Sparse-matrix algorithms for global eigenvalue problems

P. Luchini, F. Giannetti, J. Pralits

DIMEC, Università di Salerno, Italy
luchini@unisa.it

Global Flow Stability and Control IV

Why a custom algorithm?

- Treating a discretization of the linearized Navier-Stokes equations as a full matrix is too costly to be practical.
- Most work on global stability has used inverse iteration (for a single mode) or ARPACK library with shift-and-invert (for the dominant few modes)
- The inversion part of both these methods is expensive and generally iterative.

Why a custom algorithm?

- Treating a discretization of the linearized Navier-Stokes equations as a full matrix is too costly to be practical.
- Most work on global stability has used inverse iteration (for a single mode) or ARPACK library with shift-and-invert (for the dominant few modes)
- The inversion part of both these methods is expensive and generally iterative.
- It is a waste to iterate to convergence something that is in fact a stage of another, outer iteration. Can we modify the eigenvalue algorithm so that a single step of the inversion procedure can be done per iteration?

From the beginning: direct and inverse iteration

Direct iteration

$$\mathbf{x}^{(n+1)} = (\mathbf{A} - s)\mathbf{x}_n$$

$$\sigma^{(n+1)} = \frac{\mathbf{p} \cdot \mathbf{x}^{(n+1)}}{\mathbf{p} \cdot \mathbf{x}^{(n)}} + s \quad (\mathbf{p} = \text{projector; } \textit{e.g.}\ \mathbf{p} = \mathbf{x}^{(n)*})$$

converges to the eigenvalue (if unique) farthest from the shift s.

Inverse iteration

$$\mathbf{x}^{(n+1)} = (\mathbf{A} - s)^{-1}\mathbf{x}^{(n)}$$

$$\sigma^{(n+1)} = \frac{\mathbf{p} \cdot \mathbf{x}^{(n)}}{\mathbf{p} \cdot \mathbf{x}^{(n+1)}} + s \quad (p = \text{projector; } \textit{e.g.}\ \mathbf{p} = \mathbf{x}^{(n+1)*})$$

converges to the eigenvalue (if unique) closest to the shift s.
Converges quadratically if we let $s = \sigma_n$.
Requires matrix inversion.

Connection with explicit and implicit time integration

Explicit Euler

$$\delta t^{-1}\mathbf{x}^{(n+1)} = (\mathbf{A} + \delta t^{-1})\mathbf{x}^{(n)}$$

same as direct iteration with shift $s = -1/\delta t$.

Implicit Euler

$$\mathbf{x}^{(n+1)} = (\mathbf{A} - \delta t^{-1})^{-1}\delta t^{-1}\mathbf{x}^{(n)}$$

same as inverse iteration with shift $s = +1/\delta t$.

Connection with explicit and implicit time integration

Explicit Euler

$$\delta t^{-1}\mathbf{x}^{(n+1)} = (\mathbf{A} + \delta t^{-1})\mathbf{x}^{(n)}$$

same as direct iteration with shift $s = -1/\delta t$.

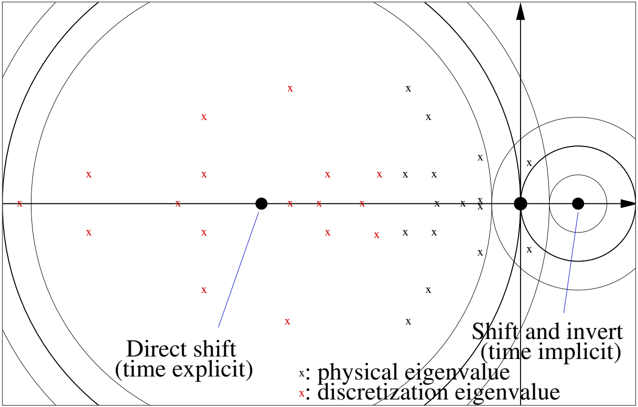
Implicit Euler

$$\mathbf{x}^{(n+1)} = (\mathbf{A} - \delta t^{-1})^{-1}\delta t^{-1}\mathbf{x}^{(n)}$$

same as inverse iteration with shift $s = +1/\delta t$.

- As many realized in the past, an already available time-integration algorithm can be used as the core of an eigenvalue iteration.
- As perhaps not as many realized, first-order time integration gives exact eigenvectors and eigenvalues.

A discretized differential equation is not just any matrix



Subspace iteration

Using a set of K basis vectors $\mathbf{x}_k$:

orthogonalize $\mathbf{x}_k^{(n)}$;
do $\mathbf{x}_k^{(n+1)} = (\mathbf{A} - s)\mathbf{x}_k^{(n)}$ for $k = 1$ to K;
let $\sigma_{hk}^{(n+1)} = (\mathbf{p}_h \cdot \mathbf{x}_k^{(n)})^{-1}(\mathbf{p}_h \cdot \mathbf{x}_k^{(n+1)}) + s$.

Notes:

- The straightforward choice $\mathbf{p}_h = \mathbf{x}_h^{(n)*}$ obviates the need to divide by $\mathbf{x}_h^{(n)*} \cdot \mathbf{x}_k^{(n)} = \delta_{hk}$.
- The eigenvalues of σ_{hk} can be extracted by a standard full-matrix library and converge to the eigenvalues of **A**.
- The rate of convergence is dictated by the first neglected eigenvalue. The leading eigenvectors will converge even if their corresponding eigenvalues are close to each other or multiple.

Inverse subspace iteration

orthogonalize $\mathbf{x}_k^{(n)}$;
do $\mathbf{x}_k^{(n+1)} = (\mathbf{A} - s)^{-1}\mathbf{x}_k^{(n)}$ for $k = 1$ to K;
$\sigma_{hk}^{(n+1)} = (\mathbf{p}_h \cdot \mathbf{x}_k^{(n+1)})^{-1}(\mathbf{p}_h \cdot \mathbf{x}_k^{(n)}) + s$.

- The standard choice $\mathbf{p}_h = \mathbf{x}_h^{(n)*}$ this time eliminates the numerator. By personal experience, to try to eliminate the denominator is a bad idea (I could not find this warning in any book).
- K matrix inversions are required. However,
- they can be performed in parallel.

Arnoldi

A smart way to organize subspace iteration so that the equation

$$\mathbf{x}_k^{(n+1)} = (\mathbf{A} - s)\mathbf{x}_k^{(n)}$$

is already satisfied for $k = 1..K - 1$ and need only be imposed for $k = K$.

- only one application of matrix **A** is needed per step instead of K, but
- K increases by 1 at every step. At a preset value K_{max} the algorithm must be *restarted*, by rotating the current approximations of the leading eigenvectors into the first K_{min} basis vectors.

Implicitly restarted Arnoldi

A smart way (incomplete QR) to rotate the basis vectors so as not to ruin the Arnoldi consistence for $k = 1..K_{min} - 1$

Our method (I): approximate-inverse subspace iteration

Orthogonalize $\mathbf{x}_k^{(n)}$ and rotate σ_{hk} accordingly;
construct $\tilde{\mathbf{x}}_h^{(n+1)} = \sigma_{hk}^{(n)} \mathbf{x}_k^{(n)}$;
with $\mathbf{B}$ being an approximate inverse of $(\mathbf{A} - s)$ do

$$\mathbf{x}_k^{(n+1)} = \tilde{\mathbf{x}}_k^{(n+1)} + \mathbf{B}\left[\mathbf{x}_k^{(n)} - (\mathbf{A} - s)\,\tilde{\mathbf{x}}_k^{(n+1)}\right] \quad \text{for} \;\; k = 1 \;\; \text{to} \;\; K;$$

$$\sigma_{hk}^{(n+1)} = (\mathbf{p}_h \cdot \mathbf{x}_k^{(n)})^{-1} (\mathbf{p}_h \cdot \mathbf{x}_k^{(n+1)}).$$

1. A multigrid algorithm (previously developed for steady-state iteration) provides $\mathbf{B}$.
2. Depending on the accuracy of the approximation, only part of the eigenvectors are usually found to converge.
3. A simultaneous iteration of the base flow achieves convergence in unstable cases.

Our method (II): towards inversion-free Arnoldi (or Jacobi-Davidson?)

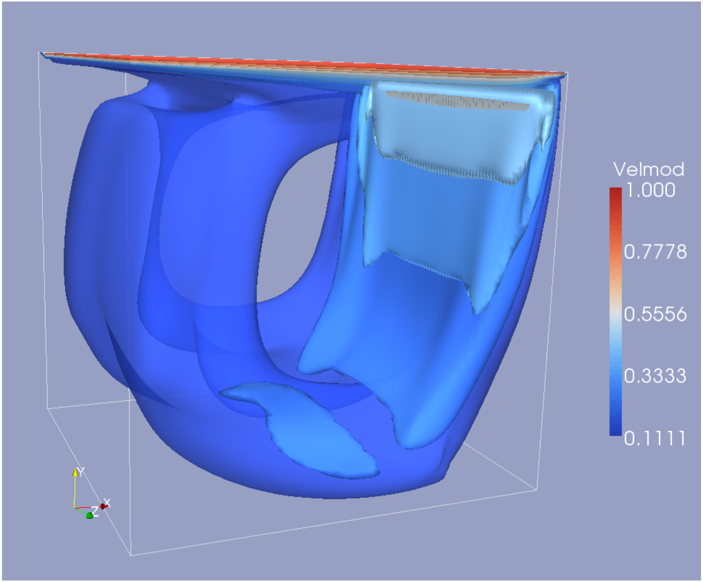
Instead of updating each basis vector, increase K by 1 and use the correction as a new basis vector;
when $K = K_{max}$, deflate the search space to K_{min} by the same incomplete QR procedure as in IRAM.

- If **B** was an exact inverse, only the last vector would generate a correction and the method would essentially coincide with IRAM.
- Only the vectors with the largest residuals need to generate corrections. Different strategies can be devised and are being experimented with.

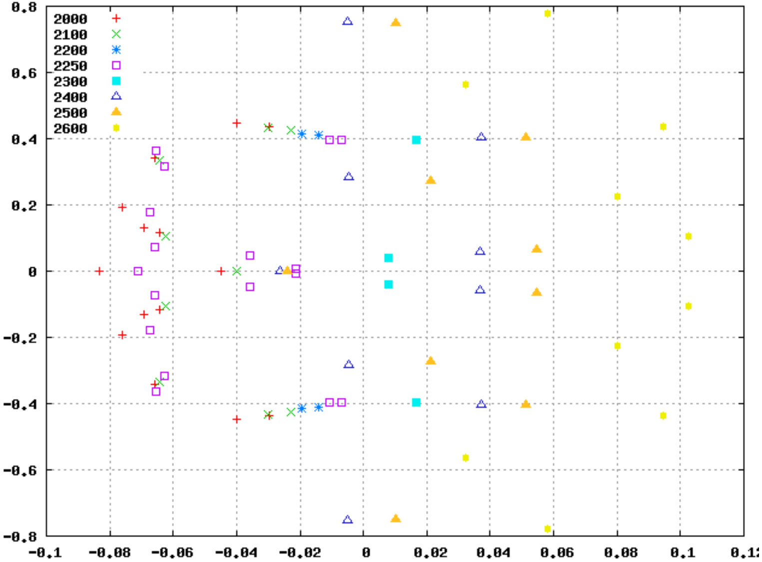
The lid-driven cavity test case

- $2\frac{1}{2}$D stability problem: suboptimal wavenumber (Ramanan & Homsy 1994).
- $2\frac{1}{2}$D stability problem: $Re_{cr} = 786$; $\beta = 15.8$ (Albensöder & Kuhlmann 2001).
- 2D stability problem: $Re_{cr} = 8018$ (Parolini, Auteri & Quartapelle 2002).
- 3D stability problem: AFAIK only solved for aspect ratio 1:1:6 (Albensöder & Kuhlmann 2001)
- 3D stability problem in a 1:1:1 cubical cavity: Re_{cr} observed to lie between 2000 and 3000 from direct simulations (Iwahatsu, Ishii & Kawanura 1989).
- *Eigenvalues to follow next...*

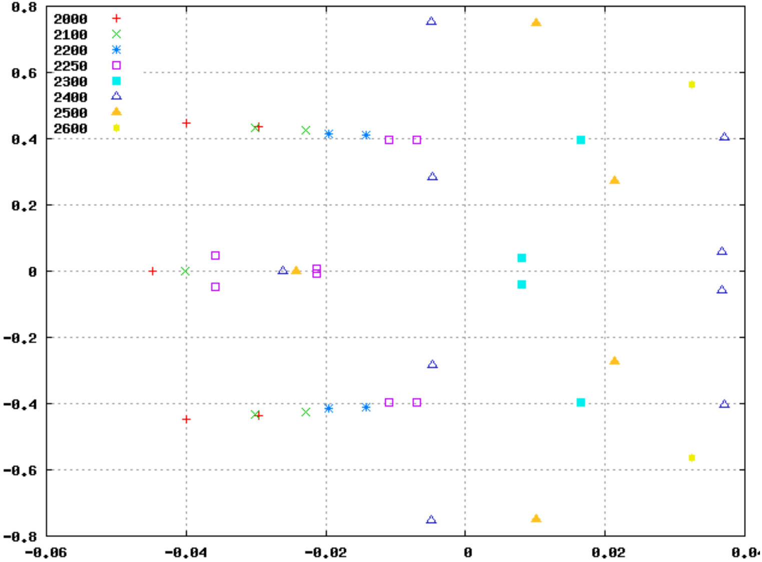
Driven cubic box base flow (Re=2000, 64^3).



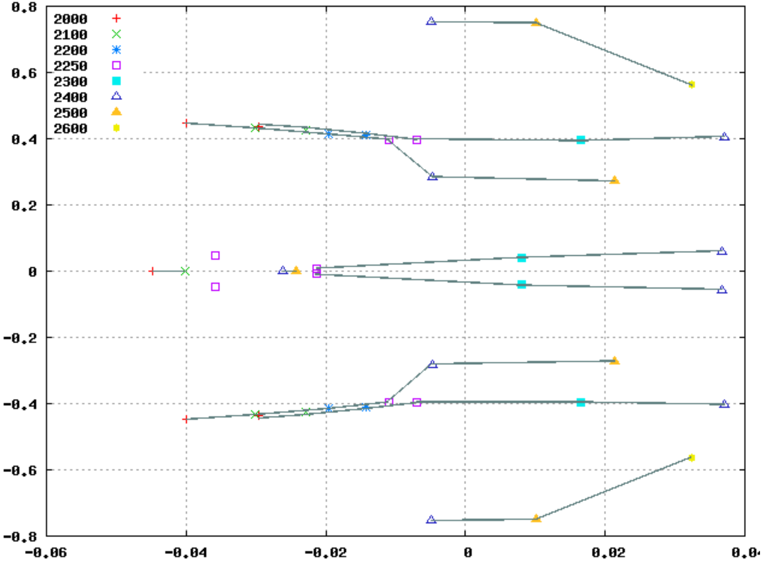
Driven cubic box spectrum (64^3).



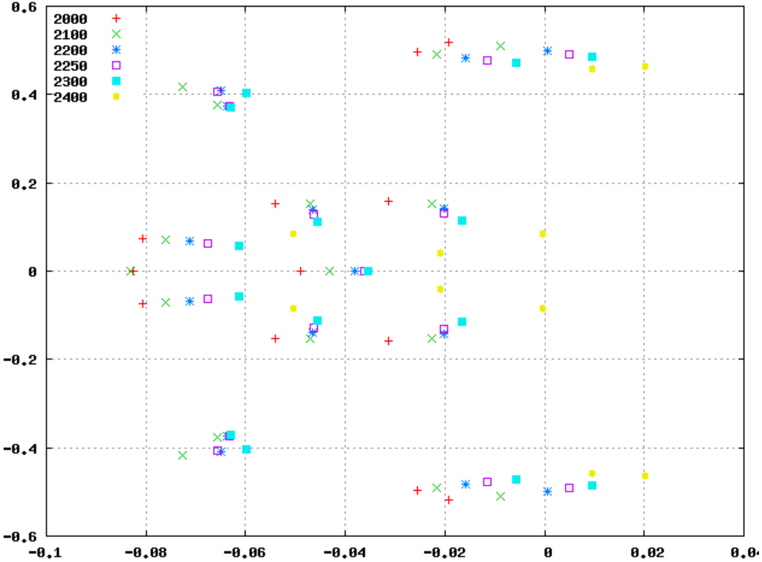
Driven cubic box spectrum (64^3).



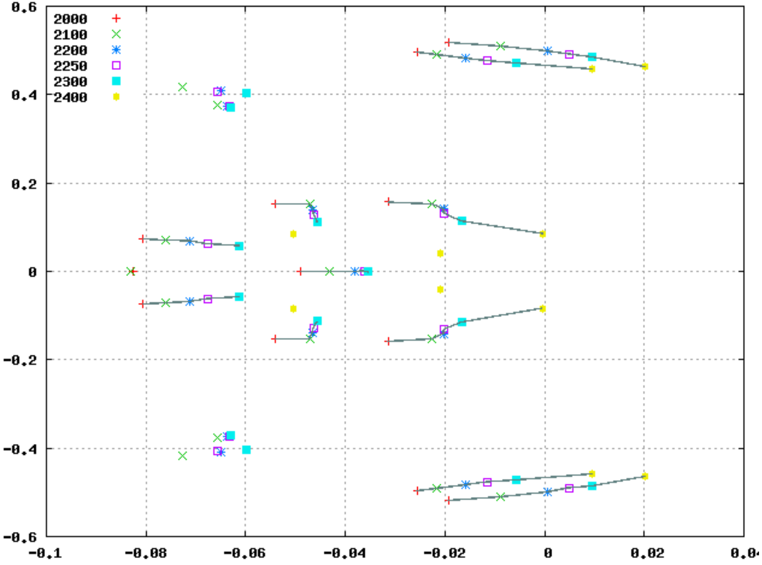
Driven cubic box spectrum (64^3).



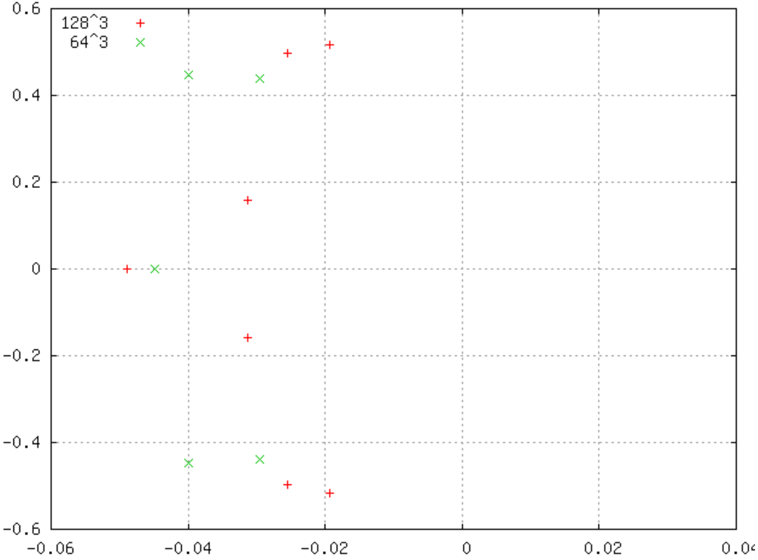
Driven cubic box spectrum (128^3).



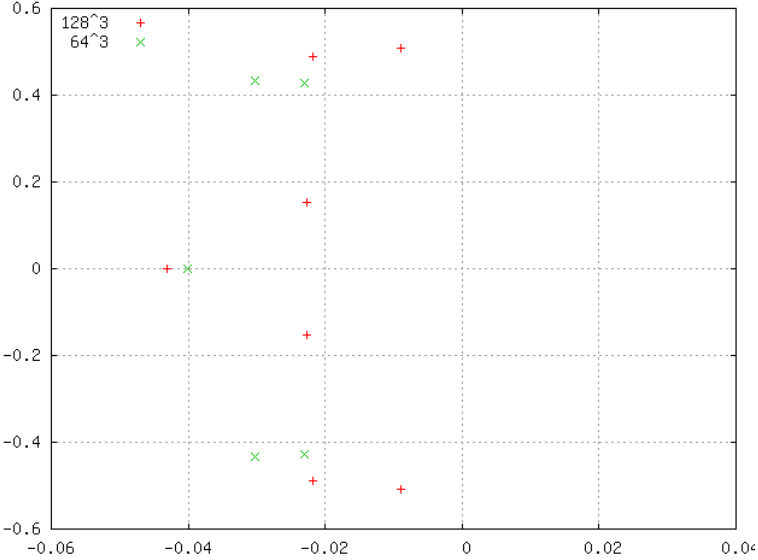
Driven cubic box spectrum (128^3).



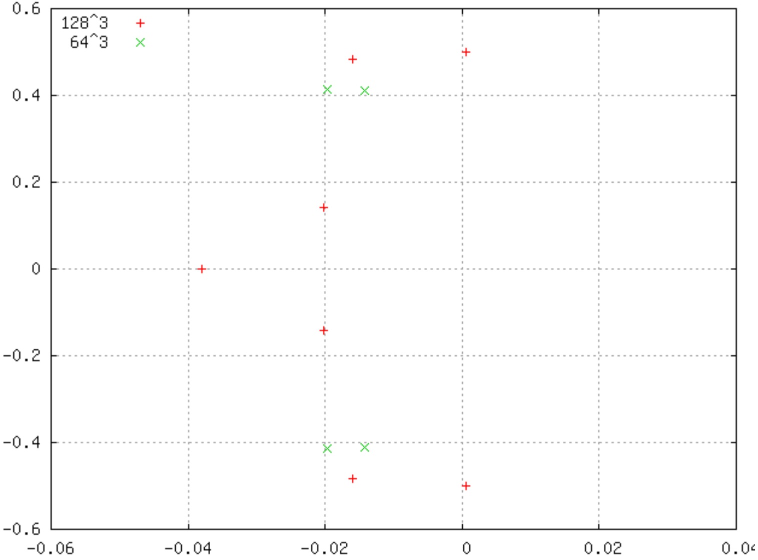
Resolution comparison. Re=2000



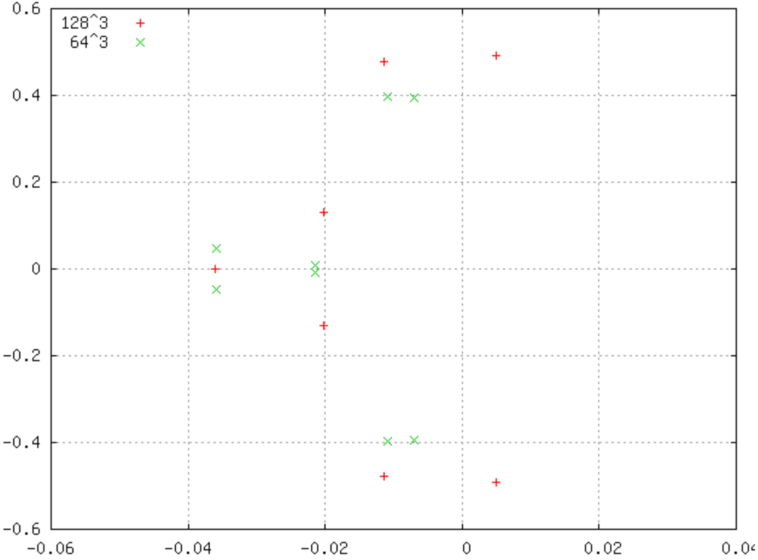
Resolution comparison. Re=2100



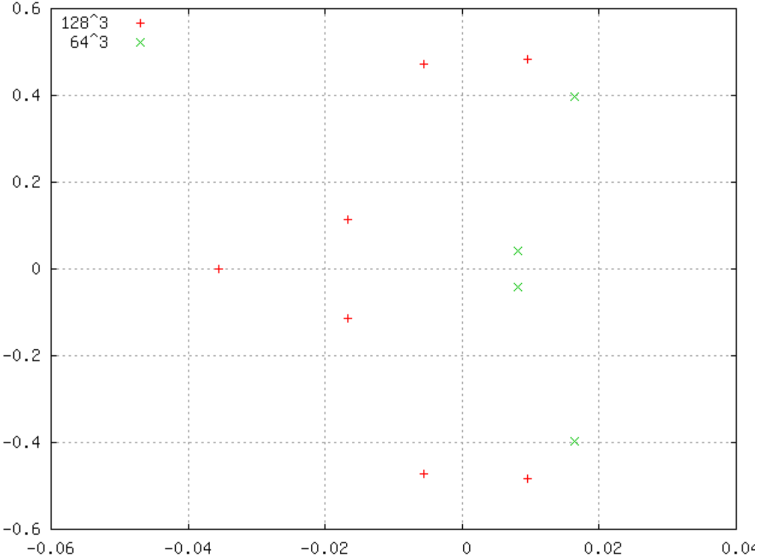
Resolution comparison. Re=2200



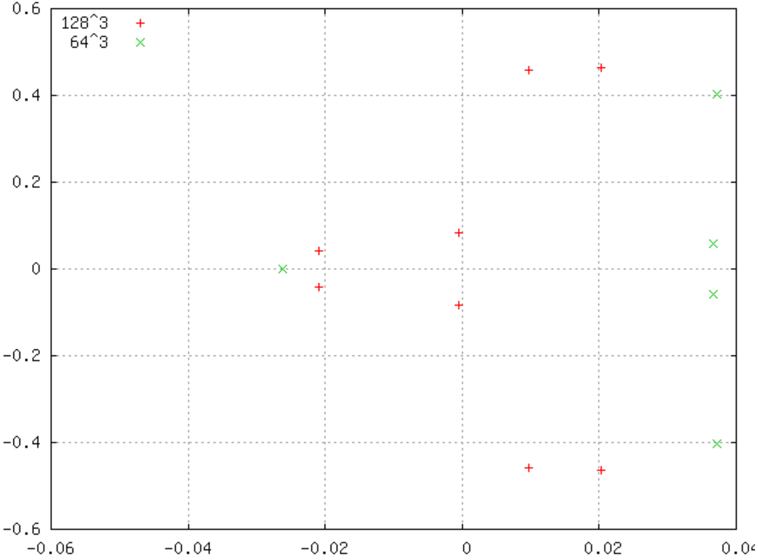
Resolution comparison. Re=2250



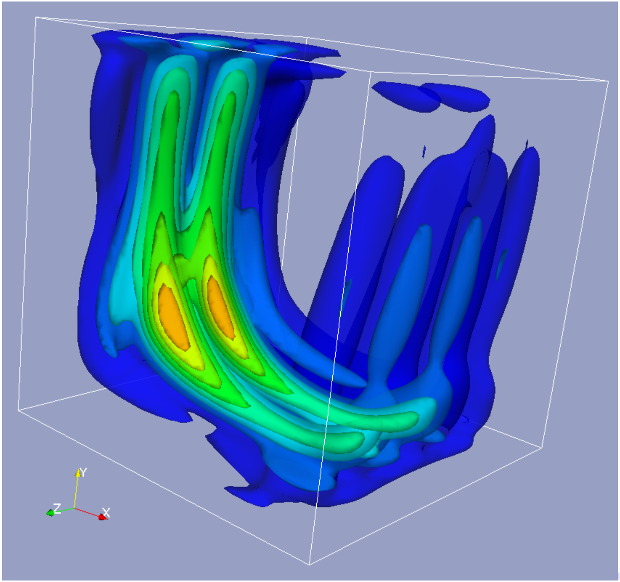
Resolution comparison. Re=2300



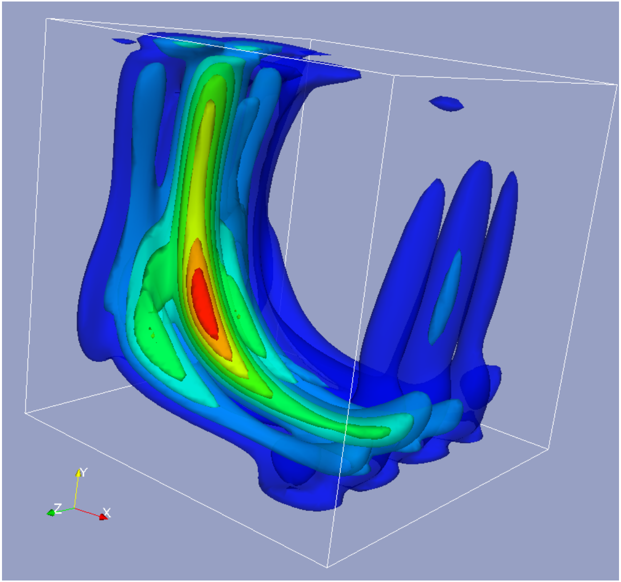
Resolution comparison. Re=2400



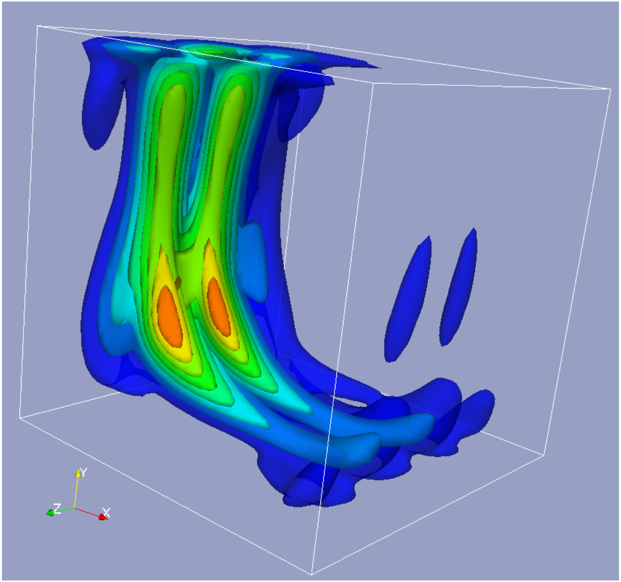
High-frequency odd mode (Re=2000, 64^3).



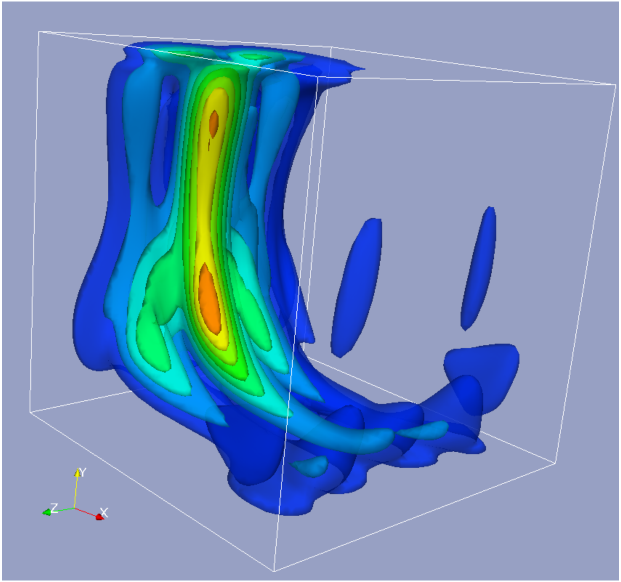
High-frequency even mode (Re=2000, 64^3).



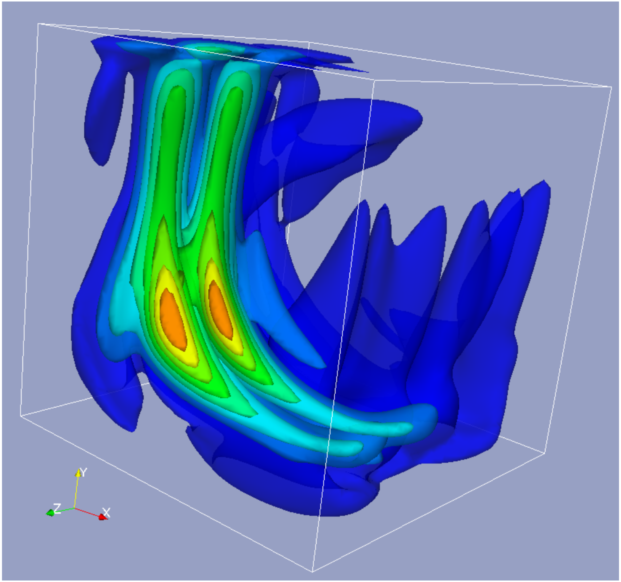
Low-frequency odd mode (Re=2000, 64^3).



Low-frequency even mode (Re=2000, 64^3).



Zero-frequency least stable mode (Re=2000, 64^3).



Conclusion

- Method I: approximate-inverse subspace iteration with multigrid.
- Method II: IRAM-like version of the above.
- Eigenvalue spectrum of the 3D 1:1:1 lid-driven cavity. $Re_{cr} = 2200, \omega = 0.50$

Ongoing developments:

- improved parallelization (256^3, 24+ modes),
- combined direct-adjoint iteration (Lanczos),
- immersed boundary, non-uniform grid and application to open flows.