Direct numerical simulation of interfacial instability in gas-liquid flows

Iain Bethune[1], Lennon Ó Náraigh*[2], David Scott[1], Peter Spelt[3,4], Prashant Valluri[5], Zlatko Solomenko[3]

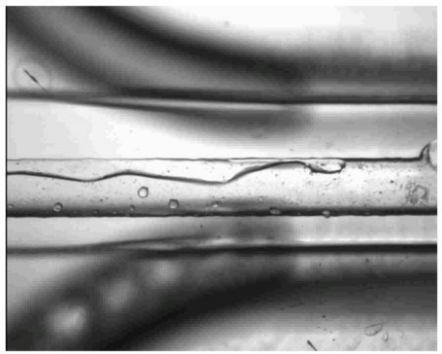
[1]Edinburgh Parallel Computing Centre, University of Edinburgh, James Clerk Maxwell Building, Mayfield Road, Edinburgh EH9 3JZ, UK
[2]School of Mathematics and Statistics, University College Dublin, Ireland
[3]Département Mécanique, Université de Lyon 1
[4]Laboratoire de Mécanique des Fluides et dAcoustique (LMFA), CNRS, Ecole Centrale de Lyon, 69134 Ecully, France
[5]Institute of Materials and Processes, Sanderson Building, School of Engineering, University of Edinburgh, Kings Buildings, Edinburgh EH9 3JL, UK

15th May 2017

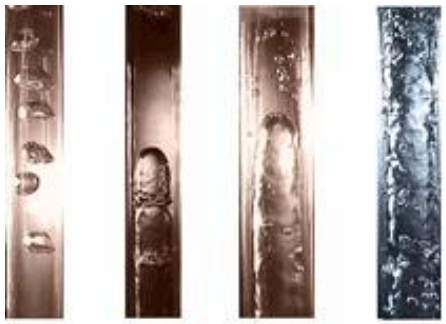
Context

Two-phase stratified flow is ubiquitous in nature.

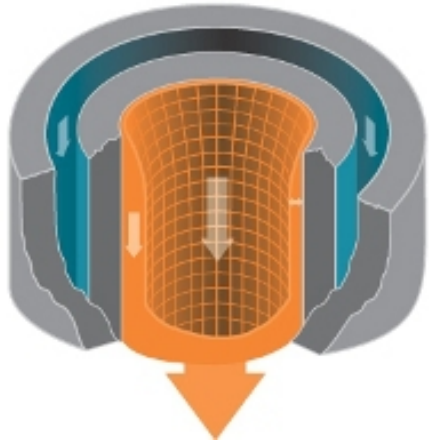
(a) Kelvin-Helmholtz instability



(b) Stratified flow in pipelines



(c) Slug flow

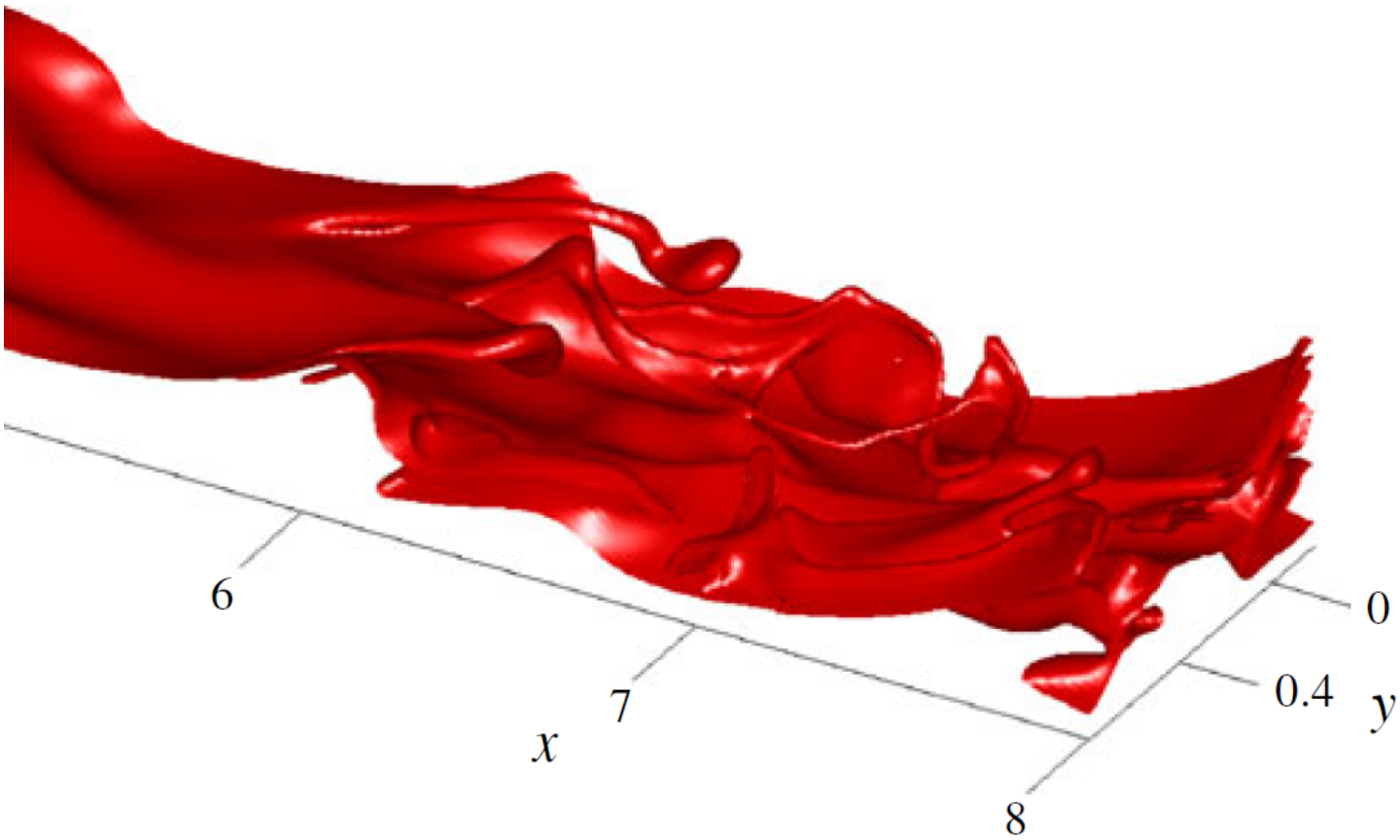


(d) Falling-film reactors

- Mathematically, and computationally, a tough problem – turbulence, extreme nonlinearity, topological change in interfaces, a range of instabilities that need to be captured.
- Even the laminar regime is tough - current focus of the research.

Context: The numerical challenge

- Flows involving many length- and time-scales
- Flows with sharp changes in interfacial topologies
- Transient three-dimensional simulations required over long periods of time, requiring **scalable** codes run at very high **resolutions**.



Context: Existing methodologies

- Existing interface-capturing methods: Levelset, Volume of Fluid, Particles, Diffuse Interface Method

Context: Existing methodologies

- Existing interface-capturing methods: Levelset, Volume of Fluid, Particles, Diffuse Interface Method
- Existing implementations: Open-source (e.g. Gerris, etc.), Commercial (CFX, etc.), in-house solvers.

Context: Existing methodologies

- Existing interface-capturing methods: Levelset, Volume of Fluid, Particles, Diffuse Interface Method
- Existing implementations: Open-source (e.g. Gerris, etc.), Commercial (CFX, etc.), in-house solvers.
- Some drawbacks (not respectively): Black-box approach, validation uncertain, artificial diffusion. Key drawbacks: resolution constraints, scalability.

Context: Existing methodologies

- Existing interface-capturing methods: Levelset, Volume of Fluid, Particles, Diffuse Interface Method
- Existing implementations: Open-source (e.g. Gerris, etc.), Commercial (CFX, etc.), in-house solvers.
- Some drawbacks (not respectively): Black-box approach, validation uncertain, artificial diffusion. Key drawbacks: resolution constraints, scalability.
- Our in-house code **TPLS** addresses these issues, in particular **resolution and scalability**.

Context: Existing methodologies

- Existing interface-capturing methods: Levelset, Volume of Fluid, Particles, Diffuse Interface Method
- Existing implementations: Open-source (e.g. Gerris, etc.), Commercial (CFX, etc.), in-house solvers.
- Some drawbacks (not respectively): Black-box approach, validation uncertain, artificial diffusion. Key drawbacks: resolution constraints, scalability.
- Our in-house code **TPLS** addresses these issues, in particular **resolution and scalability**.
- Not a silver bullet – levelset methods – tradeoff between capturing interfacial topology with great fidelity, and mass loss. But mass loss minimized at high resolution.

TPLS: Equations of motion

Numerical solution of two-phase Navier–Stokes equations with interface capturing:

$$\rho(\phi)\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=-\nabla p+\frac{1}{\mathrm{Re}}\nabla\cdot\left[\mu(\phi)\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)\right]+\boldsymbol{f}_{\mathrm{st}}(\phi)-\rho(\phi)\mathcal{G}\hat{\boldsymbol{z}},$$

where $\nabla\cdot\mathbf{u}=0$ and ϕ is the interface-capturing field:

TPLS: Equations of motion

Numerical solution of two-phase Navier–Stokes equations with interface capturing:

$$\rho(\phi)\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=-\nabla p+\frac{1}{\mathrm{Re}}\nabla\cdot\left[\mu(\phi)\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)\right]+\boldsymbol{f}_{\mathrm{st}}(\phi)-\rho(\phi)\mathcal{G}\hat{\boldsymbol{z}},$$

where $\nabla\cdot\mathbf{u}=0$ and ϕ is the interface-capturing field:

Levelset method:

$$\frac{\partial\phi}{\partial t}+\boldsymbol{u}\cdot\nabla\phi=0,\qquad \boldsymbol{f}_{\mathrm{st}}=\delta_{\epsilon}(\phi)\frac{1}{\mathrm{We}}\widehat{\boldsymbol{n}}\nabla\cdot\widehat{\boldsymbol{n}}.\qquad \widehat{\boldsymbol{n}}=\frac{\nabla\phi}{|\nabla\phi|}.$$

TPLS: Equations of motion

Numerical solution of two-phase Navier–Stokes equations with interface capturing:

$$\rho(\phi)\left(\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}\right)=-\nabla p+\frac{1}{\mathrm{Re}}\nabla\cdot\left[\mu(\phi)\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)\right]+\boldsymbol{f}_{\mathrm{st}}(\phi)-\rho(\phi)\mathcal{G}\hat{\boldsymbol{z}},$$

where $\nabla\cdot\mathbf{u}=0$ and ϕ is the interface-capturing field:

Levelset method:

$$\frac{\partial\phi}{\partial t}+\boldsymbol{u}\cdot\nabla\phi=0,\qquad \boldsymbol{f}_{\mathrm{st}}=\delta_{\epsilon}(\phi)\frac{1}{\mathrm{We}}\widehat{\boldsymbol{n}}\nabla\cdot\widehat{\boldsymbol{n}}.\qquad \widehat{\boldsymbol{n}}=\frac{\nabla\phi}{|\nabla\phi|}.$$

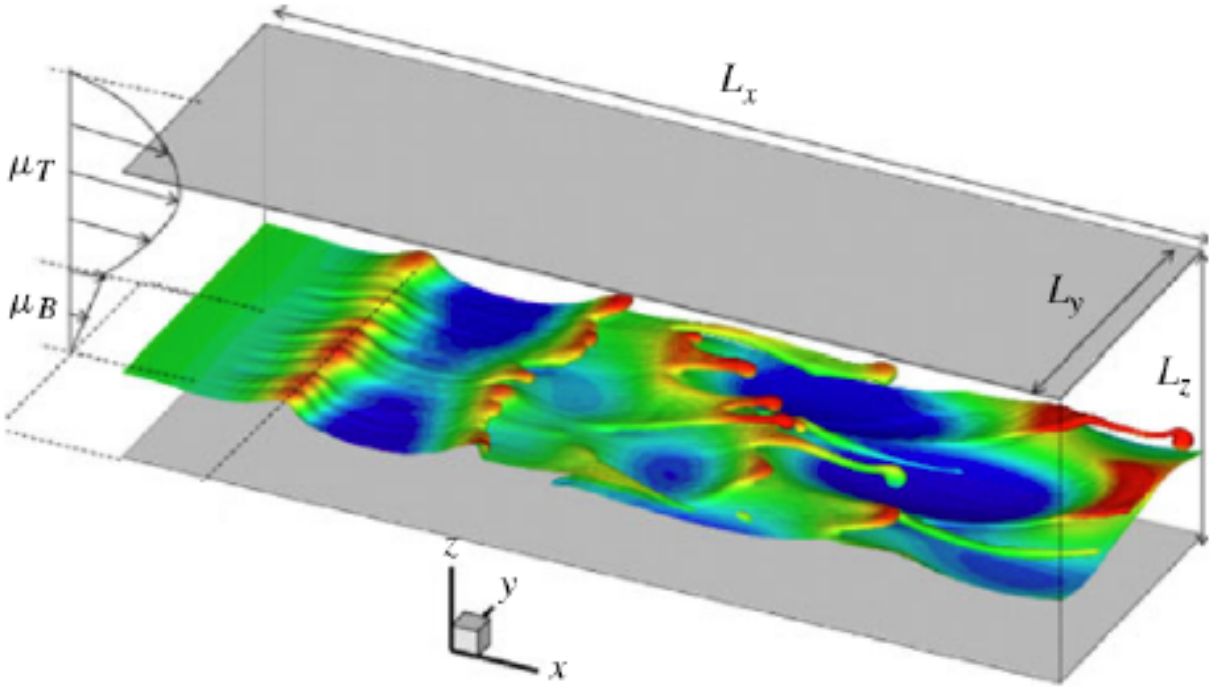
Dimensionless groups:

$$\mathrm{Re}=\frac{\rho_T V L}{\mu_T},\qquad \mathcal{G}=\frac{gL}{V^2},\qquad \mathrm{We}=\frac{\rho_T L V^2}{\gamma},$$

(I also use $\mathcal{S}=1/\mathrm{We}$, for historical reasons!)

TPLS: Problem geometry and configuration

- Simple channel geometry: periodic boundary conditions at $x = 0$, $x = L_x$; walls (no slip) at $z = 0$, $z = L_z$.
- Constant pressure drop drives flow in streamwise direction (forcing).
- Basic version involves hydrodynamics only. TPLS with physics under development, for applications including contact-line dynamics, and mass transfer.



TPLS: Numerical discretization schemes

- Marker-and-cell discretization: pressures, densities, viscosities, and ϕ at cell centres, velocities at cell faces.

TPLS: Numerical discretization schemes

- Marker-and-cell discretization: pressures, densities, viscosities, and ϕ at cell centres, velocities at cell faces.
- Finite-volumes, with flux-conservative differencing for the momentum equation.

TPLS: Numerical discretization schemes

- Marker-and-cell discretization: pressures, densities, viscosities, and ϕ at cell centres, velocities at cell faces.
- Finite-volumes, with flux-conservative differencing for the momentum equation.
- Momentum step: centred differences for the convective derivative, Crank–Nicholson treatment for the diffusion, third-order Adams–Bashforth for the time evolution.

TPLS: Numerical discretization schemes

- Marker-and-cell discretization: pressures, densities, viscosities, and ϕ at cell centres, velocities at cell faces.
- Finite-volumes, with flux-conservative differencing for the momentum equation.
- Momentum step: centred differences for the convective derivative, Crank–Nicholson treatment for the diffusion, third-order Adams–Bashforth for the time evolution.
- Projection method: Momenta are updated first, followed by a correction step involving a pressure update, thereby enforcing incompressibility.

TPLS: Numerical discretization schemes

- Marker-and-cell discretization: pressures, densities, viscosities, and ϕ at cell centres, velocities at cell faces.
- Finite-volumes, with flux-conservative differencing for the momentum equation.
- Momentum step: centred differences for the convective derivative, Crank–Nicholson treatment for the diffusion, third-order Adams–Bashforth for the time evolution.
- Projection method: Momenta are updated first, followed by a correction step involving a pressure update, thereby enforcing incompressibility.
- The levelset function $\phi(x, y, z, t)$ is carried with the flow (3rd-order WENO) but is corrected at each timestep ('redistancing').

TPLS: Parallel computing

- Typical runs involve up to 10 million gridpoints, meaning that large-scale parallel simulation is unavoidable (larger runs (up to 100 million gridpoints) are also performed).

TPLS: Parallel computing

- Typical runs involve up to 10 million gridpoints, meaning that large-scale parallel simulation is unavoidable (larger runs (up to 100 million gridpoints) are also performed).
- Code is parallelized using hybrid MPI technology; parallelization scheme takes account of problem geometry (2D domain decomposition)

TPLS: Parallel computing

- Typical runs involve up to 10 million gridpoints, meaning that large-scale parallel simulation is unavoidable (larger runs (up to 100 million gridpoints) are also performed).
- Code is parallelized using hybrid MPI technology; parallelization scheme takes account of problem geometry (2D domain decomposition)
- Data is outputted to files periodically using parallel I/O – NetCDF data storage.
- Current version of code with density contrast uses simple hand-coded algorithms for linear algebra steps (e.g. presusre step). Work is ongoing to replace these with GMRES by repeated calls to the PETSc library.

TPLS: Parallel computing

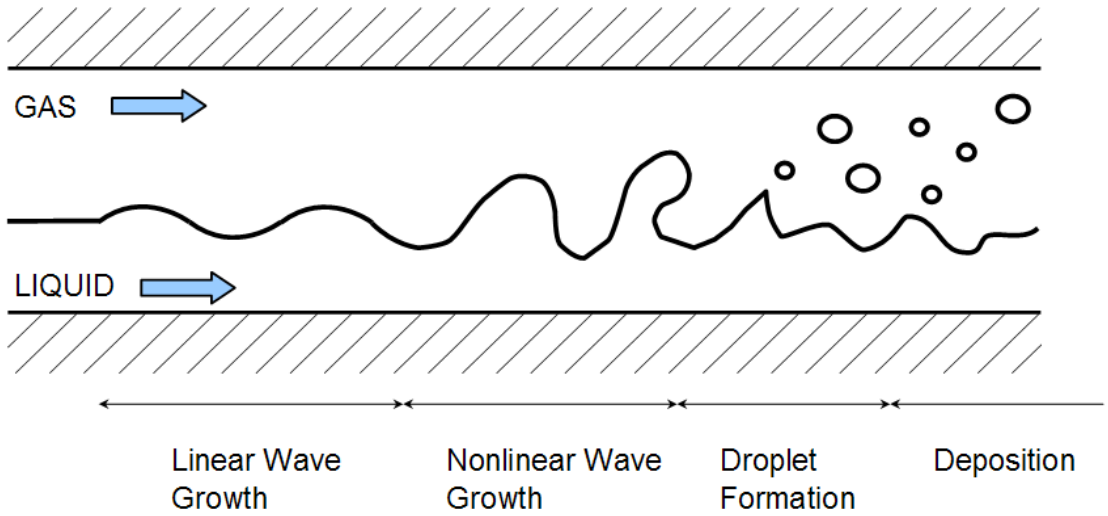
- Typical runs involve up to 10 million gridpoints, meaning that large-scale parallel simulation is unavoidable (larger runs (up to 100 million gridpoints) are also performed).
- Code is parallelized using hybrid MPI technology; parallelization scheme takes account of problem geometry (2D domain decomposition)
- Data is outputted to files periodically using parallel I/O – NetCDF data storage.
- Current version of code with density contrast uses simple hand-coded algorithms for linear algebra steps (e.g. presusre step). Work is ongoing to replace these with GMRES by repeated calls to the PETSc library.

TPLS: Parallel computing

- Typical runs involve up to 10 million gridpoints, meaning that large-scale parallel simulation is unavoidable (larger runs (up to 100 million gridpoints) are also performed).
- Code is parallelized using hybrid MPI technology; parallelization scheme takes account of problem geometry (2D domain decomposition)
- Data is outputted to files periodically using parallel I/O – NetCDF data storage.
- Current version of code with density contrast uses simple hand-coded algorithms for linear algebra steps (e.g. presusre step). Work is ongoing to replace these with GMRES by repeated calls to the PETSc library.
- Parallel efficiency with 2000 MPI processes is only 0.6 – there is a tradeoff between robustness/simplicity and performance. Underscores the need to replace hand-coded linear-algebra solvers with libraries.

Strict benchmarks for code's accuracy

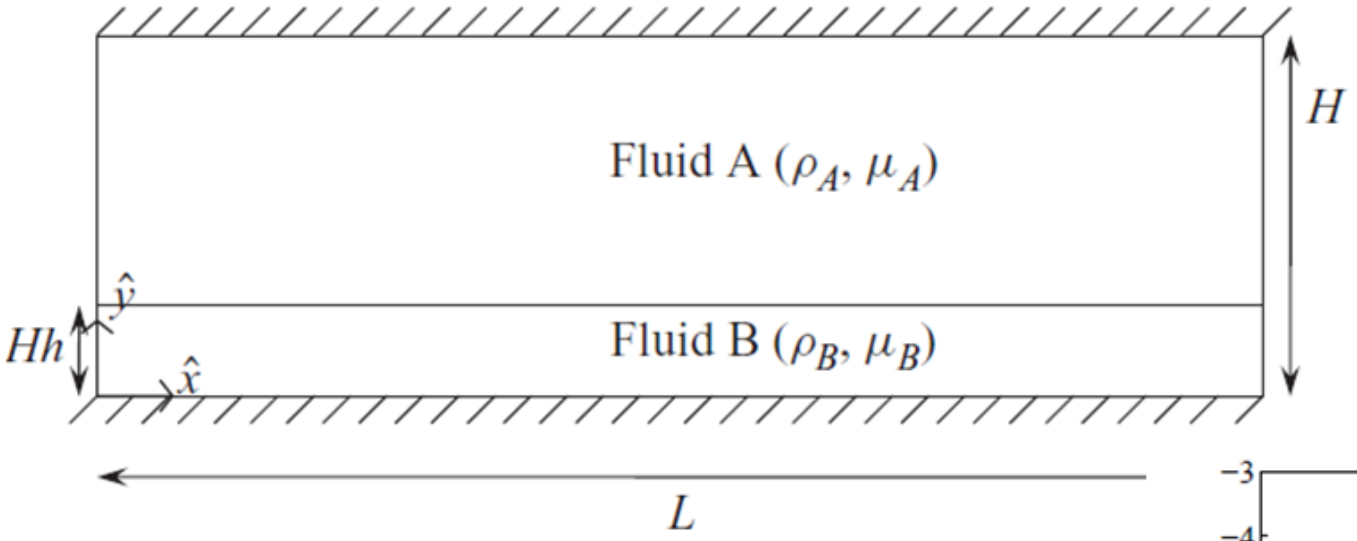
- Introduce a tiny sinusoidal perturbation at the interface.
- Produces pressure and velocity fluctuations that satisfy linear equations of motion.
- Linearized equations of motion solved via eigenvalue analysis (independent, quasi-analytical).
- Gives growth rate and wave speed of wave-like fluctuations.



Focus on finding agreement between OS analysis and wave growth in the code.

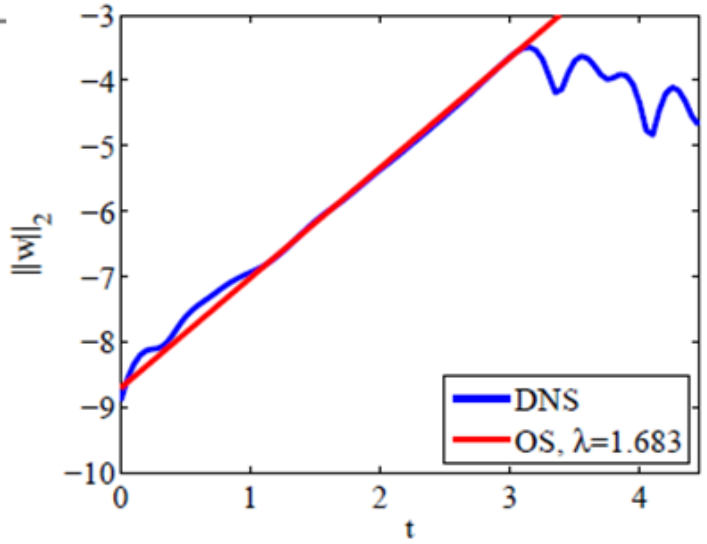
Orr–Sommerfeld analysis – Results

Stratified co-flow test case (h=0.3)

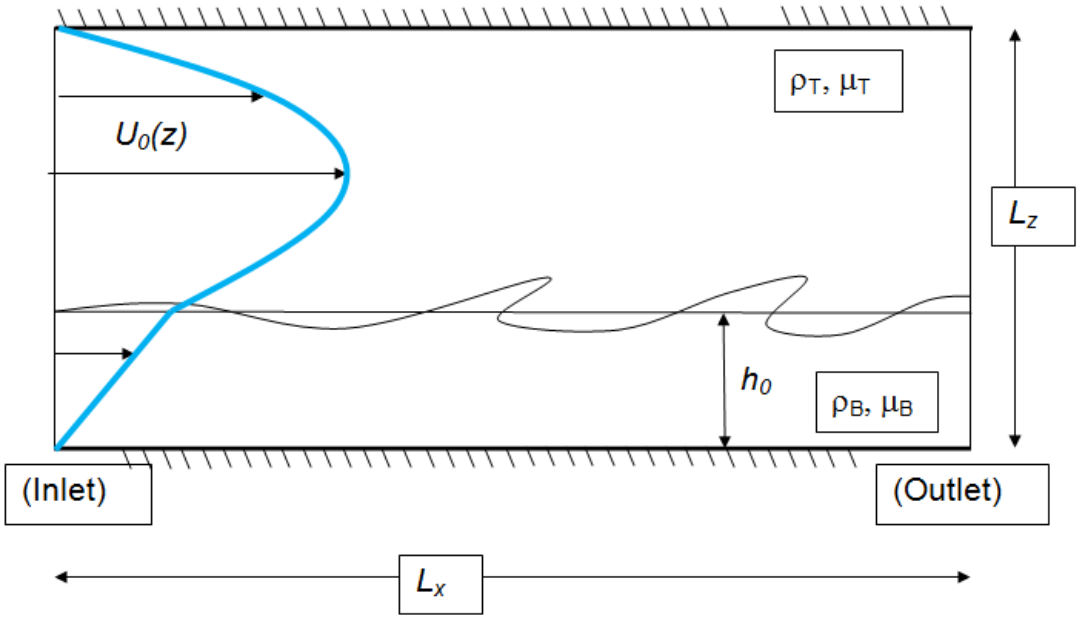


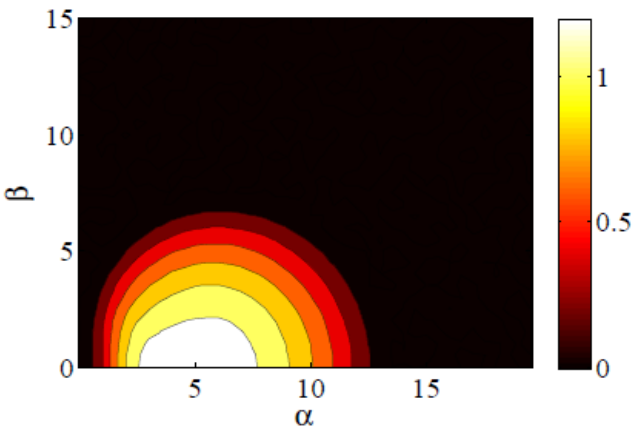
$$(Re, r, m, \mathcal{S}) = (100, 1, 30, 0.01)$$

- *L x W x H = (3 x 1 x 1)*
- *12 million grid points*
- *1024 processors, 12 h*



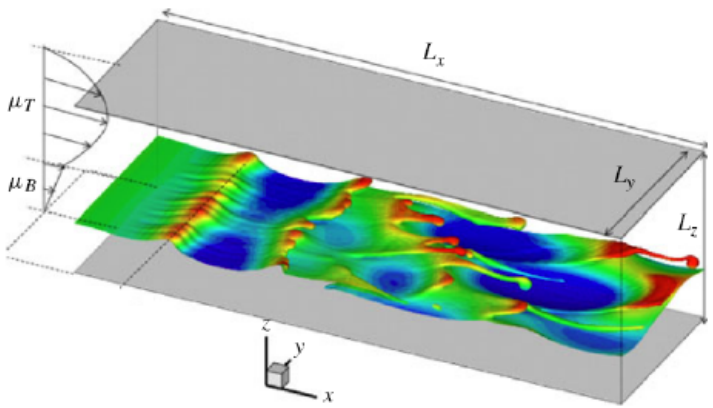
Application of TPLS: where do 3D waves in parallel flows come from?





$$r = 1, \mathcal{S} = 0.1$$

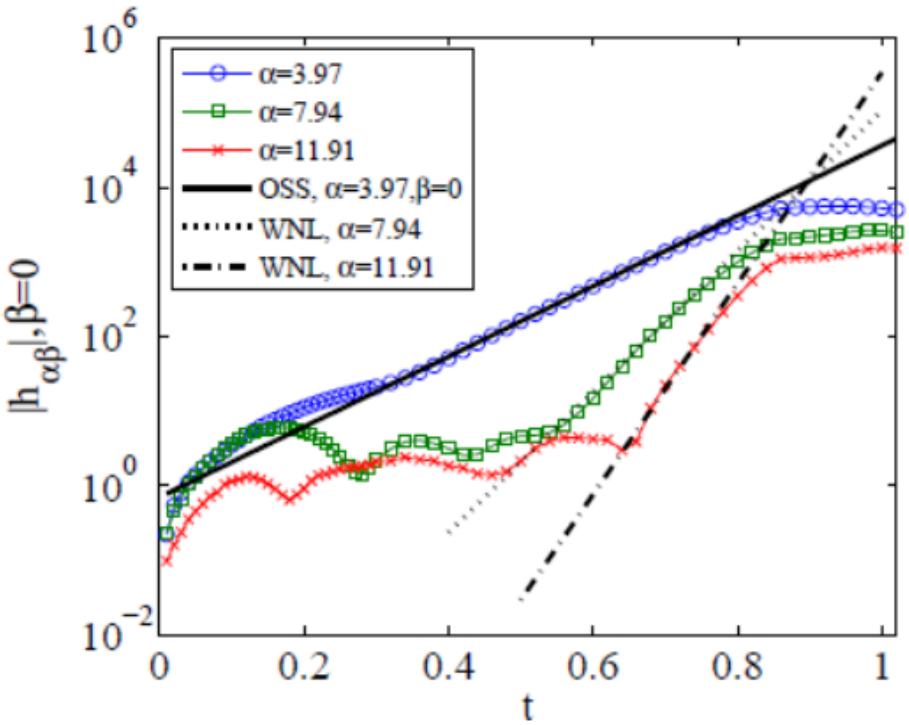
Linear instability of 2D parallel flow is dominated by 2D waves.
So how do 3D structures form?

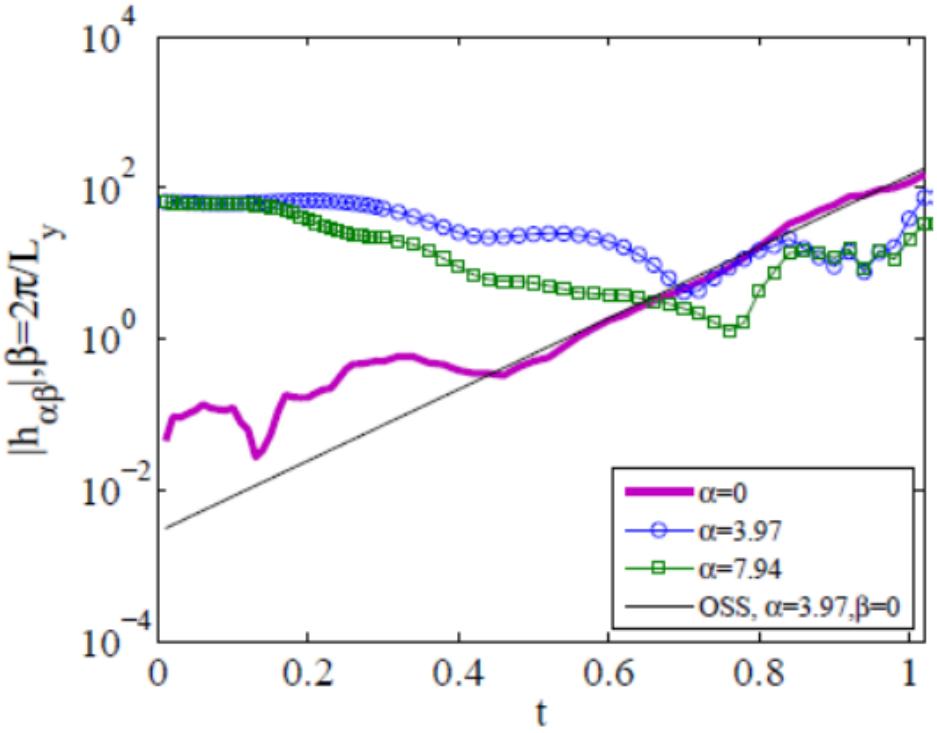


Brief review for liquid-liquid flows

We know the answer for liquid-liquid flows ($r = 1$) – it is weakly nonlinear analysis.

Streamwise waves – Large temporal growth, Spanwise waves – No temporal growth rate



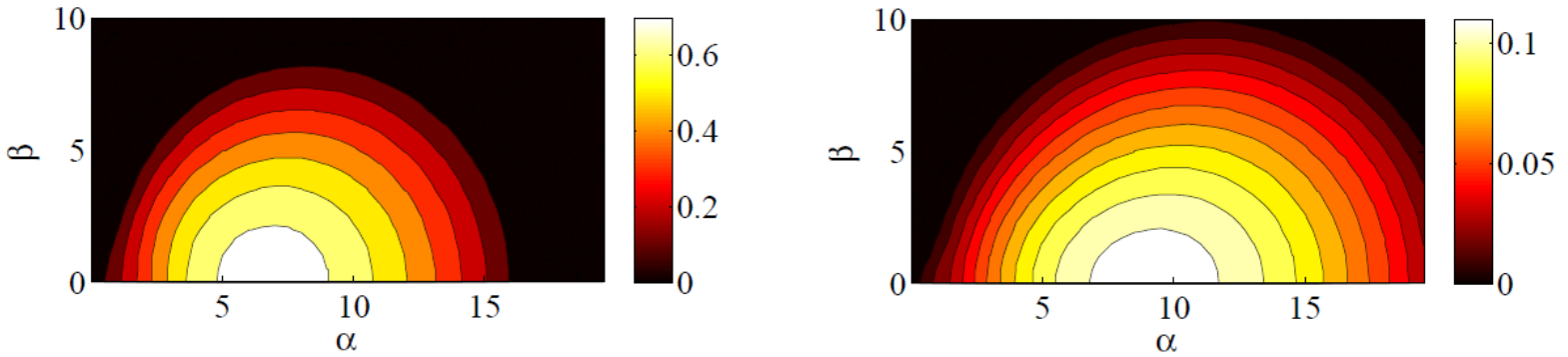


- Streamwise overtones are enslaved to the streamwise dominant mode
- Purely spanwise mode enslaved to the dominant streamwise mode

Periodic boundary conditions, $(Re, m, r, \mathcal{S}) = (300, 30, 1, 0.3)$.

New study required for gas-liquid flows

For gas-liquid flows, linear theory predicts a **direct route**.

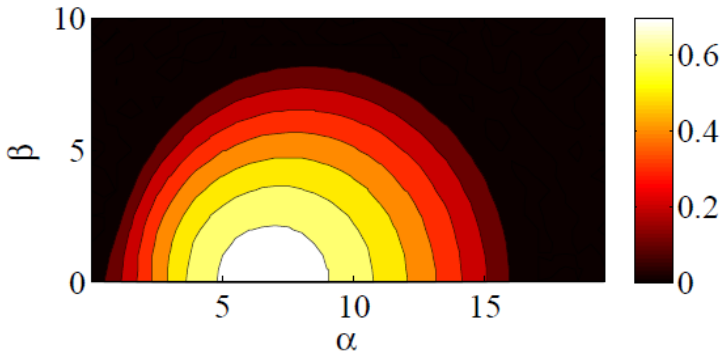


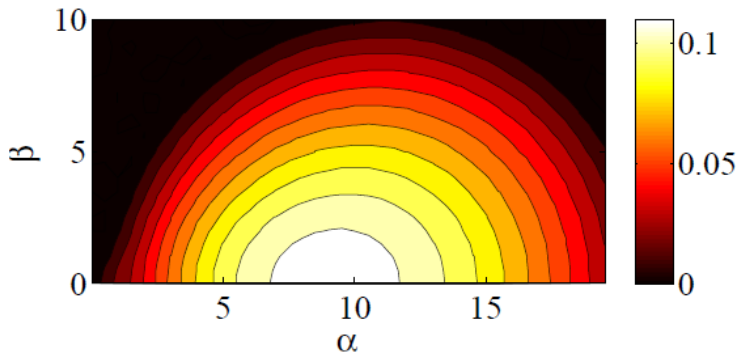
(a) $r = 100, \mathcal{S} = 0.1$ (b) $r = 1000, \mathcal{S} = 0.1$

Eigenvalue analysis of the two-phase Orr–Sommerfeld–Squire equations for $Re = 100$, $m = 30$, $h_0 = 0.3$, and $\mathcal{S} = 0.1$, and $\mathcal{G} = 0.1$.

New study required for gas-liquid flows

For gas-liquid flows, linear theory predicts a **direct route**.



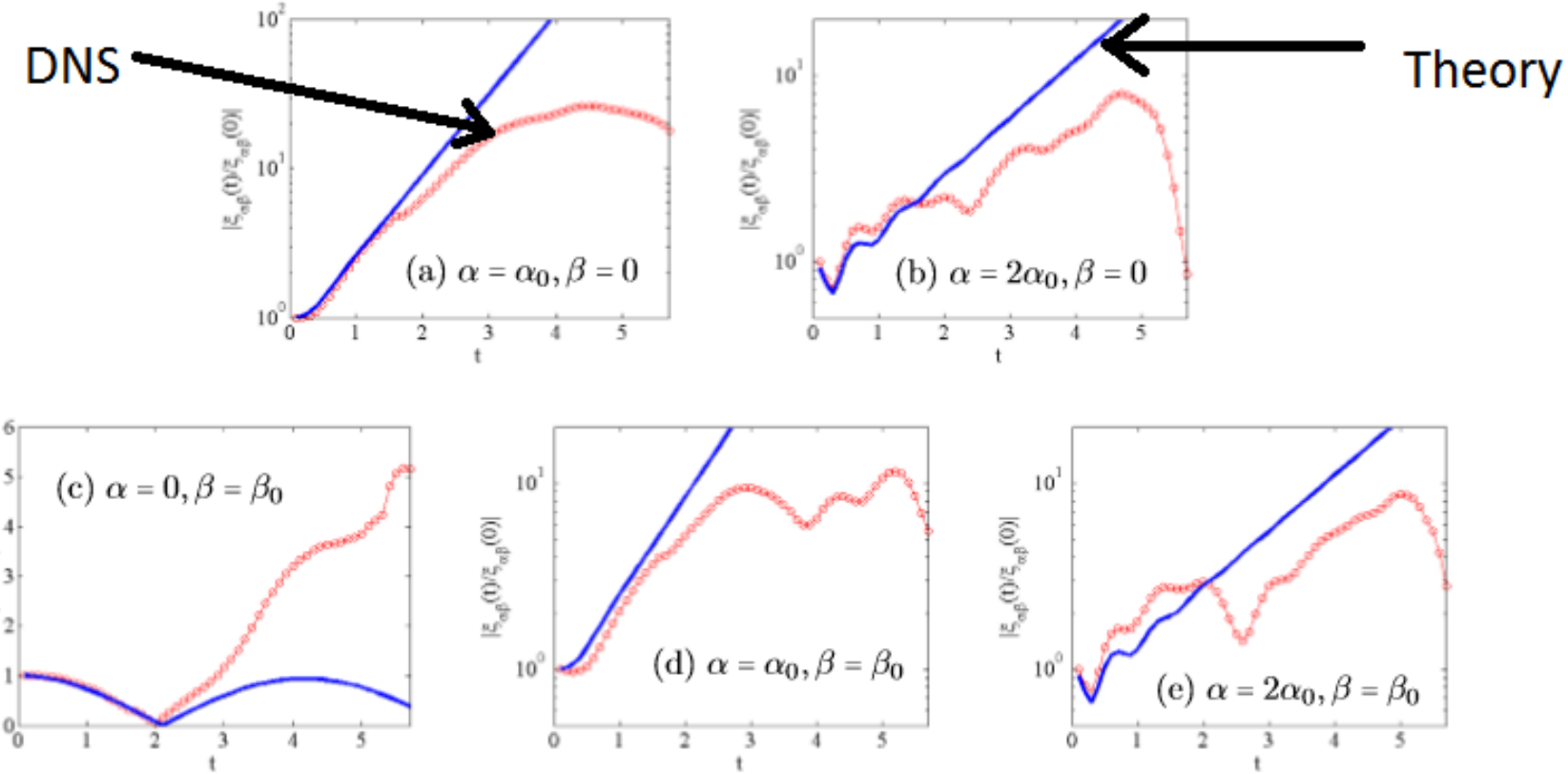


(a) $r = 100, \mathcal{S} = 0.1$ (b) $r = 1000, \mathcal{S} = 0.1$

Eigenvalue analysis of the two-phase Orr–Sommerfeld–Squire equations for $Re = 100$, $m = 30$, $h_0 = 0.3$, and $\mathcal{S} = 0.1$, and $\mathcal{G} = 0.1$.

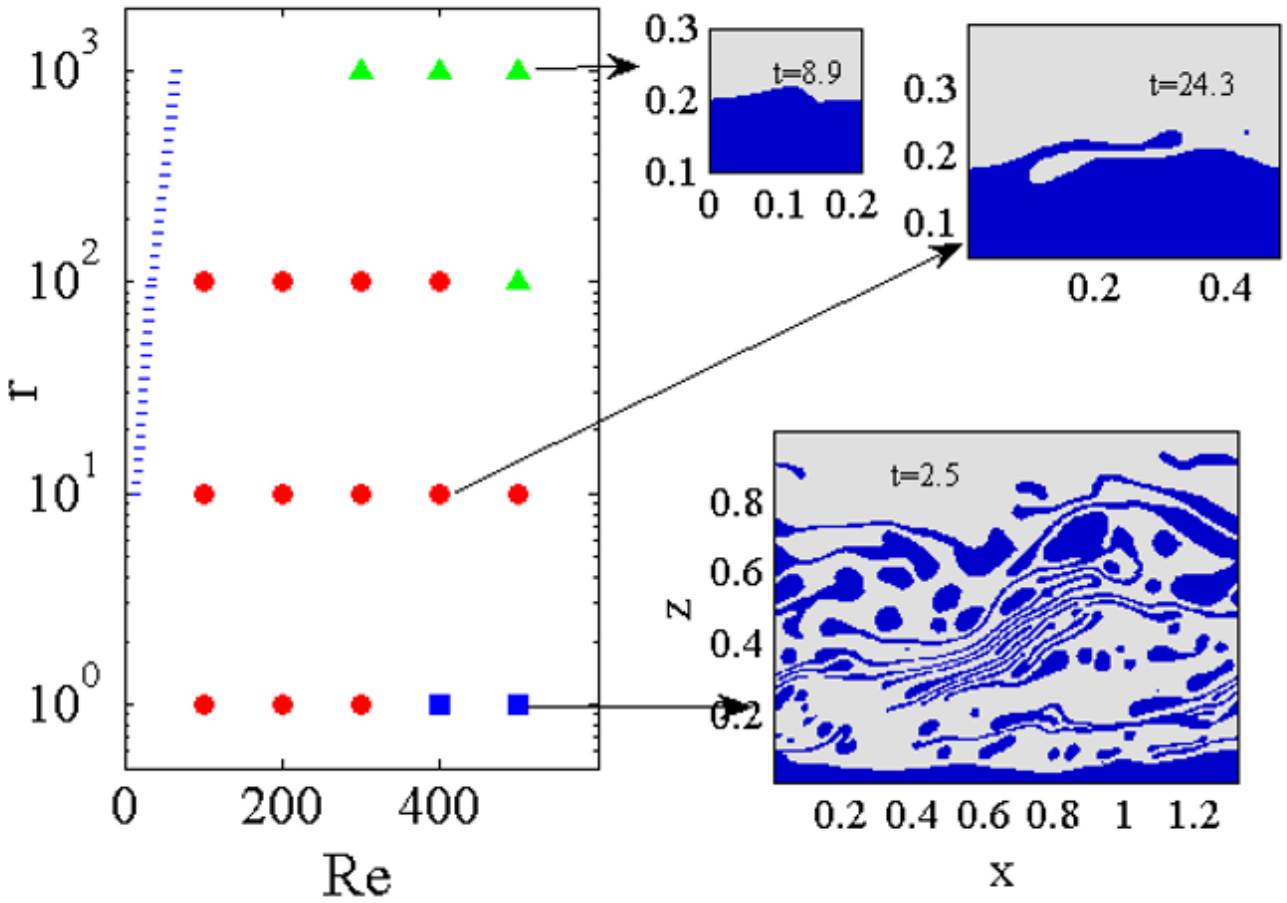
Overall trend: increasing r means that more modes become unstable (both streamwise and spanwise), but with a smaller growth rate.

Theoretical Prediction confirmed by DNS



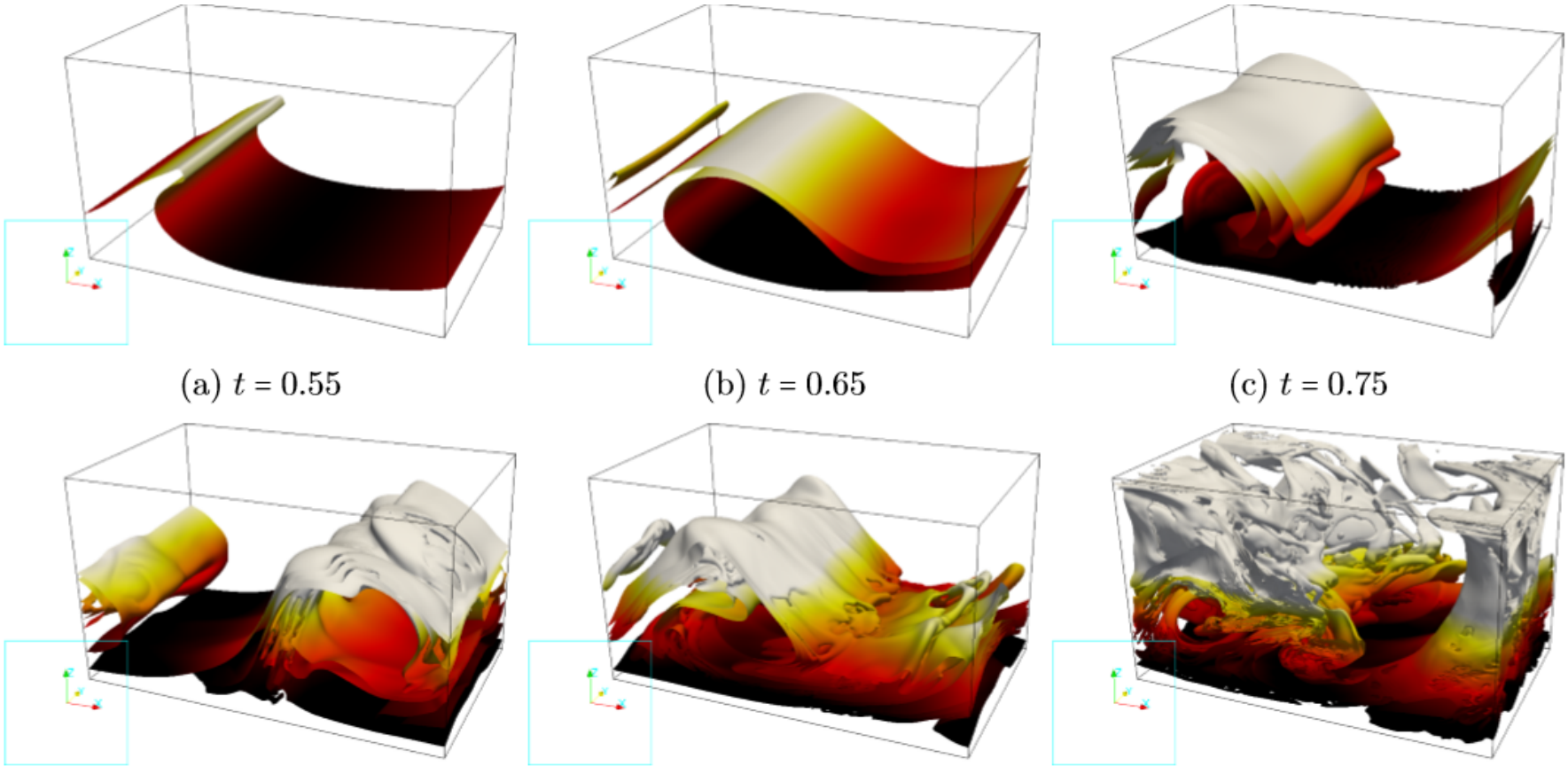
DNS results (lines with circles) for the case ($m = 50, h_0 = 0.2, \mathcal{G} = 0.1, \mathrm{We} = 10$), with $r = 1000$ and $\mathrm{Re} = 500$. Shown also is a comparison with linearized DNS (unadorned lines). Here, $\alpha_0 = 2\pi/L_x$ and $\beta_0 = 2\pi/L_y$ denote the fundamental wavenumber in the streamwise and spanwise directions respectively. In panel (c) the growth of the relevant amplitude is modest and a vertical linear (as opposed to logarithmic) scale is used. Also, the 'kink' at $t = 2$ in the same panel simply corresponds to a a zero of $\xi_{0\beta_0}(t)$, as this particular Fourier amplitude does not grow exponentially.

2D-DNS used to construct a flow-pattern map



Flow-pattern map for the two-dimensional simulations. The non-dimensionalization is based on the upper-layer properties, with $(m = 50, h_0 = 0.2, \mathcal{G} = 0.1, \mathrm{We} = 10)$. Squares – Dispersed liquid phase. Circles – ligaments. Triangle – saturated travelling wave. The insets show snapshots of the three different flow regimes.

Carefully-chosen 3D simulations show the results carry over

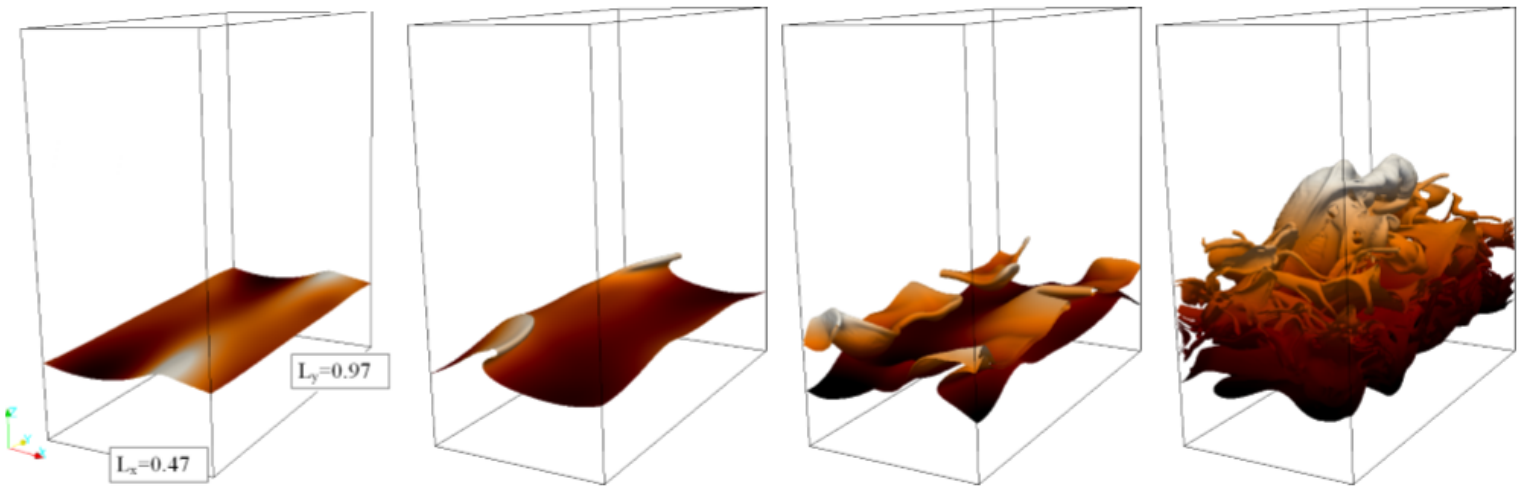


(a) $t = 0.55$ (b) $t = 0.65$ (c) $t = 0.75$

(d) $t = 0.8$ (e) $t = 0.85$ (f) $t = 0.95$

DNS results for the case $(m = 50, h_0 = 0.2, \mathcal{G} = 0.1, \mathrm{We} = 10)$, with $r = 1$ and $\mathrm{Re} = 500$.

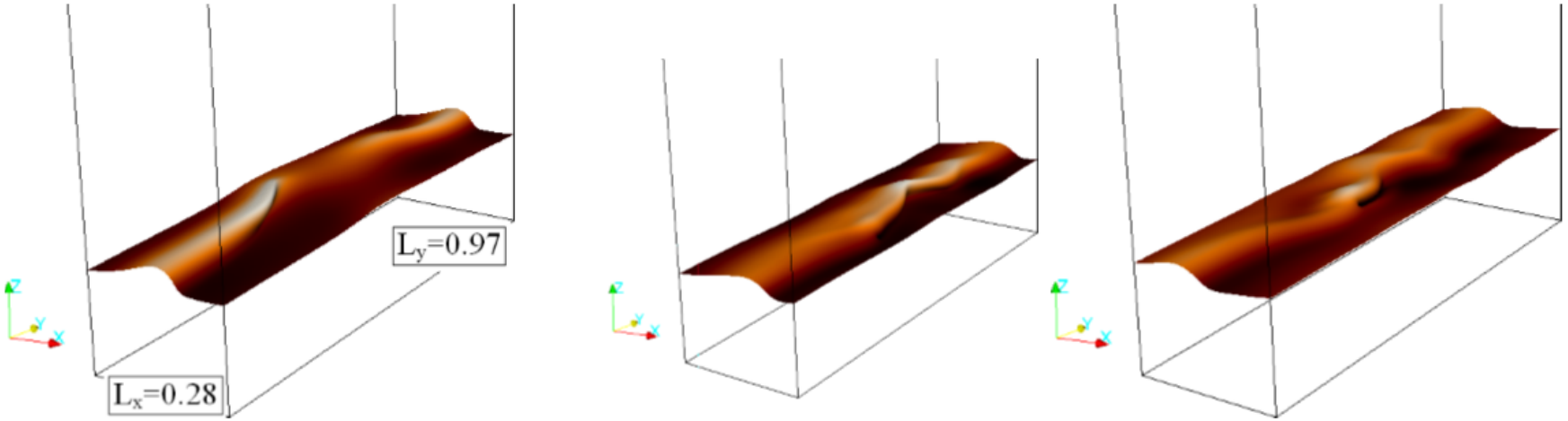
The case $r = 10$



(a) $t = 0.4$ (b) $t = 0.5$ (c) $t = 0.6$ (d) $t = 0.7$

FIG. 30. DNS results for the case $(m = 50, h_0 = 0.2, \mathcal{G} = 0.1, \mathrm{We} = 10)$, with $r = 10$ and $\mathrm{Re} = 500$.

The case $r = 100$



(a) $t = 1.1$ (b) $t = 1.4$ (c) $t = 1.6$

DNS results for the case $(m = 50, h_0 = 0.2, \mathcal{G} = 0.1, \mathrm{We} = 10)$, with $r = 100$ and $\mathrm{Re} = 500$.

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.
- Using TPLS, we answer the question, where do 3D waves in parallel flows come from?

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.
- Using TPLS, we answer the question, where do 3D waves in parallel flows come from?
- For liquid-liquid flows an indirect (weakly nonlinear) mechanism prevails.

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.
- Using TPLS, we answer the question, where do 3D waves in parallel flows come from?
- For liquid-liquid flows an indirect (weakly nonlinear) mechanism prevails.
- Using theory and DNS, our current work shows a different scenario at work in gas-liquid flows:
  - The interfacial waves in gas-liquid ows grow much more slowly than those in corresponding liquid-liquid flows.
  - However, a wider range of wavenumbers (both streamwise and spanwise) are unstable for the gas-liquid case.

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.
- Using TPLS, we answer the question, where do 3D waves in parallel flows come from?
- For liquid-liquid flows an indirect (weakly nonlinear) mechanism prevails.
- Using theory and DNS, our current work shows a different scenario at work in gas-liquid flows:
  - The interfacial waves in gas-liquid ows grow much more slowly than those in corresponding liquid-liquid flows.
  - However, a wider range of wavenumbers (both streamwise and spanwise) are unstable for the gas-liquid case.

Therefore, three-dimensional waves form in gas-liquid ows via a **direct route**: by waiting long enough, streamwise and spanwise modes form as a result of small-amplitude perturbations.

Conclusions

- TPLS has been introduced as a computational methodology for two-phase simulations in an idealized channel geometry.
- Using TPLS, we answer the question, where do 3D waves in parallel flows come from?
- For liquid-liquid flows an indirect (weakly nonlinear) mechanism prevails.
- Using theory and DNS, our current work shows a different scenario at work in gas-liquid flows:
  - The interfacial waves in gas-liquid ows grow much more slowly than those in corresponding liquid-liquid flows.
  - However, a wider range of wavenumbers (both streamwise and spanwise) are unstable for the gas-liquid case.

Therefore, three-dimensional waves form in gas-liquid ows via a **direct route**: by waiting long enough, streamwise and spanwise modes form as a result of small-amplitude perturbations.

- Beyond this early-stage wave growth, a zoo of dierent phenomena is possible, depending on the particular ow parameters involved.