Still water and other hard-to-model systems

Martin Büttner
Supervisor: Prof. Ted Johnson

16 March, 2015

The Shallow Water Equations

- Simplified fluid model:
  - Incompressible
  - Inviscid
  - *Shallow* (horizontal length scale $\gg$ typical depth)
- Used to model oceans
  and atmosphere

The Shallow Water Equations

- Simplified fluid model:
  - Incompressible
  - Inviscid
  - *Shallow* (horizontal length scale $\gg$ typical depth)
- Used to model oceans ($\sim$ 4km deep)
  and atmosphere ($\sim$ 10km deep)

The Shallow Water Equations

- Simplified fluid model:
  - Incompressible
  - Inviscid
  - *Shallow* (horizontal length scale $\gg$ typical depth)
- Used to model oceans ($\sim$ 4km deep) and atmosphere ($\sim$ 10km deep)

$$h_t + (hu)_x = 0 \tag{1}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0 \tag{2}$$

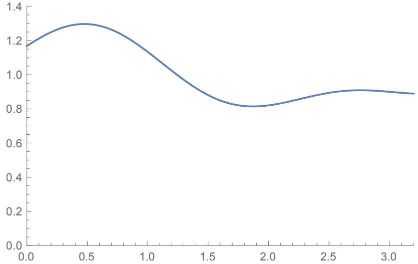
$$(hv)_t + (huv)_x = 0 \tag{3}$$

Finite Volume Methods

- Developed for *hyperbolic conservation laws*:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{4}$$

- Discretisation:

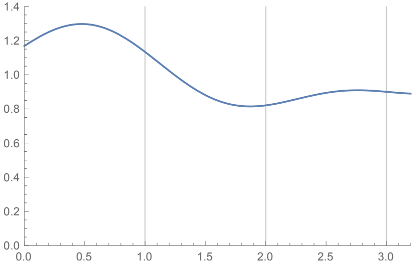


Finite Volume Methods

- Developed for *hyperbolic conservation laws*:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{4}$$

- Discretisation:

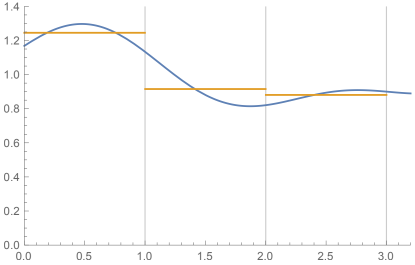


Finite Volume Methods

- Developed for *hyperbolic conservation laws*:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{4}$$

- Discretisation:

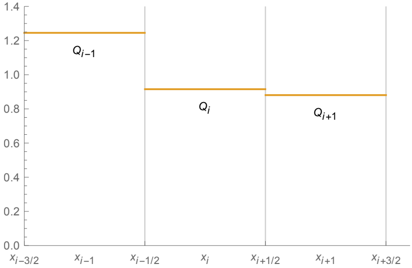


Finite Volume Methods

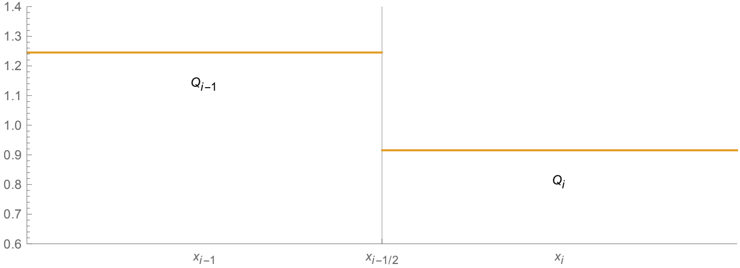
- Developed for *hyperbolic conservation laws*:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \qquad (4)$$

- Discretisation:

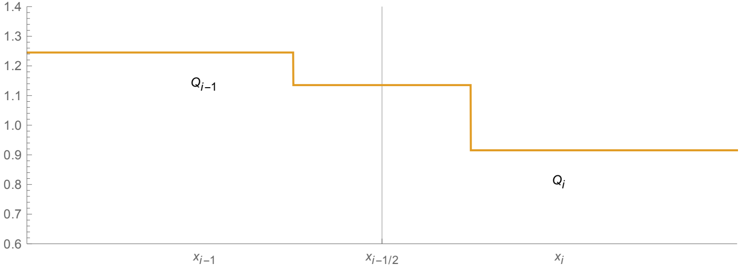


Riemann Problems



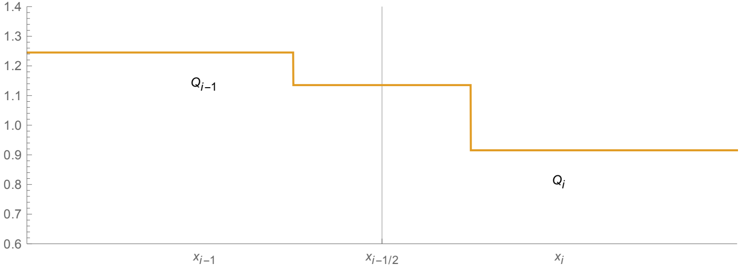
- Initial value problem: step function

Riemann Problems



- Initial value problem: step function
- Decompose into waves

Riemann Problems



- Initial value problem: step function
- Decompose into waves
- Finite speeds $\Rightarrow$ only affect neighbouring cells

Godunov's Method

- Wave propagation form:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}) \qquad (5)$$

Godunov's Method

- Wave propagation form:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(\mathcal{A}^+\Delta Q_{i-1/2} + \mathcal{A}^-\Delta Q_{i+1/2}) \qquad (5)$$

$$\mathcal{A}^-\Delta Q_{i-1/2} = \sum_{p=1}^{M_w}(s_{i-1/2}^p)^-\mathcal{W}_{i-1/2}^p \qquad (6)$$

$$\mathcal{A}^+\Delta Q_{i+1/2} = \sum_{p=1}^{M_w}(s_{i+1/2}^p)^+\mathcal{W}_{i+1/2}^p \qquad (7)$$

Implementation: Clawpack

- “**C**onservation **law pack**age”, by Randall J. LeVeque, 1994

Implementation: Clawpack

- "**C**onservation **law pack**age", by Randall J. LeVeque, 1994
- Riemann solvers in Fortran 90
  - Take grid data, compute wave decomposition

Implementation: Clawpack

- "**C**onservation **law pack**age", by Randall J. LeVeque, 1994
- Riemann solvers in Fortran 90
  - Take grid data, compute wave decomposition
- Configuration in Python 2
  - Grid setup
  - Initial conditions
  - Boundary conditions
  - Plotting instructions

Adding Source Terms

- Modelling additional effects:

$$h_t + (hu)_x = 0 \tag{8}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0 \tag{9}$$

$$(hv)_t + (huv)_x = 0 \tag{10}$$

Adding Source Terms

- Modelling additional effects:

$$h_t + (hu)_x = 0 \tag{8}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = -hB_x \tag{9}$$

$$(hv)_t + (huv)_x = 0 \tag{10}$$

Adding Source Terms

- Modelling additional effects:

$$h_t + (hu)_x = 0 \tag{8}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = -hB_x + Khv \tag{9}$$

$$(hv)_t + (huv)_x = -Khu \tag{10}$$

Adding Source Terms

- Modelling additional effects:

$$h_t + (hu)_x = 0 \tag{8}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = -hB_x + Khv \tag{9}$$

$$(hv)_t + (huv)_x = -Khu \tag{10}$$

- Full form of conservation law:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \mathbf{s}(\mathbf{q}, x) \tag{11}$$

Adding Source Terms

- Split into two steps:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{12}$$

$$\mathbf{q}_t = \mathbf{s}(\mathbf{q}, x) \tag{13}$$

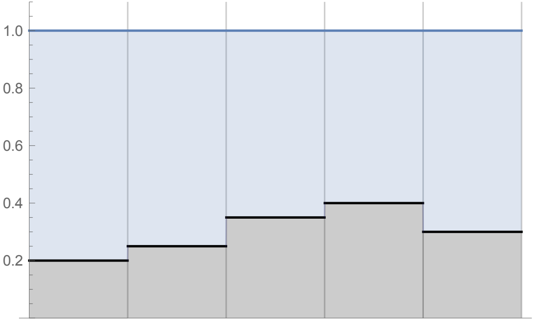
Adding Source Terms

- Split into two steps:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{12}$$

$$\mathbf{q}_t = \mathbf{s}(\mathbf{q}, x) \tag{13}$$

- The still water system:



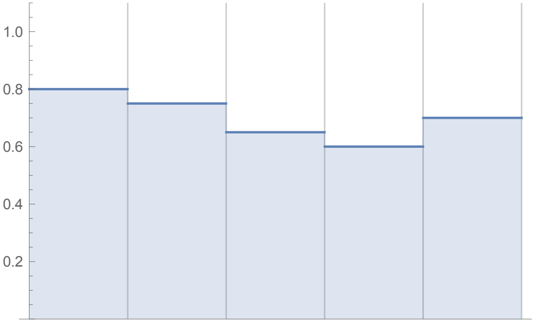
Adding Source Terms

- Split into two steps:

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0 \tag{12}$$

$$\mathbf{q}_t = \mathbf{s}(\mathbf{q}, x) \tag{13}$$

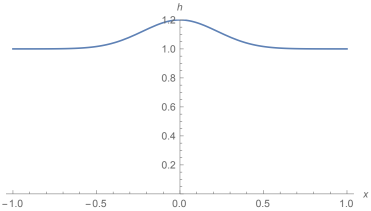
- The still water system:

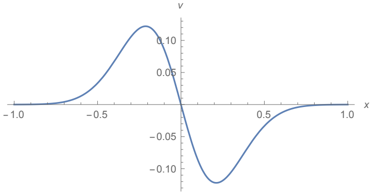


Geostrophic Equilibria

- Coriolis force admits non-trivial equilibria when

$$u = 0, \quad v = \frac{h(h_s)_x}{K} \tag{14}$$





- Geophysical flows close to geostrophic equilibrium at all times

Well-Balanced Methods

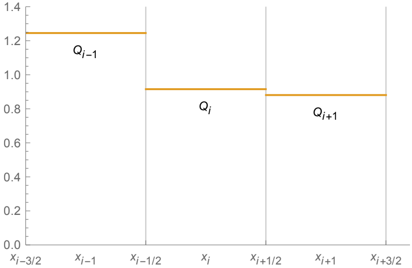
LeVeque

- LeVeque, 1998, “Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-Propagation Algorithm”

Well-Balanced Methods

LeVeque

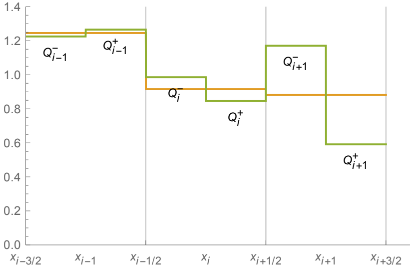
- LeVeque, 1998, “Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-Propagation Algorithm”



Well-Balanced Methods

LeVeque

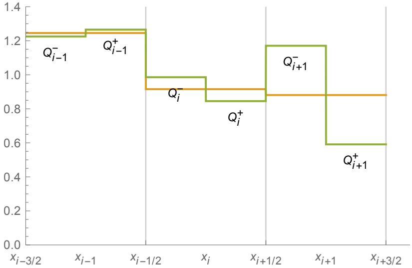
- LeVeque, 1998, “Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-Propagation Algorithm”



Well-Balanced Methods

LeVeque

- LeVeque, 1998, "Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-Propagation Algorithm"



$$\mathbf{f}(Q_i^+) - \mathbf{f}(Q_i^-) = \mathbf{s}(Q_i, x_i)\Delta x \tag{15}$$

Well-Balanced Methods

LeVeque

- Method originally developed for bathymetry term

Well-Balanced Methods

LeVeque

- Method originally developed for bathymetry term
- Derived analogous method including Coriolis terms

Well-Balanced Methods

LeVeque

- Method originally developed for bathymetry term
- Derived analogous method including Coriolis terms
- Pro:
  - Well-balanced for all equilibria

Well-Balanced Methods

LeVeque

- Method originally developed for bathymetry term
- Derived analogous method including Coriolis terms
- Pro:
  - Well-balanced for all equilibria
- Con:
  - Requires solving cubic for offset
  - Problematic away from equilibrium (transcritical flow)

Well-Balanced Methods

Rogers et al.

- Rogers, Borthwick, Taylor, 2003, “Mathematical balancing of flux gradient and source terms prior to using Roe’s approximate Riemann solver”

Well-Balanced Methods

- Rogers, Borthwick, Taylor, 2003, "Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver"
- Change of variables: deviations from equilibrium

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \mathbf{s}(\mathbf{q}, x) \tag{16}$$

$$\rightarrow (\mathbf{q} - \mathbf{q}_{\mathrm{eq}})_t + (\mathbf{f}(\mathbf{q}) - \mathbf{f}(\mathbf{q}_{\mathrm{eq}}))_x = \mathbf{s}(\mathbf{q}, x) - \mathbf{s}(\mathbf{q}_{\mathrm{eq}}, x) \tag{17}$$

$$\text{or} \quad \mathbf{q}_t^* + \mathbf{f}_x^* = \mathbf{s}^* \tag{18}$$

Well-Balanced Methods

Rogers et al.

- Rogers, Borthwick, Taylor, 2003, "Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver"
- Change of variables: deviations from equilibrium

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \mathbf{s}(\mathbf{q}, x) \tag{16}$$

$$\rightarrow (\mathbf{q} - \mathbf{q}_{\text{eq}})_t + (\mathbf{f}(\mathbf{q}) - \mathbf{f}(\mathbf{q}_{\text{eq}}))_x = \mathbf{s}(\mathbf{q}, x) - \mathbf{s}(\mathbf{q}_{\text{eq}}, x) \tag{17}$$

$$\text{or} \quad \mathbf{q}_t^* + \mathbf{f}_x^* = \mathbf{s}^* \tag{18}$$

- At equilibrium, all terms vanish

Well-Balanced Methods

Rogers et al.

- Rogers, Borthwick, Taylor, 2003, "Mathematical balancing of flux gradient and source terms prior to using Roe's approximate Riemann solver"
- Change of variables: deviations from equilibrium

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \mathbf{s}(\mathbf{q}, x) \tag{16}$$

$$\rightarrow (\mathbf{q} - \mathbf{q}_{\mathrm{eq}})_t + (\mathbf{f}(\mathbf{q}) - \mathbf{f}(\mathbf{q}_{\mathrm{eq}}))_x = \mathbf{s}(\mathbf{q}, x) - \mathbf{s}(\mathbf{q}_{\mathrm{eq}}, x) \tag{17}$$

$$\text{or} \quad \mathbf{q}_t^* + \mathbf{f}_x^* = \mathbf{s}^* \tag{18}$$

- At equilibrium, all terms vanish
- Jacobian of $\mathbf{f}$ remains unchanged
  - easy to adapt existing solver

Well-Balanced Methods

Rogers et al.

- Derivation in original paper leads to

$$\mathbf{q}_t^* + \mathbf{f}'(\mathbf{q})\mathbf{q}_x^* = \mathbf{s}^* \tag{19}$$

- Implementation based on this failed away from equilibrium

Well-Balanced Methods

Rogers et al.

- Derivation in original paper leads to

$$\mathbf{q}_t^* + \mathbf{f}'(\mathbf{q})\mathbf{q}_x^* = \mathbf{s}^* \tag{19}$$

- Implementation based on this failed away from equilibrium
- Derived alternative form:

$$\mathbf{q}_t^* + \mathbf{f}'(\mathbf{q})\mathbf{q}_x^* = \mathbf{s}^* - \frac{\partial \mathbf{f}^*}{\partial \mathbf{q}_{\text{eq}}}(\mathbf{q}_{\text{eq}})_x \tag{20}$$

- Works, but less accurate than unbalanced method

Well-Balanced Methods

Rogers et al.

- Derivation in original paper leads to

$$\mathbf{q}^*_t + \mathbf{f}'(\mathbf{q})\mathbf{q}^*_x = \mathbf{s}^* \tag{19}$$

- Implementation based on this failed away from equilibrium
- Derived alternative form:

$$\mathbf{q}^*_t + \mathbf{f}'(\mathbf{q})\mathbf{q}^*_x = \mathbf{s}^* - \frac{\partial \mathbf{f}^*}{\partial \mathbf{q}_{\text{eq}}}(\mathbf{q}_{\text{eq}})_x \tag{20}$$

- Works, but less accurate than unbalanced method
- Again, extended original work to support Coriolis term

Well-Balanced Methods

Rogers et al.

- Derivation in original paper leads to

$$\mathbf{q}_t^* + \mathbf{f}'(\mathbf{q})\mathbf{q}_x^* = \mathbf{s}^* \tag{19}$$

- Implementation based on this failed away from equilibrium
- Derived alternative form:

$$\mathbf{q}_t^* + \mathbf{f}'(\mathbf{q})\mathbf{q}_x^* = \mathbf{s}^* - \frac{\partial \mathbf{f}^*}{\partial \mathbf{q}_{\mathrm{eq}}}(\mathbf{q}_{\mathrm{eq}})_x \tag{20}$$

- Works, but less accurate than unbalanced method
- Again, extended original work to support Coriolis term
- Also derived solver for geostrophic equilibria
    - Implementation does not work yet

Well-Balanced Methods

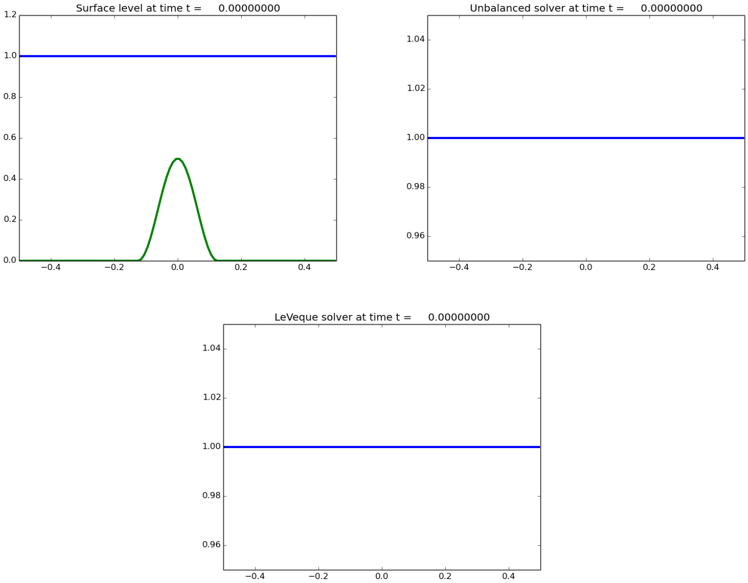
Rogers et al.

- Pro:
  - Works away from equilibrium
  - Fairly simple to derive and implement
- Con:
  - Tailored to a specific equilibrium
  - Need to know equilibrium a priori
  - Less accurate than unbalanced method away from equilibrium

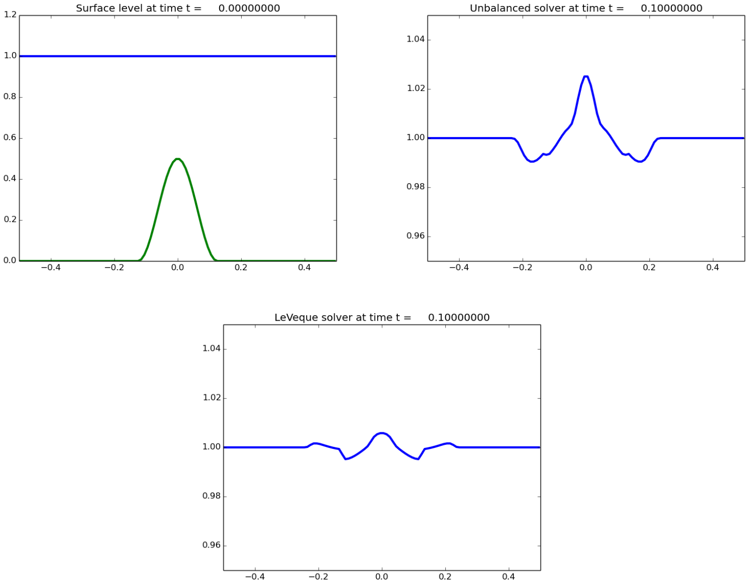
Evaluation

- Implemented evaluation framework in Clawpack (Python)
- Supports 7 different bathymetries and 5 initial conditions
- Implemented 4 Riemann solvers in Fortran (unbalanced, LeVeque, Rogers for still water, Rogers for geostrophic equilibrium)

Results

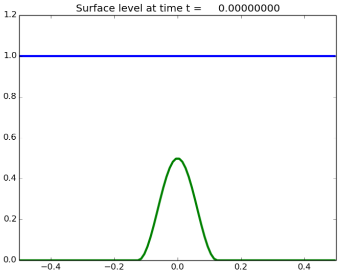

Surface level at time t = 0.00000000
Unbalanced solver at time t = 0.00000000
LeVeque solver at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

Results

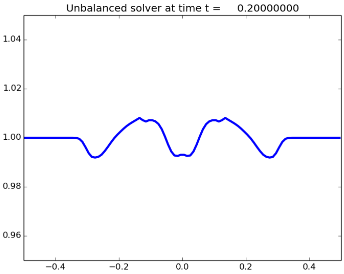


Surface level at time t = 0.00000000
Unbalanced solver at time t = 0.10000000
LeVeque solver at time t = 0.10000000

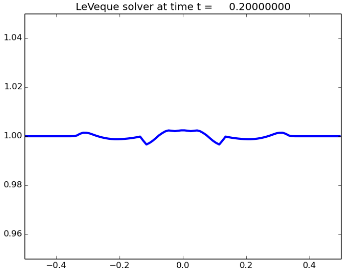
Results



Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

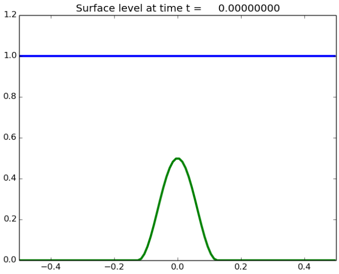


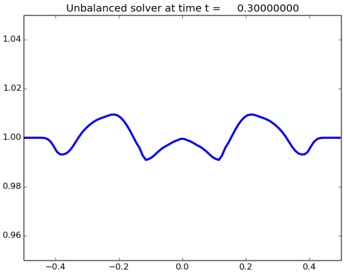
Unbalanced solver at time t = 0.20000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

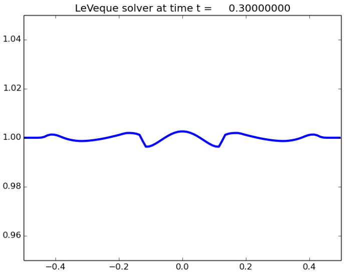


LeVeque solver at time t = 0.20000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

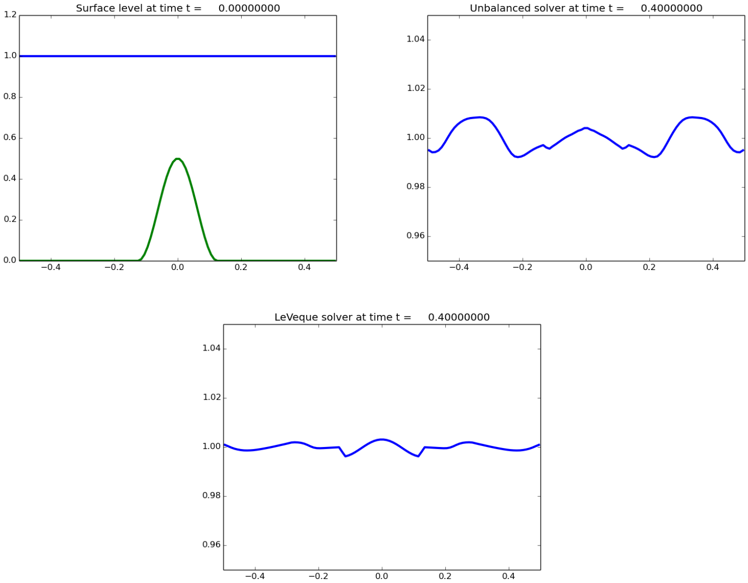
Results


Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

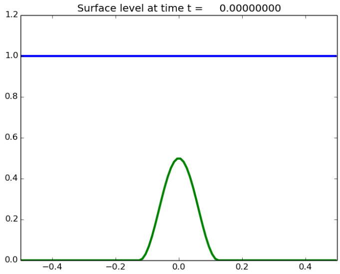

Unbalanced solver at time t = 0.30000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 0.30000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

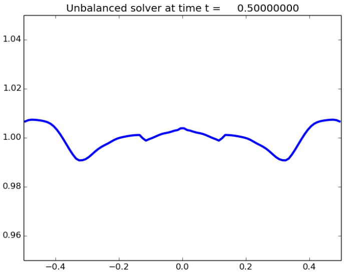
Results


Surface level at time t = 0.00000000
Unbalanced solver at time t = 0.40000000
LeVeque solver at time t = 0.40000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

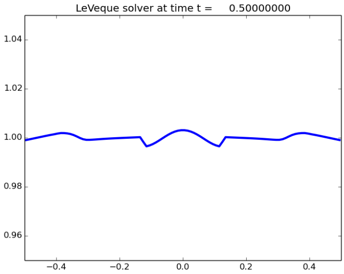
Results



Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

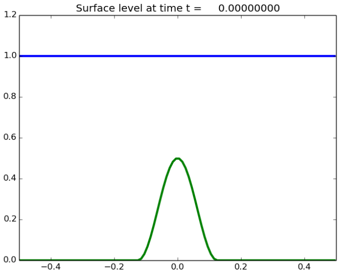


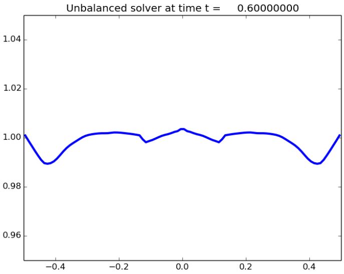
Unbalanced solver at time t = 0.50000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

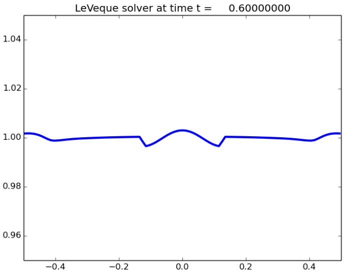


LeVeque solver at time t = 0.50000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

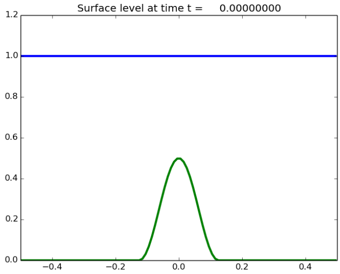
Results

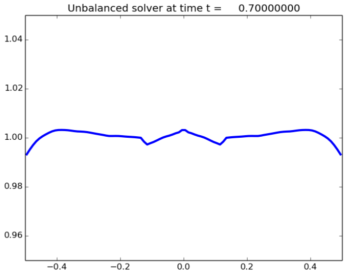

Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

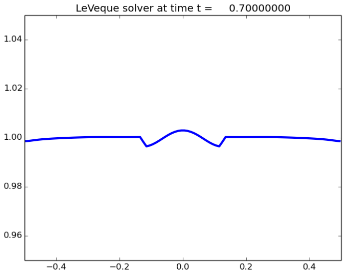

Unbalanced solver at time t = 0.60000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 0.60000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

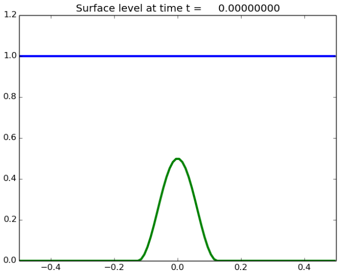
Results

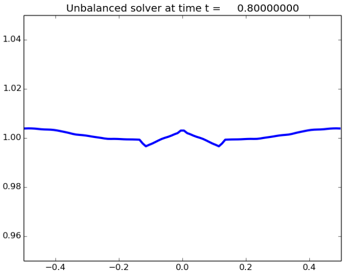

Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

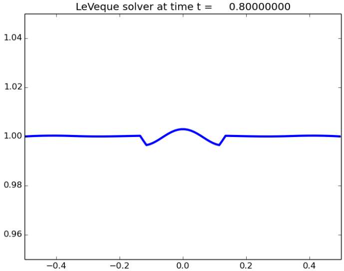

Unbalanced solver at time t = 0.70000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 0.70000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

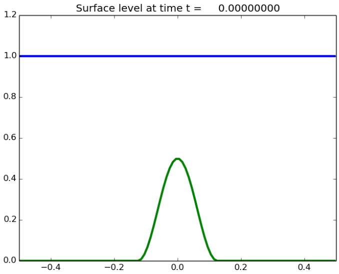
Results

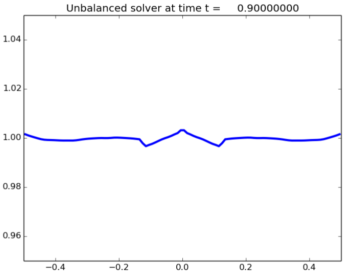

Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

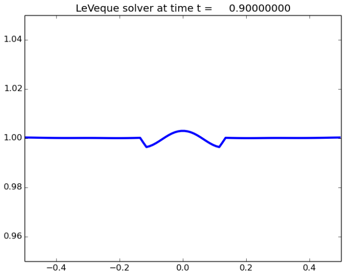

Unbalanced solver at time t = 0.80000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 0.80000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

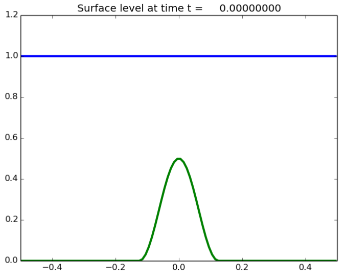
Results

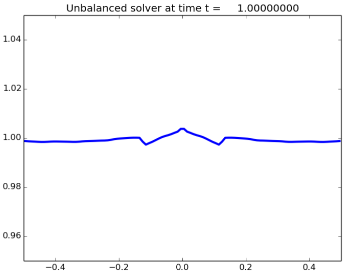

Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4

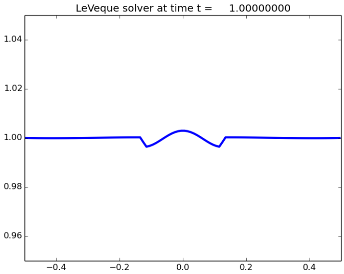

Unbalanced solver at time t = 0.90000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 0.90000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4

Results


Surface level at time t = 0.00000000
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.4
−0.2
0.0
0.2
0.4


Unbalanced solver at time t = 1.00000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4


LeVeque solver at time t = 1.00000000
1.04
1.02
1.00
0.98
0.96
−0.4
−0.2
0.0
0.2
0.4