

# Efficient Computation of Transparent Boundary Conditions

Algirdas Grybas  
School of Engineering and Science  
Jacobs University Bremen

16 May, 2007

# Outline

Problem statement

Solution

Ornstein-Uhlenbeck process

Outlook

# Outline

Problem statement

Solution

Ornstein-Uhlenbeck process

Outlook

# Set-up

- ▶ Fourier representation of travelling waves in shear flow

# Set-up

- ▶ Fourier representation of travelling waves in shear flow
- ▶ Equivalent to computing dispersive waves on a lattice

# Set-up

- ▶ Fourier representation of travelling waves in shear flow
- ▶ Equivalent to computing dispersive waves on a lattice
- ▶ Boundary introduced due to finiteness of computation

# Set-up

- ▶ Fourier representation of travelling waves in shear flow
- ▶ Equivalent to computing dispersive waves on a lattice
- ▶ Boundary introduced due to finiteness of computation
- ▶  $S$  right-most grid points before the boundary to be linearly extrapolated to  $M$  left-most off-grid points

# Set-up

- ▶ Fourier representation of travelling waves in shear flow
- ▶ Equivalent to computing dispersive waves on a lattice
- ▶ Boundary introduced due to finiteness of computation
- ▶  $S$  right-most grid points before the boundary to be linearly extrapolated to  $M$  left-most off-grid points
- ▶ Extrapolation is projection onto a subspace of outgoing waves

# Transparent boundary condition

*Solve*

$$B^H z = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_S \end{pmatrix}$$

*and then compute*

$$\begin{pmatrix} \theta_{S+1} \\ \vdots \\ \theta_{S+M} \end{pmatrix} = C^H z,$$

*where*

$$B = \begin{pmatrix} \kappa_1^1 & \cdots & \kappa_1^S \\ \vdots & & \vdots \\ \kappa_{Sout}^1 & \cdots & \kappa_{Sout}^S \end{pmatrix}, \quad C = \begin{pmatrix} \kappa_1^{S+1} & \cdots & \kappa_1^{S+M} \\ \vdots & & \vdots \\ \kappa_{Sout}^{S+1} & \cdots & \kappa_{Sout}^{S+M} \end{pmatrix}$$

## Least-norm problem

- ▶ Lack of optimal choice for  $S^{out}$  samples

# Least-norm problem

- ▶ Lack of optimal choice for  $S^{out}$  samples
- ▶ In general, underdetermined system

## Least-norm problem

- ▶ Lack of optimal choice for  $S^{out}$  samples
- ▶ In general, underdetermined system
- ▶ Find the smallest  $z$  satisfying the boundary condition

## Least-norm problem

- ▶ Lack of optimal choice for  $S^{out}$  samples
- ▶ In general, underdetermined system
- ▶ Find the smallest  $z$  satisfying the boundary condition

**Least-norm problem:** *Minimize*

$$\|z\|_W^2 = z^H W z \text{ such that } B^H z = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_S \end{pmatrix}$$

where  $W = \text{diag}(|\omega'(\xi_1)|^{-1}, \dots, |\omega'(\xi_{S^{out}})|^{-1})$

# Outline

Problem statement

**Solution**

Ornstein-Uhlenbeck process

Outlook

# Gradient method: Outline

Iterative method for solving underdetermined systems

# Gradient method: Outline

Iterative method for solving underdetermined systems

- ▶ Function as having level surfaces

# Gradient method: Outline

Iterative method for solving underdetermined systems

- ▶ Function as having level surfaces
- ▶ Choose an initial point and compute the gradient

# Gradient method: Outline

Iterative method for solving underdetermined systems

- ▶ Function as having level surfaces
- ▶ Choose an initial point and compute the gradient
- ▶ Gradient is orthogonal to the level surface

# Gradient method: Outline

Iterative method for solving underdetermined systems

- ▶ Function as having level surfaces
- ▶ Choose an initial point and compute the gradient
- ▶ Gradient is orthogonal to the level surface
- ▶ Find minimum in the direction of the gradient

# Gradient method: Outline

Iterative method for solving underdetermined systems

- ▶ Function as having level surfaces
- ▶ Choose an initial point and compute the gradient
- ▶ Gradient is orthogonal to the level surface
- ▶ Find minimum in the direction of the gradient
- ▶ Take the minimum point as the new guess

# Gradient method: Properties

Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)

# Gradient method: Properties

Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)

# Gradient method: Properties

## Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)
- ▶ Matrix  $B$  well-conditioned

# Gradient method: Properties

## Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)
- ▶ Matrix  $B$  well-conditioned
- ▶ Expect to have a relatively good solution after single iteration

# Gradient method: Properties

## Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)
- ▶ Matrix  $B$  well-conditioned
- ▶ Expect to have a relatively good solution after single iteration
- ▶ Confident to have good initial guess for next computations (small perturbations of data)

# Gradient method: Properties

## Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)
- ▶ Matrix  $B$  well-conditioned
- ▶ Expect to have a relatively good solution after single iteration
- ▶ Confident to have good initial guess for next computations (small perturbations of data)

## Drawbacks:

- ▶ Computation of the stepsize costly (at least for non-linear gradient)

# Gradient method: Properties

## Advantages:

- ▶ Potentially small number of iterations (if initial guess is good)
- ▶  $O(n^2)$  operations needed (vs.  $O(n^3)$  for QR-decomposition)
- ▶ Matrix  $B$  well-conditioned
- ▶ Expect to have a relatively good solution after single iteration
- ▶ Confident to have good initial guess for next computations (small perturbations of data)

## Drawbacks:

- ▶ Computation of the stepsize costly (at least for non-linear gradient)
- ▶ Potentially many iterations if initial guess bad (if  $B$  sparse or ill-conditioned)

# Penalty method

**Constrained optimization:** *Minimize the function  $f(x)$  under the constraint that  $g(x) = 0$ .*

# Penalty method

**Constrained optimization:** *Minimize the function  $f(x)$  under the constraint that  $g(x) = 0$ .*

Iterative method used for transforming problems of constrained optimization to those of unconstrained optimization

$$\mathcal{L}(x_k, \delta_k) = f(x_k) + \delta_k \|g(x_k)\|^2$$

# Penalty method

**Constrained optimization:** *Minimize the function  $f(x)$  under the constraint that  $g(x) = 0$ .*

Iterative method used for transforming problems of constrained optimization to those of unconstrained optimization

$$\mathcal{L}(x_k, \delta_k) = f(x_k) + \delta_k \|g(x_k)\|^2$$

- ▶ Penalty parameter  $\delta$  penalizes for not satisfying the constraint condition

# Penalty method

**Constrained optimization:** *Minimize the function  $f(x)$  under the constraint that  $g(x) = 0$ .*

Iterative method used for transforming problems of constrained optimization to those of unconstrained optimization

$$\mathcal{L}(x_k, \delta_k) = f(x_k) + \delta_k \|g(x_k)\|^2$$

- ▶ Penalty parameter  $\delta$  penalizes for not satisfying the constraint condition
- ▶ If the constraint condition satisfied, equivalent to minimizing  $f(x)$

# Penalty method

**Constrained optimization:** *Minimize the function  $f(x)$  under the constraint that  $g(x) = 0$ .*

Iterative method used for transforming problems of constrained optimization to those of unconstrained optimization

$$\mathcal{L}(x_k, \delta_k) = f(x_k) + \delta_k \|g(x_k)\|^2$$

- ▶ Penalty parameter  $\delta$  penalizes for not satisfying the constraint condition
- ▶ If the constraint condition satisfied, equivalent to minimizing  $f(x)$
- ▶ In general, increase  $\delta$  in every iteration step

# Gradient-Penalty method

**Constrained optimization:** *Minimize  $\|z\|_W^2$  under the constraint that  $B^H z = \theta$*

# Gradient-Penalty method

**Constrained optimization:** Minimize  $\|z\|_W^2$  under the constraint that  $B^H z = \theta$

**Idea:** Apply Penalty method to the constrained optimization at hand, use Gradient method to minimize the resulting Lagrangian function

$$\mathcal{L}(z_k, \delta_k) = z_k^H W z_k + \frac{1}{2} \delta_k \|B^H z_k - \theta\|_2^2$$

# Other methods

(Preconditioned) Conjugate Gradient method

## Other methods

(Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions

## Other methods

(Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

### Lagrange multiplier method

- ▶ Minimize the *augmented Lagrangian*
$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2} \delta_k \|h(x)\|_2^2, \text{ where } \lambda_k \in \mathbb{R}^m$$

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

### Lagrange multiplier method

- ▶ Minimize the *augmented Lagrangian*
$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2} \delta_k \|h(x)\|_2^2, \text{ where } \lambda_k \in \mathbb{R}^m$$
- ▶ Additional degree of freedom  $\lambda$

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

### Lagrange multiplier method

- ▶ Minimize the *augmented Lagrangian*
$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2} \delta_k \|h(x)\|_2^2, \text{ where } \lambda_k \in \mathbb{R}^m$$
- ▶ Additional degree of freedom  $\lambda$
- ▶ More than linear convergence, if  $\delta_k \rightarrow \infty$  as  $k \rightarrow \infty$  (linear if  $\delta_k$  bounded)

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

### Lagrange multiplier method

- ▶ Minimize the *augmented Lagrangian*
$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2}\delta_k \|h(x)\|_2^2, \text{ where } \lambda_k \in \mathbb{R}^m$$
- ▶ Additional degree of freedom  $\lambda$
- ▶ More than linear convergence, if  $\delta_k \rightarrow \infty$  as  $k \rightarrow \infty$  (linear if  $\delta_k$  bounded)
- ▶ Ill-conditioning of the problem limited ( $\delta_k \rightarrow \infty$  not necessary)

## Other methods

### (Preconditioned) Conjugate Gradient method

- ▶ Subsequent direction orthogonal to all previous directions
- ▶ Significantly faster convergence rate
- ▶ Since in this set-up no iteration used, unnecessary increase in computational cost
- ▶ For matrices of specific structure, solve the system
$$P^{-1}Ax = P^{-1}b$$

### Lagrange multiplier method

- ▶ Minimize the *augmented Lagrangian*
$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2} \delta_k \|h(x)\|_2^2, \text{ where } \lambda_k \in \mathbb{R}^m$$
- ▶ Additional degree of freedom  $\lambda$
- ▶ More than linear convergence, if  $\delta_k \rightarrow \infty$  as  $k \rightarrow \infty$  (linear if  $\delta_k$  bounded)
- ▶ Ill-conditioning of the problem limited ( $\delta_k \rightarrow \infty$  not necessary)
- ▶ Convergence rate independent of the growth rate of the penalty parameter

# Outline

Problem statement

Solution

Ornstein-Uhlenbeck process

Outlook

## Ornstein-Uhlenbeck: Overview

The Ornstein-Uhlenbeck process  $\{V(t); t \geq 0\}$  with a drift coefficient  $\beta > 0$  and a diffusion parameter  $\sigma^2$  is defined in terms of a standard Brownian motion  $B(t)$  by scale changes in space and time:

$$V(t) = \nu e^{-\beta t} + \frac{\sigma e^{-\beta t}}{\sqrt{2\beta}} B(e^{2\beta t} - 1) \text{ for } t \geq 0$$

$$V(0) = \nu \text{ if } B(0) = 0.$$

# Ornstein-Uhlenbeck: Overview

The Ornstein-Uhlenbeck process  $\{V(t); t \geq 0\}$  with a drift coefficient  $\beta > 0$  and a diffusion parameter  $\sigma^2$  is defined in terms of a standard Brownian motion  $B(t)$  by scale changes in space and time:

$$V(t) = \nu e^{-\beta t} + \frac{\sigma e^{-\beta t}}{\sqrt{2\beta}} B(e^{2\beta t} - 1) \text{ for } t \geq 0$$

$V(0) = \nu$  if  $B(0) = 0$ .

- ▶ Fourier coefficients follow the Ornstein-Uhlenbeck process

# Ornstein-Uhlenbeck: Overview

The Ornstein-Uhlenbeck process  $\{V(t); t \geq 0\}$  with a drift coefficient  $\beta > 0$  and a diffusion parameter  $\sigma^2$  is defined in terms of a standard Brownian motion  $B(t)$  by scale changes in space and time:

$$V(t) = \nu e^{-\beta t} + \frac{\sigma e^{-\beta t}}{\sqrt{2\beta}} B(e^{2\beta t} - 1) \text{ for } t \geq 0$$

$V(0) = \nu$  if  $B(0) = 0$ .

- ▶ Fourier coefficients follow the Ornstein-Uhlenbeck process
- ▶ To-do: estimate hitting time of Ornstein-Uhlenbeck process

$$\tau_S := \inf\{t > 0 \mid V(t) \in S\}$$

# Outline

Problem statement

Solution

Ornstein-Uhlenbeck process

Outlook

# Outlook

- ▶ Implementation of Gradient-Penalty method in Python

# Outlook

- ▶ Implementation of Gradient-Penalty method in Python
- ▶ Analytic efficiency comparison

# Outlook

- ▶ Implementation of Gradient-Penalty method in Python
- ▶ Analytic efficiency comparison
- ▶ Numerical estimation of OU process hitting time

Thank you!