Computing Flexural-Gravity Waves in Three Dimensions

Olga Trichtchenko

Applied Mathematics
University of Washington
ota6@uw.edu

November 3, 2016

Acknowledgements

This is joint work with

- **Jean-Marc Vanden-Broeck** at University College London
- **Emilian Părău** at UEA
- **Paul Milewski** at University of Bath

Outline

Outline

Goal

Compute solutions to Euler's equations as efficiently and as accurately as possible.

Outline

Model for Water Waves

For an inviscid, incompressible fluid with velocity potential $\phi(x, y, z, t)$, the forced Euler's equations are given by

$$\begin{cases} \triangle\phi = 0, & (x, y, z) \in \Omega, \\ \phi_z = 0, & z = -h, \\ \eta_t + \eta_x\phi_x + \eta_y\phi_y = \phi_z, & z = \eta(x, y, t), \\ \phi_t + \frac{1}{2}|\nabla\phi|^2 + \frac{1}{F^2}\eta + P(x, y, t) = -D\frac{\delta H}{\delta\eta}, & z = \eta(x, y, t), \end{cases}$$

where

h: depth

F: Froude number

D: flexural rigidity

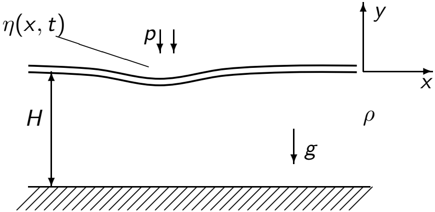
$\eta(x, y, t)$: variable surface

$P(x, y, t)$: external pressure distribution

$\frac{\delta H}{\delta\eta}$: condition at the interface.

$\Omega = \{-\infty < x < \infty, -\infty < y < \infty, -\infty < z < \eta(x, y, t)\}$

Models For a Thin Sheet of Ice



We consider two models

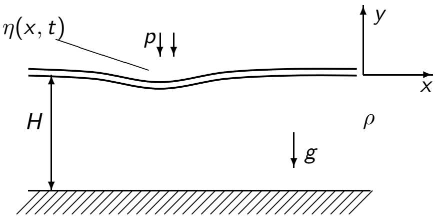
- Biharmonic (linear) model

$$H_L = D\frac{1}{2}\int (\triangle\eta)^2 dA$$

- Cosserat (nonlinear) model

$$H_N = D\frac{1}{2}\int (\kappa_1+\kappa_2)^2 dS \quad \text{with } \kappa_1, \kappa_2 \text{ principle curvatures}$$

Models For a Thin Sheet of Ice



We consider two models

- Biharmonic (linear) model

$$H_L = D\frac{1}{2}\int (\triangle\eta)^2 dA$$

- Cosserat (nonlinear) model

$$H_N = D\frac{1}{2}\int (\kappa_1 + \kappa_2)^2 dS \quad \text{with } \kappa_1, \kappa_2 \text{ principle curvatures}$$

Reformulation: Bernoulli Equation

- Switch into surface variables via the velocity potential at the surface

$$q(x, y, t) = \phi(x, y, z = \eta, t)$$

- Go into a moving frame of reference
- Combine the dynamic and kinematic boundary conditions

Then the steady-state Bernoulli equation becomes

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2 + (1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2} + \frac{\eta}{F^2} + P + D\frac{\delta H}{\delta \eta} = \frac{1}{2}$$

Reformulation: Bernoulli Equation

- Switch into surface variables via the velocity potential at the surface

$$q(x, y, t) = \phi(x, y, z = \eta, t)$$

- Go into a moving frame of reference
- Combine the dynamic and kinematic boundary conditions

Then the steady-state Bernoulli equation becomes

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2 + (1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2} + \frac{\eta}{F^2} + P + D\frac{\delta H}{\delta \eta} = \frac{1}{2}$$

Reformulation: Bernoulli Equation

- Switch into surface variables via the velocity potential at the surface

$$q(x, y, t) = \phi(x, y, z = \eta, t)$$

- Go into a moving frame of reference
- Combine the dynamic and kinematic boundary conditions

Then the steady-state Bernoulli equation becomes

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2 + (1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2} + \frac{\eta}{F^2} + P + D\frac{\delta H}{\delta \eta} = \frac{1}{2}$$

Reformulation: Bernoulli Equation

- Switch into surface variables via the velocity potential at the surface

$$q(x, y, t) = \phi(x, y, z = \eta, t)$$

- Go into a moving frame of reference
- Combine the dynamic and kinematic boundary conditions

Then the steady-state Bernoulli equation becomes

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2 + (1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2} + \frac{\eta}{F^2} + P + D\frac{\delta H}{\delta \eta} = \frac{1}{2}$$

Reformulation: Bernoulli Equation

- Switch into surface variables via the velocity potential at the surface

$$q(x, y, t) = \phi(x, y, z = \eta, t)$$

- Go into a moving frame of reference
- Combine the dynamic and kinematic boundary conditions

Then the steady-state Bernoulli equation becomes

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2 + (1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2} + \frac{\eta}{F^2} + P + D\frac{\delta H}{\delta \eta} = \frac{1}{2}$$

Models for Ice

The two different models are considered

- Biharmonic (linear) model

$$\frac{\delta H}{\delta \eta} = \nabla^4 \eta$$

- Cosserat (nonlinear) model

$$\frac{\delta H}{\delta \eta} = \frac{2}{\sqrt{a}} \left[\partial_x \left(\frac{1+\eta_y^2}{\sqrt{a}} \partial_x H \right) - \partial_x \left(\frac{\eta_x \eta_y}{\sqrt{a}} \partial_y H \right) - \partial_y \left(\frac{\eta_x \eta_y}{\sqrt{a}} \partial_x H \right) + \partial_y \left(\frac{1+\eta_x^2}{\sqrt{a}} \partial_y H \right) \right] + 4H^3 - 4KH$$

where

$$a = 1 + \eta_x^2 + \eta_y^2$$

$$H = \frac{1}{2} a^{3/2} \left[(1+\eta_y^2)\eta_{xx} - 2\eta_{xy}\eta_x\eta_y + (1+\eta_x^2)\eta_{yy} \right]$$

$$K = \frac{1}{a^2} \left[\eta_{xx}\eta_{yy} - \eta_{xy}^2 \right]$$

Models for Ice

The two different models are considered

- Biharmonic (linear) model

$$\frac{\delta H}{\delta \eta} = \nabla^4 \eta$$

- Cosserat (nonlinear) model

$$\frac{\delta H}{\delta \eta} = \frac{2}{\sqrt{a}} \left[\partial_x \left(\frac{1+\eta_y^2}{\sqrt{a}} \partial_x H \right) - \partial_x \left(\frac{\eta_x \eta_y}{\sqrt{a}} \partial_y H \right) - \partial_y \left(\frac{\eta_x \eta_y}{\sqrt{a}} \partial_x H \right) + \partial_y \left(\frac{1+\eta_x^2}{\sqrt{a}} \partial_y H \right) \right] + 4H^3 - 4KH$$

where

$$a = 1 + \eta_x^2 + \eta_y^2$$

$$H = \frac{1}{2} a^{3/2} \left[(1+\eta_y^2)\eta_{xx} - 2\eta_{xy}\eta_x\eta_y + (1+\eta_x^2)\eta_{yy} \right]$$

$$K = \frac{1}{a^2} \left[\eta_{xx}\eta_{yy} - \eta_{xy}^2 \right]$$

Reformulation: Boundary Integral Method

Following the formulation by Forbes (1989), use Green's second identity

$$\int_V (\alpha \Delta \beta - \beta \Delta \alpha) dV = \oint_{S(V)} \left(\alpha \frac{\partial \beta}{\partial n} - \beta \frac{\partial \alpha}{\partial n} \right) dS$$

where in three dimensions, β is the fundamental solution given by the Green's function

$$\frac{1}{4\pi} \frac{1}{((x - x^*)^2 + (y - y^*)^2 + (z - z^*)^2)^{1/2}}$$

and $\alpha = \phi - x$, which satisfies Laplace's equation.

System of Equations

The final form of equations to solve for flexural-gravity waves in infinite depth is

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2+(1+\eta_y^2)q_x^2-2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2}+\frac{\eta}{F^2}+P+D\frac{\delta H}{\delta\eta}=\frac{1}{2}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[(q-q^*-x+x^*)K_1+\eta_x K_2\right]dxdy=2\pi(q^*-x^*)$$

where

$$K_1=\frac{1}{d^{3/2}}(\eta-\eta^*-(x-x^*)^2\eta_x-(y-y^*)^2\eta_y)$$

$$K_2=\frac{1}{d^{1/2}}$$

with

$$d(x,y,x^*,y^*,\eta)=(x-x^*)^2+(y-y^*)^2+(\eta-\eta^*)^2$$

.

Symmetry

Symmetry in y direction

$$\eta(x,y) = \eta(x,-y)$$

and

$$q(x,y) = q(x,-y)$$

implies additional terms

$$\frac{1}{2}\frac{(1+\eta_x^2)q_y^2+(1+\eta_y^2)q_x^2 - 2\eta_x\eta_y q_x q_y}{1+\eta_x^2+\eta_y^2}+\frac{\eta}{F}-\frac{1}{2} = F(\eta)$$

$$\int_0^\infty \int_{-\infty}^\infty \left[(q-q^*-x+x^*)\tilde{K}_1 + \eta_x \tilde{K}_2\right] dxdy = 2\pi(q^*-x^*)$$

where

$$\tilde{K}_1 = \bar{K}_1(x,y,\eta,x^*,y^*,\eta^*) + \bar{K}_1(x,-y,\eta,x^*,y^*,\eta^*)$$
$$\tilde{K}_2 = \bar{K}_2(x,y,\eta,x^*,y^*,\eta^*) + \bar{K}_2(x,-y,\eta,x^*,y^*,\eta^*)$$

Outline

Discretisation

- Let x_i and y_j be equally spaced points such that $i = 1, \ldots, N$ and $j = 1, \ldots, M$.
- Let the vector of unknowns be $q_{x(i,j)}$ and $\eta_{x(i,j)}$ such that

$$u = \left[q_{x(1,1)}, \cdots, q_{x(N,1)}, \cdots, q_{x(N,M)}, \eta_{x(1,1)}, \cdots, \eta_{x(N,M)}\right]^T$$

- Use finite differences to discretise the derivatives
- Obtain 2NM equations

$$G(u) = 0$$

Discretisation

- Let x_i and y_j be equally spaced points such that $i = 1, \ldots, N$ and $j = 1, \ldots, M$.
- Let the vector of unknowns be $q_{x(i,j)}$ and $\eta_{x(i,j)}$ such that

$$u = \left[q_{x(1,1)}, \cdots, q_{x(N,1)}, \cdots, q_{x(N,M)}, \eta_{x(1,1)}, \cdots, \eta_{x(N,M)} \right]^T$$

- Use finite differences to discretise the derivatives
- Obtain 2NM equations

$$G(u) = 0$$

Discretisation

- Let x_i and y_j be equally spaced points such that $i = 1, \ldots, N$ and $j = 1, \ldots, M$.
- Let the vector of unknowns be $q_{x(i,j)}$ and $\eta_{x(i,j)}$ such that

$$u = \left[q_{x(1,1)}, \cdots, q_{x(N,1)}, \cdots, q_{x(N,M)}, \eta_{x(1,1)}, \cdots, \eta_{x(N,M)}\right]^T$$

- Use finite differences to discretise the derivatives
- Obtain 2NM equations

$$G(u) = 0$$

Discretisation

- Let x_i and y_j be equally spaced points such that $i = 1, \ldots, N$ and $j = 1, \ldots, M$.
- Let the vector of unknowns be $q_{x(i,j)}$ and $\eta_{x(i,j)}$ such that

$$u = \left[q_{x(1,1)}, \cdots, q_{x(N,1)}, \cdots, q_{x(N,M)}, \eta_{x(1,1)}, \cdots, \eta_{x(N,M)}\right]^T$$

- Use finite differences to discretise the derivatives
- Obtain 2NM equations

$$G(u) = 0$$

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n$, $0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n,\ 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n, \; 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n, \ 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n,\ 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n, \ 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Numerical Approach

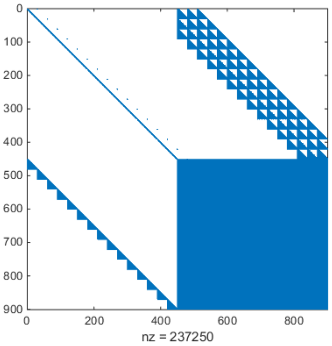
To solve the system

1. Set up an initial guess u^0
2. Until convergence
   2.1 Solve $J(u^n)\delta^n = -G(u^n)$
   2.2 Set $u^{n+1} = u^n + \lambda\delta^n,\ 0 < \lambda < 1$
   2.3 Test for convergence

This method relies on an initial guess u^0 and the Jacobian J.

Jacobian

The sparsity of the linearised Jacobian for flexural-gravity waves



Solving the System of Equations

We consider two ways of solving the system of equations

1. Inexact Newton Method: (direct method) uses an inexact Jacobian (not computed at each step).
2. Modified Newton Method: (iterative method) using a preconditioned Krylov method to construct the solution, not keeping the full Jacobian matrix.
    - Preconditioner is constructed as shown in Pethiyagoda *et al* (2014)
    - Krylov subspace methods implemented using Sundials solver KINSOL implemented in Matlab and C.

Solving the System of Equations

We consider two ways of solving the system of equations

1. Inexact Newton Method: (direct method) uses an inexact Jacobian (not computed at each step).
2. Modified Newton Method: (iterative method) using a preconditioned Krylov method to construct the solution, not keeping the full Jacobian matrix.
   - Preconditioner is constructed as shown in Pethiyagoda *et al* (2014)
   - Krylov subspace methods implemented using Sundials solver KINSOL implemented in Matlab and C.

Solving the System of Equations

We consider two ways of solving the system of equations

1. Inexact Newton Method: (direct method) uses an inexact Jacobian (not computed at each step).
2. Modified Newton Method: (iterative method) using a preconditioned Krylov method to construct the solution, not keeping the full Jacobian matrix.
    - Preconditioner is constructed as shown in Pethiyagoda *et al* (2014)
    - Krylov subspace methods implemented using Sundials solver KINSOL implemented in Matlab and C.

Solving the System of Equations

We consider two ways of solving the system of equations

1. Inexact Newton Method: (direct method) uses an inexact Jacobian (not computed at each step).
2. Modified Newton Method: (iterative method) using a preconditioned Krylov method to construct the solution, not keeping the full Jacobian matrix.
   - Preconditioner is constructed as shown in Pethiyagoda *et al* (2014)
   - Krylov subspace methods implemented using Sundials solver KINSOL implemented in Matlab and C.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Initial Condition

Newton's method is very sensitive to initial conditions. In order to compute different wave amplitudes, generate a bifurcation diagram

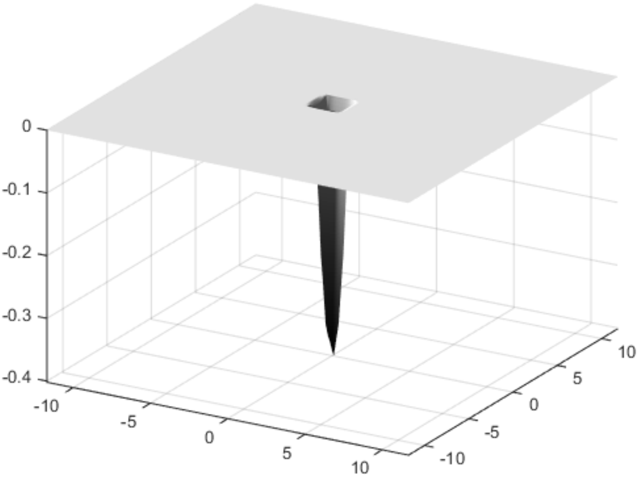
- Guess a small amplitude solution
- Use this guess in Newton's method to compute the true solution.
- Scale the previous solution to get a guess for a larger amplitude solution
- Apply Newton's method to find the true solution.

Can use the Jacobians from previous steps in the bifurcation branch as preconditioners.

Outline

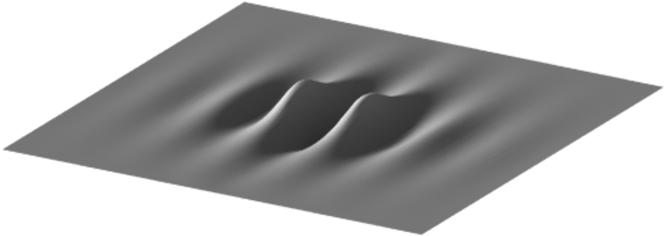
Forcing Term

We use the following pressure as a forcing for depression waves



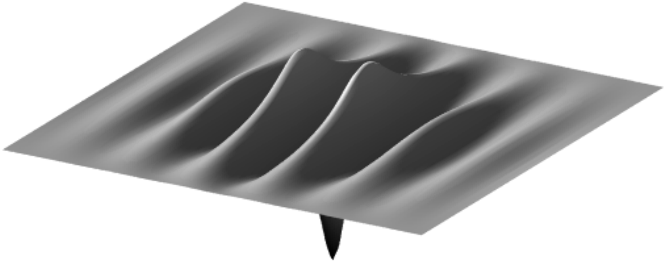
Sample Solutions

Solutions for forced waves underneath an ice sheet



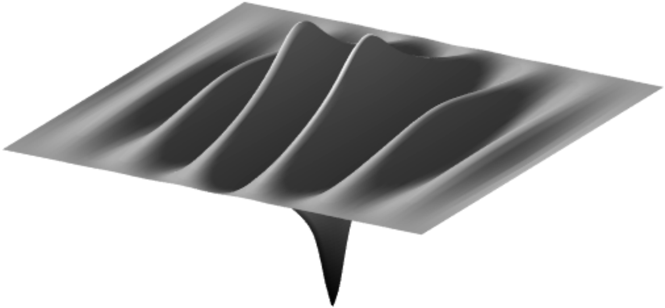
Sample Solutions

Solutions for forced waves underneath an ice sheet



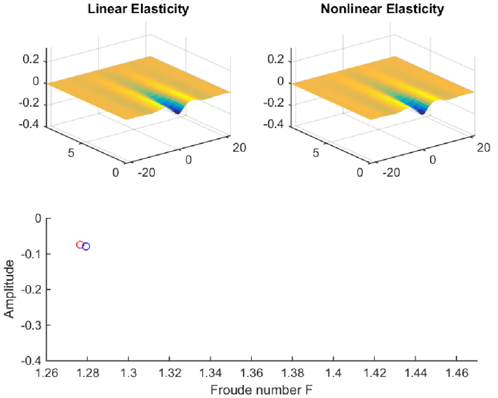
Sample Solutions

Solutions for forced waves underneath an ice sheet



Bifurcation Branch

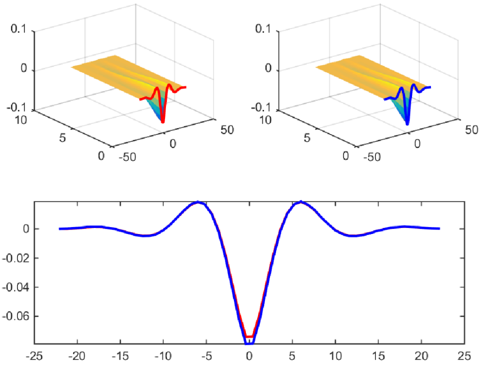
Comparison of the bifurcation branches for flexural-gravity waves with the linear and the nonlinear elasticity models



Note: both models give the same wave amplitude, but different Froude numbers

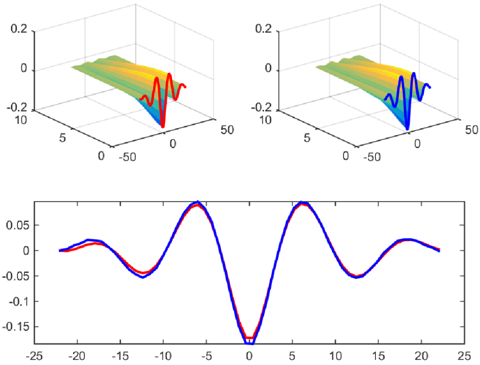
Flexural-Gravity Wave Profiles

Comparison of the solution profiles for linear elasticity model and the nonlinear elasticity model.



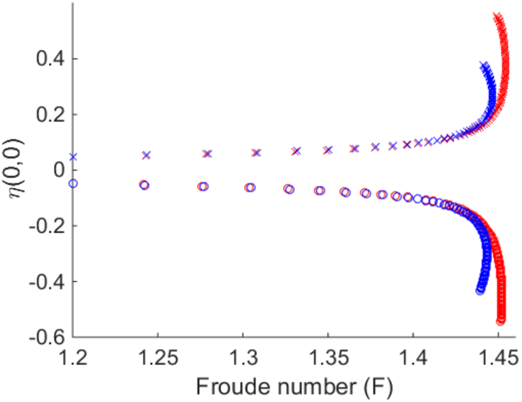
Flexural-Gravity Wave Profiles

Comparison of the solution profiles for linear elasticity model and the nonlinear elasticity model.



Flexural-Gravity Bifurcation Branch

Comparison of the bifurcation branch for linear elasticity model and the nonlinear elasticity model.



Elevation waves are represented as crosses and depression waves as circles.

Outline

Conclusions

- Can compute solutions to both models for flexural-gravity waves
- Both models for produce similar shaped profiles, but at different Froude numbers
- The code is easy to use and easy to modify
- A variety of numerical methods have been tested

Conclusions

- Can compute solutions to both models for flexural-gravity waves
- Both models for produce similar shaped profiles, but at different Froude numbers
- The code is easy to use and easy to modify
- A variety of numerical methods have been tested

Conclusions

- Can compute solutions to both models for flexural-gravity waves
- Both models for produce similar shaped profiles, but at different Froude numbers
- The code is easy to use and easy to modify
- A variety of numerical methods have been tested

Conclusions

- Can compute solutions to both models for flexural-gravity waves
- Both models for produce similar shaped profiles, but at different Froude numbers
- The code is easy to use and easy to modify
- A variety of numerical methods have been tested

Future Work

- Compute accurate free surface waves without a forcing
- Compare the different models quantitatively
- Do free surface depression or elevation waves bifurcate away from 0?
- Switch to using a Modified Newton Method for solutions

Future Work

- Compute accurate free surface waves without a forcing
- Compare the different models quantitatively
- Do free surface depression or elevation waves bifurcate away from 0?
- Switch to using a Modified Newton Method for solutions

Future Work

- Compute accurate free surface waves without a forcing
- Compare the different models quantitatively
- Do free surface depression or elevation waves bifurcate away from 0?
- Switch to using a Modified Newton Method for solutions

Future Work

- Compute accurate free surface waves without a forcing
- Compare the different models quantitatively
- Do free surface depression or elevation waves bifurcate away from 0?
- Switch to using a Modified Newton Method for solutions

Thank you for your attention