_{32} + \pi_{33} = 0.2, \tag{24}$$

and for the target distribution ν,

$$\pi_{11} + \pi_{21} + \pi_{31} = 0.4, \tag{25}$$

$$\pi_{12} + \pi_{22} + \pi_{32} = 0.3, \tag{26}$$

$$\pi_{13} + \pi_{23} + \pi_{33} = 0.3. \tag{27}$$

We aim to minimize the total transportation cost:

$$C(\pi) = \pi_{12} + 2\pi_{13} + \pi_{21} + \pi_{23} + 2\pi_{31} + \pi_{32} \tag{28}$$

This is a **linear programming (LP) problem**, which can be solved via a simplex-based method. The resulting transport plan is the optimal transport plan π^*.

2 Deterministic and Stochastic Gradient Descent

Deterministic gradient descent is an iterative optimization algorithm used to find the minimum of a function. It is commonly used in machine learning and statistical modeling to minimize a cost function, such as the error between predicted and actual values.

Suppose we have a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where x_i are the input features and y_i are the corresponding target values. We want to

minimize a cost function $F(\theta)$, which measures the difference between predicted values $\hat{y}_i = f(x_i, \theta)$ and the actual target values y_i.

The objective is to find the value of θ that minimizes $F(\theta)$.

The **gradient** of a function $F(\theta)$ is a vector that points in the direction of the steepest ascent. For optimization, we want to move in the opposite direction (downhill) to minimize the function.

The gradient $\nabla F(\theta)$ is computed as the vector of partial derivatives with respect to each parameter:

$$\nabla F(\theta) = \left(\frac{\partial F(\theta)}{\partial \theta_1}, \frac{\partial F(\theta)}{\partial \theta_2}, \ldots, \frac{\partial F(\theta)}{\partial \theta_n} \right) \tag{29}$$

In gradient descent, we update the parameters iteratively to move towards the minimum of an objective function. The update rule is,

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - \alpha \nabla J(\boldsymbol{\theta}_{\text{old}}) \tag{30}$$

where:

- $\boldsymbol{\theta}_{\text{old}}$ is the current value of the parameter.
- $\boldsymbol{\theta}_{\text{new}}$ is the updated parameter.
- α is the **learning rate**, a small positive scalar that determines how large each step will be.
- $\nabla F(\theta_{\text{old}})$ is the gradient of the cost function at the current parameter value.

You should note that the **data set remains unchanged during the gradient descent. Only hyperparameters get updated at each step. This is important when we work on back propagation in deep learning.**

The process is repeated iteratively until the parameter $\boldsymbol{\theta}$ converges to a minimum (or a local minimum). Convergence is often determined when the change in the cost function value or parameters between iterations is sufficiently small.

Stochastic Gradient Descent is a variation of the Gradient Descent algorithm used for optimization. Unlike standard Gradient Descent, which computes the gradient using the entire dataset, Stochastic Gradient Descent updates the parameters using only one data point at a time. This makes SGD more efficient for large datasets and enables faster convergence.

Suppose we have a cost function $f(x)$ that we want to minimize, where x represents the parameters (or weights) of the model. In traditional Gradient Descent, the gradient is calculated using the entire dataset:

$$\nabla f(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla f(\theta, x^{(i)}) \tag{31}$$

where m is the number of data points and $x^{(i)}$ is the i-th data point. We sum over all datasets.

In Stochastic Gradient Descent (SGD), we update the parameters using a single data point (x_i, y_i) randomly selected from the dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$. The update rule is:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla J(\theta_{\text{old}}; x_i, y_i) \tag{32}$$

where x_i is randomly chosen in each iteration. For **mini-batch** SGD, a small batch of size B is chosen instead:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla J(\theta_{\text{old}}, x_{i_1}, y_{i_1}, \ldots, x_{i_B}, y_{i_B}). \tag{33}$$

Shuffling the dataset at the start of each epoch is common to ensure randomness. In each **epoch**, the model will see and update the parameters based on all the data points at least once.

3 Numerical solvers for partial differential equations

A general second-order linear two-variable PDE is given by,

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu + G = O_t[u], \tag{34}$$

where on the left hand side, we have $u_{xx} = \partial_x^2 u$, and the **characteristic equation** is derived from setting the highest-order terms to zero,

$$A\lambda + 2B\lambda + C = 0, \tag{35}$$

and λ is a curve parametrization.

$O_t[u]$ is a time-dependent differential operator, and the order of the this operator is determined based on the spatial differential operator on the left-hand side.

The nature of this PDE is determined by the discriminant of the characteristic equation,

$$\Delta = B^2 - AC. \tag{36}$$

Depending on the sign of Δ, the PDE can be classified into three types, hyperbolic, parabolic, or elliptic.

- Hyperbolic Case ($\Delta > 0$). The characteristic equation has two distinct real roots. This means there exist two real and distinct characteristic curves along which the solution propagates. Therefore, we have the second order in time, $O_t[u] \sim \partial_t^2$. Hyperbolic PDEs describe wave propagation, where information travels at finite speeds. For example, we have the 1D wave equation,

$$u_{tt} - u_{xx} = 0. \tag{37}$$

- **Parabolic Case ($\Delta = 0$)**, the characteristic equation has a repeated real root, leading to a degenerate characteristic direction. Therefore, we have the second order in time, $O_t[u] \sim \partial_t$. The solution does not exhibit wave propagation and diffuses over time. Parabolic PDEs model processes such as heat conduction and diffusion,

$$u_t - u_{xx} = 0. \tag{38}$$

- **Elliptic Case ($\Delta < 0$)** The characteristic equation has complex roots, meaning there are no real characteristic curves. Therefore, we have the zero order in time (time independent), $O_t[u] \sim$ const. There is no finite-speed propagation, and the equation typically represents equilibrium or steady-state problems. Elliptic PDEs appear in physical models such as electrostatics and steady-state heat conduction, for example, the Laplace's equation we discussed previously,

$$u_{xx} + u_{yy} = 0. \tag{39}$$

3.1 Explicit forward Euler methods

$$y'(x) = F(x, y(x)), y(x_0) = y_0, \tag{40}$$

and the variable x may be understood as the time t or other dynamic variables.

The solution solution $y(x)$ can be approximated at a discrete set of nodes,

$$x_0 < x_1 < x_2 < \cdots < x_N \leq b. \tag{41}$$

and when taking these nodes to be evenly spaced,

$$x_n = x_0 + nh, \quad n = 0, 1, \ldots, N. \tag{42}$$

we have the Euler method,

$$y(x+h) = y(x) + h\frac{\partial y}{\partial x} = y(x) + hF(t, y(x)). \tag{43}$$

Euler method is explicit, since at a latter time, the function $y(x + h)$ is an explicit function of $y(x)$.

3.2 Explicit Runge-Kutta Methods

Let's look at the same PDE in ZEuler method,

$$y'(x) = F(x, y(x)), \quad x_0 \leq x \leq b, \tag{44}$$

Explicit Runge-Kutta methods are more advanced numerical techniques for solving differential equations. These methods achieve higher accuracy by combining multiple evaluations of the derivative within each step.

The general form of an explicit Runge-Kutta method is:

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j k_j, \tag{45}$$

where:

$$k_1 = F(x_n, y_n), \tag{46}$$

$$k_2 = F(x_n + c_2 h, y_n + h a_{21} k_1), \tag{47}$$

$$k_3 = F(x_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2)), \tag{48}$$

and so on until:

$$k_s = F(x_0 + c_s h, y_n + h(a_{s1} k_1 + \cdots + a_{s,s-1} k_{s-1})). \tag{49}$$

The coefficients a_{ij}, b_j, and c_i are parameters that define the specific Runge-Kutta method being used. The relationship between c_i and a_{ij} is given by:

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \quad 2 \leq i \leq s. \tag{50}$$

Runge-Kutta methods are widely used because they have a good balance between accuracy and computational efficiency. The classical **fourth-order Runge-Kutta method** ($RK4$) is one of the most commonly used methods due to its simplicity and accuracy. Higher-order methods can be used when even greater precision is required.

3.3 First Order Runge-Kutta Method

The explicit Euler method is a **first order Runge-Kutta method** of the form:

$$y_{n+1} = y_n + hF(x_n, y_n). \tag{51}$$

Here:

$$y'(x_0) = F(x_0, y) = F(x_0, y_0), \quad y(x_0) = y_0. \tag{52}$$

Expanding $y(x_0 + h)$ using Taylor series:

$$y(x_0 + h) - y_1 = y(x_0) + hy'(x_0) + \mathcal{O}(h^2) - y_0 - hF(x_0, y_0), \tag{53}$$

which simplifies to:

$$y(x_0 + h) - y_1 = y_0 + hF(x_0, y_0) - y_0 - hF(x_0, y_0) + \mathcal{O}(h^2). \tag{54}$$

Thus:

$$y(x_0 + h) - y_1 = \mathcal{O}(h^2). \tag{55}$$

This is what we have seen: the explicit Euler method.

3.4 Second Order Runge-Kutta Method (Midpoint)

The following midpoint method is a **2-order Runge-Kutta method** given by the formula:

$$y_{n+1} = y_n + hF\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) = y_n + hF\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}F(x_n, y_n)\right). \tag{56}$$

We look at the Taylor expansions of $y(x_0+h)$ and $F\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}F(x_0, y_0)\right)$:

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2}y''(x_0) + \mathcal{O}(h^3), \tag{57}$$

and we can expand the function F,

$$F\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}F(x_0, y_0)\right) = F(x_0, y_0) + \frac{h}{2}F_x(x_0, y_0) + \frac{h}{2}F_y(x_0, y_0)F(x_0, y_0) + \mathcal{O}(h^2). \tag{58}$$

Substituting into the midpoint formula:

$$y_1 = y_0 + hF\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}F(x_0, y_0)\right), \tag{59}$$

then we plug in the expression,

$$y_1 = y_0 + hF(x_0, y_0) + h\left(\frac{h}{2}F_x(x_0, y_0) + \frac{h}{2}F_y(x_0, y_0)F(x_0, y_0)\right) + \mathcal{O}(h^3). \tag{60}$$

On the other side, we know that $y'(x) = F(x, y)$. We also find that:

$$\begin{aligned} y''(x_0) &= \frac{d}{dx}(y') = \frac{d}{dx}F(x_0, y) = \frac{\partial}{\partial x}F(x_0, y_0) + \frac{d}{dx}F(x_0, y_0) \\ &= \frac{\partial}{\partial x}F(x_0, y_0) + \frac{\partial}{\partial y}F(x_0, y_0)\frac{dy}{dx} = F_x(x_0, y_0) + F_y(x_0, y_0)F(x_0, y_0). \end{aligned} \tag{61}$$

Finally:

$$y(x_0 + h) - y_1 = \mathcal{O}(h^3), \tag{62}$$

hence, the midpoint method has order 2.

3.5 4th Order Runge-Kutta Method

Historically, the 4th order Runge-Kutta method has often been the method of choice,

$$k_1 = hf(x_n, y_n), \tag{63}$$

$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1), \tag{64}$$

$$k_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2), \tag{65}$$

$$k_4 = hf(x_n + h, y_n + k_3), \tag{66}$$

and

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h^5). \tag{67}$$

3.6 Stiff equation

A **stiff equation** is a differential equation for which certain numerical methods for solving the equation are **numerically unstable, unless the step size is taken to be extremely small**.

Let's consider the following example with Euler method on the following stiff equation,

$$y' = -cy. \tag{68}$$

If we use the **explicit Euler method**,

$$y_{n+1} = y_n + hy'_n = (1 - ch)y_n. \tag{69}$$

It becomes unstable if $h > 2/c$. This is because that,

$$y_n = (1 - ch)^n y_0, \tag{70}$$

and if $|1 - ch| > 1$, y_n becomes unbounded.

This can be avoided if we consider the **implicit (also called backward Euler method)**:

$$y_{n+1} = y_n + hy'_{n+1}, \tag{71}$$

which gives:

$$y_{n+1} = \frac{y_n}{1 + ch}. \tag{72}$$

As we can see, implicit methods are numerically stable, but in general, they are **harder to solve than explicit methods.** Consider the following PDE with **implicit Euler method**,

$$y' = F(y), \quad y_{n+1} = y_n + hF(y_{n+1}), \tag{73}$$

where on the right hand side, we have y_{n+1} instead of y_n in the explicit Euler.

To solve this, we can linearize F using Newton's method,

$$y_{n+1} = y_n + h\left[f(y_n) + \frac{\partial f}{\partial y}\bigg|_{y_n}(y_{n+1} - y_n)\right]. \tag{74}$$

Rearrange to invert the Jacobian:

$$y_{n+1} = y_n + h\left[I - h\frac{\partial f}{\partial y}\bigg|_{y_n}\right]^{-1} f(y_n). \tag{75}$$

This is called the **semi-implicit Euler method** as we convert back to the explicit method using the first order expansion. There are also stiff versions of higher-order ODE solvers.

4 Symplectic Integrators

Do we break the system symmetry when we numerically update to a new step? If the integrator preserves the symmetry, we call it **symplectic**. For example, if the numerical step does preserves the Hamiltonian, it leads to the conservation of phase space volume. In particular, **we require that the error does not accumulate in an unbounded (decay or growth) way, and remain constant.** We do not enforce the exact conservation.

The canonical condition is given by the Hamilton's equation,

$$\dot{p} = \frac{\partial H}{\partial q}, \quad \dot{q} = -\frac{\partial H}{\partial p}. \tag{76}$$

Hamilton's equations can be written in a compact form as:

$$\dot{z} = A\frac{\partial H}{\partial z}, \tag{77}$$

where $z = (p, q)$ and,

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{78}$$

A transformation (mapping) from $z = (p, q)$ to $z' = (p', q')$ can be written as:

$$z' = Jz, \quad J = \frac{\partial z'}{\partial z}, \tag{79}$$

where J is a 2×2 b matrix. We plug in this transformation into the Hamilton's equations for z, we get,

$$\dot{z}' = J\dot{z} = JA\frac{\partial H}{\partial z} = JAJ^T\frac{\partial H}{\partial z'}. \tag{80}$$

For this to be canonical, we need to **preserve the form of Hamilton's equations**,

$$\dot{z} = A\frac{\partial H}{\partial z}, \quad \rightarrow \quad \dot{z}' = A\frac{\partial H}{\partial z'}, \tag{81}$$

and this gives,

$$JAJ^T = J, \quad J^TAJ = J. \tag{82}$$

We can also use $z = A^Tz'$ and apply the argument in reverse to obtain the second equality.

We can also understood from the phase space volume perspective. Consider the volume differential in different coordinates,

$$dV = \det(J)dV'. \tag{83}$$

A volume element in n-dimensional space can be written as the **wedge product** of differentials of coordinate basis vectors (they are covariant vectors since they are differential),

$$dV = dx^1 \wedge dx^2 \wedge \cdots \wedge dx^n. \tag{84}$$

In here, $x = (q, p)$ and $y = (q', p')$. If we perform a coordinate transformation $x \to y$, then the differentials transform as:

$$dx^i = \sum_j \frac{\partial x^i}{\partial y^j} dy^j. \tag{85}$$

Substituting this into the wedge product:

$$dx^1 \wedge dx^2 \wedge \cdots \wedge dx^n = \left(\sum_j \frac{\partial x^1}{\partial y^j} dy^j \right) \wedge \cdots \wedge \left(\sum_j \frac{\partial x^n}{\partial y^j} dy^j \right). \tag{86}$$

Expanding this determinant-like structure and using the antisymmetry property of the wedge product, we obtain:

$$dx^1 \wedge \cdots \wedge dx^n = \det(J)\, dy^1 \wedge \cdots \wedge dy^n. \tag{87}$$

Thus, the volume element transforms as:

$$dV = \det(J) dV', \tag{88}$$

where $dV' = dy^1 \wedge dy^2 \wedge \cdots \wedge dy^n$ is the volume element in the new coordinates. Since we know $\det(J) = 1$ in canonical transformation, the **volume must conserve**.

4.1 Non-Symplectic Methods

Euler's method is a typical example of a Non-symplectic Method. Let's consider the phase space evolution of a simple harmonic oscillator. The Hamiltonian is given by,

$$H(p, q) = \frac{1}{2}(p^2 + q^2). \tag{89}$$

For Euler's method, we see that the determinant of the transformation matrix is non unity,

$$\det A = 1 + (\Delta t)^2 \neq 1, \tag{90}$$

so this transform is not symplectic.

We can look at the Hamiltonian after one step,

$$H = \frac{1}{2}(p^2 + q^2) = \frac{1}{2}\left((q + p\Delta t)^2 + (p - q\Delta t)^2\right) = \frac{1}{2}(p^2 + q^2)(1 + (\Delta t)^2). \tag{91}$$

From here **energy is not conserved**. it increases by a factor of $1 + (\Delta t)^2$ at each integration step, which leads to instability in the long run.

4.2 Symplectic Euler method

The advantage of symplectic methods is that they possess global stability over time. **Symplectic integrator are equal to the exact dynamics of a "close by" Hamiltonian,** $\tilde{H}(H,h)$ from the orgional Hamiltonian H.

The transformation must be chosen to preserve the canonical structure (i.e., the Poisson bracket) and give the Hamilton's equations in an invariant way.

Let's consider the **symplectic Euler method**,

$$q' = q + \frac{\partial H}{\partial p}\Delta t, \quad p' = p, \tag{92}$$

$$q' = q, \quad p' = p - \frac{\partial H}{\partial q}\Delta t, \tag{93}$$

are each symplectic, and to the first order, the Hamilton's equations become,

$$z' = \begin{pmatrix} 1 & \Delta t \\ -\Delta t & 1 - \Delta t^2 \end{pmatrix} z, \tag{94}$$

$$\tilde{H} = \frac{1}{2}\left(p^2 + q^2 + \Delta t p' q'\right) \tag{95}$$

$$= \frac{1}{2}\left((q + p\Delta t)^2 + (p - q\Delta t)^2 + \Delta t(q + p\Delta t)(p - q\Delta t)\right) \tag{96}$$

$$= \frac{1}{2}\left(p^2 + q^2 + \Delta t p q\right), \tag{97}$$

so $\tilde{H}$ is conserved.

For a fixed initial value of Δt, the above equation describes an ellipse in (p, q) plane. and the evolution will lie on an ellipse that differs from the true solution by a term of order Δt. This term cannot grow after repeated mappings, and there is no long-term energy error accumulation. Therefore, it is symplectic.

We should note that **symplectic Euler is only first-order accurate**. We should note the difference between the word accurate and symplectic. The former measures the truncation in a numerical approximation with respect to the exact solution at each step. The latter enforces a particular canonical structure. For example, in 4-th order Runge-Kutta Method (RK4), we have 4-th order accuracy, but it does not preserve the phase space volume, therefore, it is not symplectic. Over the long run, error accumulates, leading to instability.

4.3 leap-frog method (Verlet)

The **leap-frog method** has the following stepping,

$$p_{n+\frac{1}{2}} = p_n - \frac{\Delta t}{2} q_n \tag{98}$$

$$q_{n+1} = q_n + \Delta t p_{n+\frac{1}{2}} \tag{99}$$

$$p_{n+1} = p_{n+\frac{1}{2}} - \frac{\Delta t}{2} q_{n+1} \tag{100}$$

We can expression $n+1$ step in terms of n without using the half-step.

- Half-step for p,

$$p_{n+\frac{1}{2}} = p_n - \frac{\Delta t}{2} q_n. \tag{101}$$

- **Full step for q:**

$$q_{n+1} = q_n + \Delta t\, p_{n+\frac{1}{2}}, \tag{102}$$

and substituting the half-step value,

$$q_{n+1} = q_n + \Delta t \left(p_n - \frac{\Delta t}{2} q_n \right) = \left(1 - \frac{\Delta t^2}{2} \right) q_n + \Delta t\, p_n. \tag{103}$$

- **Half-step for p:**

$$p_{n+1} = p_{n+\frac{1}{2}} - \frac{\Delta t}{2} q_{n+1}, \tag{104}$$

and substitute $p_{n+\frac{1}{2}} = p_n - \frac{\Delta t}{2} q_n$ and the expression for q_{n+1}:

$$p_{n+1} = \left(p_n - \frac{\Delta t}{2} q_n \right) - \frac{\Delta t}{2} \left[\left(1 - \frac{\Delta t^2}{2} \right) q_n + \Delta t\, p_n \right]. \tag{105}$$

The above sequence gives the following stepping expression,

$$p_{n+1} = p_n - \frac{\Delta t}{2} q_n - \frac{\Delta t}{2} \left(1 - \frac{\Delta t^2}{2} \right) q_n - \frac{\Delta t^2}{2} p_n \tag{106}$$

$$p_{n+1} = \left(1 - \frac{\Delta t^2}{2} \right) p_n - \Delta t \left(1 - \frac{\Delta t^2}{4} \right) q_n. \tag{107}$$

These can be written in a matrix form as,

$$\mathbf{z}_{n+1} = M \mathbf{z}_n, \tag{108}$$

where $\mathbf{z}_n = \begin{pmatrix} q_n \\ p_n \end{pmatrix}$ and the matrix M is:

$$M = \begin{pmatrix} 1 - \frac{\Delta t^2}{2} & \Delta t \\ -\Delta t \left(1 - \frac{\Delta t^2}{4} \right) & 1 - \frac{\Delta t^2}{2} \end{pmatrix} \tag{109}$$

The determinant of M is:

$$\det(M) = 1 \tag{110}$$

This demonstrates that while Euler's method is simple to implement, it does not preserve the symplectic structure or conserve energy in Hamiltonian systems. This is **almost a rotation matrix**, other than a third order term in the off-diagonal term.

One can show that the lowest order, the leap-frog method is second order accurate, but the symplectic Euler method is only first-order accurate.

$$p_{n+1}^2 + q_{n+1}^2 = p_n^2 + q_n^2 + \mathcal{O}(\Delta t^2) \tag{111}$$