ATM 265, Spring 2019
Lecture 4
Numerical Methods: Temporal Discretizations
April 10, 2019

Paul A. Ullrich (HH 251)
paullrich@ucdavis.edu

CESM Project

- The CESM project has been posted on the course webpage. Please have a look and let me know if you have any questions. The project will be due on April 26th, 2019.

- Office hours for the class: Monday 1:30pm-2:30pm. Also available for virtual meetings on Tuesdays, Thursdays and Fridays.

Outline

1. Introduction / Motivation
2. Explicit / Implicit Methods
3. Runge-Kutta Methods
4. Lagrangian / Semi-Lagrangian Methods
5. Numerical Stability

Introduction

Last Time: Spatial Discretizations

Atmospheric Modeling – Question One

- How do we best represent continuous data when only a (very) limited amount of information can be stored?
- Equivalently, what is the best way to represent continuous data discretely?

1D Advection Equation

Eulerian Frame

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0$$

We considered this term.

This Time: Temporal Discretizations

Atmospheric Modeling – Question Two

- How do we best represent the dynamic evolution of the atmosphere? (how to deal with time?)

Let's look at this term.

1D Advection Equation

Eulerian Frame

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0$$

Spatial and Temporal Discretizations

Atmospheric Modeling – Question One

- How do we best represent continuous data when only a (very) limited amount of information can be stored?

These questions are inherently linked

Atmospheric Modeling – Question Two

- How do we best represent the dynamic evolution of the atmosphere? (how to deal with time?)

Spatial and Temporal Discretizations

So far we have discretized the spatial component of the equations:

$$\frac{\partial q_j}{\partial t} = F_j(\mathbf{q})$$

Time evolution of data point j

Some function applied to all other data points

$\mathbf{q}$ is the vector of all discrete data values

Example from finite-differences:

$$\frac{\partial q_j}{\partial t} = \frac{u}{2\Delta x} q_{j-1} - \frac{u}{2\Delta x} q_{j+1}$$

Spatial and Temporal Discretizations

All the methods discussed in Lecture 3 are linear:
For a linear differential equation (e.g. advection equation) the function f can be represented as a matrix multiply.

$$\frac{\partial \mathbf{q}}{\partial t} = \mathrm{A}\mathbf{q}$$

Time evolution of data point j.

"Spatial discretization" matrix.

$\mathbf{q}$ is the vector of all discrete data values.

Spatial and Temporal Discretizations

Example from finite-differences:

$$\frac{\partial q_j}{\partial t} = \frac{u}{2\Delta x} q_{j-1} - \frac{u}{2\Delta x} q_{j+1}$$

1D Evolution Equation

$$\frac{\partial \mathbf{q}}{\partial t} = \mathsf{A}\mathbf{q}$$

$$\mathsf{A} = \frac{u}{2\Delta x} \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & +1 \\ +1 & 0 & -1 & 0 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 & 0 \\ 0 & 0 & +1 & 0 & -1 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 \\ -1 & 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

Spatial and Temporal Discretizations

- Example from finite-differences:

$$\frac{\partial q_j}{\partial t} = \frac{u}{2\Delta x} q_{j-1} - \frac{u}{2\Delta x} q_{j+1}$$

1D Evolution Equation

$$\frac{\partial \mathbf{q}}{\partial t} = \mathsf{A}\mathbf{q}$$

Matrix is mostly zeroes!

$$\mathsf{A} = \frac{u}{2\Delta x} \begin{pmatrix} 0 & -1 & 0 & 0 & 0 & +1 \\ +1 & 0 & -1 & 0 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 & 0 \\ 0 & 0 & +1 & 0 & -1 & 0 \\ 0 & 0 & 0 & +1 & 0 & -1 \\ -1 & 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

Banded structure!

Applications to Spatial Discretizations

- An evolution matrix which consist mostly of zeroes are referred to as sparse matrices.

- Finite-difference, finite-volume, spectral element methods all (typically) lead to a sparse evolution matrix.

- The spectral transform method leads to a dense evolution matrix. That is, there are very few zeros.

Applications to Spatial Discretizations

- As the accuracy of a numerical method increases, there are fewer zeros in the evolution matrix (they make use of more information).

- That is, accuracy implies the need for a dense matrix.

- But! A more dense matrix is more computationally expense to apply in calculations.

- Hence, there is a trade-off between accuracy and efficiency.

Explicit / Implicit Methods

The Time Step

Linear Discretization

$$\frac{\partial \mathbf{q}}{\partial t} = \mathrm{A}\mathbf{q}$$

Non-Linear Discretization

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{F}(\mathbf{q})$$

Integrate these discretizations with respect to time:

Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \int_{t^n}^{t^{n+1}} \mathrm{A}\mathbf{q}\,dt$$

Current value of q (known)

Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})\,dt$$

Value of q in the future (unknown)

The Time Step

Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \int_{t^n}^{t^{n+1}} \mathrm{A}\mathbf{q}\,dt$$

Value of q in the future (unknown)

Current value of q (known)

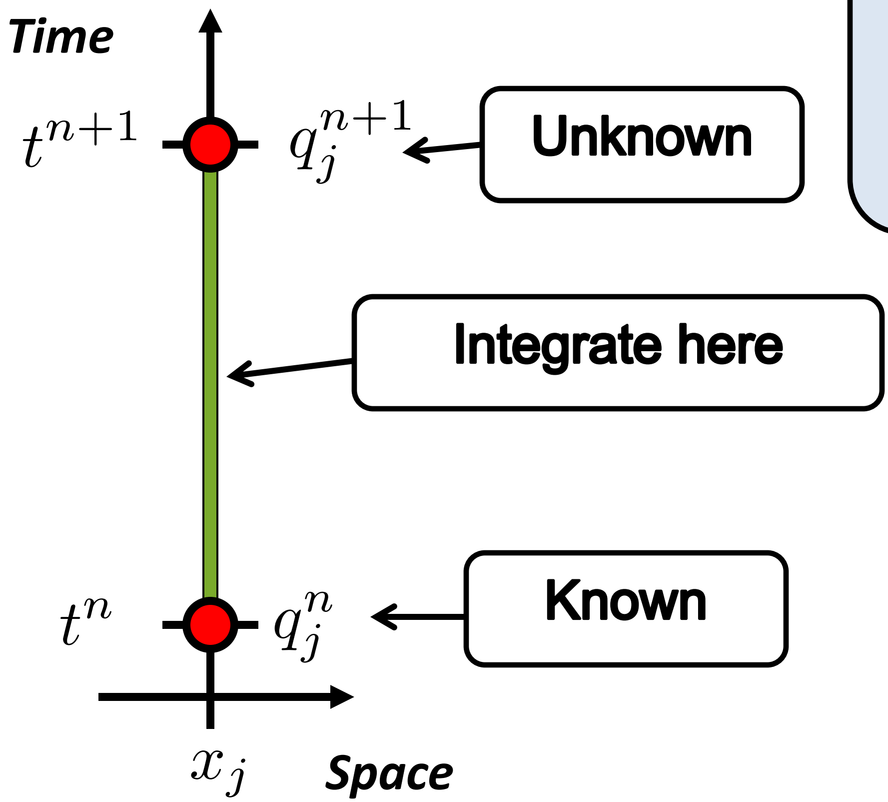
To connect present and future, need to know the value of this integral!

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})\,dt$$

Non-Linear Discretization

The Time Step

Consider the non-linear discretization of the evolution equation.



Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

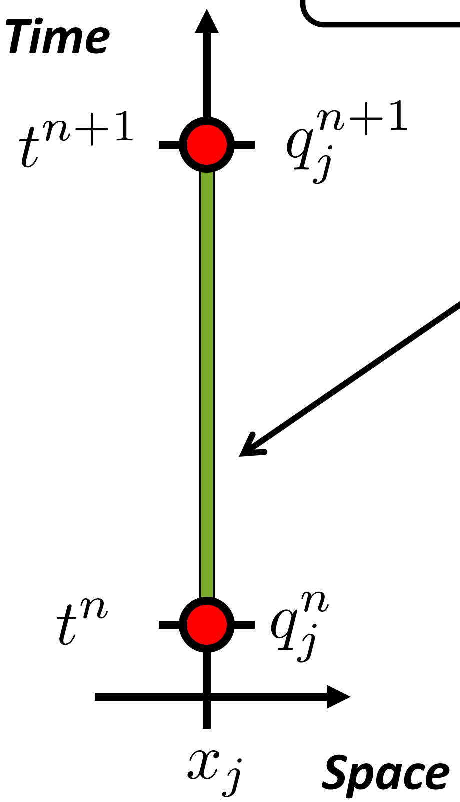
A First Explicit Scheme

Non-linear discretization.

Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

$$\Delta t = t^{n+1} - t^{n}$$



$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt \approx \Delta t\ \mathbf{F}(\mathbf{q^n})$$

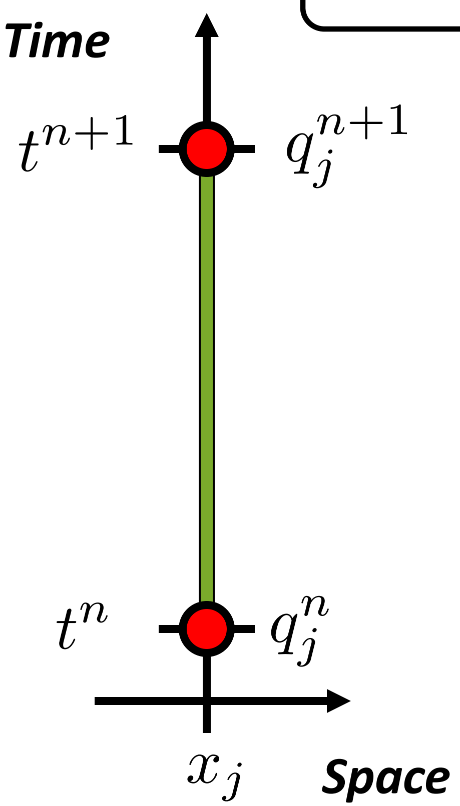
Forward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + \Delta t\ \mathbf{F}(\mathbf{q}^{n})$$

A First Explicit Scheme

Non-linear discretization.

$$\Delta t = t^{n+1} - t^n$$



Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q}) dt$$

Forward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \; \mathbf{F}(\mathbf{q}^n)$$

Unknown

Known

Under the explicit discretization, the unknown is written explicitly in terms of known values.

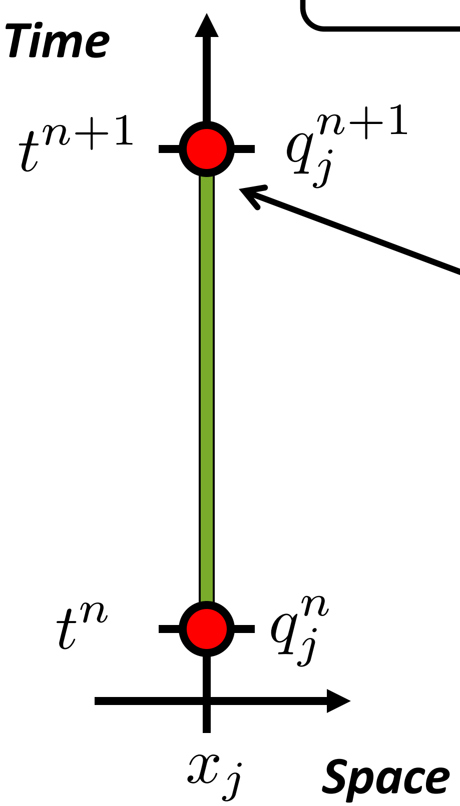
A First Implicit Scheme

Non-linear discretization.

$$\Delta t = t^{n+1} - t^n$$

Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$



$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt \approx \Delta t \; \mathbf{F}(\mathbf{q}^{n+1})$$

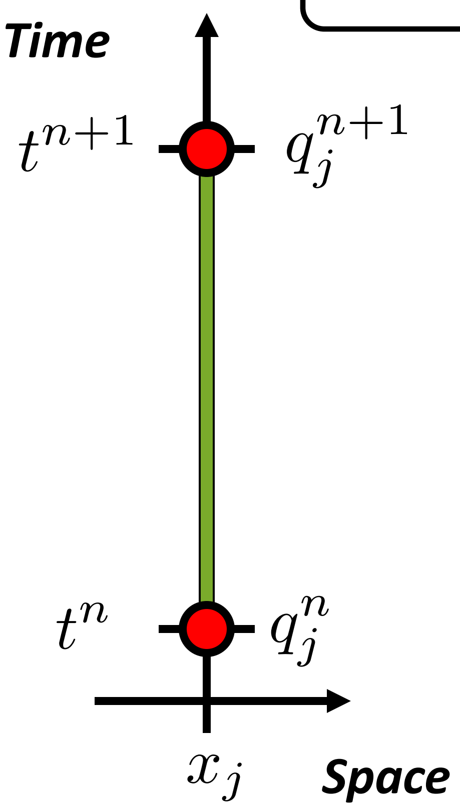
Backward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \; \mathbf{F}(\mathbf{q}^{n+1})$$

A First Implicit Scheme

Non-linear discretization.

$$\Delta t = t^{n+1} - t^n$$



Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

Backward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \ \mathbf{F}(\mathbf{q}^{n+1})$$

Unknown **Known** **Unknown**

Under the implicit discretization, one needs to solve a system of equations to find $\mathbf{q}^{n+1}$.

A First Implicit Scheme

In the linear case, the backward Euler method simplifies to

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \ \mathsf{A} \ \mathbf{q}^{n+1}$$

Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathrm{A}\mathbf{q} dt$$

We can directly rewrite this in terms of $\mathbf{q}^{n+1}$:

$$\mathbf{q}^{n+1} = \underbrace{(I - \Delta t \ \mathsf{A})}_{\text{Need to solve a linear system!}}{}^{-1}\mathbf{q}^n$$

Need to solve a linear system!

Solving the linear system is potentially an expensive operation.

A First Implicit Scheme

Backward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t\ \mathbf{F}(\mathbf{q}^{n+1})$$

Non-Linear Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^n = \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

In the non-linear case, we can also linearize the update:

$$\mathbf{F}(\mathbf{q}^{n+1}) \approx \mathbf{F}(\mathbf{q}^n) + \frac{d\mathbf{F}}{d\mathbf{q}}(\mathbf{q}^{n+1} - \mathbf{q}^n)$$

Linearly Implicit Backward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \left(I - \Delta t \frac{d\mathbf{F}}{d\mathbf{q}}(\mathbf{q}^n) \right)^{-1} \mathbf{F}(\mathbf{q}^n)$$

Again need to solve a linear system!

Explicit / Implicit Schemes

Q: Clearly the explicit method is significantly more straightforward to evaluate. Why would we choose an implicit method?

A: Stability! We will see this more later, but the basic difference is as follows:

Forward Euler Method

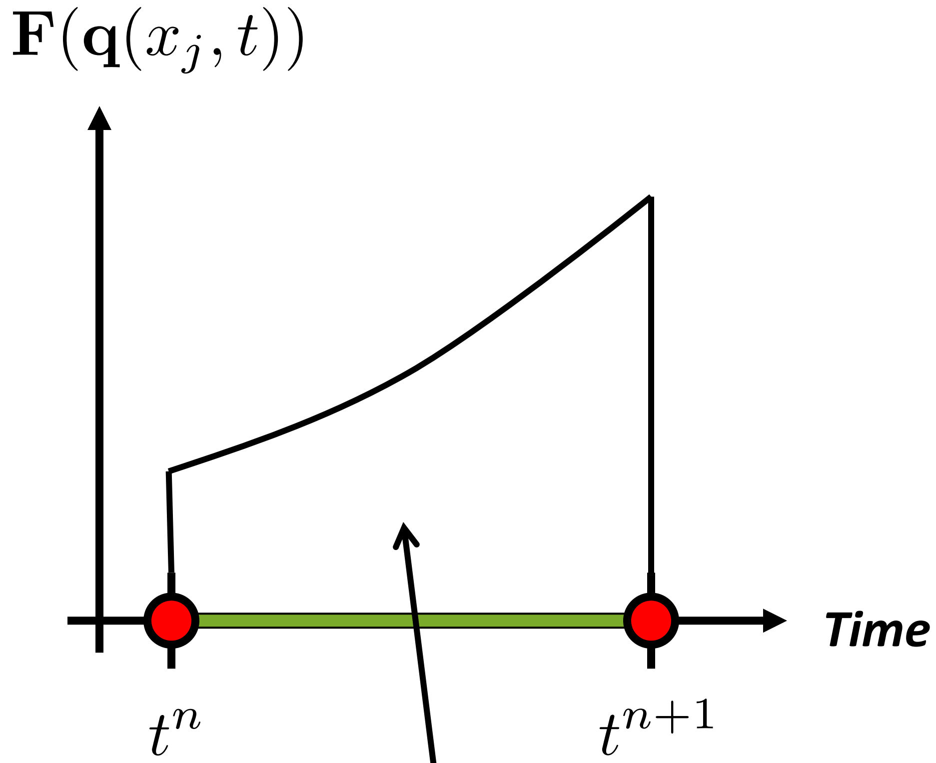
$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \; \mathbf{F}(\mathbf{q}^n)$$

Backward Euler Method

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \; \mathbf{F}(\mathbf{q}^{n+1})$$

- Implicit schemes have no limit on the size of the time step size Δt. However, a larger time step size is less accurate. Also: Implicit schemes generally require global communication.
- Explicit schemes impose a (strict) limit on the time step size Δt. Exceeding this limit will cause the method to "blow up".

Accuracy



Area under the F curve determines how to update q

Time Integral

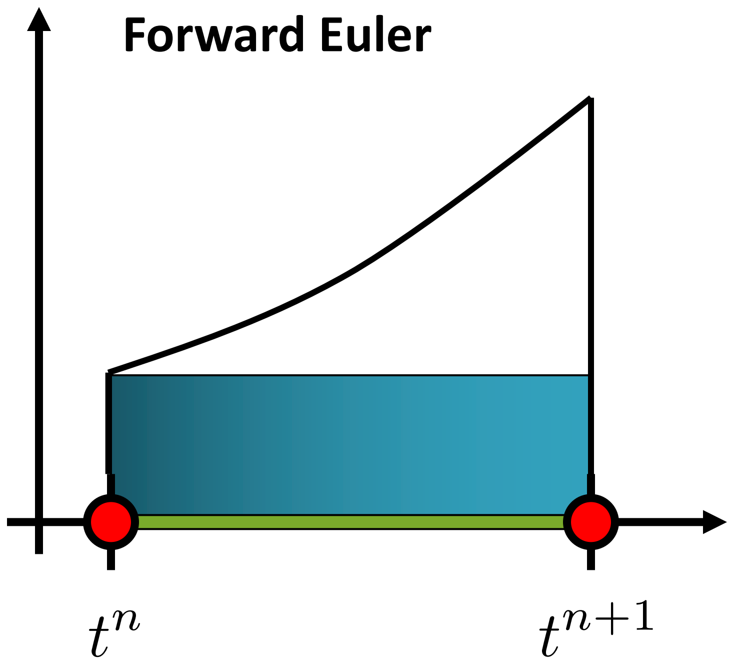
$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q}(x_j, t))dt$$

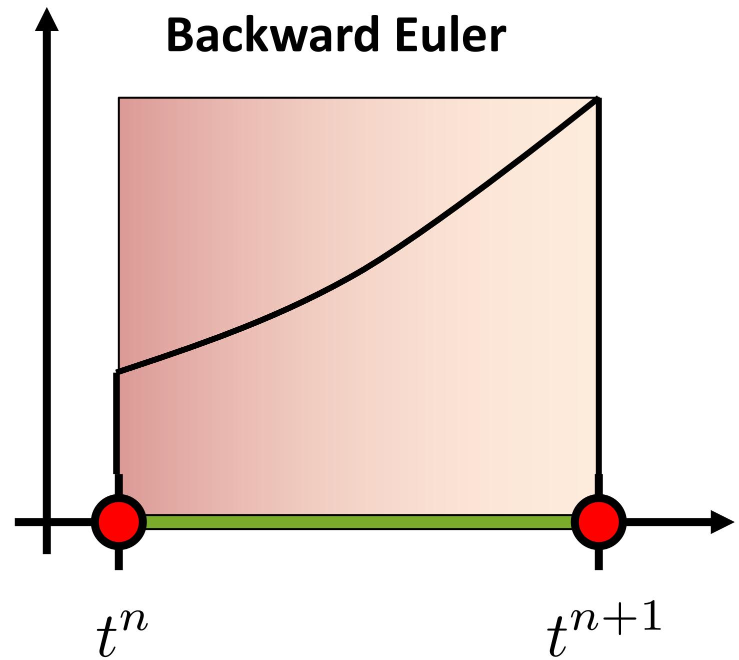
Accuracy

Time Integral

$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q}(x_j, t))dt$$

$\mathbf{F}(\mathbf{q}(x_j, t))$

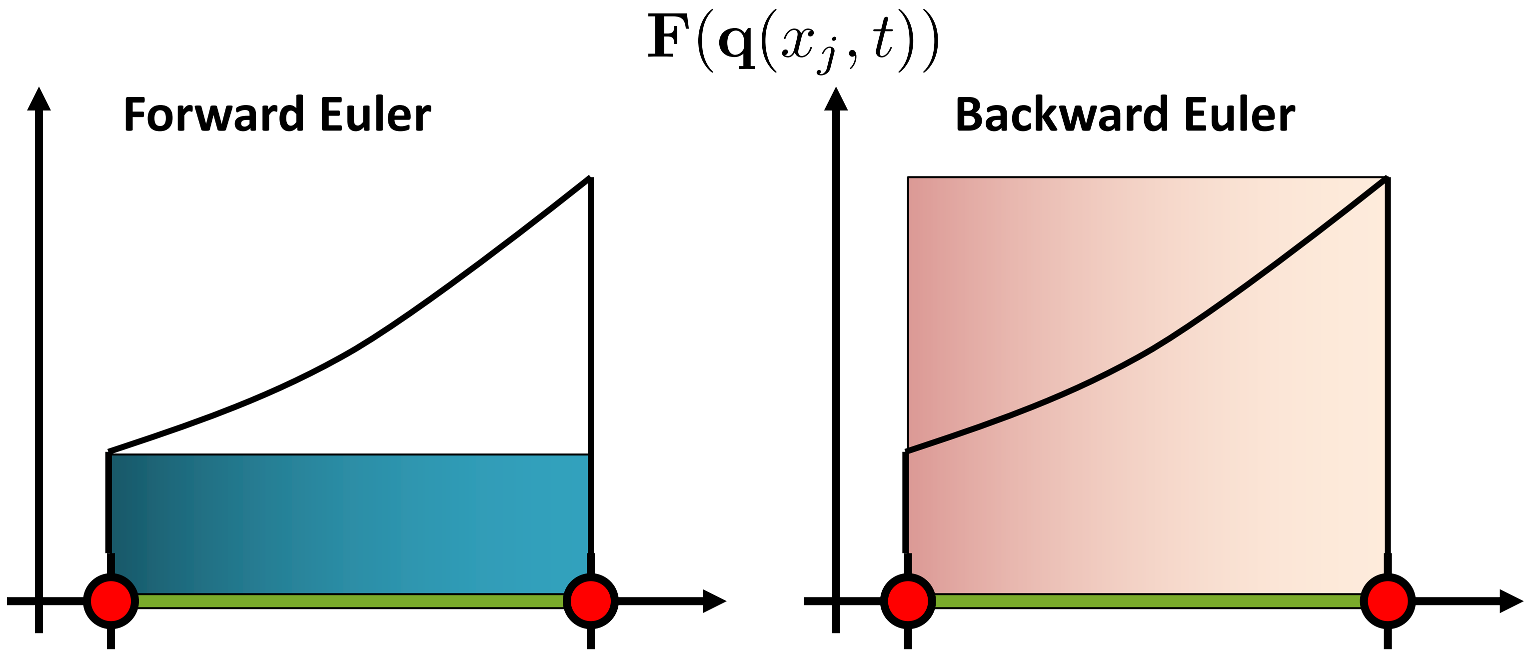




$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt \approx \Delta t \; \mathbf{F}(\mathbf{q}^n)$$

$$\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt \approx \Delta t \; \mathbf{F}(\mathbf{q}^{n+1})$$

Accuracy

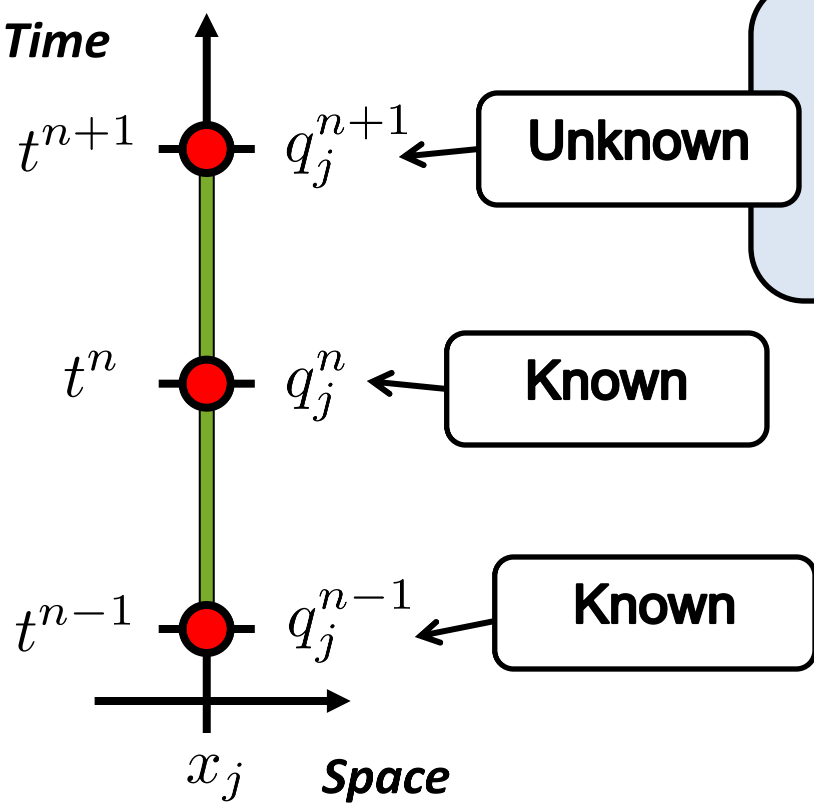


Both the forward Euler method and backward Euler method are first-order accurate: They are only exact when $\mathbf{F}$ is a constant.

First-order accuracy is typically insufficient. We need to do better.

Leap Frog

The leap frog scheme is a traditional second-order accurate explicit method. This means that the integral is exact if $\mathbf{F}$ is either constant or linear in time.



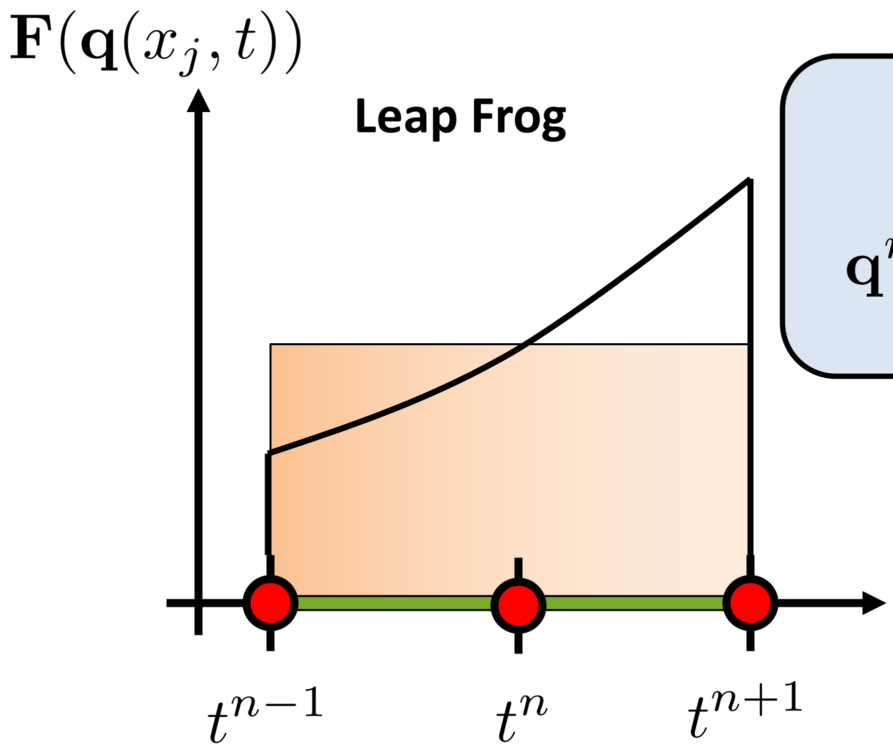
Two-Step Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n-1} = \int_{t^{n-1}}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

Leap frog requires knowledge of $\mathbf{q}$ from two time levels prior to the unknown level.

Accuracy of Leap Frog

Second-order accuracy for the leap frog method is attained by using the midpoint value.

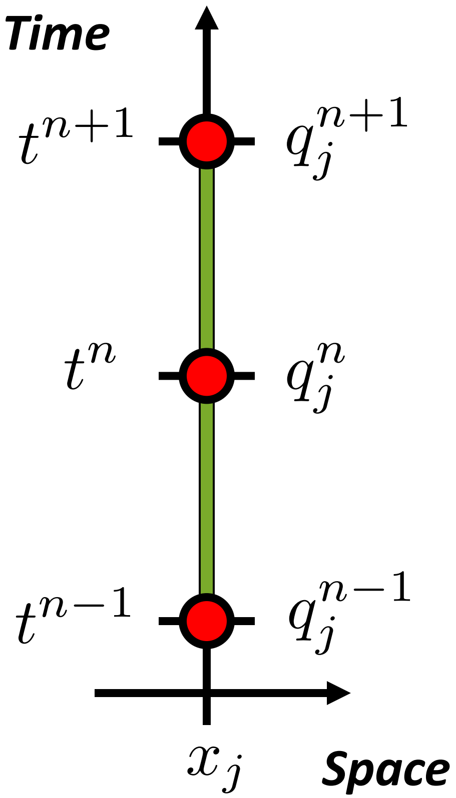


Two-Step Discretization

$$\mathbf{q}^{n+1} - \mathbf{q}^{n-1} = \int_{t^{n-1}}^{t^{n+1}} \mathbf{F}(\mathbf{q}) dt$$

Leap Frog

The leap frog scheme has traditionally been used in combination with the spectral transform method.



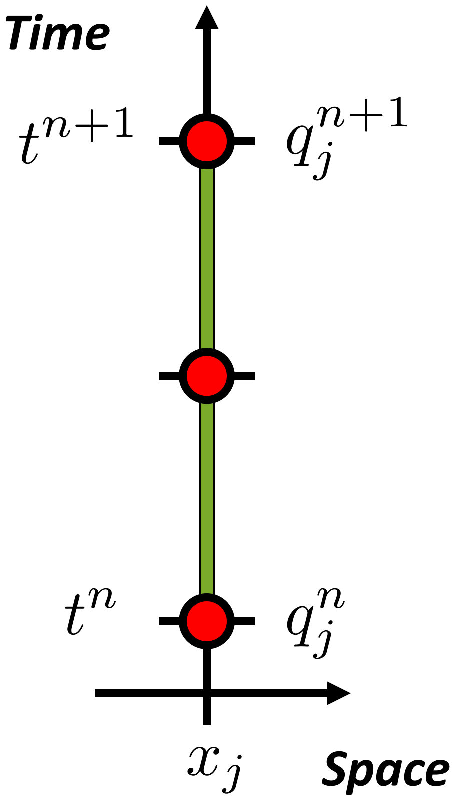
The leap frog scheme possesses a computational mode since the odd and even time levels can separate.

This is usually fixed by using off-centering (Asselin filtering)

Runge-Kutta Methods

Runge-Kutta Methods

Runge-Kutta methods are a popular method for attaining high-order accuracy in time without the need to store data from multiple time steps.



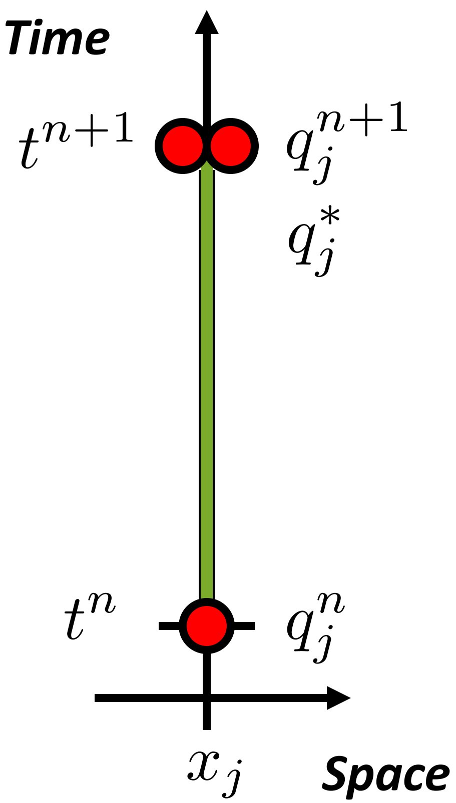
Non-Linear Discretization

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{F}(\mathbf{q})$$

- Runge-Kutta methods are multi-stage, which means in order to advance by Δt the function $\mathbf{F}$ must be evaluated multiple times.

Predictor / Corrector

The second-order accurate predictor-corrector method is one of the most basic Runge-Kutta methods.



A first-order approximation to q_j^{n+1} is first computed (prediction step):

$$\mathbf{q}^* = \mathbf{q}^n + \Delta t \ \mathbf{F}(\mathbf{q}^n)$$

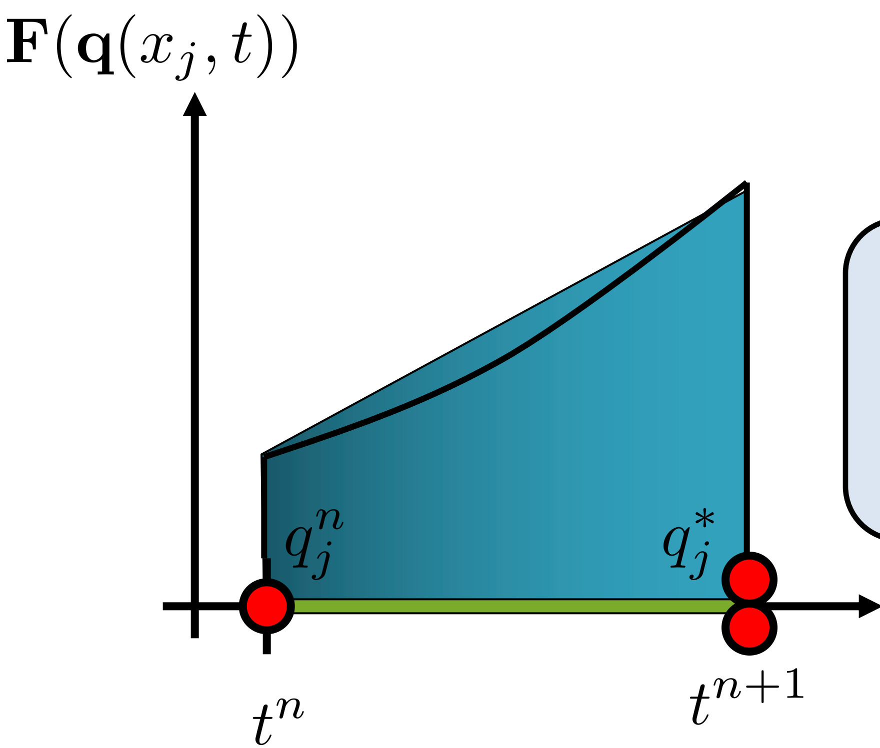
A second-order correction is then computed:

$$\mathbf{q}^{n+1} = \frac{1}{2}\mathbf{q}^n + \frac{1}{2}\mathbf{q}^* + \frac{\Delta t}{2} \ \mathbf{F}(\mathbf{q}^*)$$

Predictor / Corrector

The predictor corrector scheme can also be written as follows:

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \frac{\Delta t}{2}\mathbf{F}(\mathbf{q}^n) + \frac{\Delta t}{2}\mathbf{F}(\mathbf{q}^*)$$



Trapezoid rule for integration

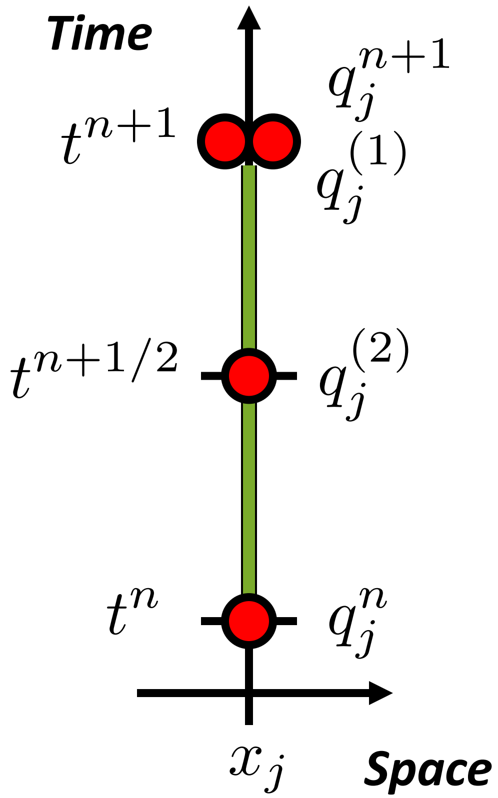
$$\mathbf{q}^{n+1} = \mathbf{q}^n + \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

Non-Linear Discretization

Strong Stability Preserving RK3 (SSPRK3)

MPAS

One of the more popular Runge-Kutta methods is the SSPRK3 scheme, which is a third-order accurate, three stage Runge-Kutta method.



Stage one:

$$\mathbf{q}_j^{(1)} = \mathbf{q}_j^n + \Delta t \mathbf{F}(\mathbf{q}^n)$$

Stage two:

$$\mathbf{q}_j^{(2)} = \frac{3}{4}\mathbf{q}_j^n + \frac{1}{4}\mathbf{q}_j^{(1)} + \frac{\Delta t}{4}\mathbf{F}(\mathbf{q}^{(1)})$$

Final update:

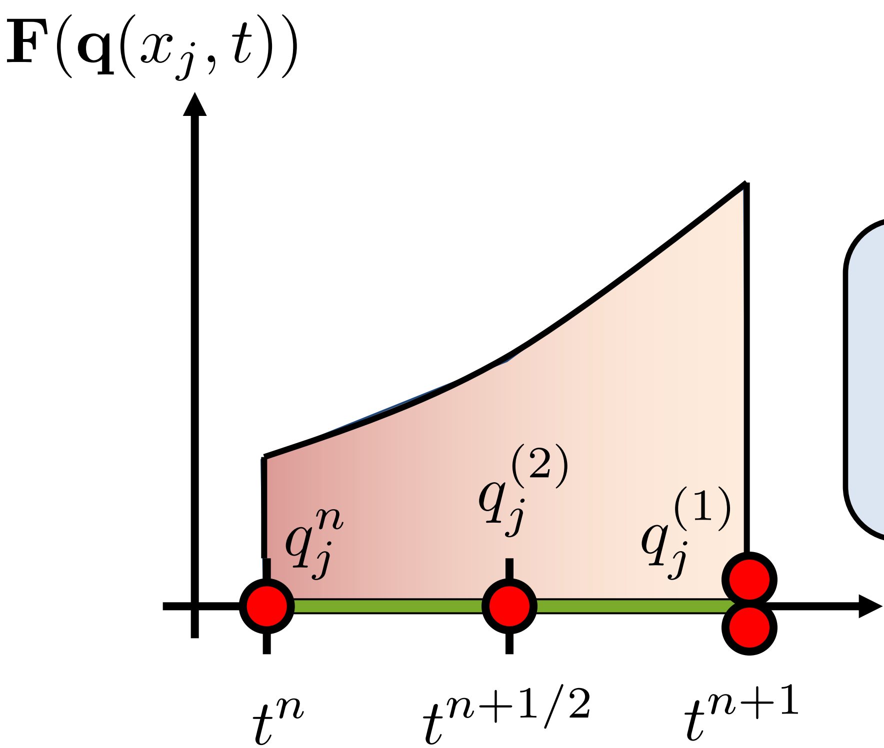
$$\mathbf{q}_j^{n+1} = \frac{1}{3}\mathbf{q}_j^n + \frac{2}{3}\mathbf{q}_j^{(2)} + \frac{2\Delta t}{3}\mathbf{F}(\mathbf{q}^{(2)})$$

Strong Stability Preserving RK3 (SSPRK3)

MPAS

Writing as a one-stage update equation:

$$\mathbf{q}_j^{n+1} = \mathbf{q}_j^n + \frac{\Delta t}{6}\left[\mathbf{F}(\mathbf{q}^n) + 4\,\mathbf{F}(\mathbf{q}^{(2)}) + \mathbf{F}(\mathbf{q}^{(1)})\right]$$



Simpson's rule for integration

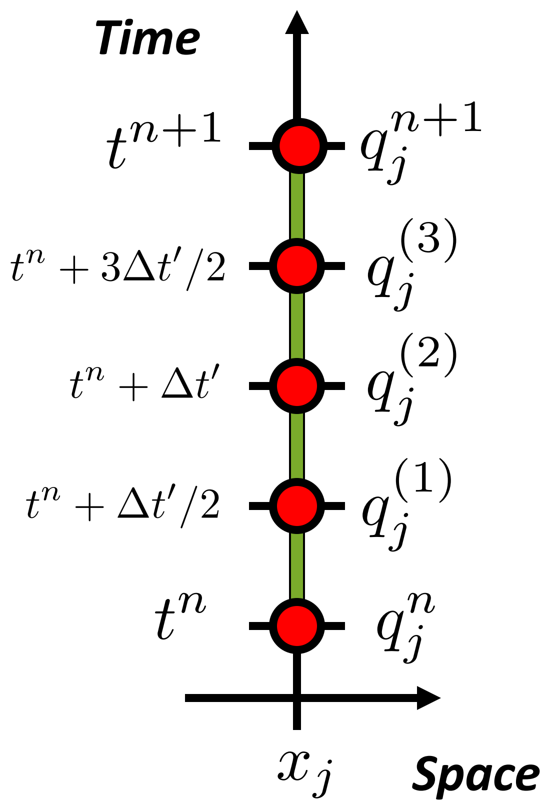
$$\mathbf{q}^{n+1} = \mathbf{q}^n + \int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{q})dt$$

Non-Linear Discretization

Synchronized Leap Frog

CAM-SE (CESM1)

The CAM Spectral Element (CAM-SE) model uses a Runge-Kutta scheme closely modeled on the leap frog scheme discussed earlier:



$$\boxed{\mathbf{q}^{(0)} = \mathbf{q}_j^n}$$

Stage one:

$$\mathbf{q}_j^{(1)} = \mathbf{q}_j^n + \frac{\Delta t'}{2}\mathbf{F}(\mathbf{q}_j^n)$$

Stage k+1:

$$\mathbf{q}^{(k+1)} = \mathbf{q}_j^{(k-1)} + \Delta t' \mathbf{F}(\mathbf{q}_j^{(k)})$$

Final update:

$$\mathbf{q}^{n+1} = \mathbf{q}^{(N)}$$

Overall 2nd order

Lagrangian and Semi-Lagrangian Methods

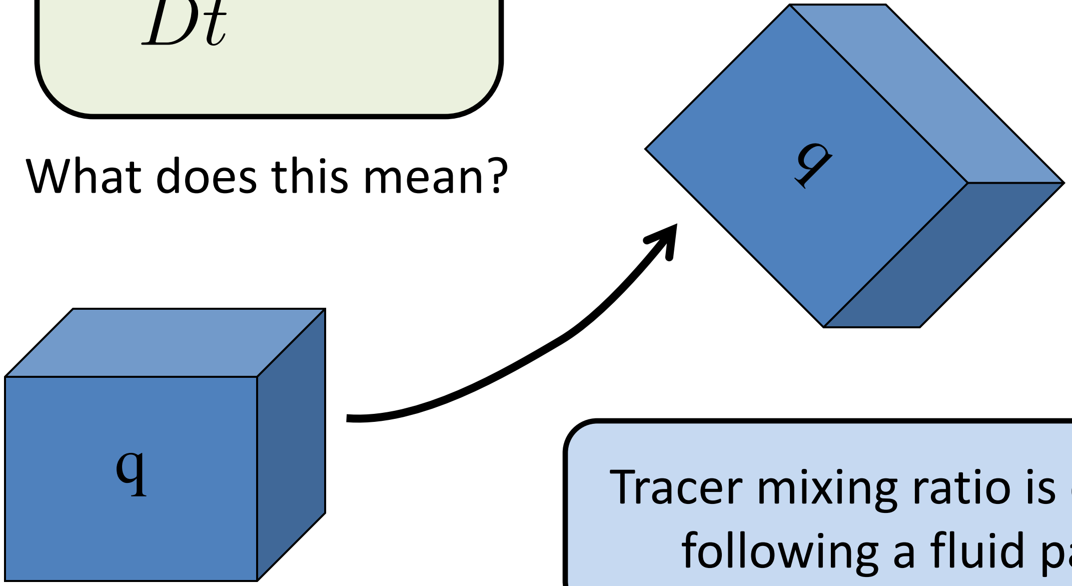
Lagrangian Methods

Recall Lagrangian reference frame (follows a fluid parcel).

Lagrangian Frame

$$\frac{Dq}{Dt} = 0$$

What does this mean?



Tracer mixing ratio is constant following a fluid parcel.

Lagrangian Methods

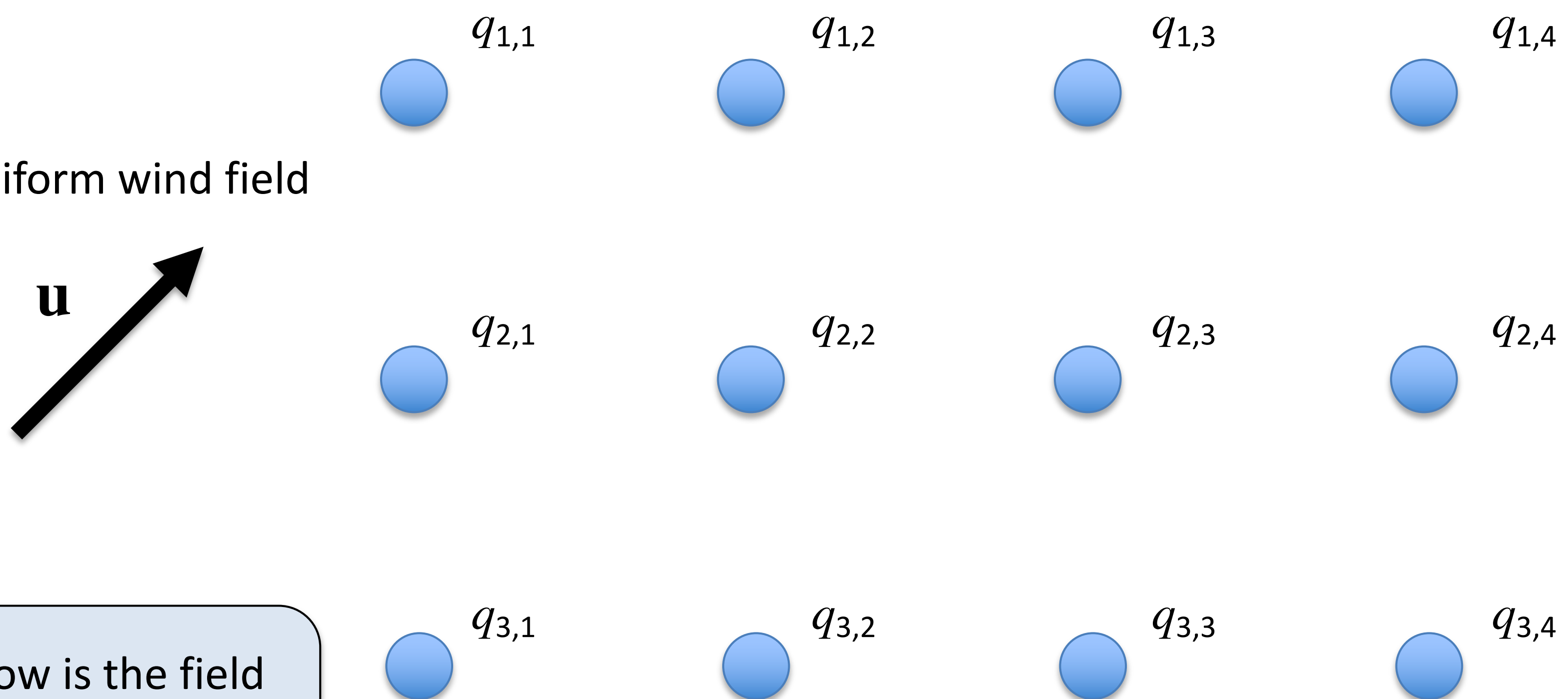
…in the finite difference context

Lagrangian Frame

Continuous: $\dfrac{Dq}{Dt} = 0$

Discrete: $q^n_{i,j} = q^{n-1}_{\hat{i},\hat{j}}$

Uniform wind field



How is the field propagating in time?

Lagrangian Methods

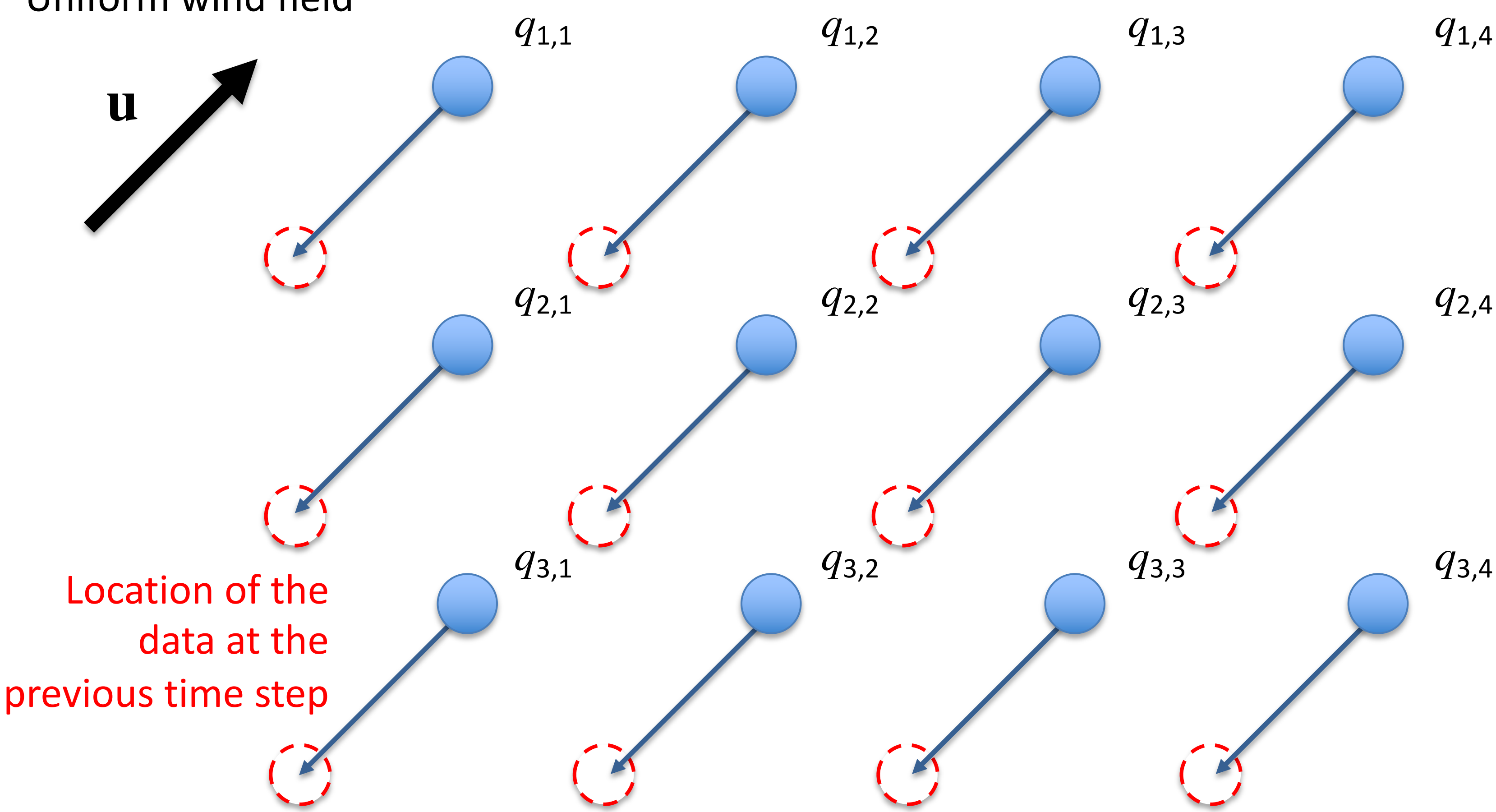
…in the finite difference context

Lagrangian Frame

Continuous: $\frac{Dq}{Dt} = 0$

Discrete: $q_{i,j}^{n} = q_{\hat{i},\hat{j}}^{n-1}$

Uniform wind field



Semi-Lagrangian Methods

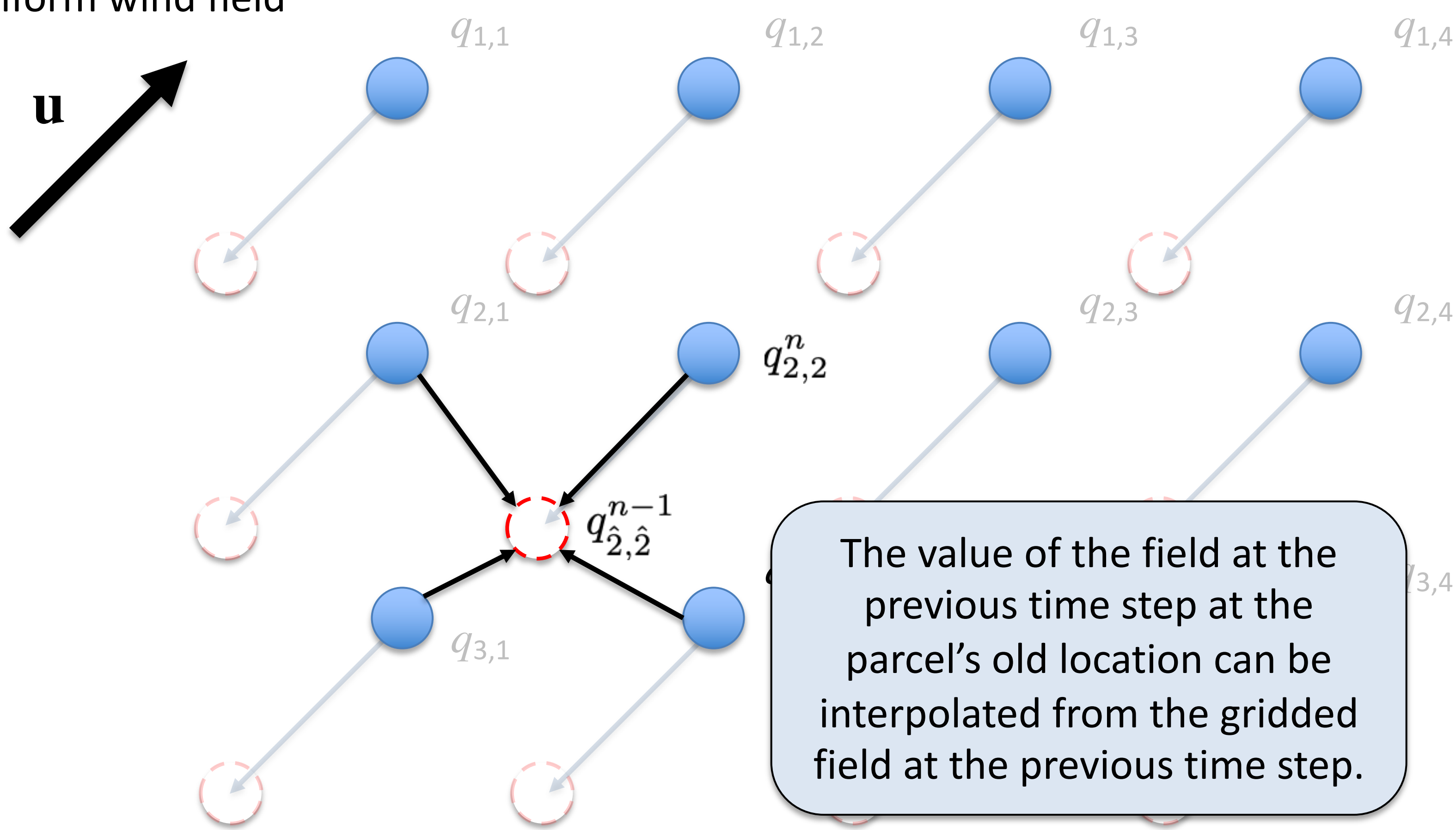
…in the finite difference context

Lagrangian Frame

Continuous: $\frac{Dq}{Dt} = 0$

Discrete: $q_{i,j}^{n} = q_{\hat{i},\hat{j}}^{n-1}$

Uniform wind field



The value of the field at the previous time step at the parcel's old location can be interpolated from the gridded field at the previous time step.

Lagrangian Methods

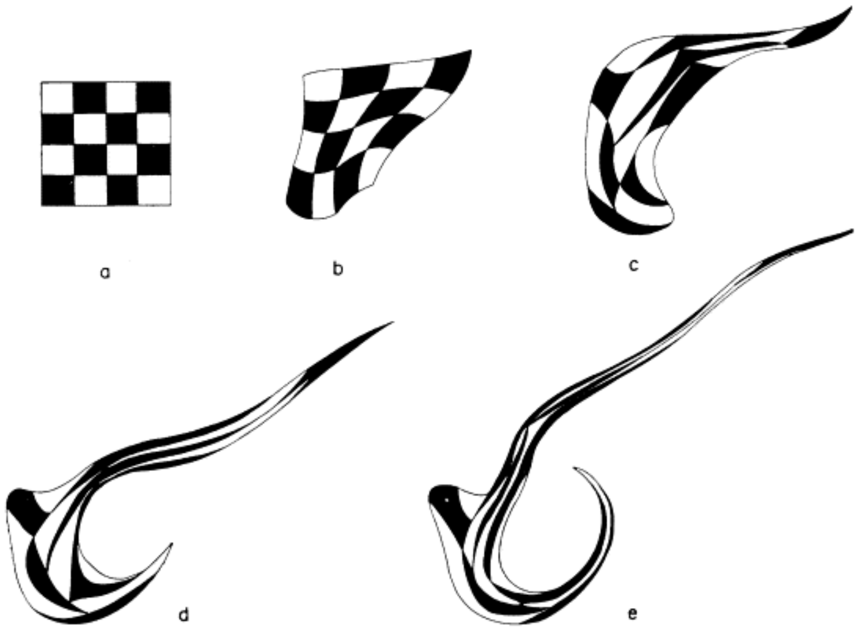
…in the finite volume context

- The Lagrangian frame is, in some sense, the most natural way to think about the advection equation.

- But: In practice it is difficult to follow around fluid parcels in presence of deforming flow.

Lagrangian Frame

$$\frac{Dq}{Dt} = 0$$

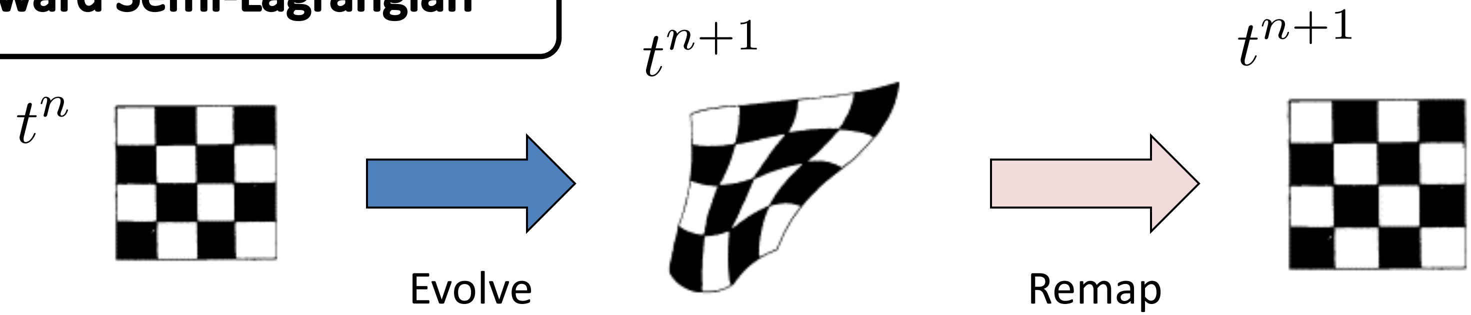


Source: R.A. Pielke and M. Uliasz (1997).

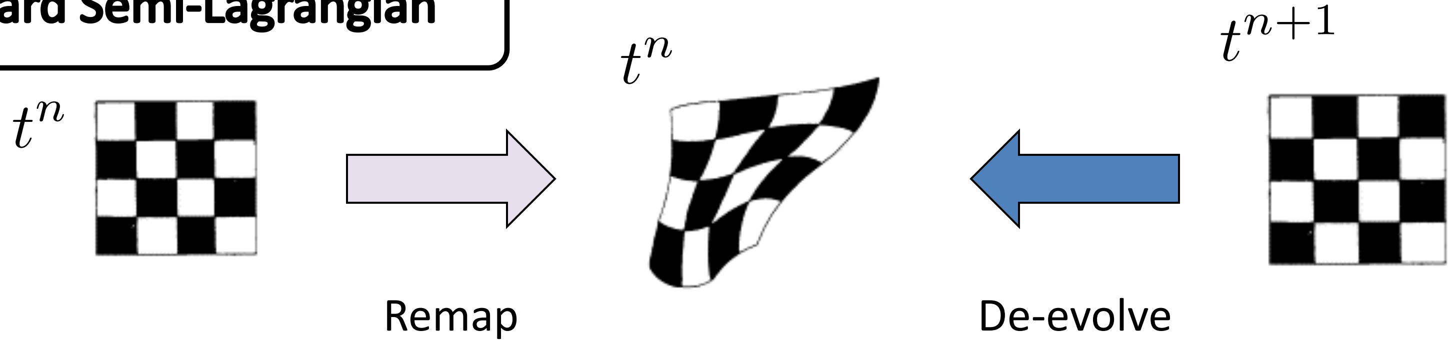
Semi-Lagrangian Methods

- Instead: Semi-Lagrangian methods follow a fluid parcel in time, then remap to a regular mesh.

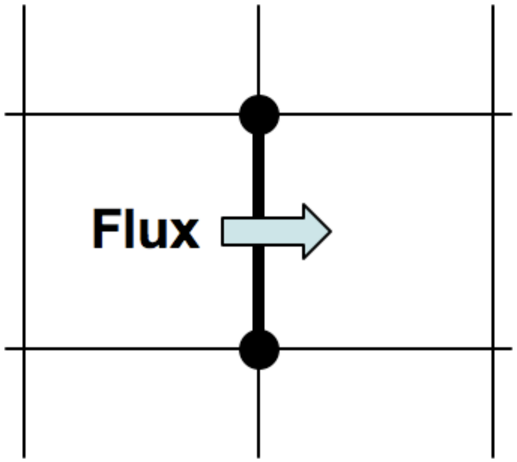
Forward Semi-Lagrangian



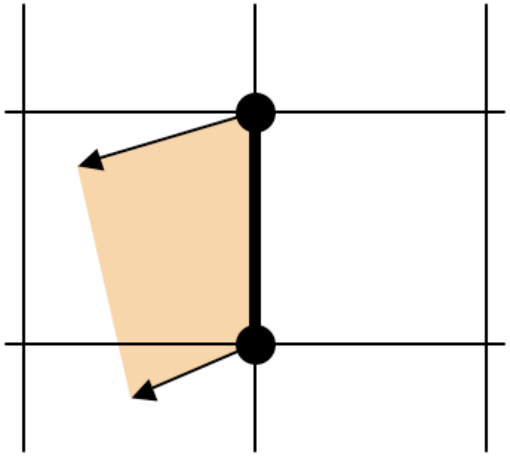
Backward Semi-Lagrangian



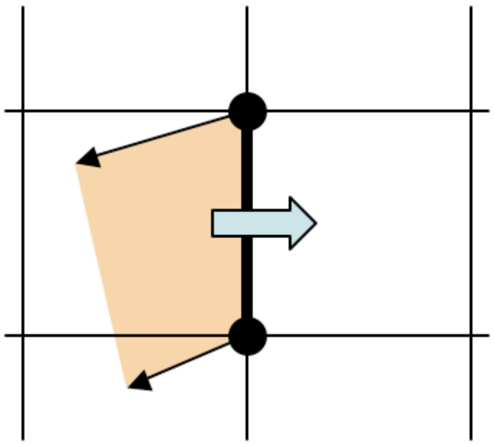
Flux-Form Lagrangian Transport



The flux across the highlighted edge is desired.

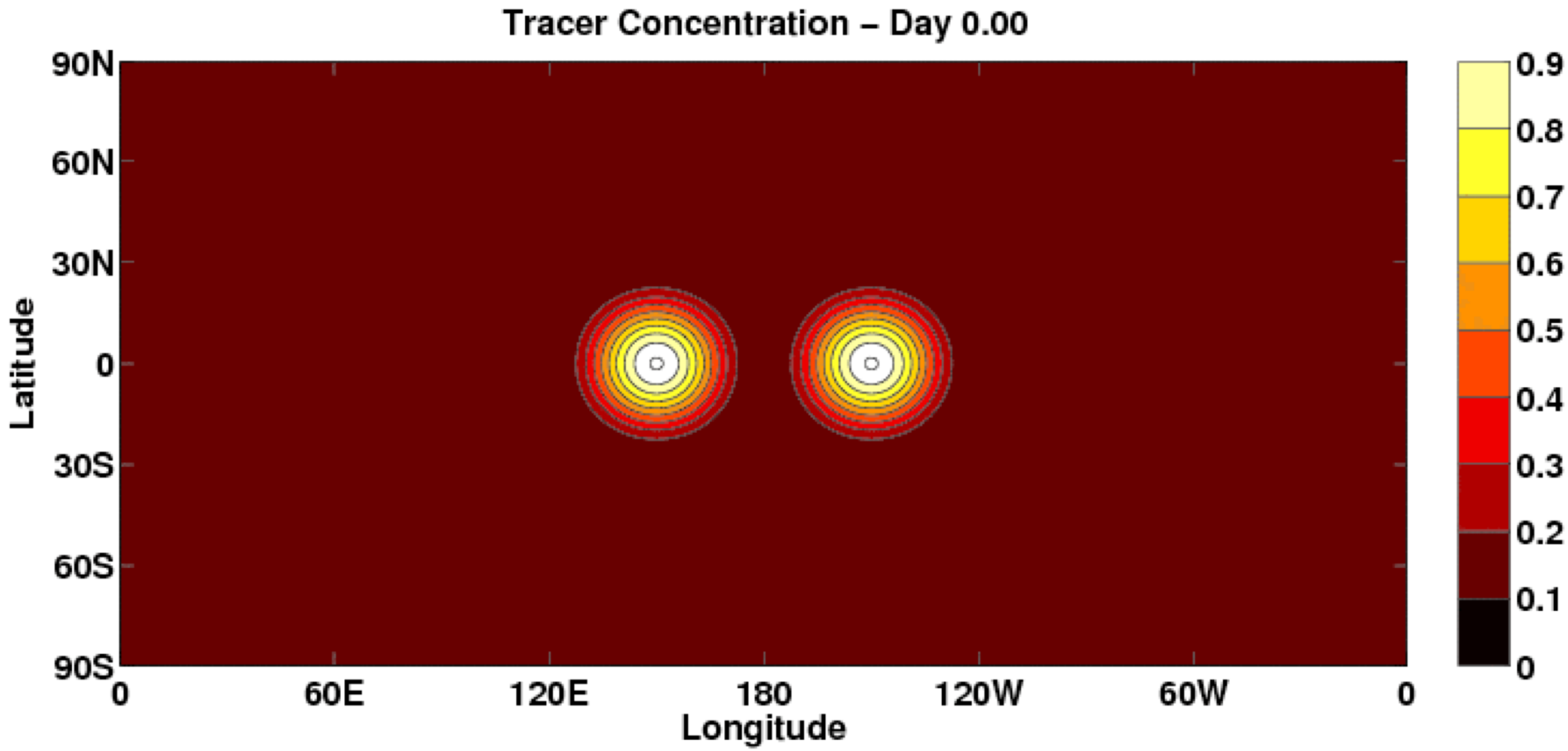


Step 1: Project velocity field backwards in time to obtain a "flux area."



Step 2: Integrate over the flux area to obtain the flux through the edge.

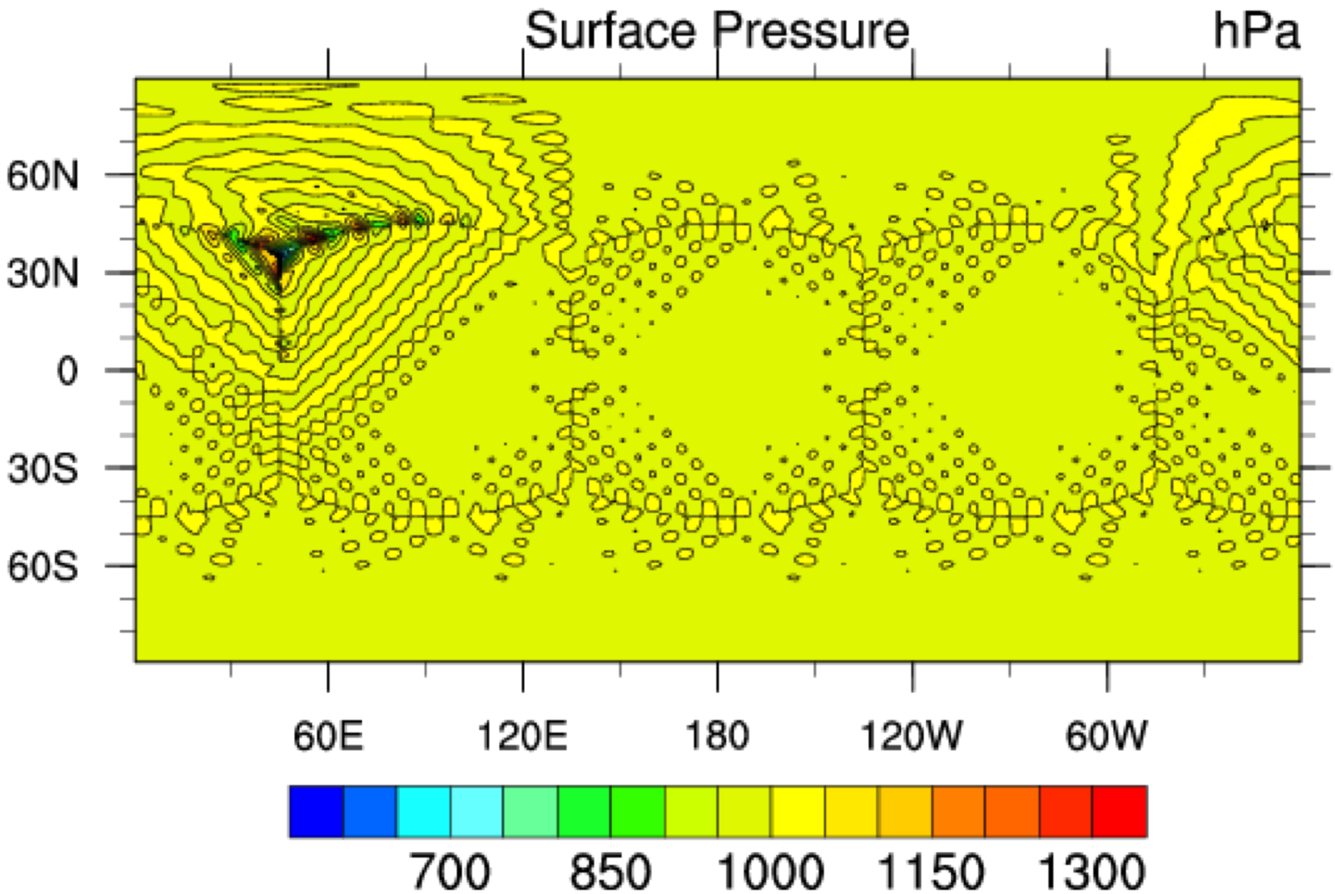
Deformational Flow Test



Stability

Introduction to Stability

Numerical instability in a high-frequency computational mode:



Introduction to Stability

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^n$$

Apply both time and space discretization:

- Recall definition of eigenvectors of B: If $\mathbf{v}$ is an eigenvector of B, then it satisfies $\mathrm{B}\mathbf{v} = \lambda\mathbf{v}$ where λ is the (complex) eigenvalue associated with $\mathbf{v}$.

Computational modes

- Theory: If B is well behaved, then it will have N eigenvector / eigenvalue pairs, where N is the number of free parameters.

$$\mathbf{q}^n = \sum_{i=1}^{N} a_i^n \mathbf{v}_i$$

Introduction to Stability

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^n$$

$\mathbf{v}_i$ eigenvectors of B, with associated eigenvalues λ_i.

$$\mathbf{q}^n = \sum_{i=1}^{N} a_i^n \mathbf{v}_i$$

- Substitute this solution into the update equation:

$$\mathbf{q}^{n+1} = \sum_{i=1}^{N} a_i^n \mathrm{B}\mathbf{v}_i$$

- Use properties of eigenvectors:

$$\mathbf{q}^{n+1} = \sum_{i=1}^{N} \lambda_i a_i^n \mathbf{v}_i$$

Introduction to Stability

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^n$$

$\mathbf{v}_i$ eigenvectors of B, with associated eigenvalues λ_i.

$$\mathbf{q}^n = \sum_{i=1}^{N} a_i^n \mathbf{v}_i \qquad \mathbf{q}^{n+1} = \sum_{i=1}^{N} a_i^{n+1} \mathbf{v}_i$$

where $a_i^{n+1} = \lambda_i a_i^n$

Each mode is amplified by its corresponding eigenvalue.

Introduction to Stability

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^n$$

$\mathbf{v}_i$ eigenvectors of B, with associated eigenvalues λ_i.

$$\mathbf{q}^n = \sum_{i=1}^{N} a_i^n \mathbf{v}_i \qquad \mathbf{q}^{n+1} = \sum_{i=1}^{N} a_i^{n+1} \mathbf{v}_i$$

where $a_i^{n+1} = \lambda_i a_i^n$

Take absolute values: $|a_i^{n+1}| = |\lambda_i| \; |a_i^n|$

What happens if

$|\lambda_i| > 1?$ $|\lambda_i| < 1?$

Introduction to Stability

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^{n}$$

$\mathbf{v}_i$ eigenvectors of B, with associated eigenvalues λ_i.

$|\lambda_i| > 1$ Instability! The corresponding computational mode will blow up.

$|\lambda_i| \leq 1$ Stable! The corresponding computational mode will either maintain its amplitude, or will decay with time (lose energy?)

Stability: An Example

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathsf{B}\mathbf{q}^n$$

Example: Forward Euler plus upwinding (first-order finite volume).

$$q_j^{n+1} = q_j^n + \frac{u\Delta t}{\Delta x}(q_j^n - q_{j-1}^n) \qquad \nu = \frac{u\Delta t}{\Delta x}$$

Corresponding evolution matrix:

$$\mathsf{B} = \begin{pmatrix} 1-\nu & & & \dots & \nu \\ \nu & 1-\nu & & & \\ & \nu & 1-\nu & & \\ & & \ddots & \ddots & \end{pmatrix}$$

Eigenvectors and eigenvalues:

$$(\mathbf{v}_k)_j = \exp(ijk) \qquad \lambda_k = 1 - \nu(1 + \exp(-ik))$$

Stability: An Example

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^n$$

Example: Forward Euler plus upwinding (first-order finite volume).

$$q_j^{n+1} = q_j^n + \frac{u\Delta t}{\Delta x}(q_j^n - q_{j-1}^n) \qquad \nu = \frac{u\Delta t}{\Delta x}$$

Eigenvectors and eigenvalues:

$$(\mathbf{v}_k)_j = \exp(ijk) \qquad \lambda_k = 1 - \nu(1 + \exp(-ik))$$

Absolute value of eigenvalues:

$$|\lambda_k|^2 = 1 - 2\nu(\nu - 1)(\cos(k) - 1)$$

Maximum eigenvalue:

$$\max_k |\lambda_k|^2 = 1 + 4\nu(\nu - 1)$$

Stability: An Example

Linear Update Equation

$$\mathbf{q}^{n+1} = \mathrm{B}\mathbf{q}^{n}$$

Example: Forward Euler plus upwinding (first-order finite volume).

$$q_j^{n+1} = q_j^n + \frac{u\Delta t}{\Delta x}(q_j^n - q_{j-1}^n) \qquad \nu = \frac{u\Delta t}{\Delta x}$$

Maximum eigenvalue:

$$\max_k |\lambda_k|^2 = 1 + 4\nu(\nu - 1)$$

Stable as long as

$$0 \leq \nu \leq 1$$

(CFL Condition)