Acceleration of the 2D Helmholtz model HARES

Gemma van de Sande

Delft University of Technology

May 23, 2012

Outline

Outline

HARES

- HARES $\rightarrow$ HArbour RESonance.

HARES

- HARES $\rightarrow$ HArbour RESonance.
- Determines wave penetration into harbours.

HARES

- HARES $\rightarrow$ HArbour RESonance.
- Determines wave penetration into harbours.
- Uses the non-linear Mild-Slope equation.

HARES

- HARES $\rightarrow$ HArbour RESonance.
- Determines wave penetration into harbours.
- Uses the non-linear Mild-Slope equation.
- Developed by Svašek Hydraulics.
  - ◇ Consultant in coastal, harbour and river engineering.
  - ◇ Specialized in numerical fluid dynamics.

HARES
Example

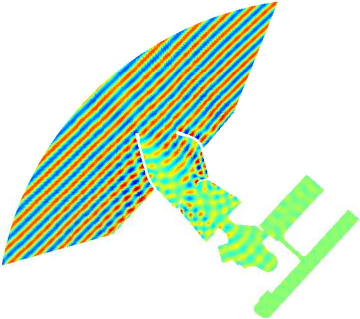


Figure: The harbour of Scheveningen

Project description

PROBLEM

For large domains, when the number of unknowns is large, the computing time becomes undesirably lengthy.

Project description

PROBLEM

For large domains, when the number of unknowns is large, the computing time becomes undesirably lengthy.

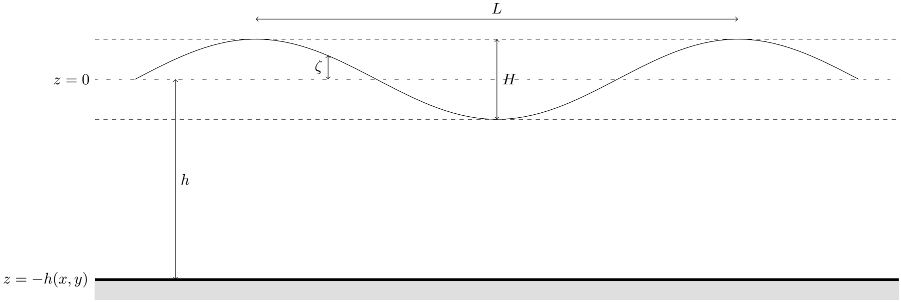
TASK

Accelerate HARES, decrease the computing time.

Outline

1. Introduction
2. Mild-Slope equation
3. Initial implementation
4. Proposed improvements
5. Numerical experiments
6. Conclusions & Recommendations
7. Future research

Wave motion



$h(x, y)$	Water depth	L	Wave length
H	Wave height	$\zeta(x, y, t)$	Elevation of the free surface

Wave motion transforming effects

$$\text{Objects in the domain} \implies \begin{cases} - & \text{Diffraction} \\ - & \text{Reflection} \end{cases}$$

$$\text{Decreasing water depth} \implies \begin{cases} - & \text{Refraction} \\ - & \text{Shoaling} \end{cases}$$

Wave motion transforming effects

$$\left.\begin{array}{ll} - & \text{Diffraction} \\ - & \text{Reflection} \\ & \\ - & \text{Refraction} \\ - & \text{Shoaling} \end{array}\right\} \Longrightarrow \text{Linear Mild-Slope equation}$$

Wave motion transforming effects

$$\left.\begin{array}{ll} - & \text{Diffraction} \\ - & \text{Reflection} \\ & \\ - & \text{Refraction} \\ - & \text{Shoaling} \end{array}\right\} \Longrightarrow \text{Linear Mild-Slope equation}$$

$$\left.\begin{array}{ll} - & \text{Wave breaking} \\ - & \text{Bottom friction} \end{array}\right\} \Longrightarrow \text{Non-linear term in the Mild-Slope equation}$$

Wave motion transforming effects

$$\left.\begin{array}{ll} - & \text{Diffraction} \\ - & \text{Reflection} \\ & \\ - & \text{Refraction} \\ - & \text{Shoaling} \\ & \\ - & \text{Wave breaking} \\ - & \text{Bottom friction} \end{array}\right\} \Longrightarrow \text{Non-linear Mild-Slope equation}$$

To derive the non-linear Mild-Slope equation we make the following assumptions:

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.
- Pressure at the free surface is constant and uniform.

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.
- Pressure at the free surface is constant and uniform.
- Wave slope $\epsilon_s = \frac{2\pi A}{L}$ is small.

Non-linear Mild-Slope equation

Assumptions

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.
- Pressure at the free surface is constant and uniform.
- Wave slope $\epsilon_s = \frac{2\pi A}{L}$ is small.
- Wave motion is harmonic in time.

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.
- Pressure at the free surface is constant and uniform.
- Wave slope $\epsilon_s = \frac{2\pi A}{L}$ is small.
- Wave motion is harmonic in time.
- Surface tension and Coriolis effect can be neglected.

To derive the non-linear Mild-Slope equation we make the following assumptions:

- Water is an ideal fluid, i.e. homogeneous, inviscid, irrotational and incompressible flow.
- Pressure at the free surface is constant and uniform.
- Wave slope $\epsilon_s = \frac{2\pi A}{L}$ is small.
- Wave motion is harmonic in time.
- Surface tension and Coriolis effect can be neglected.
- Changes in bottom topography are small.

Non-linear Mild-Slope equation

The non-linear Mild-Slope equation is given by

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0.$$

With

$n_0(x, y)$	Parameter $n_0 \in [\frac{1}{2}, 1]$
$k_0(x, y)$	Wave number
$\tilde{\zeta}(x, y)$	Elevation of the free surface
$W(x, y, \tilde{\zeta})$	Dissipation of wave energy

ω	Wave frequency
i	$= \sqrt{-1}$
∇	$= \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T$

Non-linear Mild-Slope equation

The non-linear Mild-Slope equation is given by

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0.$$

Non-linearity

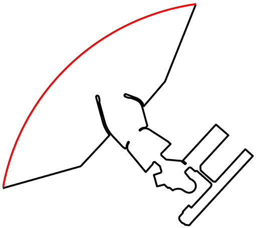
$$W(x, y, \tilde{\zeta})\tilde{\zeta} = \left(\mathcal{A}|\tilde{\zeta}| + \frac{\mathcal{B}}{|\tilde{\zeta}|^2} \right) \tilde{\zeta}$$

We make the distinction between two types of boundaries, i.e.

- The *open boundary* with an incoming wave from the exterior and an outgoing wave from the interior.
- The *closed boundary* where (partial) reflection occurs.

We make the distinction between two types of boundaries, i.e.

- The *open boundary* with an incoming wave from the exterior and an outgoing wave from the interior.
- The *closed boundary* where (partial) reflection occurs.



The condition for the *open boundary* is given by

$$\frac{\partial\tilde{\zeta}}{\partial n} = -i\left\{\hat{p}(\tilde{\zeta}-\tilde{\zeta}_{in}) + \frac{1}{2\hat{p}}\left(\frac{\partial^2\tilde{\zeta}}{\partial s^2} - \frac{\partial^2\tilde{\zeta}_{in}}{\partial s^2}\right) - \hat{p}(\boldsymbol{e}_{in}\cdot\boldsymbol{n})\tilde{\zeta}_{in}\right\}.$$

The condition for the *open boundary* is given by

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left\{ \hat{p}(\tilde{\zeta} - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) - \hat{p}(\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \tilde{\zeta}_{in} \right\}.$$

The condition for the *closed boundary* is given by

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\tilde{\zeta} + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \right\}.$$

The condition for the *open boundary* is given by

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left\{ \hat{p}(\tilde{\zeta} - \tilde{\zeta}_{in}) + \frac{1}{2\hat{p}} \left(\frac{\partial^2 \tilde{\zeta}}{\partial s^2} - \frac{\partial^2 \tilde{\zeta}_{in}}{\partial s^2} \right) - \hat{p}(\boldsymbol{e}_{in} \cdot \boldsymbol{n}) \tilde{\zeta}_{in} \right\}.$$

The condition for the *closed boundary* is given by

$$\frac{\partial \tilde{\zeta}}{\partial n} = -i \left(\frac{1-R}{1+R} \right) \left\{ \hat{p}\tilde{\zeta} + \frac{1}{2\hat{p}} \frac{\partial^2 \tilde{\zeta}}{\partial s^2} \right\}.$$

With

$\hat{p}(x, y, \tilde{\zeta})$	Modified wave number	R	Reflection coefficient
$\tilde{\zeta}_{in}$	Incoming wave	i	$= \sqrt{-1}$

Outline

Structure of HARES

HARES consist of three parts, i.e.

1. Outer loop to deal with the non-linearity of the equation.
   $\rightarrow$ Non-linear Mild-Slope equation is linearised.
2. Spatial discretization of the linearised Mild-Slope equation.
   $\rightarrow$ Results in a system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$.
3. Inner loop to determine the solution of $\boldsymbol{S\zeta} = \boldsymbol{b}$.

Initial implementation

The current programme has the following implementation:

1. Outer loop: Picard iteration.
2. Spatial discretization: Ritz-Galerkin finite element method.
3. Inner loop: ILU(0) - Bi-CGSTAB.

Linearising the non-linear equation

Picard iteration

Using Picard iteration the non-linear Mild-Slope equation is linearised with the following steps:

Using Picard iteration the non-linear Mild-Slope equation is linearised with the following steps:

1. Use the previous iterative solution $\tilde{\zeta}^k$ to compute a value for $W(x, y, \tilde{\zeta})$ and $\hat{p}(x, y, \tilde{\zeta})$.

Using Picard iteration the non-linear Mild-Slope equation is linearised with the following steps:

1. Use the previous iterative solution $\tilde{\zeta}^k$ to compute a value for $W(x, y, \tilde{\zeta})$ and $\hat{p}(x, y, \tilde{\zeta})$.
2. Determine the next iterative solution $\tilde{\zeta}^{k+1}$.

Linearising the non-linear equation

Picard iteration

Using Picard iteration the non-linear Mild-Slope equation is linearised with the following steps:

1. Use the previous iterative solution $\tilde{\zeta}^k$ to compute a value for $W(x, y, \tilde{\zeta})$ and $\hat{p}(x, y, \tilde{\zeta})$.
2. Determine the next iterative solution $\tilde{\zeta}^{k+1}$.
3. Repeat steps 1 & 2 until convergence is reached.

Using Picard iteration the non-linear Mild-Slope equation is linearised with the following steps:

1. Use the previous iterative solution $\tilde{\zeta}^k$ to compute a value for $W(x, y, \tilde{\zeta})$ and $\hat{p}(x, y, \tilde{\zeta})$.
2. Determine the next iterative solution $\tilde{\zeta}^{k+1}$.
3. Repeat steps 1 & 2 until convergence is reached.

The current programme repeats steps 1 & 2 25 times without knowing whether convergence has been reached.

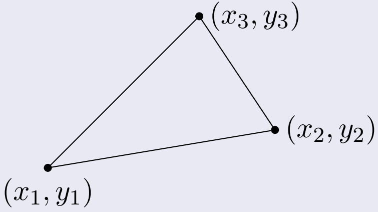
The Ritz-Galerkin finite element method consist of the following steps:

Spatial discretization

Ritz-Galerkin finite element method

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.



- Two types of elements:
  - Internal elements.
  - Boundary elements.
- Number of nodes N = Number of unknowns.

Spatial discretization

Ritz-Galerkin finite element method

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.

Multiply the PDE by a test function $\eta(x, y)$, integrate it over the domain Ω and apply the boundary conditions.

Spatial discretization

Ritz-Galerkin finite element method

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.
3. Approximate the solution by a linear combination of basis functions.

$$\tilde{\zeta}(x,y) \approx \sum_{j=1}^{N} \zeta_j \psi_j(x,y),$$

- $\psi_j(x,y)$ piecewise linear basis function.
- N unknown coefficients ζ_j.

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.
3. Approximate the solution by a linear combination of basis functions.
4. Replace the test function by each of the basis function separately.

$$\eta(x, y) \rightarrow \psi_m(x, y)$$

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.
3. Approximate the solution by a linear combination of basis functions.
4. Replace the test function by each of the basis function separately.
5. Determine the element matrix $\boldsymbol{S}^e$ and element vector $\boldsymbol{b}^e$ for each element, with $\boldsymbol{S}^e \in \mathbb{C}^{3\times 3}$ and $\boldsymbol{b}^e \in \mathbb{C}^3$.

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.
3. Approximate the solution by a linear combination of basis functions.
4. Replace the test function by each of the basis function separately.
5. Determine the element matrix $\boldsymbol{S}^e$ and element vector $\boldsymbol{b}^e$ for each element, with $\boldsymbol{S}^e \in \mathbb{C}^{3\times 3}$ and $\boldsymbol{b}^e \in \mathbb{C}^3$.
6. Obtain the global matrix $\boldsymbol{S}$ and global vector $\boldsymbol{b}$, with $\boldsymbol{S} \in \mathbb{C}^{N\times N}$ and $\boldsymbol{b} \in \mathbb{C}^N$.

$$\boldsymbol{S}^e \to \boldsymbol{S} \quad \text{and} \quad \boldsymbol{b}^e \to \boldsymbol{b}$$

The Ritz-Galerkin finite element method consist of the following steps:

1. Divide the domain into linear triangular elements.
2. Derive the weak formulation of the PDE.
3. Approximate the solution by a linear combination of basis functions.
4. Replace the test function by each of the basis function separately.
5. Determine the element matrix $\boldsymbol{S}^e$ and element vector $\boldsymbol{b}^e$ for each element, with $\boldsymbol{S}^e \in \mathbb{C}^{3\times 3}$ and $\boldsymbol{b}^e \in \mathbb{C}^3$.
6. Obtain the global matrix $\boldsymbol{S}$ and global vector $\boldsymbol{b}$, with $\boldsymbol{S} \in \mathbb{C}^{N\times N}$ and $\boldsymbol{b} \in \mathbb{C}^N$.
7. Compute the solution in each node by solving $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$.

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

Ritz-Galerkin finite element method

Non-linear Mild-Slope equation

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) \quad \Longrightarrow \quad -\frac{n_0}{k_0^2} \boldsymbol{L}^e$$

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

$$\left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} \quad \Longrightarrow \quad \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e$$

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

Boundary conditions $\implies$ $-i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e$

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

- Global matrix $\boldsymbol{S}$ is a symmetric, non-Hermitian, sparse matrix.

$$\nabla \cdot \left(\frac{n_0}{k_0^2} \nabla \tilde{\zeta} \right) + \left(n_0 - \frac{iW}{\omega} \right) \tilde{\zeta} = 0 \quad \text{and BC's}$$

Application of the Ritz-Galerkin finite element method results in element matrices of the following form:

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2} \boldsymbol{L}^e + \left(n_0 - \frac{iW}{\omega} \right) \boldsymbol{M}^e - i \frac{n_0}{k_0^2} \left(\frac{1-R}{1+R} \right) \boldsymbol{C}^e.$$

- Global matrix $\boldsymbol{S}$ is a symmetric, non-Hermitian, sparse matrix.
- Global vector $\boldsymbol{b}$ is completely determined by the incoming wave $\tilde{\zeta}_{in}$.

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S\zeta} = \boldsymbol{b}.$$

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S\zeta} = \boldsymbol{b}.$$

$\boldsymbol{S}$ is a general matrix $\implies$ Krylov subspace methods

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

$\boldsymbol{S}$ is a general matrix $\Longrightarrow$ Krylov subspace methods

- Iterative solution method.

Starting vector ζ_0, iterations ζ_1, $\zeta_2, \ldots, \zeta_m$ until the stopping criterion is satisfied.

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

$\boldsymbol{S}$ is a general matrix $\implies$ Krylov subspace methods

- Iterative solution method.
- Krylov subspace of dimension m is given by

$$\mathcal{K}_m(\boldsymbol{S}; \boldsymbol{r}_0) = span\{\boldsymbol{r}_0, \boldsymbol{S}\boldsymbol{r}_0, \ldots, \boldsymbol{S}^{m-1}\boldsymbol{r}_0\},$$

with $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_0$.

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

$\boldsymbol{S}$ is a general matrix $\implies$ Krylov subspace methods

- Iterative solution method.
- Krylov subspace of dimension m is given by

$$\mathcal{K}_m(\boldsymbol{S}; \boldsymbol{r}_0) = span\{\boldsymbol{r}_0, \boldsymbol{S}\boldsymbol{r}_0, \ldots, \boldsymbol{S}^{m-1}\boldsymbol{r}_0\},$$

  with $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_0$.
- Number of matrix-vector products is an important measure.

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

To accelerate the convergence we can apply a preconditioner $\boldsymbol{K}$ to the system of equations, i.e.

$$\boldsymbol{K}^{-1}\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{K}^{-1}\boldsymbol{b}$$

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

To accelerate the convergence we can apply a preconditioner $\boldsymbol{K}$ to the system of equations, i.e.

$$\boldsymbol{K}^{-1}\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{K}^{-1}\boldsymbol{b}$$

- Preconditioner $\boldsymbol{K}$ is a good approximation of matrix $\boldsymbol{S}$

Solving a system of equations

After linearisation and spatial discretization we obtain the system of equations

$$\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}.$$

To accelerate the convergence we can apply a preconditioner $\boldsymbol{K}$ to the system of equations, i.e.

$$\boldsymbol{K}^{-1}\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{K}^{-1}\boldsymbol{b}$$

- Preconditioner $\boldsymbol{K}$ is a good approximation of matrix $\boldsymbol{S}$
- Constructing the preconditioner $\boldsymbol{K}$ is not too expensive.

- Proposed by H.A. van der Vorst in 1992.

- Proposed by H.A. van der Vorst in 1992.
- Krylov subspace method.

- Proposed by H.A. van der Vorst in 1992.
- Krylov subspace method.
- Finite method, one iterations has two matrix-vector products.

- Proposed by H.A. van der Vorst in 1992.
- Krylov subspace method.
- Finite method, one iterations has two matrix-vector products.
- Stopping criterion for Bi-CGSTAB

$$\frac{\|\boldsymbol{b} - \boldsymbol{S}\zeta_m\|_2}{\|\boldsymbol{b} - \boldsymbol{S}\zeta_0\|_2} \leq \text{TOL}.$$

The system of equations is preconditioned with the incomplete LU decomposition of matrix S.

Solving a system of equations

Preconditioner - Incomplete LU decomposition

The system of equations is preconditioned with the incomplete LU decomposition of matrix $\boldsymbol{S}$.

- $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U} - \boldsymbol{R}$.
  - $\boldsymbol{L}$ lower triangular matrix.
  - $\boldsymbol{U}$ upper triangular matrix.
  - $\boldsymbol{R}$ residual matrix.

The system of equations is preconditioned with the incomplete LU decomposition of matrix $\boldsymbol{S}$.

- $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U} - \boldsymbol{R}$.
- The elements of matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are determined by
  - $\boldsymbol{L}$ and $\boldsymbol{U}$ have the same zero-pattern as $\boldsymbol{S}$, i.e. if $s_{i,j} = 0$ then $u_{i,j} = l_{i,j} = 0$ and if $s_{i,j} \neq 0$ then $u_{i,j} \neq 0$ and $l_{i,j} \neq 0$.
  - diag$(\boldsymbol{L}) = 1$ and diag$(\boldsymbol{U})$ is determined by the algorithm.

The system of equations is preconditioned with the incomplete LU decomposition of matrix $\boldsymbol{S}$.

- $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U} - \boldsymbol{R}$.
- The elements of matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are determined by
  - $\boldsymbol{L}$ and $\boldsymbol{U}$ have the same zero-pattern as $\boldsymbol{S}$, i.e. if $s_{i,j} = 0$ then $u_{i,j} = l_{i,j} = 0$ and if $s_{i,j} \neq 0$ then $u_{i,j} \neq 0$ and $l_{i,j} \neq 0$.
  - diag$(\boldsymbol{L}) = 1$ and diag$(\boldsymbol{U})$ is determined by the algorithm.
- Preconditioning is done by $\boldsymbol{L}^{-1}\boldsymbol{S}\boldsymbol{U}^{-1}\boldsymbol{y} = \boldsymbol{L}^{-1}\boldsymbol{b}$ with $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{x}$.

Outline

Proposed improvements

To reduce the computing time we propose the following solution methods

Proposed improvements

To reduce the computing time we propose the following solution methods

1. Outer loop:
   - Implement a stopping criterion for Picard iteration.
   - Inexact Picard iteration.

Proposed improvements

To reduce the computing time we propose the following solution methods

1. Outer loop:
   - Implement a stopping criterion for Picard iteration.
   - Inexact Picard iteration.
2. Inner loop:
   - IDR(s) combined with the shifted Laplace preconditioner.
   - Direct method `MUMPS`.

Improvement of the outer loop

Stopping criterion for Picard iteration

- Current programme performs 25 outer iterations.
- A suitable stopping criterion is needed to determine when and whether the non-linear solution is obtained.

$$\frac{\|F(\zeta^k)\|_2}{\|F(\zeta^{\mathbf{0}})\|_2} \leq \mathsf{TOL}_{\mathsf{residual}}$$

- Value for $\mathsf{TOL}_{\mathsf{residual}}$ depends on the test case.

Each iteration of Picard iteration we need to determine the solution of the system of equations $\boldsymbol{S\zeta} = \boldsymbol{b}$. This can be done exactly.

Each iteration of Picard iteration we need to determine the solution of the system of equations $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$. This can be done exactly.

However, we can relax this condition with the following stopping criterion

$$\|\boldsymbol{S}\boldsymbol{\zeta}^k - \boldsymbol{b}\|_2 \leq \eta_k \|\boldsymbol{b}\|_2,$$

with

$$\eta_k = \mathsf{TOL} \cdot \frac{\|\boldsymbol{\zeta}^k - \boldsymbol{\zeta}^{k-1}\|_2}{\|\boldsymbol{\zeta}^0\|_2}.$$

- IDR is proposed by P. Sonneveld in 1980.

- IDR is proposed by P. Sonneveld in 1980.
- Krylov subspace method.

- IDR is proposed by P. Sonneveld in 1980.
- Krylov subspace method.
- Generate residuals $\boldsymbol{r}_n$ that are in the subspace $\mathcal{G}_j$ with decreasing dimension.

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A}) \left(\mathcal{G}_{j-1} \cap \boldsymbol{P}^{\perp} \right),$$

with $\mathcal{G}_0 = \mathcal{K}^N(\boldsymbol{A}; \boldsymbol{v}_0)$ and $\boldsymbol{P} \in \mathbb{C}^{N \times s}$.

- IDR is proposed by P. Sonneveld in 1980.
- Krylov subspace method.
- Generate residuals $\boldsymbol{r}_n$ that are in the subspace $\mathcal{G}_j$ with decreasing dimension.
- Based on the IDR theorem

(i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ for all $\mathcal{G}_{j-1} \neq \{\mathbf{0}\}, j > 0$,

(ii) $\mathcal{G}_j = \{\mathbf{0}\}$ for some $j \leq N$.

- IDR is proposed by P. Sonneveld in 1980.
- Krylov subspace method.
- Generate residuals $\boldsymbol{r}_n$ that are in the subspace $\mathcal{G}_j$ with decreasing dimension.
- Based on the IDR theorem

(i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$ for all $\mathcal{G}_{j-1} \neq \{\mathbf{0}\}, j > 0$,

(ii) $\mathcal{G}_j = \{\mathbf{0}\}$ for some $j \leq N$.

$\Longrightarrow$ Finite method, requires at most $N + \frac{N}{s}$ matrix-vector multiplications.

- IDR is proposed by P. Sonneveld in 1980.
- Krylov subspace method.
- Generate residuals $\boldsymbol{r}_n$ that are in the subspace $\mathcal{G}_j$ with decreasing dimension.
- Based on the IDR theorem
- Stopping criterion implemented in IDR(s)

$$\frac{\|\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\zeta}_m\|_2}{\|\boldsymbol{b}\|_2} \leq \text{TOL}.$$

For each element the shifted Laplace preconditioner is given by

$$\boldsymbol{K}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e$$

with $\boldsymbol{K}^e \in \mathbb{C}^{3\times 3}$ and ξ^2 the shift parameter.

For each element the shifted Laplace preconditioner is given by

$$\boldsymbol{K}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2 \boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e$$

with $\boldsymbol{K}^e \in \mathbb{C}^{3\times 3}$ and ξ^2 the shift parameter.

Very similar to the element matrices

$$\boldsymbol{S}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^2 + \left(n_0 - \frac{iW}{\omega}\right)\boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e.$$

For each element the shifted Laplace preconditioner is given by

$$\boldsymbol{K}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e$$

with $\boldsymbol{K}^e \in \mathbb{C}^{3\times 3}$ and ξ^2 the shift parameter.

- The global preconditioner $\boldsymbol{K} \in \mathbb{C}^{N\times N}$ is computed from the matrices $\boldsymbol{K}^e$.

For each element the shifted Laplace preconditioner is given by

$$\boldsymbol{K}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2 \boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e$$

with $\boldsymbol{K}^e \in \mathbb{C}^{3\times 3}$ and ξ^2 the shift parameter.

- The global preconditioner $\boldsymbol{K} \in \mathbb{C}^{N\times N}$ is computed from the matrices $\boldsymbol{K}^e$.
- Approximate inverse of $\boldsymbol{K}$ by its incomplete LU decomposition.

For each element the shifted Laplace preconditioner is given by

$$\boldsymbol{K}^e = -\frac{n_0}{k_0^2}\boldsymbol{L}^e - \xi^2\boldsymbol{M}^e - i\frac{n_0}{k_0^2}\left(\frac{1-R}{1+R}\right)\boldsymbol{C}^e$$

with $\boldsymbol{K}^e \in \mathbb{C}^{3\times 3}$ and ξ^2 the shift parameter.

- The global preconditioner $\boldsymbol{K} \in \mathbb{C}^{N\times N}$ is computed from the matrices $\boldsymbol{K}^e$.
- Approximate inverse of $\boldsymbol{K}$ by its incomplete LU decomposition.
- Use the shift $\xi^2 = i\left|n_0 - \frac{iW}{\omega}\right|$.

Solving a system of equations
Direct method `MUMPS`

- `MUMPS` - MUltifrontal Massively Parallel Solver.

- MUMPS - MUltifrontal Massively Parallel Solver.
- Determines the solution of the system of equations $\boldsymbol{S}\zeta = \boldsymbol{b}$, where $\boldsymbol{S}$ is a square sparse matrix.

Solving a system of equations
Direct method MUMPS

- MUMPS - MUltifrontal Massively Parallel Solver.
- Determines the solution of the system of equations $\boldsymbol{S}\zeta = \boldsymbol{b}$, where $\boldsymbol{S}$ is a square sparse matrix.
- Computes the LU factorization of the matrix $\boldsymbol{S}$, i.e. $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U}$.

- MUMPS - MUltifrontal Massively Parallel Solver.
- Determines the solution of the system of equations $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$, where $\boldsymbol{S}$ is a square sparse matrix.
- Computes the LU factorization of the matrix $\boldsymbol{S}$, i.e. $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U}$.
- Obtains the solution by $\zeta = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{b}$.

- `MUMPS` - MUltifrontal Massively Parallel Solver.
- Determines the solution of the system of equations $\boldsymbol{S}\boldsymbol{\zeta} = \boldsymbol{b}$, where $\boldsymbol{S}$ is a square sparse matrix.
- Computes the LU factorization of the matrix $\boldsymbol{S}$, i.e. $\boldsymbol{S} = \boldsymbol{L}\boldsymbol{U}$.
- Obtains the solution by $\boldsymbol{\zeta} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{b}$.
- Available in a sequential and parallel version.

Outline

1. Introduction
2. Mild-Slope equation
3. Initial implementation
4. Proposed improvements
5. Numerical experiments
6. Conclusions & Recommendations
7. Future research

Four test cases are considered:

1. Harbour of Scheveningen
    - 63,253 unknowns

Numerical experiments

Test cases

Four test cases are considered:

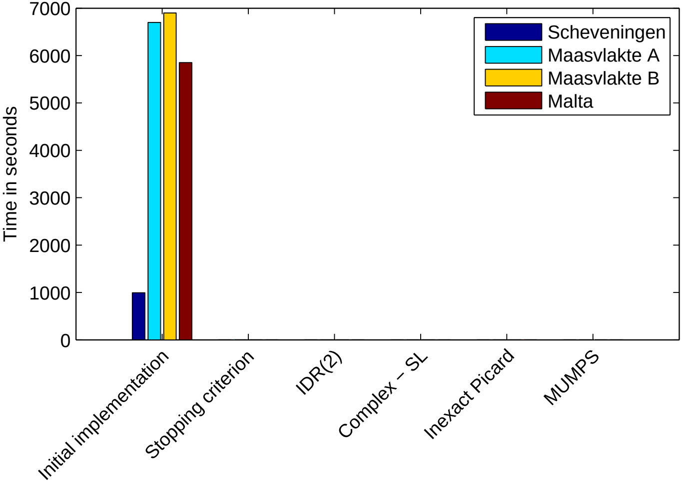
1. Harbour of Scheveningen
   - 63,253 unknowns
2. Maasvlakte - bottom topography A
   - 173,612 unknowns
3. Maasvlakte - bottom topography B
   - 173,612 unknowns

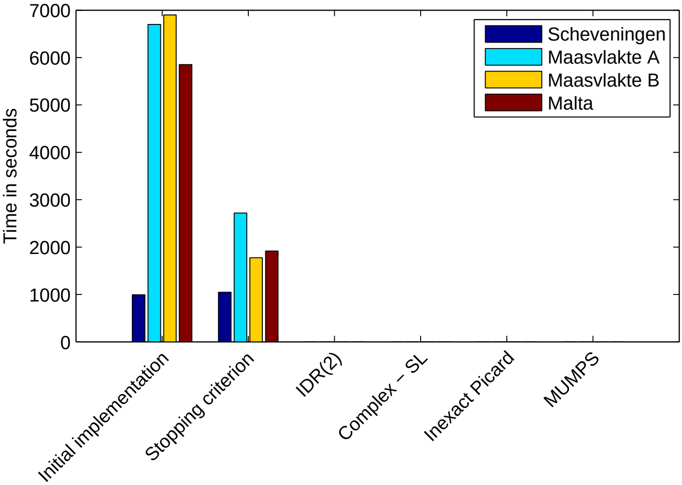
Four test cases are considered:

1. Harbour of Scheveningen
   - 63,253 unknowns
2. Maasvlakte - bottom topography A
   - 173,612 unknowns
3. Maasvlakte - bottom topography B
   - 173,612 unknowns
4. Harbour of Marsaxlokk - Malta
   - 170,423 unknowns

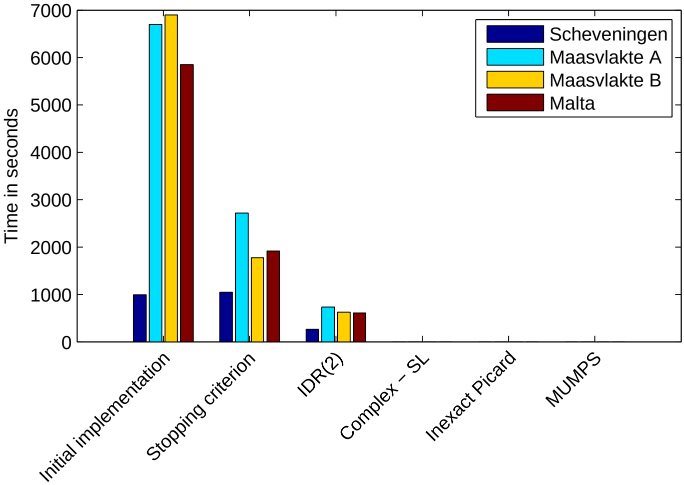
Numerical experiments

Results - computing time



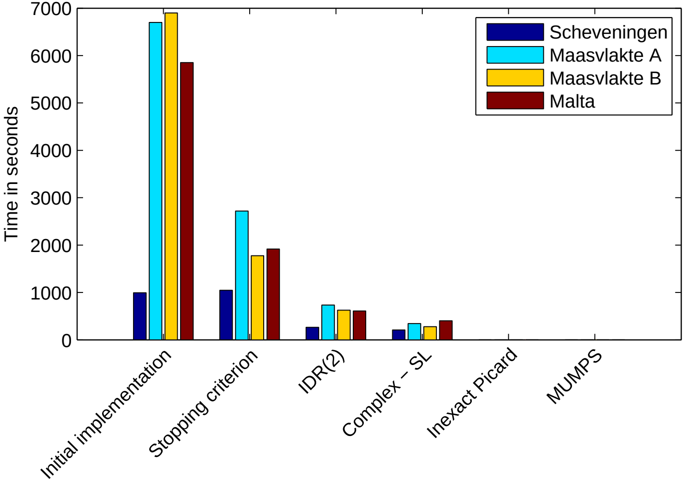

7000
6000
5000
4000
3000
2000
1000
0
Time in seconds
Scheveningen
Maasvlakte A
Maasvlakte B
Malta
Initial implementation
Stopping criterion
IDR(2)
Complex – SL
Inexact Picard
MUMPS

Numerical experiments
Results - computing time

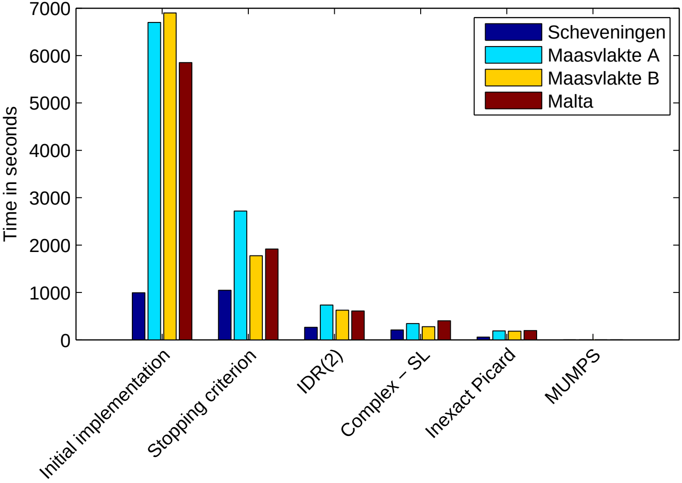


Numerical experiments

Results - computing time

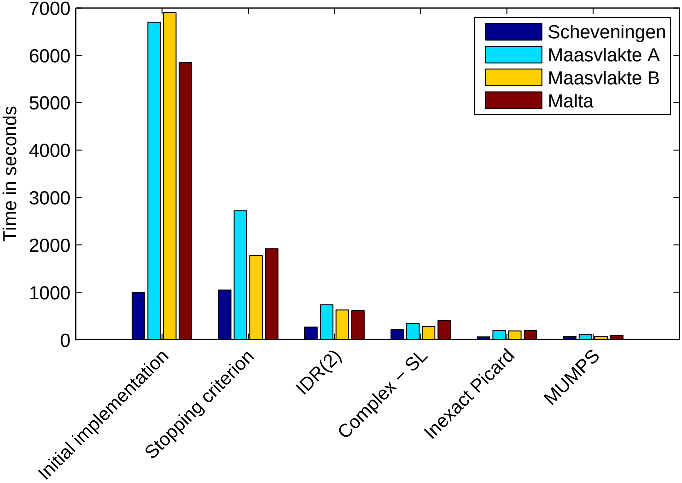


Numerical experiments
Results - computing time



Numerical experiments

Results - computing time



After implementing the proposed improvements we need the following percentages of the computing time of the initial implementation.

	Scheveningen	Maasvlakte A	Maasvlakte B	Malta
Iterative	5.8 %	2.8 %	2.7 %	3.4 %
Direct	7.0 %	1.6 %	1.0 %	1.5 %

Outline

Conclusions & Recommendations

- Proposed improvements for the iterative solver are upto 35 times faster than the initial implementation.

Conclusions & Recommendations

- Proposed improvements for the iterative solver are upto 35 times faster than the initial implementation.
- The number of matrix-vector products is reduced by a factor 58.

Conclusions & Recommendations

- Proposed improvements for the iterative solver are upto 35 times faster than the initial implementation.
- The number of matrix-vector products is reduced by a factor 58.
- Using the direct method `MUMPS` the computing time is upto 100 times faster than the original implementation.

Conclusions & Recommendations

- Proposed improvements for the iterative solver are upto 35 times faster than the initial implementation.
- The number of matrix-vector products is reduced by a factor 58.
- Using the direct method `MUMPS` the computing time is upto 100 times faster than the original implementation.

- Use a direct method, e.g. `MUMPS`, to determine the solution of the system of equations.

Conclusions & Recommendations

- Proposed improvements for the iterative solver are upto 35 times faster than the initial implementation.
- The number of matrix-vector products is reduced by a factor 58.
- Using the direct method `MUMPS` the computing time is upto 100 times faster than the original implementation.

- Use a direct method, e.g. `MUMPS`, to determine the solution of the system of equations.
- If the dimension of the sparse matrix is considerably larger we propose inexact Picard iteration, where the system of equations is solved using IDR(s) preconditioned with the shifted Laplace preconditioner.

Outline

Future research

- Parallel version of the direct method `MUMPS`.

Future research

- Parallel version of the direct method `MUMPS`.
- Parallel computation of the global matrix $\boldsymbol{S}$.

Future research

- Parallel version of the direct method `MUMPS`.
- Parallel computation of the global matrix $\boldsymbol{S}$.
- Approximation of the complete LU factorization of the shifted Laplace preconditioner.

Future research

- Parallel version of the direct method `MUMPS`.
- Parallel computation of the global matrix $\boldsymbol{S}$.
- Approximation of the complete LU factorization of the shifted Laplace preconditioner.
- Inexact Picard iteration based on a different forcing sequence.

Numerical experiments

Computing time - logarithmic scale