Outline:

• Basic definitions and methods for transport-dominated problems

• Particle distortion: problems and solutions

• Particle remeshing and adaptive particle methods
Many fluid and solid mechanics dynamical systems can be described by conservation laws

\[
\frac{\partial U}{\partial t} + \text{div} (a : U) + AU = F
\]

U is the advected/conserved quantity(ies), a is the advection field, A measures absorption (or amplification) and F are external forces or additional terms.

U can be a scalar (density, passive markers, level set function ..) or a vector (vorticity, momentum ..) - examples follow

\text{div} (a:U) \text{ is the vector with i-component } \frac{\partial (a_j u_i)}{\partial x_j}

a (and A, F) can be given (\rightarrow \text{ linear equations}) or functions of U (nonlinear equations)
Conservation law in the sense that

\[
\frac{d}{dt} \left[ \int_{\Omega(t)} U \, dx \right] + \int_{\Omega(t)} A U \, dx = \int_{\Omega(t)} F \, dx
\]

where \( \Omega(t) = a(\Omega(t), t) \)

Idea of particle methods is

- to concentrate mass on simple sets: points (sometimes curves), called particles
- to follow these particles along flow trajectories -> solve differential equations
- to increment mass of these particles to account for absorption and/or force terms
- to recover needed information from particles

Comparison with a recent work of Smereka about approximation of delta-functions: computation of the arc-length of an ellipse

\[
(x, y) = a^2 x^2 + b^2 y^2
\]

1

\[
L_0 = \int_\mathcal{D} (\varepsilon)^{\mathcal{D}} \nabla U \cdot \nabla \psi \, d\mathcal{D}
\]

\[
t = 0.3125 + s L_1.03125 \sin \frac{\pi}{2} s L_1.03125
\]

\[
y(s, t) = 0.125 L_0.03125 + s L_1.03125
\]

\[
\theta^\alpha(a^2 x^2 + b^2 y^2)
\]

\[
\sum_{i=1}^{N} \left| \mathcal{R}_i \right|^2 = 1
\]
\[ U(x, t) = \sum_p \Xi_p \delta(x - x_p(t)) \quad \Xi_p = U_p v_p \]

\[ \text{U}_p \text{ local values of } U, \text{ v}_p \text{ local volumes} \]

\[ \frac{dx_p}{dt} = a(x_p, t) \]

\[ \frac{d\Xi_p}{dt} + A(x_p, t)\Xi_p = F(x_p(t)) \]

\[ \frac{dv_p}{dt} = \text{div}a(x_p, t)v_p \]
Consequence:

• conservation properties
• localization of computational effort
• no projection of the original equation involved
  (unlike spectral, FEM, FV)
Essential feature: transport of Dirac masses directly translates conservative formulation

Four classical + one less classical examples:

• Vlasov-Maxwell equations
• Gas dynamics
• Incompressible Navier-Stokes equations
• Interface capturing and variable density flows
• Linear elasticity
Example 1: Vlasov-Maxwell equations

Distribution function for ions (or electrons) subject to electric and magnetic fields

\[ f = f(x, v, t) \in [0, 1] \quad E = E(x, t) \quad B = B(x, t) \]

Conservation of charge:

\[ \frac{\partial f}{\partial t} + (v \cdot \nabla_x)f + ((E + v \times B) \cdot \nabla_v)f = 0 \]

\[ U = \begin{bmatrix} v \\ E + v \times B \end{bmatrix} \quad \text{satisfies} \quad \text{div}_{x,v} U = 0 \]

hence, conservative advection equation for \( f \) with velocity field \( U \)
Example 2: Compressible flows and SPH methods

\[ \mathbf{U} = (\rho, \rho \mathbf{u}, \rho E) \] where \( \rho \) is the density, \( \mathbf{u} \) is the velocity, \( E \) the energy.

\[ \mathbf{a} = \mathbf{u}, \, \mathbf{A} = 0, \, \mathbf{F} = (0, \text{pressure and diffusion terms}). \]

In 1 space dimension:

\[
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho \mathbf{u})}{\partial x} = 0
\]

\[
\frac{\partial (\rho \mathbf{u})}{\partial t} + \frac{\partial (\rho \mathbf{u} \mathbf{u})}{\partial x} = \frac{\partial p}{\partial x}
\]

\[
\frac{\partial (\rho E)}{\partial t} + \frac{\partial (\rho E \mathbf{u})}{\partial x} = \frac{\partial (p \mathbf{u})}{\partial x}
\]

supplemented by \( p = p(\rho) \).

Particles \( x_p \) carry \( \mathbf{U} = (\rho, (\rho \mathbf{u}), (\rho E)) \):

\[
\rho(x) = \sum_p \alpha_p \delta(x - x_p) \quad \rho \mathbf{u}(x) = \sum_p \beta_p(t) \delta(x - x_p) \quad \rho E(x) = \cdots
\]
Example 3: Vorticity conservation for incompressible flows

Incompressible Navier-Stokes equations in velocity-pressure formulation

\[ \frac{\partial u}{\partial t} + (u \cdot \nabla)u + \nabla p = \nu \Delta u \quad \text{div}u = 0 \]

\[ \omega = \nabla \times u \quad \frac{\partial \omega}{\partial t} + \text{div}(u\omega) - [\nabla u]\omega = \nu \Delta \omega \]

\[ \omega = \sum_p \alpha_p \delta(x - x_p) \quad \frac{dx_p}{dt} = u(x_p) \quad \frac{d\alpha_p}{dt} = [\nabla u(x_p)]\alpha_p + \text{diffusion} \]
Other types of particles: vorticity contours and filaments

Vorticity filament: a curve that concentrates a vortex tube

\[ \iint_{\Sigma_1} \omega \cdot n_1 \, dx = - \iint_{\Sigma_2} \omega \cdot n_2 \, dx \]

Circulation of vortex tube
Vortex filament: particle with support an oriented curve (centerline of vortex tube) and strength = its circulation.

Helmholtz theorem: the circulation is preserved in time

In mathematical words, filament of circulation $\Gamma$ supported by curve $F$ parameterized by $\gamma(\xi)$:

$$\langle \mu, \varphi \rangle = \alpha \oint_{F} \varphi(\gamma(\xi)) \cdot \frac{\partial \gamma}{\partial \xi} \, d\xi$$

Same notions apply for contours of 2D vortex patches (method of contour dynamics):

$$\nabla \times \omega \quad \text{a 1D Dirac mass, tangent to the contour, satisfying a transport-stretching equation identical to the 3D vorticity equation}$$
Important issues in particle methods:

recover fields or non advection-related terms, from particles,

for example:

• velocity in vorticity particle methods, electric/magnetic fields in plasma
• pressure gradients in compressible flows
• diffusion
• surface tension (curvature) in level set particle methods for multiphase flows

General approach: mollify particles

\[
U(x, t) = \sum_p \Xi_p \delta(x - x_p(t)) \quad \longrightarrow \quad \sum_p \Xi_p W_h(x - x_p(t))
\]

\[
W_h(x) = h^{-d}W(x/h) \quad \int W(x) \, dx = 1, \int x^\alpha W(x) \, dx = 0 \quad |\alpha| = 1, 2, \ldots
\]
Examples of field evaluation

Velocity of vortex particles:

Biot-Savart law, unbounded flow, rest at infinity

\[ u = K \ast \omega = \int K(x - y) \times \omega(y) \, dy \quad ; \quad K = \nabla(1/4\pi|x|) \]

\[ K_\varepsilon = K \ast \zeta_\varepsilon \quad \zeta_\varepsilon = W_h \cdots \]

\[ \frac{dx_p}{dt} = \sum_q \alpha_q K_\varepsilon (x_p - x_q) \]

Pressure gradient in compressible flows

\[ \nabla \Pi(x) \sim \sum_p \Pi_q \nabla_h (x - x_p) \]
Diffusion in viscous flows or for diffusive scalar:

based on integral representation of diffusion

\[ \Delta_\varepsilon \omega(x) = \varepsilon^{-2} \int (\omega(y) - \omega(x)) \eta_\varepsilon(y - x) \, dy \]

where kernel \( \eta \) satisfies moment properties \( \int x_i^2 \eta(x) \, dx = 2 \)

Resulting particle scheme for diffusion of vorticity and viscosity \( \nu \):

\[ \frac{d\omega^h_p}{dt} = \nu \varepsilon^{-2} \sum_q (v_q \omega_q^h - v_q \omega_p^h) \eta_\varepsilon(x_q^h - x_p^h) \]
Main features: conservation properties, CFL free, particles adapt to zones of interest

Convergence requirement (and trouble ..): overlapping of particles necessary to ensure consistency \( \Delta x \ll h \)

Hard to satisfy for long time calculations in practice and/or strong strain

Accumulation/lack of particles can lead to numerical artefacts

Steady-state solution of 2D incompressible Euler equation
To account/correct for particle distortion, 2 classes of techniques:

- **adapt**
  - particle volumes/strengths,
  - kernel shape/radius/derivatives

- **control/adapt** particle locations
Adapt kernel formulas: renormalization (P.W. Randles, L.D. Libersky, 1996)

Idea: ensure that at the discrete particle level the gradient evaluations are exact for linear profiles -> modified kernel $W^R$ satisfying

$$\sum_{i=1}^{N} (\vec{x}_j - \vec{x}) \otimes \nabla^R W(\vec{x} - \vec{x}_j) \omega_j = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$W^R$ obtained from $W$ through multiplication by renormalization matrix

$$\nabla^R W(\vec{x} - \vec{x}_j) = L(\vec{x}) \nabla W(\vec{x} - \vec{x}_j)$$

To fulfill desired property, $L$ given by

$$L(\vec{x}) = \left[ \sum_j \nabla W_h(x_i - x_j) \otimes (x_j - x_i) v_j \right]^{-1}$$
Test of rotating circular spot
Gas dynamics model
Different implementation of renormalization and mollifier range (Oger et al., JCP 2007)

\[
\frac{d\vec{v}_i}{dt} = - \sum_j \frac{m_j}{\rho_i\rho_j} (P_i + P_j) \nabla W(\vec{x}_i - \vec{x}_j)
\]

\[
\frac{d\vec{v}_i}{dt} = - \sum_j \frac{m_j}{\rho_i\rho_j} (-P_i + P_j) L(\vec{x}_i) \nabla W(\vec{x}_i - \vec{x}_j)
\]

\( h = 1.3 \Delta x \)

Later time, \( h = 3 \Delta x \) (about 200 neighbors)
Reconstructing interfaces from lagrangian particle distribution is also challenging

Marrone et al., 2010, method based on looking at smallest eigenvalues of the renormalization matrix
Adami et al., 2010, computation of curvature
Other (related) approach:
reproducing kernel methods (Liu et al. 95) and variants (discrete corrected methods, Sbalzarini et al. 2010)
idea: adapt kernel shapes to satisfy at the discrete level moment properties (up to a desired order).
Need to solve linear systems at each particle location + lin-list of particles
Implicit subgrid-scale models in particle methods

Focus on 3D Euler equations (inviscid flows) in vorticity formulation

\[
\frac{\partial \omega}{\partial t} + \text{div}(u\omega) - [\nabla u]\omega = 0
\]

Particle solution given by

\[
\omega(x, t) = \sum_p \Gamma_p \delta(x - x_p)
\]

\[
\frac{dx_p}{dt} = \bar{u}(x_p, t), \quad \frac{d\Gamma_p}{dt} = \nabla \bar{u}(x_p, t) \Gamma_p
\]

Where \(\bar{u}\) is a mollified velocity field, weak solution to

\[
\frac{\partial \omega_i}{\partial t} + \text{div} \bar{u}\omega_i - \text{div} \omega \bar{u}_i = 0
\]
Mollified particles (blobs) thus satisfy

\[
\frac{\partial \omega_i}{\partial t} + \text{div} \, \overline{u \omega_i} - \text{div} \, \overline{\omega u_i} = 0
\]

This is an **averaged** Euler equations

This means that the particle method is achieving some subgrid scale (implicit) model

The transfer from larges scales to small scales can be illustrated by the following sketch
More precise analysis needed to understand enstrophy transfers
For simplicity, consider 2D case and start from

$$\frac{\partial \omega_\varepsilon}{\partial t} + \text{div}(u_\varepsilon \omega_\varepsilon) = E.$$ 

Where $\omega_\varepsilon = \overline{\omega} = \omega \ast \zeta_\varepsilon$

$$E(x) = \text{div}_x \int \omega(y)[u_\varepsilon(x) - u_\varepsilon(y)]\zeta_\varepsilon(x - y) \, dy$$

$$E = E_1 + E_2$$

$$E_1(x) = \text{div}_x \left( \omega(x) \int [u_\varepsilon(x) - u_\varepsilon(y)]\zeta_\varepsilon(x - y) \, dy \right)$$

$$E_2(x) = \text{div}_x \int [\omega(y) - \omega(x)][u_\varepsilon(x) - u_\varepsilon(y)]\zeta_\varepsilon(x - y) \, dy$$
First term is a drift which does not contribute to enstrophy balance

\[ E_1 = \text{div}(\tilde{u}_\varepsilon \omega) \]

Second term can be rewritten after Taylor expansions as

\[ E_2(x) = \text{div}_x \sum_{i,j} \int [(y_i - x_i) \partial_i u(x)] [(x_j - y_j) \partial_j \omega(x)] \zeta(x - y) \, dy \]

For symmetry reasons, cross terms disappear and we are left with

\[ E_2 = m_2 \varepsilon^2 \text{div}([Du_{\varepsilon}] \nabla \omega) + O(\varepsilon^4) \]

with

\[ m_2 = \frac{1}{2} \int |x|^2 \zeta(x) \, dx \]

In other words, « equivalent » equation for large scales (blobs) is Euler + diffusion with anisotropic eddy viscosity
Back to the enstrophy balance: compute \( \int \omega E dx \) (drop \( \varepsilon \) for simplicity)

\[
\frac{1}{2} \frac{d}{dt} \int \omega^2 \, dx = \int \int \omega(x) \omega(y) [u(x) - u(y)] \cdot \nabla \zeta(x - y) \, dx \, dy
\]

\[
\omega(x) = \omega(y) + \omega(x) - \omega(y)
\]

\[
\frac{1}{2} \frac{d}{dt} \int \omega^2 \, dx = \int \int \omega^2(y) [u(x) - u(y)] \cdot \nabla \zeta(x - y) \, dx \, dy
\]

\[
+ \int \int [\omega(x) - \omega(y)] \omega(y) [u(x) - u(y)] \cdot \nabla \zeta(x - y) \, dx \, dy
\]

First term vanishes (\( \text{div} \, u = 0 \)) and for second term write

\[
2 \omega(y) = [\omega(x) + \omega(y)] - [\omega(x) - \omega(y)]
\]
For symmetry reasons, \((\omega(x) + \omega(y))\) does not contribute, left with

\[
\frac{d}{dt} \int \omega^2 dx = - \int \int [\omega(x) - \omega(y)]^2 [u(x) - u(y)] \cdot \nabla \zeta(x - y) \, dx \, dy
\]

Positive contributions come from points where

\[
[u(x) - u(y)] \cdot \nabla \zeta(x - y) < 0
\]

In practice, cut-off is a decreasing function of radius, and condition becomes

\[
[u(x) - u(y)] \cdot (x - y) > 0
\]

Coherent with the first sketch: backscatter through diverging particles
In practice, codes have always trouble to handle backscatter: Need to compensate, at least partially.

Two ways to prevent excess backscatter in a particle code:

• remesh particles or
• use above calculations to tune the proper diffusion model:

\[
\frac{d\omega_p}{dt} = \sum_q (\omega_p - \omega_q) v_q \left\{ [u(x_p) - u(x_q)] \right. \\
\left. \cdot [x_p - x_q] f'_\varepsilon(|x_p - x_q|) |x_p - x_q|^{-1} \right\}
\]

where \( \zeta(x) = f(|x|) \)

Non-linear diffusion, acting only on directions of diverging particles (positive eigenvalues of strain tensor)
pros and cons of totally grid-free particle methods:

**pro:** intuitive, fully lagrangian, adaptive, sharp interface in free surface flows

**cons:** expensive (link list for neighbors, construction of renormalization matrix), incompressible simulations by SPH at moderate to high Re (2D and 3D) still an open problem

**missing** (in my view): systematic validations (accuracy, $\Delta x$, CPU ..) against conventional methods on reference flows and diagnostics
Remeshed particle methods

Idea goes back to the 80’s:
Krasny’s 2D vortex sheet, Meiburg’s 3D jets, and Chorin’s and Leonard’s hairpin removal
Insert fresh particles «in between» old particles when needed
Specific to problems with topology control

More generic approach: remesh particles on regular grids through standard 3D interpolation formulas.

Conservation of the moments of the particle distribution:

\[
\text{conservation of } \int f \, dx, \int xf \, dx, \int x^2 f \, dx
\]
In practice work with tensor products of 1D formulas

Typical interpolation formulas:

• conservation of 3 moments (third order truncation error) use 3 points in each direction
  smooth version uses an additional grid point → 4 grid points

• conservation of 5 moments (5th order truncation error) use 5 points
  smooth version spread particle on 6 nearest grid points

• resulting stencils in 3D: 27, 64, 125, 216 points
  if advection of particles is split direction by direction, reduces to one-dimensional stencils

example of interpolation function (4 points) (Monaghan, 1995)

\[ M_4^r (x) = \begin{cases} 
1 - 5x^2 / 2 + 3|x|^3 / 2 & \text{if } 0 \leq |x| \leq 1 \\
(2 - |x|)^2(1 - |x|) / 2 & \text{if } 1 \leq |x| \leq 2 \\
0 & \text{if } 2 \leq |x| 
\end{cases} \]
pros and cons of remeshing

pros:
• maintains desired accuracy
• cheap
• allows to easily combine with grid-based solvers and technology (FFT-based field evaluation, domain decomposition, AMR, boundary conditions, limiters ..)
• regularity helps for load balancing in parallel implementation

drawbacks:
• against lagrangian nature of particles
• loose sharp interfaces in multiphase flows and self adaptation
• looks like classical semi-lagrangian method
Remeshed particle methods have been validated in a number of compressible and incompressible flows against Eulerian methods (FD, FV or spectral).
Classical example of the advection of a level set in an off-center rotating field

Implementation of grid-based methods with particles for corrections

Enright et al, JCP 2002
3r order Weno
N=100 + 64 ppc
CFL=1 (?)
CPU =??

Vincent et al, JCP 2010
VOF
N=64 + 9 ppc
CFL=0.1

N=100, CFL=8

N=160, CFL=12
CPU time :
1 s per iteration
remeshed particle method, 4th order remeshing,
2nd order in time

mercredi 11 juillet 2012
How to choose the remeshing frequency?

in principle: $\Delta t_{\text{remesh}} \gg \Delta t_{\text{push}}$

in practice $\Delta t_{\text{remesh}} \sim \Delta t_{\text{push}}$ because both driven flow strain (and not maximum velocity value et grid size)

So it is advisable to remesh at each time step with a time step given by $\Delta t = C/\text{Max}|\nabla a|$ (a=velocity field) - Independent of $\Delta x$

$\Rightarrow$ need to revisit analysis (and implementation) of remeshed particle methods ($\Rightarrow$ precise value for constant $C$ above)
**Starting point:** particle method with remeshing at every time step can be seen as combination of exact solution with finite-difference method.

Case of 1D linear advection: \( u_t + au_x = 0 \ (a > 0) \)

remeshing with linear interpolation

\[
\alpha = 1 - \lambda \\
\beta = \lambda \\
(\lambda = a\Delta t / \Delta x)
\]

\[
u_{j}^{n+1} = u_{j}^{n}(1 - \lambda) + \lambda u_{j-1}^{n}
\]

1st order upwind scheme
Third order 3 points remeshing formula if CFL<1:

$$u_{j}^{n+1} = \alpha u_{j}^{n} + \beta u_{j-1}^{n} + \gamma u_{j+1}^{n}$$

conservation of mass: $\alpha + \beta + \gamma = 1$
conservation of linear impulse: $\beta - \gamma = \lambda$
conservation of angular impulse: $\beta + \gamma = \lambda^2$

$$\alpha = 1 - \lambda^2$$
$$\beta = \frac{\lambda}{2} (1 + \lambda)$$
$$\gamma = \frac{\lambda}{2} (-1 + \lambda)$$

Lax-Wendroff scheme
If CFL > 1: exact solution for $\Delta t' = \text{Int}(\text{CFL}) \ast \Delta x/a$ followed by FD scheme for $\Delta t - \Delta t'$

Difference of Remeshed Particle Methods with semi-lagrangian schemes:

- forward instead of backwards
- keep easily localization property
- conservative in mass instead of pointwise values
- extends to nonlinear case (see next)
- allows lagrangian adaptive refinement
- allows to derive limiters (control of overshoots)
Nonlinear case

\[ u_t + (f(u))_x = 0 \ ; \ f(u) = u g(u) \]

2nd order requires 3 points remeshing formula AND 2nd order particle pusher

Centered RK2: compute particle velocity at time \( t_{n+1/2} \):

\[ u_p + \frac{\Delta t}{2} \frac{du_p}{dt} \]

but, from equation:

\[ \frac{du}{dt} + u(g(u))_x = 0 \]

\[ \tilde{g}_j = g \left( u_j - \frac{\Delta t}{2} u_j \left( g(u_j)_x \right) \right) \]

by centered finite-difference
Equivalent 2nd order FD scheme:

\[ u_{i}^{n+1} = u_{i}^{n} - \frac{\lambda}{2} (\tilde{g}_{i+1}^{n} u_{i+1}^{n} - \tilde{g}_{i-1}^{n} u_{i-1}^{n}) + \frac{\lambda^2}{2} (\tilde{g}_{i+1}^{n})^2 u_{i+1}^{n} - 2(\tilde{g}_{i}^{n})^2 u_{i}^{n} + (\tilde{g}_{i-1}^{n})^2 u_{i-1}^{n} \]

\[(\lambda = \frac{\Delta t}{\Delta x})\]

Equivalent equation:

\[ u_{i} + (f(u))_{x} + \Delta t^2 \left[ \frac{u_{ttt}}{6} + \frac{1}{6\lambda^2} f(u)_{xxx} \right] + \Delta t^2 \left[ \frac{1}{8} (g^2(u)_{x} g''(u)u^3)_{x} + \frac{1}{2} (g(u)g'(u)g(u)_{x}u^2)_{xx} \right] = 0 \]

Difference with L-W turns out to make it entropy consistent (Weynans, Magni 2013)
This interpretation allows to borrow limiting techniques from «Finite-Difference world», and derive remeshing techniques which avoid oscillations near shocks or discontinuities.

General idea:
• compute slopes of the solution on the grid
• push particles
• interpret remeshing in terms of FD fluxes
• derive limiters for the FD fluxes
• interpret limited fluxes back to remeshing formulas

Illustration for Burgers (shock and wave)
- green: original 2nd order RPM
- red: TVD correction
- blue: exact solution
Comparison of 2nd order RPM with TVD limitation at CFL 12 with 5th order Weno scheme at CFL 2 on a 1D test with steep gradients and deformation

initial condition: double top hat
advection field: \( a(x) = 1 + 0.5 \sin(\pi x) \)
Important to adapt time step to (local) flow strain independently of spatial resolution:

CFL 0.5 vs CFL 12 for same 2nd order remeshing formula

Reason: for CFL=0.5, RPM equivalent to Lax-Wendroff: TVD limitation performs poorly

Conclusion: large time-steps in remeshed particle methods not only reduce cost but also improve accuracy!
Next: how to enhance adaptivity in particle methods

2 class of methods:

- grid free method: self organizing Lagrangian particles
- remeshed particle method with Adaptive Mesh Refinement
Self organizing Lagrangian particles (Reboux et al., 2012)

To represent a given function \( f \), adapt particle locations to a criterion of the form

\[
\tilde{D}(\mathbf{x}) = \frac{D_0}{\sqrt{1 + |\nabla f(\mathbf{x})|^2}} \quad D_p = D(\mathbf{x}_p) = \min_{|\mathbf{x}_q-\mathbf{x}_p| \leq r_{c,p}} \tilde{D}(\mathbf{x}_q)
\]

Rearrange particles through pseudo-force, to minimize potential of the form

\[
V_{pq} = D_{pq}^2 V(|\mathbf{x}_p - \mathbf{x}_q|/D_{pq}) \quad D_{pq} = \min(D_p, D_q)
\]

ensures a low discrepancy repartition of particles with \(|f(\mathbf{x}_p)-f(\mathbf{x}_q)|\) constant for neighboring particles
Illustration in 2D viscous Burger’s equation

error as a function of average particle spacing (white circles) compared to standard 2nd order remeshed particle methods (black circles)

drawbacks: not straightforward, no 3D or large deformations so far
Remeshed particle methods with AMR (Bergdorff et al. 2005, 2009)

Works very much like FD AMR:

• define different level of refinement on the grid
• initialize corresponding particles
• push particles
• remesh particles at the corresponding grid sizes
• ensure buffer of articles to ensure consistent remeshing at a given level
For each level of refinement and every time step

- Select «active» particles on the grid (tag=1) (for instance using a wavelet-based MRA).
- Create a buffer around these particles (tag=0).
- Advect particles and tag.
- Remesh particles and tag them; keep particles with tag > 0.
Lagrangian AMR, consistent if $CFL > 1$, provided $\Delta t < \frac{C}{Max|\nabla a|}$

$c$ and the size of the remeshing kernel will give the size of the buffer

Illustration: simulation of 3D curvature-driven flow: **Collapsing Dumbbell**

$$\frac{\partial \phi}{\partial t} + \kappa \mathbf{n} \cdot \nabla \phi = 0. \quad \kappa = \nabla \cdot \mathbf{n}$$

distribution of active particles
A few words about algorithms and parallel implementation of remeshed particle methods

For computation of long range interaction, having particle on a grid can significantly reduce cost

Example of vorticity form of NS incompressible Navier-Stokes equations:

- In totally grid-free PM, Biot Savart leads to a N-body problem for the particle velocities

\[ O(N^2) \] complexity can be reduced to \[ O(N) \] using multipole expansions and tree codes

- In Remeshed particle methods, can rely on FFT-based Poisson solvers
Typical tree-code:
Divide recursively into boxes containing about the same number of particles

Upward pass:
form multipole expansions, from finer to coarser level (using shifts of previously computed expansions)

Downward pass:
accumulate contributions of well-separated boxes, from coarser to finer level
At finest level, complete with direct summation of nearby particles
Comparison of CPU times for velocity evaluations in 3D
(Krasny tree-code vs RPM with Fishpack Poisson solver and 64 points interpolation formulas)

VIC1: cartesian grid with 100% particles
VIC2: polar grid with 65% particles
VIC3: polar grid with 25% particles
Things to consider in parallel implementation of remeshed particle methods

Three type of operations:

Particle to Mesh (PM) : remeshing (local)
Mesh to Mesh : local (gradients, diffusion, boundary terms ..) or non-local (velocity from vorticity)
Mesh to Particle (MP) : interpolation (local) of velocity on particles (and forces if you wish)

Under Lagrangian CFL, localization of particles in MPI topology natural -> local PM or MP operations are OK. Global operations rely mostly on parallel FFT.
Examples of GPU implementations of remeshed particle methods

Navier-Stokes: 2D wakes
2nd order remeshing, 1 million particles, 21 fps
(Rossinelli et al. 2010)

Scalar advection: 3D sphere in strain flow
4th order remeshing, 16 million grid points, 0.5 fps
(Etancelin, 2012)

Still room for improvement: far from peak perf and particles have to be everywhere.
GPU/CPU speed up in the 3D case: 20 w.r. # grid points, but only 2 w.r. Δx