High performance computing and numerical modeling

Volker Springel

Plan for my lectures

Heidelberg Institute for Theoretical Studies HITS

UNIVERSITÄT HEIDELBERG

43rd Saas Fee Course
Villars-Sur-Ollon, March 2013

Signal propagation with hyperbolic equations

THE ADVECTION EQUATION

Let's take the continuity equation and simplify it a bit by assuming constant velocity and 1 dimension:

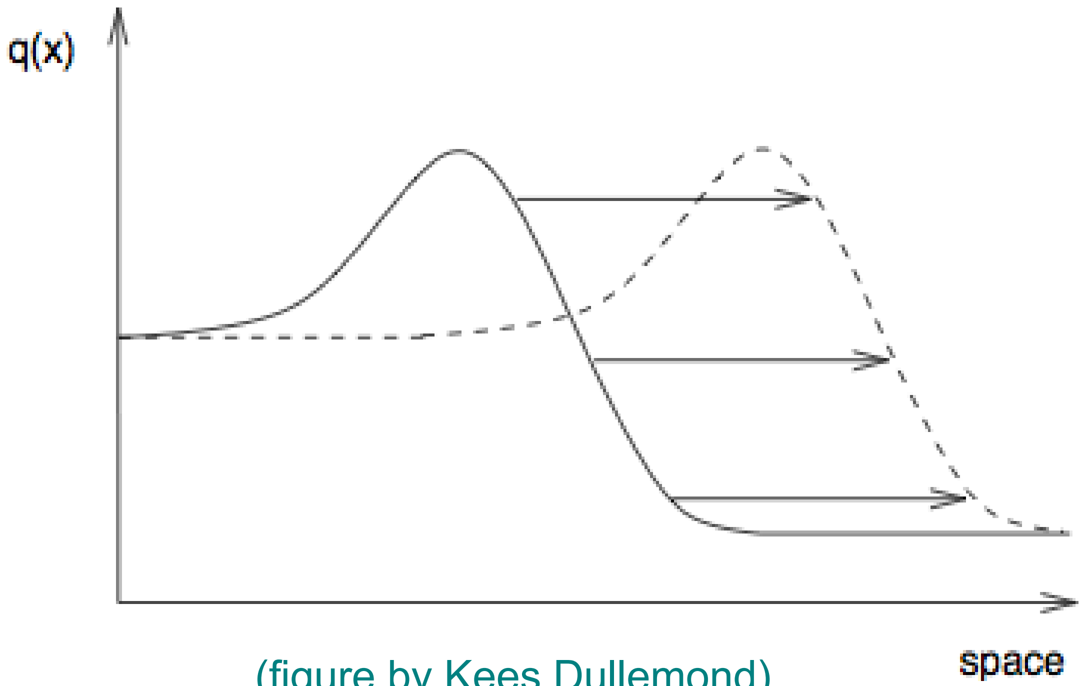
$$\frac{\partial \rho}{\partial t} + \nabla(\rho \mathbf{v}) = 0$$

We get a simple **advection equation**:

$$\partial_t q + u \partial_x q = 0$$

The solution is:

$$q(x,t) = q(x - ut, 0)$$

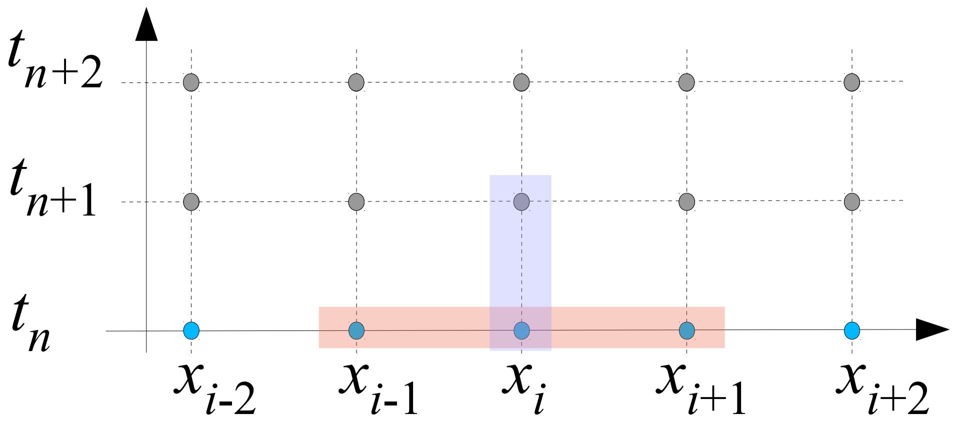


(figure by Kees Dullemond)

Straightforward finite difference discretization schemes for the advection problem can be tried....

CENTERED DIFFERENCE APPROCH FOR THE ADVECTION EQUATION

$$\partial_t q + u\partial_x q = 0$$



Centered difference discretization:

$$\frac{q_i^{n+1} - q_i^n}{t_{n+1} - t_n} + u\frac{q_{i+1}^n - q_{i-1}^n}{x_{i+1} - x_{i-1}} = 0$$

This yields the update formula:

$$q_i^{n+1} = q_i^n - \frac{\Delta t}{2\Delta x}u(q_{i+1}^n - q_{i-1}^n)$$

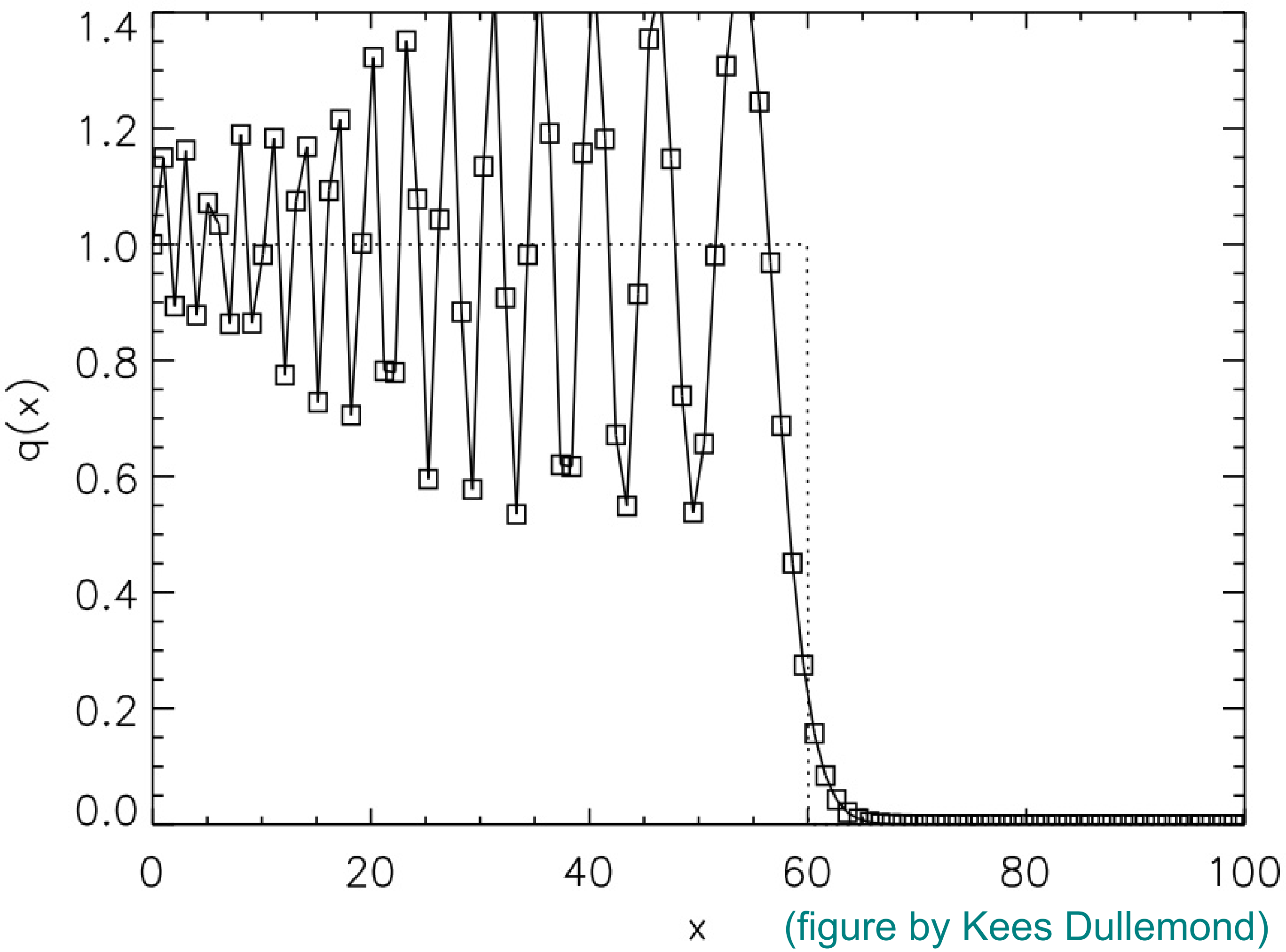
Such attempts tend to fail, often being spectacularly unstable

CENTERED DIFFERENCE ADVECTION OF A STEP FUNCTION

Initial conditions:

$$q(x, t = t_0) = \begin{cases} 1 & \text{for} \quad x \leq 30 \\ 0 & \text{for} \quad x > 30 \end{cases}$$

Evolved state:



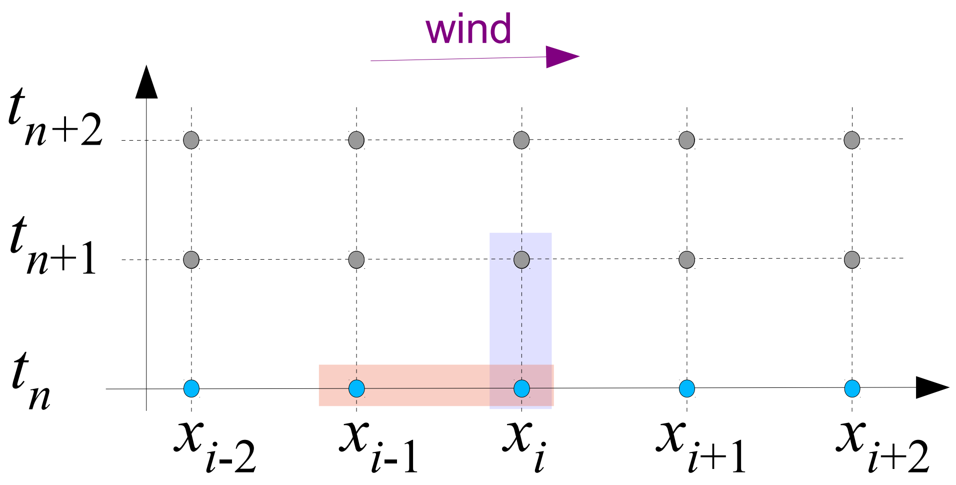
(figure by Kees Dullemond)

Upwinding can render the simple advection algorithm stable

UPWIND DIFFERENCING APPROACH APPLIED TO THE ADVECTION EQUATION

Let's assume:

$$u > 0$$



Now modify the finite difference approximation to be skewed to the upwind side:

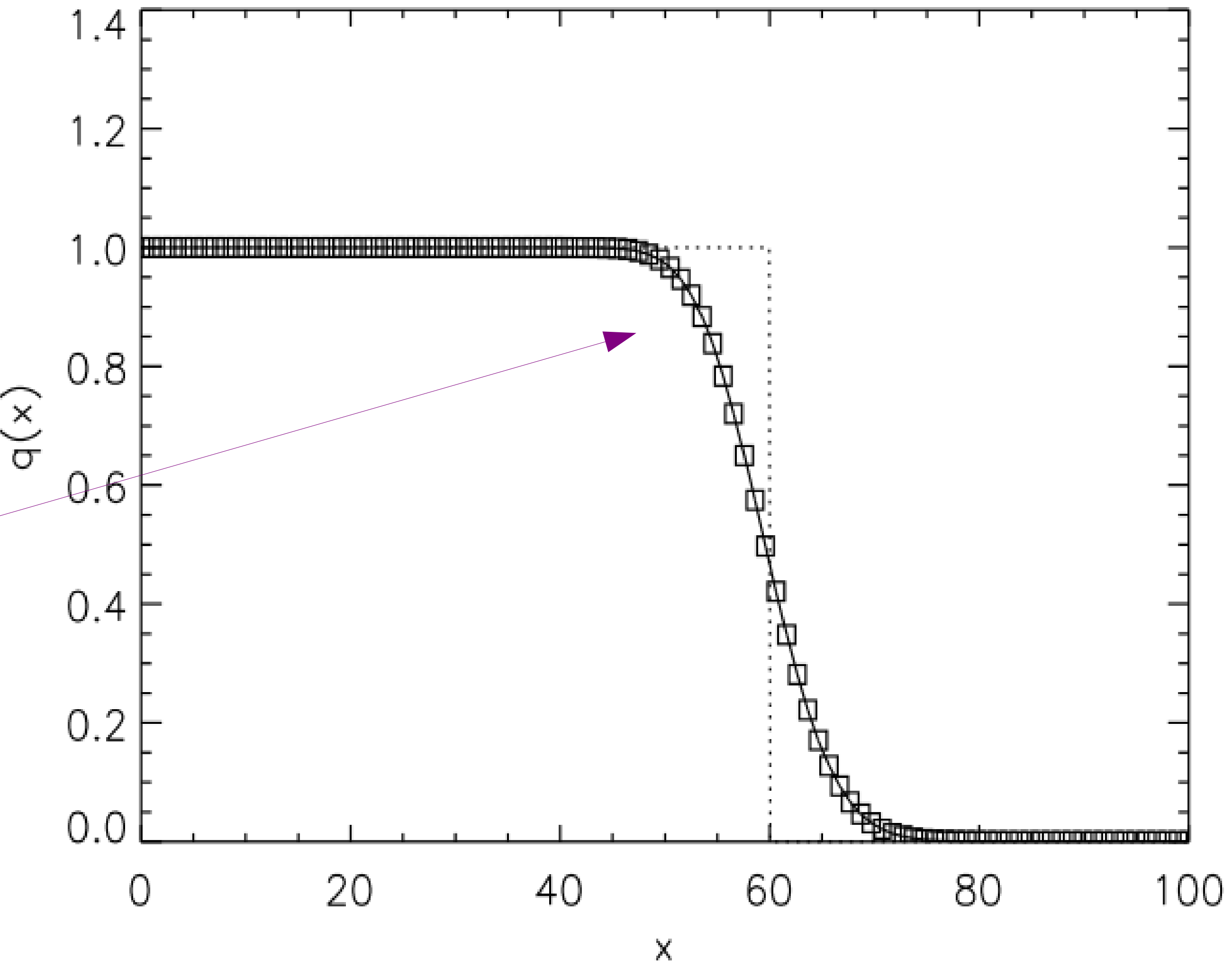
$$\frac{q_i^{n+1} - q_i^n}{t_{n+1} - t_n} + u\frac{q_i^n - q_{i-1}^n}{x_i - x_{i-1}} = 0$$

This yields the update formula:

$$q_i^{n+1} = q_i^n - \frac{\Delta t}{\Delta x} u(q_i^n - q_{i-1}^n)$$

Using upwind differencing, the simple update formula yields stable and robust results

UPWIND ADVECTION OF A STEP FUNCTION



However,
there is considerable
“numerical diffusion”

(figure by Kees Dullemond)

Effectively, upwind differencing adds some diffusion to the central differencing scheme

ANALYSING UPWIND DIFFERENCING

$$\frac{q_i - q_{i-1}}{\Delta x} = \frac{q_{i+1} - q_{i-1}}{2\Delta x} - \Delta x \frac{q_{i+1} - 2q_i + q_{i-1}}{2\Delta x^2}$$

upwind difference

centered difference

diffusion term with diffusion constant

$$D = \frac{\Delta x u}{2}$$

- Once a bit of diffusion is added to the centered scheme, it becomes stable!

- The diffusion vanishes in the limit of $\Delta x \rightarrow 0$.
  It is purely **numerical diffusion**.

The Euler equations as a set of hyperbolic conservation laws

FLUX FORMULATION OF THE EULER EQUATIONS

State vector

$$\boxed{\mathbf{U} = \begin{pmatrix} \rho \\ \rho\mathbf{v} \\ \rho e \end{pmatrix}}$$

$$e = u + \mathbf{v}^2/2$$

Flux vector

Euler equations

$$\boxed{\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0}$$

$$\boxed{\mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\mathbf{v}^T + P \\ (\rho e + P)\mathbf{v} \end{pmatrix}}$$

$$P = (\gamma - 1)\rho u$$

Finite volume discretization:

Cell averages

$$\boxed{\mathbf{Q}_i = \begin{pmatrix} M_i \\ \mathbf{p}_i \\ E_i \end{pmatrix} = \int_{V_i} \mathbf{U}\,\mathrm{d}V}$$

Evolution equation

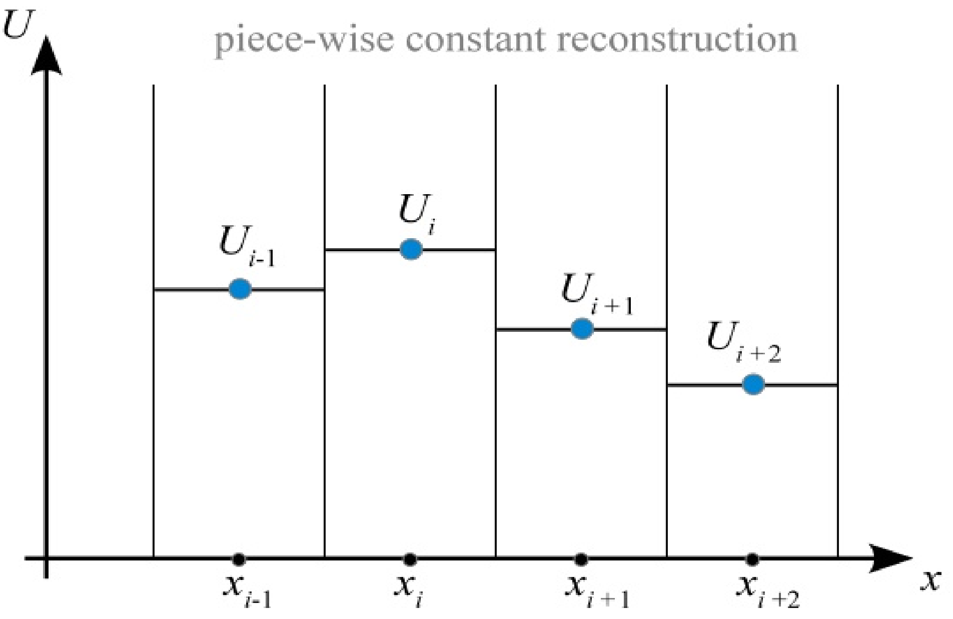
$$\boxed{\frac{\mathrm{d}\mathbf{Q}_i}{\mathrm{d}t} = -\int_{\partial V_i} \mathbf{F}(\mathbf{U})\ \mathrm{d}\mathbf{n}}$$

But how to compute the fluxes through cell surfaces?

The timesteps in many finite volume scheme can be viewed as a sequence of Reconstruct-Evolve-Average (REA) steps

REA SCHEMES

Reconstruct: Using the cell-averaged quantities, determine the run of these quantities everywhere in the cell.

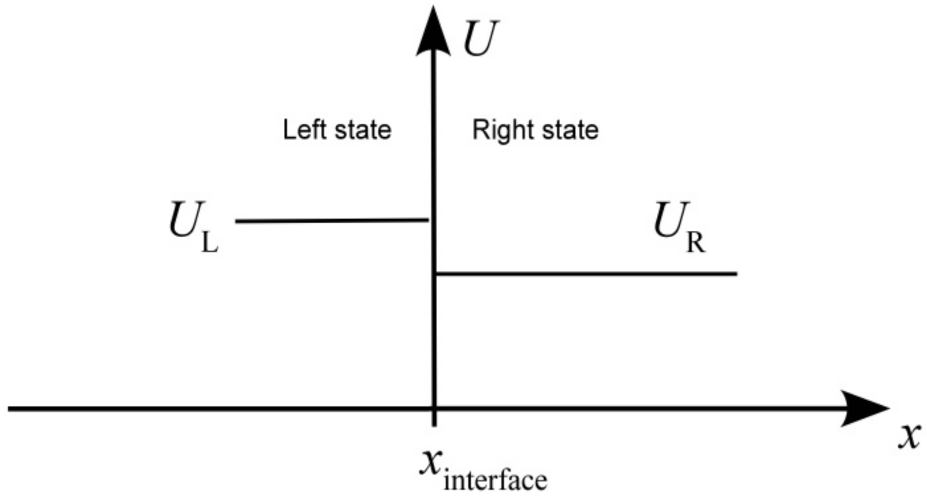


Evolve: The reconstructed state is then evolved forward in time by Δt. In **Godunov's approach**, this is done by treating each cell interface as a piece-wise constant initial value problem which is solved with a **Riemann solver**.

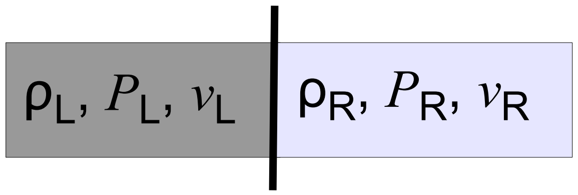
Average: The evolved solution of the previous step is averaged at time Δt to compute new average states U^{n+1} in each cell in a conservative fashion. Then the cycle repeats.

The Riemann problem as basis for Godunov schemes

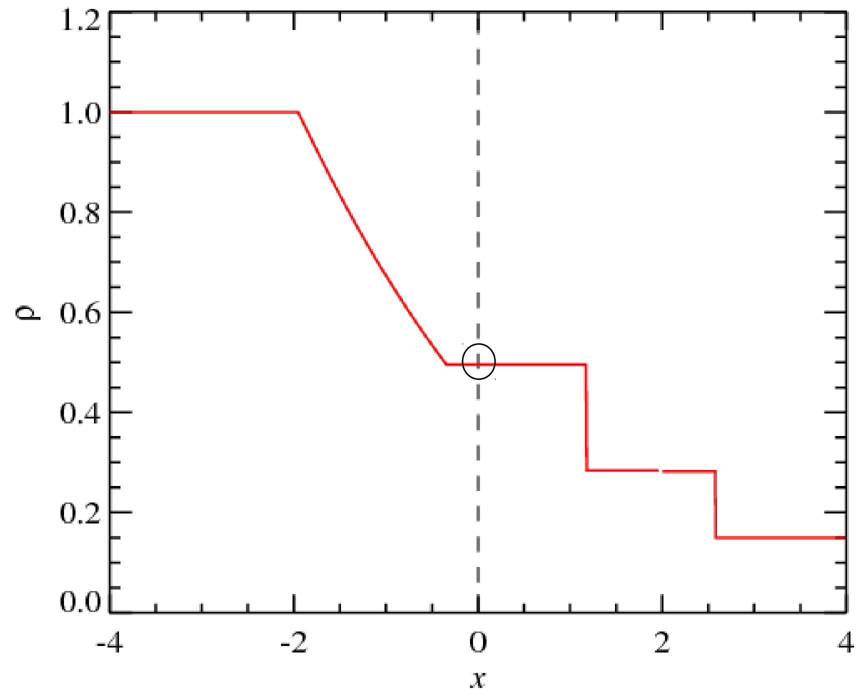
CALCULATION OF THE GODUNOV FLUX

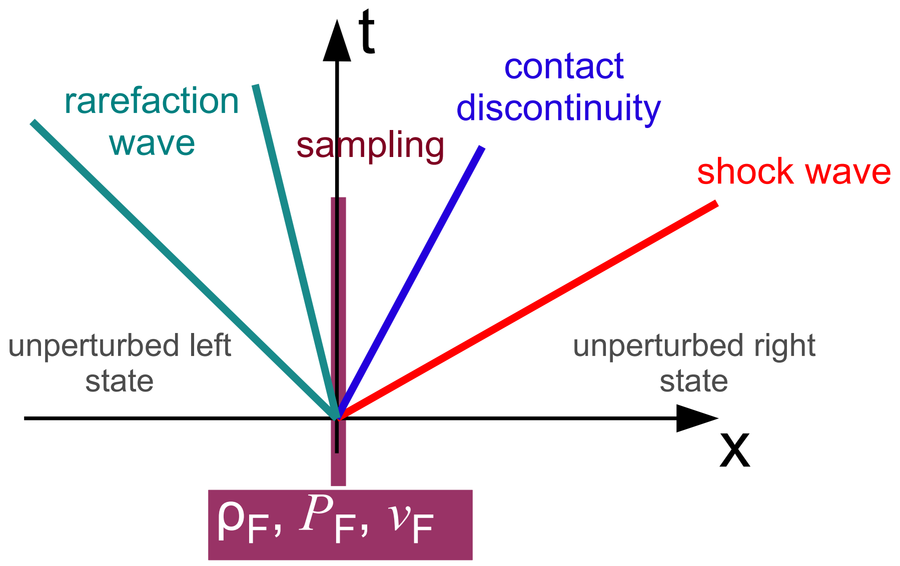


Assume piece-wise constant left and right states for the fluid



Calculate the self-similar time evolution **(Riemann problem)**





Sample the solution along x/ t=0, which yields the **Godunov flux**

Fortunately, the **averaging step** doesn't have to be done explicitly

OBTAINING THE AVERAGED STATE

Let's integrate the Euler equation over a cell and in time:

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathrm{d}x \int_{t_n}^{t_{n+1}} \mathrm{d}t \left(\partial_t \mathbf{U} + \partial_x \mathbf{F}\right) = 0$$

This yields:

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}(x, t_{n+1})\mathrm{d}x - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}(x, t_n)\mathrm{d}x + \int_{t_n}^{t_{n+1}} \mathbf{F}(x_{i+\frac{1}{2}}, t)\mathrm{d}t - \int_{t_n}^{t_{n+1}} \mathbf{F}(x_{i-\frac{1}{2}}, t)\mathrm{d}t = 0$$

But the Riemann solution is self-similar, with the Godunov flux being independent of time:

$$\mathbf{F}(x_{i+\frac{1}{2}}, t) = \mathbf{F}^\star_{i+\frac{1}{2}} \qquad \mathbf{F}^\star = \mathbf{F}_{\mathrm{Riemann}}(\mathbf{U}_L, \mathbf{U}_R)$$

Hence the **new spatially average state** is simply given by:

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \frac{\Delta t}{\Delta x}\left(\mathbf{F}^\star_{i-\frac{1}{2}} - \mathbf{F}^\star_{i+\frac{1}{2}}\right)$$

Two issues are left open:

- How can this be extended to multiple dimensions?
- How can we reach an accuracy higher than 1st order in space and time?

Operator splitting can be used to extend the scheme to multiple dimensions

FROM 1D TO 3D WITH LITTLE EFFORT

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0$$

Let's write out the full Euler for Cartesian coordinates:

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho u v \\ \rho u w \\ \rho u(\rho e + P) \end{pmatrix} + \partial_y \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + P \\ \rho v w \\ \rho v(\rho e + P) \end{pmatrix} + \partial_z \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + P \\ \rho w(\rho e + P) \end{pmatrix} = 0,$$

This can be written also in the following form:

$$\mathbf{v} = (u, v, w)$$

$$\partial_t \mathbf{U} + \partial_x \mathbf{F} + \partial_y \mathbf{G} + \partial_z \mathbf{H} = 0.$$

Now let's split up the equation into three separate equations:

$\partial_t \mathbf{U} + \partial_x \mathbf{F} = 0$ $\quad \mathcal{X}(\Delta t)$ generates evolution under F

$\partial_t \mathbf{U} + \partial_y \mathbf{G} = 0$ $\quad \mathcal{Y}(\Delta t)$

$\partial_t \mathbf{U} + \partial_z \mathbf{H} = 0$ $\quad \mathcal{Z}(\Delta t)$

Approximate solution can be obtained as:

$$\boxed{\mathbf{U}^{n+1} \simeq \mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n}$$

The sequence of dimensionally split sweeps must be alternated to reach higher-order accuracy in time

TIME-ADVANCE IN DIMENSIONALLY SPLIT SCHEMES

Simple operator split in two dimensions:

$$\mathbf{U}^{n+1} \simeq \mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n$$

For second-order accuracy in time, symmetrize the action of the operators:

$$\mathbf{U}^{n+1} = \frac{1}{2}[\mathcal{X}(\Delta t)\mathcal{Y}(\Delta t) + \mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)]\mathbf{U}^n$$

One may also, e.g., use:

$$\mathbf{U}^{n+1} = \mathcal{X}(\Delta t/2)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t/2)\mathbf{U}^n$$

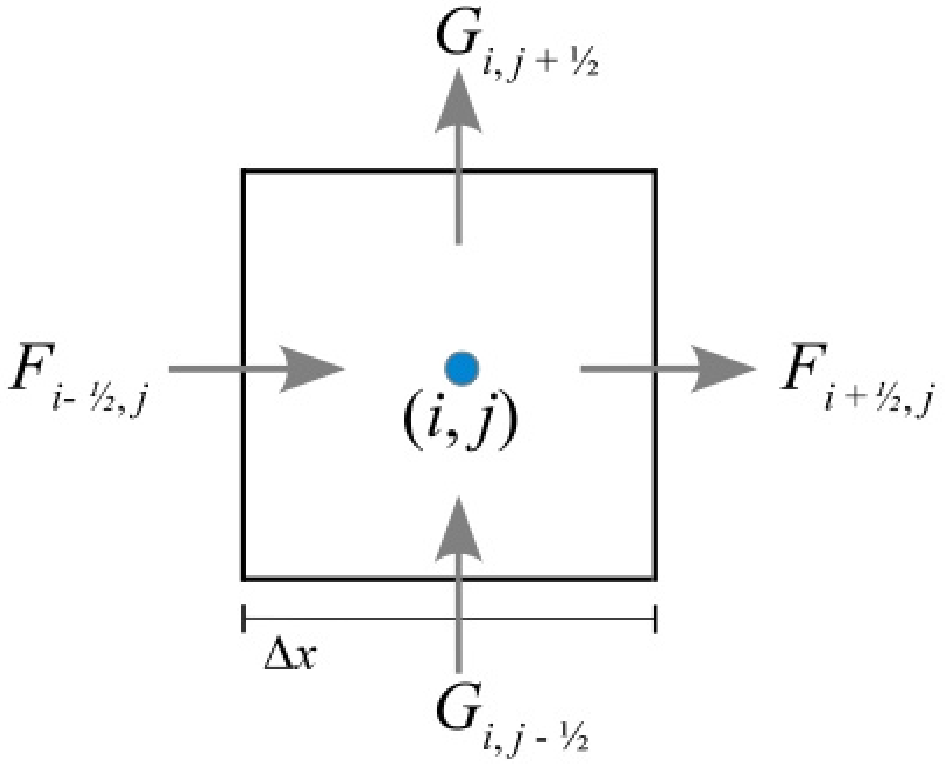
This readily generalizes to three dimensions:

$$\mathbf{U}^{n+2} = \mathcal{X}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{Z}(\Delta t)\mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n$$

$$\mathbf{U}^{n+1} = \mathcal{X}(\Delta t/2)\mathcal{Y}(\Delta t/2)\mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t/2)\mathcal{X}(\Delta t/2)\mathbf{U}^n$$

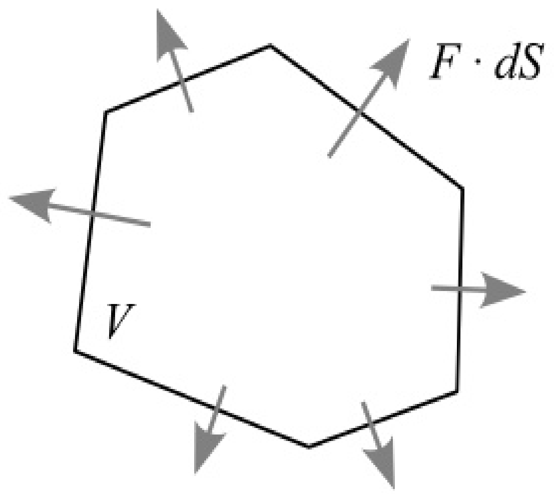
One can also calculate the fluxes directly in multi-D and arrive at an *unsplit* scheme

UNSPLIT FINITE-VOLUME UPDATES



Unstructured mesh case

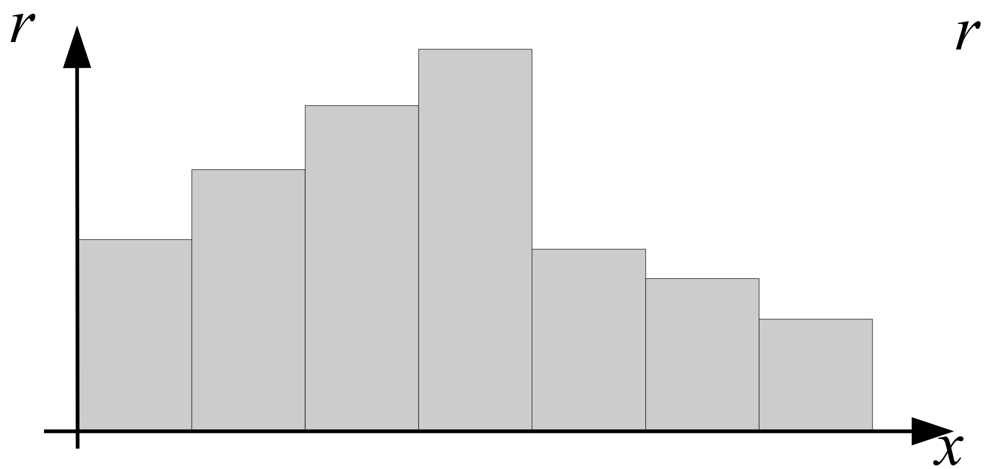
$$\mathbf{U}^{n+1} = \mathbf{U}^n - \frac{\Delta t}{V}\int \mathbf{F}\cdot d\mathbf{S}$$

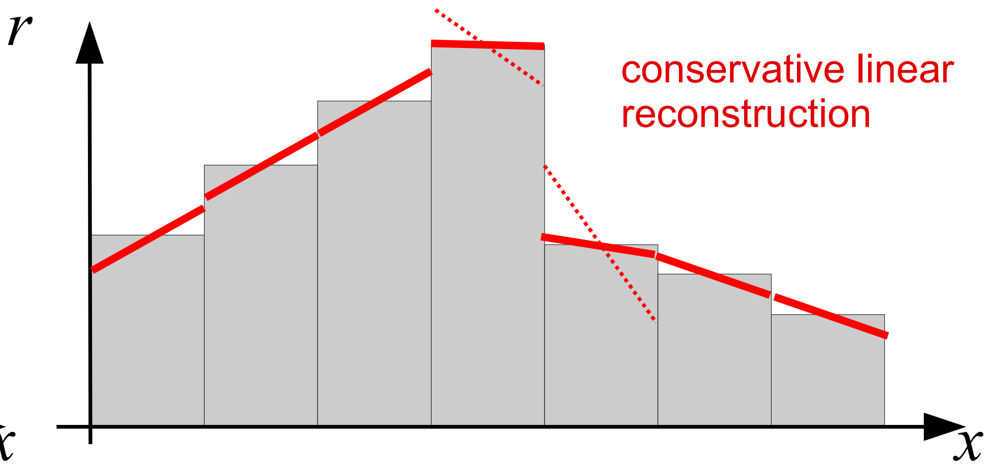


$$U_{i,j}^{n+1} = U_{i,j}^n + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i-\frac{1}{2},j} - \mathbf{F}_{i+\frac{1}{2},j}\right) + \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j-\frac{1}{2}} - \mathbf{G}_{i,j+\frac{1}{2}}\right)$$

To achieve second-order accuracy, one can use a **piece-wise linear** reconstruction

LINEAR RECONSTRUCTION AND GRADIENT LIMITATION





Slope limiting procedure:

$$\langle \nabla \phi \rangle_i' = \alpha_i \langle \nabla \phi \rangle_i$$

- Needed to prevent creation of new extrema, preventing spurious oscillations
- Reduces the order of the scheme at discontinuities

Example slope limiter:

$$\alpha_i = \min(1, \psi_{ij})$$

$$\psi_{ij} = \begin{cases} (\phi_i^{\max} - \phi_i)/\Delta\phi_{ij} & \text{for} \quad \Delta\phi_{ij} > 0 \\ (\phi_i^{\min} - \phi_i)/\Delta\phi_{ij} & \text{for} \quad \Delta\phi_{ij} < 0 \\ 1 & \text{for} \quad \Delta\phi_{ij} = 0 \end{cases}$$

$$\phi_i^{\max} = \max(\phi_j) \qquad \phi_i^{\min} = \max(\phi_j)$$

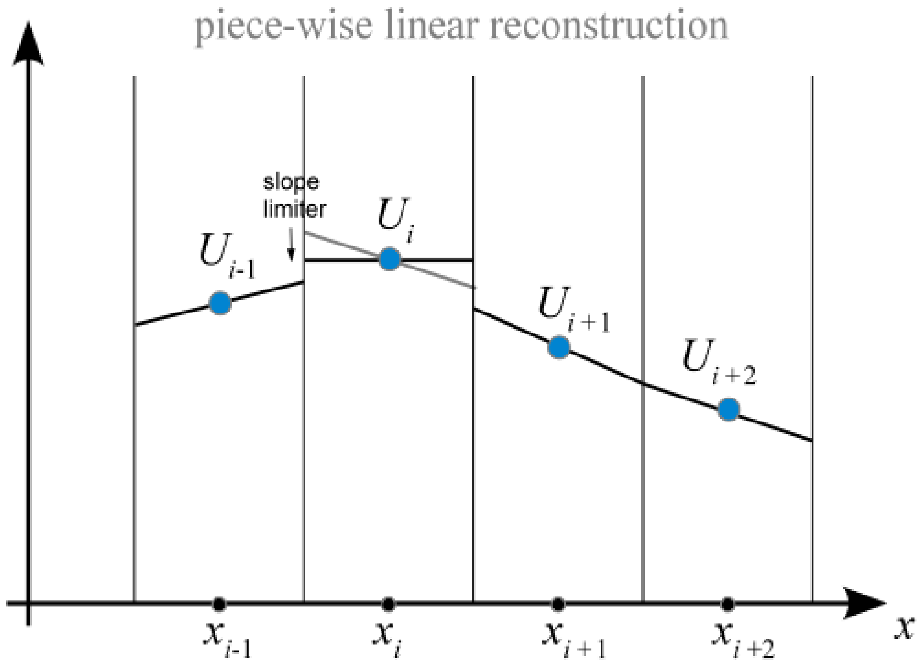
$$\Delta\phi_{ij} = \langle \nabla \phi \rangle_i \cdot (\boldsymbol{f}_{ij} - \boldsymbol{s}_i)$$

The gradients can be used to predict the fluid state directly at the interfaces

LINEAR EXTRAPOLATION TO CELL BOUNDARIES

Spatial extrapolation:

$$\begin{aligned}\rho^L_{i+\frac{1}{2}} &= \rho_i + (\nabla\rho)_i \frac{\Delta x}{2}, \\ \rho^R_{i+\frac{1}{2}} &= \rho_{i+1} - (\nabla\rho)_{i+1} \frac{\Delta x}{2}\end{aligned}$$



Temporal extrapolation to mid-step:

$$\begin{aligned}\rho^L_{i+\frac{1}{2}} &= \rho_i + (\nabla\rho)_i \frac{\Delta x}{2} + \left(\frac{\partial\rho}{\partial t}\right)_i \frac{\Delta t}{2} \\ \rho^R_{i+\frac{1}{2}} &= \rho_{i+1} - (\nabla\rho)_{i+1} \frac{\Delta x}{2} + \left(\frac{\partial\rho}{\partial t}\right)_{i+1} \frac{\Delta t}{2}\end{aligned}$$

This combined extrapolation needs to be done for the full fluid state for second-order accuracy:

$$\begin{aligned}\mathbf{U}^L_{i+\frac{1}{2}} &= \mathbf{U}_i + (\partial_x\mathbf{U})_i \frac{\Delta x}{2} + (\partial_t\mathbf{U})_i \frac{\Delta t}{2} \\ \mathbf{U}^R_{i+\frac{1}{2}} &= \mathbf{U}_{i+1} - (\partial_x\mathbf{U})_{i+1} \frac{\Delta x}{2} + (\partial_t\mathbf{U})_{i+1} \frac{\Delta t}{2}\end{aligned}$$

But how do we get the time derivative?

The MUSCL-Hancock scheme is a particularly simple second-order extension of Godunov's method

GETTING THE TIME EXTRAPOLATION FROM THE SPATIAL GRADIENTS

Euler equation:

$$\partial_t \mathbf{U} = -\partial_x \mathbf{F}(\mathbf{U}) = -\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\,\partial_x \mathbf{U} = -\mathbf{A}(\mathbf{U})\,\partial_x \mathbf{U}$$

This means we can estimate the time derivate based on the current state of a cell and the spatial gradient estimates:

$$(\partial_t \mathbf{U})_i = -\mathbf{A}(\mathbf{U}_i)\,\underbrace{(\partial_x \mathbf{U})_i}_{\text{gradient estimate}}.$$

MUSCL-Hancock prediction:

$$\boxed{\begin{aligned} \mathbf{U}^L_{i+\frac{1}{2}} &= \mathbf{U}_i + \left[\frac{\Delta x}{2} - \frac{\Delta t}{2}\mathbf{A}(\mathbf{U}_i)\right](\partial_x \mathbf{U})_i, \\ \mathbf{U}^R_{i+\frac{1}{2}} &= \mathbf{U}_{i+1} + \left[-\frac{\Delta x}{2} - \frac{\Delta t}{2}\mathbf{A}(\mathbf{U}_{i+1})\right](\partial_x \mathbf{U})_{i+1} \end{aligned}}$$

- At discontinuities, the slope limiter will suppress the derivative, making the scheme automatically first order
- We recall also Godunov's theorem, according to which *any linear algorithm that does not produce new extrema (aka TVD) can be at most first order*.

Thermal energy, temperature and entropy are treated in a fundamentally different way in Eulerian codes compared to SPH

QUESTIONS ABOUT ENTROPY AND FRIENDS IN EULERIAN CODES

Why is there no artificial viscosity needed in the MUSCL scheme?

How is entropy produced?

Some notes may help to clarify this:

- Temperature is defined as difference between total energy and kinetic energy (note: *cold flow problem*)
- Entropy is not followed explicitly – one simply assigns the entropy according to what the conservation laws dictate
- The break-down of the differential form of the equations at shocks is circumvented by using the integrated flux-form of the equations

Approximate Riemann solvers are often used for computational efficiency

EXAMPLE: ROE SOLVER FOR ISOTHERMAL GAS IN TWO DIMENSIONS

2d state vector for isothermal gas:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \end{pmatrix}$$

The x-sweep part of the equations is:

$$\partial_t \mathbf{U} + \partial_x \mathbf{F} = 0 \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + \rho c_s^2 \\ \rho u v \end{pmatrix}$$

We want to solve for the x-sweep:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{i-\frac{1}{2}} - \mathbf{F}_{i+\frac{1}{2}}\right]$$

To derive an approximate Riemann solver, we want to linearize the equation:

$$\partial_t \mathbf{U} + \mathbf{A}(\mathbf{U})\,\partial_x \mathbf{U} = 0 \qquad \mathbf{A} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$$

Roe's suggestion:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{A}}(\mathbf{U}_L, \mathbf{U}_R).$$

$$\tilde{\mathbf{A}} = \begin{pmatrix} 0 & 1 & 0 \\ c_s^2 - \tilde{u}^2 & 2\tilde{u} & 0 \\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} \end{pmatrix}$$

The Jacobian is approximated as being constant and only a function of left and right states.

$$\tilde{u} = (\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R)/(\sqrt{\rho_L} + \sqrt{\rho_R})$$

Eigenstructure of the linearized isothermal equations:

Eigenvalues:

$$\lambda_1 = \tilde{u} - c_s; \qquad \lambda_2 = \tilde{u} + c_s \qquad \lambda_3 = \tilde{u}$$

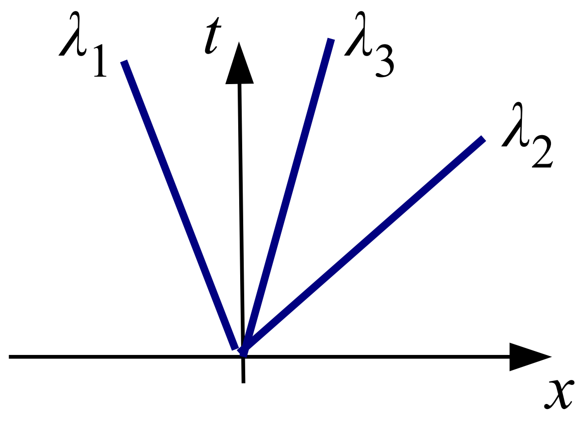
Eigenvectors:

$$\mathbf{K}_1 = \begin{pmatrix} 1 \\ \tilde{u} - c_s \\ \tilde{v} \end{pmatrix} \qquad \mathbf{K}_2 = \begin{pmatrix} 1 \\ \tilde{u} + c_s \\ \tilde{v} \end{pmatrix} \qquad \mathbf{K}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Expand the jump in states in the eigenbasis:

$$\Delta \mathbf{U} = (u_1, u_2, u_3) = \mathbf{U}_R - \mathbf{U}_L$$

$$\Delta \mathbf{U} = \textstyle\sum_i \alpha_i \mathbf{K}_i;$$



Linearized solution is easily obtained.

Roe's flux:

$$\mathbf{F}^* = \frac{1}{2}(\mathbf{F}_L + \mathbf{F}_R) - \frac{1}{2}\sum_i \alpha_i \left|\lambda_i\right| \mathbf{K}_i$$