The Phase Flow Method

Emmanuel Candès, California Institute of Technology

Workshop on High Frequency Waves
CSCAMM, University of Maryland, September 2005

Collaborators: Lexing Ying (Caltech)
Agenda

• The phase flow method

• Applications in computational high frequency wave propagation
  – Wave front propagation
  – Amplitude computations
  – Multiple arrival times computations

• Numerical results

• Epilogue
Problem Statement

- Nonlinear autonomous ODE

\[
\frac{dy}{dt} = F(y), \quad t > 0,
\]

where \( y : \mathbb{R} \to \mathbb{R}^d \) and \( F : \mathbb{R}^d \to \mathbb{R}^d \) is smooth.

- Compute \( y(T) = y(T, y_0) \) for many initial conditions \( y_0 \).

- Standard approach: time step \( \tau \) and local integration rule for each \( y_0 \).

- Not very efficient.
Terminology

• Phase map: $g_t : \mathbb{R}^d \rightarrow \mathbb{R}^d$ defined by $g_t(y_0) = y(t, y_0)$.

• Phase flow: collection of all phase maps $\{g_t, t \in \mathbb{R}\}$.

• A manifold $M \subset \mathbb{R}^d$ is invariant if $g_t(M) \subset M$. 
Example: Bicharacteristic Flow

Ray equations in phase space $\mathbb{R}^d \times \mathbb{R}^d, d = 2, 3$

$$\frac{dx}{dt} = \nabla_p H(x, p), \quad \frac{dp}{dt} = -\nabla_x H(x, p),$$

with Hamiltonian $H(x, p)$

$$H(x, p) = c(x)|p|,$$

- Autonomous
- Wish to integrate for each $y_0 = (x_0, p_0) \in \Sigma_0$ (initial wave front)
- Invariant manifold $M$:
  - $\mathbb{R}^d \times \mathbb{R}^d$
  - $\mathbb{R}^d \times S^{d-1}$
  - $[0, 1]^3 \times S^{d-1}$
Key Structure

Rapid construction of the complete phase map $g_T$ at time $T$.

1. **Discretization.** Start with a uniform or quasi-uniform grid $M_h$ on $M$.

2. **Initialization.** Fix a small time step $\tau$ and compute an approximation of $g_\tau$.
   - For each $y_0 \in M_h$, $g_\tau(y_0)$ is computed by a standard ODE integration rule
   - The value of $g_\tau$ at any other point is defined via local interpolation.

3. **Loop.** Construct $g_{2^k_\tau}$ from $g_{2^{k-1}_\tau}$
   - For each $y_0 \in M_h$, $g_\tau(y_0)$

\[ g_{2^k_\tau}(y_0) = g_{2^{k-1}_\tau}(g_{2^{k-1}_\tau}(y_0)) \]

   - Otherwise, local interpolation.

**Key point:** Systematic use of already computed information
Peek at the results

• Very efficient

• Surprisingly accurate
Algorithm 1 (Basic Version)

- **Parameter selection.** Select a grid size $h > 0$, a time step $\tau > 0$, and an integer constant $S \geq 1$ such that $B = (T/\tau)^{1/S}$ is an integer power of 2.

- **Discretization.** Select a uniform or quasi-uniform grid $M_h \subset M$ of size $h$.

- **Burn-in.** Compute $\tilde{g}_\tau$.
  - For a gridpoint $y_0$, $\tilde{g}_\tau(y_0)$ is calculated by applying the ODE integrator.
  - Construct an interpolant and compute $\tilde{g}_\tau(y_0)$ by evaluating the interpolant outside of the grid.

- **Loop.** For $k = 1, \ldots, S$, evaluate $\tilde{g}_{B^k \tau}$.
  - $\tilde{g}_{B^k \tau}(y_0) = \left(\tilde{g}_{B^{k-1} \tau}\right)^{(B)}(y_0)$ for each $y_0$ on the grid.
  - Construct an interpolant which and use it for out-of-grid evaluation.

- **Terminate.** When $k = S$, we hold $\tilde{g}_t$, for $t = \tau, 2\tau, 4\tau, 8\tau, \ldots, T$ and more.
Main Result

- ODE integrator is of order $\alpha$.
- Local interpolation scheme is of order $\beta \geq 2$.
- Size of grid is $O(h^{-d_M})$

Approximation error at time $t$

$$\varepsilon_t = \max_{b \in M} |g_t(b) - \tilde{g}_t(b)|.$$  

(i) The approximation error obeys

$$\varepsilon_T \leq C \cdot (\tau^\alpha + h^\beta)$$

(ii) The complexity is $O(\tau^{-1/S} \cdot h^{-d_M})$.

(iii) For each $y \in M$, $\tilde{g}_T(y)$ can be computed in $O(1)$ operations.

(iv) For any intermediate time $t = m\tau \leq T$ and $y \in M$, $\tilde{g}_t(y)$ is evaluated in $O(\log(1/\tau))$ operations.
Asymptopia

Balancing of errors $h^\beta \sim \tau^\alpha$

- Accuracy $O(\tau^\alpha)$
- Complexity $O(\tau^{-r})$, $r = d_M \alpha / \beta + 1/S$.

Suppose that $M$ and $F$ are sufficiently smooth, and choose $\beta$ and $S$ s.t. $r < 1$.

In an asymptotic sense, one can compute an approximation to the entire phase map $g_T$ much faster than one computes—with the same order of accuracy—a single solution with the standard ODE integration rule.
Variation I: Time-doubling

Select $B = 2$, and construct $g_{2^k \tau}$ from $g_{2^{k-1} \tau}$ via

$$g_{2^k \tau}(y_0) = g_{2^{k-1} \tau}(g_{2^{k-1} \tau}(y_0))$$

- Complexity is lower $O(h^{-d_M} \log(1/\tau))$
- Accuracy is reduced $O((\tau^\alpha + h^\beta)/\tau)$
Variation II: Algorithm 2 (Practical Version)

For large times, $g_T$ may become quite oscillatory, and one would need a very fine initial spatial resolution.

(a) Choose $T_0 = O(1)$, $T = mT_0$, such that $g_{T_0}$ remains non-oscillatory and pick $h$ so that the grid is sufficiently dense to approximate $g_{T_0}$ accurately.

(b) Construct $\tilde{g}_{T_0}$ using Algorithm 1.

(c) For any $y_0$, define $\tilde{g}_T(y_0)$ by $\tilde{g}_T(y_0) = (\tilde{g}_{T_0})^{(m)}(y_0)$. 
Problem specific components

- Discretization of $M$
- ODE integration rule
- Local interpolation scheme
Geometrical Optics

- Inhomogeneous scalar wave equation in 2D and 3D:
  \[ u_{tt} - c^2(x) \Delta u = 0, \quad t > 0. \]

- High-frequency expansion (WKB)
  \[ u(t, x) = e^{i \lambda \Phi(t, x)} \sum_{n \geq 0} A_n(t, x) (i \lambda)^{-n} \]
  where \( \Phi \) and \( A_n \) are smooth.

- Eikonal equations
  \[ \Phi_t \pm c(x) |\nabla \Phi| = 0. \]
• Bicharacteristics equations:

\[
\frac{dx}{dt} = c(x) \frac{p}{|p|}, \quad \frac{dp}{dt} = -\nabla c(x)|p|
\]

• Reduced Hamiltonian flow, \( p = |p|\nu \)

\[
\frac{dx}{dt} = c(x)\nu, \quad \frac{d\nu}{dt} = -\nabla c(x) + (\nabla c(x) \cdot \nu) \nu
\]

or compactly \( dy/dt = F(y) \).

• Assume \( c(x) \) is periodic on \([0, 1]^d\) (can be relaxed).

• \( M = \{(x, \nu) \in [0, 1]^d \times S^{d-1}\} \) compact and smooth.
The Phase Flow Method for HFWP (2D)

- $M = [0, 1]^2 \times [0, 2\pi]$

\[
\frac{dx}{dt} = c(x, y) \cos \theta, \quad \frac{dy}{dt} = c(x, y) \sin \theta, \quad \frac{d\theta}{dt} = c_x \sin \theta - c_y \cos \theta
\]

- ODE integrator: 4th order Runge-Kutta

- Cartesian uniform grid on $M = [0, 1]^2 \times [0, 2\pi]$

- Local interpolation:
  - Interpolate the periodic shift $g_t(y) - y$ instead of $g_t(y)$.
  - Interpolation of a periodic function on a Cartesian grid.
  - Solution: tensor-product Cardinal B-spline (interpolant is constructed by means of FFT’s).
The Phase Flow Method for HFWP (3D)

- $M = [0, 1]^3 \times S^2$
- ODE integrator: 4th order Runge-Kutta
- Discretization
  - Uniform Cartesian grid in $x \in [0, 1]^3$
  - Spherical coordinates in $\nu \in S^2$
  
  $$\nu(\theta, \phi) = (\cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi)$$
  
  with sample points $(0, h, \cdots, 2\pi - h) \times (h/2, \cdots, \pi - h/2)$.
- Local interpolation
  - Cardinal B-splines in $x$.
  - Cardinal B-splines in $\nu$ (after periodic extension trick)
  
  $$f^e(\theta, \phi) = \begin{cases} f(\theta, \phi) & \phi \in [0, \pi) \\ f(\theta + \pi, -\phi) & \phi \in [-\pi, 0). \end{cases}$$

$f^e$ is periodic on $[0, 2\pi] \times [-\pi, \pi]$. 
Wave Front Construction

Initial wave front $y_0(r) = (x_0(r), \nu_0(r))$ propagated up to time $T$. Basic algorithm:

- Choose $T_0$ and construct $\tilde{g}_{T_0}$.
- Discretize the wave front by sampling $y_0(r)$ at the points $r_i$.
- For each $r_i$, approximate $y(T, r_i)$ with $\tilde{y}(T, r_i) = (\tilde{g}_{T_0})^{(m)}(y_0(r_i))$ where $T = mT_0$.
- Connect $\tilde{x}(T, r_i)$ to construct the final wave front.
2D Adaptive Wave Front Construction

Choose a tolerance $\lambda$, and sample the initial wave front with $R = \{r_i\}$ s.t.

$$|y_0(r_i) - y_0(r_{i+1})| \leq \lambda$$

For $k = 1, \cdots, T/T_0$

- For any $r_i \in R$, $\tilde{y}(kT_0, r_i) = \tilde{g}_{T_0}(\tilde{y}((k - 1)T_0, r_i))$.

- For any interval $I_i := [r_i, r_{i+1}]$ s.t. $|\tilde{y}(kT_0, r_i) - \tilde{y}(kT_0, r_{i+1})| > \lambda$:
  - Insert $N_i$ new samples $\{r_\ell\}$ evenly distributed in $I_i$;

$$N_i = \lceil |\tilde{y}(kT_0, r_i) - \tilde{y}(kT_0, r_{i+1})|/\lambda \rceil.$$  

  - The values $\tilde{y}(kT_0, r_\ell)$ at the new points are computed using

$$\tilde{y}(kT_0, r_\ell) = (\tilde{g}_{T_0})^{(k)}(y_0(r_\ell))$$
Inserting Rays

Standard Lagrange type methods insert new rays by interpolating nearby sampled values.

- Difficult (unstructured grid)
- Low accuracy

Effortless and accurate with the phase flow method.

Refinement condition.

- Standard methods need to use

\[ |\tilde{y}(kT_0, r_i) - \tilde{y}(kT_0, r_j)| > \lambda. \]

- Here,

\[ |x(kT_0, r_i) - x(kT_0, r_j)| > \lambda \]

is sufficient since interpolation is not used. Increased efficiency.
Amplitude Computation, I

Squeezing and spreading of rays

\[
\frac{A_0(x(t, r))}{A_0(x(0, r))} = \sqrt{\left| \frac{\partial_r x(0, r)}{\partial_r x(t, r)} \right| \frac{c(x(t, r))}{c(x(0, r))}} \quad (2D)
\]

\[
\frac{A_0(x(t, r, s))}{A_0(x(0, r, s))} = \sqrt{\left| \frac{\partial_r x(0, r, s) \times \partial_s x(0, r, s)}{\partial_r x(t, r, s) \times \partial_s x(t, r, s)} \right| \frac{c(x(t, r, s))}{c(x(0, r, s))}} \quad (3D)
\]

- Additional information is needed: \( \partial_r x(t, r) \) in 2D and \( \partial_r x(t, r, s) \) and \( \partial_s x(t, r, s) \) in 3D.

- It is sufficient to know \( \nabla_b y(t, b) \).

- Approximate the gradient of the phase map along with the phase map.
Amplitude Computation, II

- Linear equation for $\nabla_b y(t, b)$.

$$\frac{d\nabla_b y(t, b)}{dt} = \nabla_y F(y(t, b)) \cdot \nabla_b y(t, b), \quad \nabla_b y(0, b) = I.$$

with

$$\nabla_y F(y) = \begin{pmatrix} \nu \nabla c^T & cI \\ -\nabla^2 c + \nu \nu^T \nabla^2 c & (\nabla c \cdot \nu)I + \nu \nabla c^T \end{pmatrix}$$

- Group property:

$$\nabla_b y(2t, b) = \nabla_b y(t, g_t(b)) \cdot \nabla_b y(t, b).$$

- Build the approximation to $\nabla_b y(t, b)$ along with $g_t(b)$ in Algorithms 1 & 2.
Multiple Arrival Times

- **Source:**
  - 2D: smooth curve in 3D phase space.
  - 3D: smooth surface in 5D phase space.

- **Target:** point in physical space.

- **Trace:** family of wave fronts from time index by \( t \in [t_0, t_1] \).
  - 2D: smooth 2D surface in 3D phase space.
  - 3D: smooth 3D manifold in 5D phase space.

Problem: compute the number of arrivals and the arrival times (at the targets) up to time \( T \).
Single Source / Multiple Targets

- Choose a time step $\Delta T$ and a tolerance $\lambda > 0$.
- Apply the adaptive wave front algorithm with time step $\Delta T$ to construct the final wave front at time $T$; the algorithm provides the values $\tilde{y}(k\Delta T, r_i)$ for $0 \leq k \leq T/\Delta T$.
- Approximate the trace by linear interpolation of the sampled values $\tilde{y}(k\Delta T, r_i)$—represented by a triangle mesh in phase space. The computed trace is piecewise linear.
- Project the approximate trace $\tilde{y}$ onto physical space (i.e. discard $\nu$).
- For each target point, check whether it is covered by nearby projected triangles. If so, the arrival and arrival time are recorded.
Single Source and Single Target

Adaptive version

- Wasteful to compute the full trace
- Basic idea is to throw away large parts of the trace before construction
- Efficient algorithm
Details

- The nearby triangles can be collected efficiently using a bounding box test.
- Inside/outside test for each triangle is carried out using the determinant test (standard in computational geometry).
- The discretization of the source is not fixed and is refined as the wave front evolves.
- The parameters $\Delta T$ and $\lambda$ control the accuracy.
  - $\Delta T$ and $\lambda$ are of the order of $\sqrt{\varepsilon}$ suffice for an error tolerance of $\varepsilon$ for the trace approximation.
  - Accuracy of arrival times is then $O(\varepsilon)$. 
Example 1

• Velocity field (2D waveguide)

\[ c(x, y) = \frac{1}{1 + e^{-64 \cdot (y - 1/2)^2}}. \]

• The initial wave front is a planar wave at \( x = 0 \).
Example 1: wave front construction
Example 1: wave front construction
MATLAB implementation on a desktop computer with a 2.6GHz CPU and 1GB of memory.

- Uniform grid with 64 points in $y$ and 128 points in $\theta$.
- The construction of $\tilde{g}_{T_0}$ takes about 2 seconds.
- Accuracy of computed phase map is about $10^{-5}$.
- Adaptive wave front propagation up to $T = 4$
  - Final wave front has about 600 samples
  - Takes about 0.064 second
  - MATLAB’s ODE solver takes about 0.08 second to trace a single ray
  - Speedup factor is about 750
Example 1: accuracy

<table>
<thead>
<tr>
<th>Discretization vs. $T_0$</th>
<th>0.0625</th>
<th>0.125</th>
<th>0.25</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16,32)</td>
<td>4.991e-04</td>
<td>1.034e-03</td>
<td>2.316e-03</td>
<td>5.252e-03</td>
</tr>
<tr>
<td>(32,64)</td>
<td>2.301e-05</td>
<td>4.563e-05</td>
<td>8.344e-05</td>
<td>3.787e-04</td>
</tr>
<tr>
<td>(64,128)</td>
<td>1.274e-06</td>
<td>2.759e-06</td>
<td>5.195e-06</td>
<td>7.343e-06</td>
</tr>
<tr>
<td>(128,256)</td>
<td>1.133e-07</td>
<td>1.755e-07</td>
<td>3.901e-07</td>
<td>6.016e-07</td>
</tr>
</tbody>
</table>

High accuracy with small sample size.
Example 1: amplitude computation

amplitude at $t=0.5$

amplitude at $t=1$

amplitude at $t=2$

amplitude at $t=4$
Example 1: multiple arrivals

![Graph showing total number of arrivals, arrival times for points with x=1.96094, time of first arrival, and time of last arrival before T.](image-url)
Example 2

- Velocity field: \( a = (1/4, 1/4), \ b = (3/4, 3/4) \)

\[
c(x, y) = \frac{1}{1 + 3e^{-64|x-a|^2} + 3e^{-64|x-b|^2}}^2
\]

- The initial wave front is a small circle centered at \((1/2, 1/2)\).
Example 2 (wave front construction)

Speed up factor is 200
Example 2 (amplitude computation)
Example 2 (multiple arrivals)
Example 3

- Velocity field (3D waveguide)

\[ c(x, y, z) = \frac{1}{1 + e^{-64 \cdot (x-1/2)^2 - 64 \cdot (y-1/2)^2}} \]

- The initial wave front is a plane wave at \( z = 0 \).
Example 3 (wave front construction)
Example 3 (wave front construction)
Example 3 (amplitude computation)
amplitude at $t=0.5$

amplitude at $t=1$

amplitude at $t=1.5$

amplitude at $t=2$
Example 3 (multiple arrivals)
Example 4

- Velocity field (3D waveguide)

\[ c(x, y, z) = \frac{1}{1 + e^{-64(t(x-1/2)^2 + (y-1/2)^2)}} \]

- The initial wave front is a small sphere centered at \((1/2, 1/2, 1/2)\).
Example 4
Example 4
Example 5

- Velocity field: \( a = (1/4, 1/4, 1/2), b = (3/4, 3/4, 1/2) \)

\[
c(x, y) = \frac{1}{1 + 3e^{-64|x-a|^2} + 3e^{-64|x-b|^2}}
\]

- The initial wave front is a small sphere centered at \((1/2, 1/2, 1/2)\).
Example 5
• $T_0 = 0.0625$ and $\tau = 2^{-10}$ in the wave front construction algorithm.
• Cartesian grid with 16, 16, 16, 32 and 16 points in $x$, $y$, $z$, $\theta$ and $\phi$.
• $\tilde{g}_{T_0}$ is constructed within 900 second and has accuracy around $10^{-4}$.
• Adaptive wave front propagation up to $T = 1.5$. The final wave front is resolved with 37,000 samples.
Summary

The phase flow method: novel approach to integrate ODEs.

- Bootstrapping in time domain using the group property of the phase flow.
- Efficient and accurate
- Inserting rays is effortless
- Many applications: e.g. geodesic flows on surfaces
- Further developments: piecewise smooth velocity fields
Epilogue: Curvelet and Wave Equations

New curvelet multiscale pyramid

- Multiscale
- Multi-orientations
- Parabolic (anisotropy) scaling

\[ width \approx length^2 \]

- Indexed by phase space

Curvelet expansion

\[
f = \sum_{\mu} \langle f, \varphi_\mu \rangle \varphi_\mu \quad \| f \|_2^2 = \sum_{\mu} \langle f, \varphi_\mu \rangle^2
\]
Digital Curvelets

[Graph showing digital curvelets with axes labeled from 50 to 500]
\[ u_{tt} - c^2(x) \Delta u = 0 \]

The action of the wave propagator on a curvelet is well-approximated by a rigid motion along the Hamiltonian flow.
Curvelets and Wave Equations, II

Wave equation

\[ u_{tt} - c^2(x) \Delta u = 0, \]

with \( u(0, x) \) and \( u_t(0, x) \) as initial data.

The curvelet matrix of a wide range of wave propagators is optimally sparse: the coefficients decay nearly exponentially fast away from a shifted diagonal.

Sketch of the curvelet representation of the wave propagator
Fast Wave Propagation?

\[ u_t = e^{-Pt}u_0 \]

\[ u_0 \xrightarrow{e^{-Pt}} u_t \]

\[ F \quad \downarrow \quad \downarrow F \]

\[ \theta_0 \xrightarrow{A(t)} \theta_t \]

For any \( t \), \( A(t) \) is sparse

Background: Fast and accurate Digital Curvelet Transform is available (with Demanet, Donoho and Ying).