Algorithmic Developments and Software Engineering for Scalable Sparse Eigensolvers in the DFG Project ESSEX

Achim Basermann    Jonas Thies*    A. Alvermann    H. Fehske†    B. Lang‡

* German Aerospace Center
  Simulation and Software Technology

† University of Greifswald
  Department of Physics

‡ University of Wuppertal
  Department of Mathematics

project ESSEX
Survey

- Use Case: Linear Stability Analysis
- Block Jacobi-Davidson Eigensolver with Preconditioning
- BEAST Framework
- Chebyshev Filter Diagonalization
- Programming Model
- Solver Toolkit PHIST
- Outlook
Use Case: Linear Stability Analysis of 3D Flow Problems

Problem setting

- incompressible Navier-Stokes
- Boussinesq approximation
- structured C-grid (‘staggered grid’)
- linear stability analysis: compute most unstable eigenmodes of the Jacobian $A$.

Numerical challenge

- solve $Ax = \lambda Bx$ for the right-most eigenpairs $(\lambda, x)$,
- $A$ is large, sparse and non-symmetric,
- $A$ is a saddle point matrix,
- $B$ is symmetric and positive semi-definite.
Block Jacobi-Davidson Eigensolver in PHIST

**Block JDQR algorithm**

- extend basis $V$ by $k$ vectors per iteration
- Rayleigh-Ritz, solve $V^T AV\tilde{s} = \tilde{\lambda} V^T BV\tilde{s}$
- correction equation: solve $(A - \tilde{\lambda}B)t = -\text{res}$, $t \perp_B V$

**Performance gain over single-vector algorithm:**

- better cache performance of SpMVM
- fewer synchronizations by collecting dot products
- blocked and fused kernels to additionally save memory traffic
SMILU: Staggered-Grid Multi-Level ILU

- Partition into many small subdomains
SMILU: Staggered-Grid Multi-Level ILU

- Partition into many small subdomains
- eliminate interior
SMILU: Staggered-Grid Multi-Level ILU

- Partition into many small subdomains
- Eliminate interior
- Aggregate and drop on separators
SMILU: Staggered-Grid Multi-Level ILU

- Partition into many small subdomains
- Eliminate interior
- Aggregate and drop on separators
- Repeat recursively
SMILU: Staggered-Grid Multi-Level ILU

- Partition into many small subdomains
- Eliminate interior
- Aggregate and drop on separators
- Repeat recursively

implies Robust, spectrally equivalent, mass and energy conserving ILU
Solving the JD Correction Equations

Let $V = [V_1, V_2]^T$, $W = BV$. A bordered Schur-Complement preconditioner

$$
\begin{pmatrix}
A_{11} & A_{12} & W_1 \\
A_{21} & A_{22} & W_2 \\
V_1^T & V_2^T & 0
\end{pmatrix}
\approx
\begin{pmatrix}
L_{11} & 0 & 0 \\
A_{21}U_{11}^{-1} & \hat{L}_{22} \\
\hat{V}_1^T & 0 & 0
\end{pmatrix}
\begin{pmatrix}
U_{11} & L_{11}^{-1}A_{12} & \hat{W}_1 \\
0 & 0 & \hat{U}_{22}
\end{pmatrix}
$$

where

$$
\hat{L}_{22} \hat{U}_{22} \approx \begin{pmatrix}
A_{22} - A_{21}A_{11}^{-1}A_{12} & W_2 - A_{21}\hat{W}_1 \\
V_2^T - \hat{V}_1^TA_{12} & -\hat{V}_1^T\hat{W}_1
\end{pmatrix}, \hat{W}_1 = L_{11}^{-1}W_1, \hat{V}_1^T = V_1^TU_{11}^{-1}.
$$

- yields vectors (B-)orthogonal to the search space $V$,
- only needs to re-factor last-level Schur complement when updating $V$,
- does not increase the fill because $V$ is already dense.
BEAST (Beyond FEAST)

An implementation of the FEAST algorithm, Chebyshev filter diagonalization, and, most recently, CIRR/SSM to solve large scale eigenproblems $AX = BX\Lambda$ (with $A = A^H$, $B$ hpd) in an interval $[\lambda, \bar{\lambda}]$ on hybrid-parallel high performance supercomputers.
BEAST Framework

Choose $Y \in \mathbb{C}^{n \times m}$

Possible projections:

- $U = S(A) Y$, where $S(\lambda) = \sum_{i=0}^{d} c_i T_i(\lambda)$ (Polynomial)

- $U \approx \frac{1}{2\pi i} \oint_{C} (zB - A)^{-1} BYdz$ (FEAST)

- $U = [U^0, U^1, \ldots, U^{l-1}]$ with $U^k = \frac{1}{2\pi i} \int_{C} z^k (zB - A)^{-1} BYdz$ (SSM)

Postprocess $U$

Rayleigh-Ritz: \[
\begin{align*}
A_U &:= U^H A U; \quad B_U := U^H B U; \\
\text{Solve } A_U W &= B_U W \Lambda; \quad X := UW
\end{align*}
\]

Repeat with $Y := X$
Traditional Polynomial Filters

\[ S(\lambda) = \sum_{i=0}^{d} c_i T_i(\lambda) \]

Dirichlet  

Jackson  

Lanczos(1)  

Fejer  

Lorentz(1)  

Wang-Zunger(1,1)
A-Priori Optimization of Separation Properties

Require

• $S(\lambda) \geq \tau_{\text{inside}}$ for $\lambda \in [\lambda, \bar{\lambda}]$
• $|S(\lambda)| \leq \tau_{\text{outside}}$ for $\lambda \notin [\lambda - \delta, \bar{\lambda} + \delta]$

Goal: minimize $\delta$; adaptively control the polynomial degree
Using BEAST with Moments

Use Sakurai-Sugiura methods (SS-RR) to reduce cost of building $U$

$U_j^0 = (z_jB-A)^{-1}$

- $U$ constructed using moments ($z_j^k$)
- Reduce overall number of right hands sides

- Improvement compared to pure FEAST scheme
- BEAST-M*: switch from SS-RR to FEAST between iterations
BEAST Framework Features

- Leverage the hybrid-parallel performance of GHoST
- A priori eigencount estimation via KPM (soon also stochastic estimation)
- Locking of converged eigenpairs (w/ re-orthogonalization)
- On-the-fly eigencount estimation and subspace reduction
- Generic linear solver interface, CARP-CG (via PHIST)
- Adaptive choice of polynomial degree (and number of integration nodes)
- Optimized polynomial coefficients
- SDC detection (also Checkpointing, in the future)
- Mixed precision
- Multi-level parallelism
- Orthogonality
ChebFD: Chebyshev Filter Diagonalization

(large sparse) symmetric / Hermitian matrix $A$

interior eigenvalues $\lambda_k, \lambda_{k+1}, \ldots, \lambda_{k+N_T} \in [a, b]$

eigenvalue density (density of states) $\rho(\lambda) = \sum_k \delta(\lambda - \lambda_k)$

$\# \text{ target values}$

$N_T \approx \int_a^b \rho(\lambda) \, d\lambda$

A. Pieper, M. Kreutzer, A. Alvermann, M. Galgon, H. Fehske, G. Hager, B. Lang, G. Wellein,
“High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations”,
ChebFD: Parallel Implementation

- uses only spMVM ⇒ suitable for very large unstructured matrices
- convergence depends on eigenvalue density ⇒ large number of spMVMs required
- search vectors ⇒ target vectors ⇒ massive block algorithm ⇒ spMMVM

Node-level performance ...

... is high with spMMVM

Parallel scalability ...

... is good due to iterative scheme (e.g., few synchronization points)

- matrix dimension ≈ 10^9 no problem even for ≈ 200 interior eigenvalues

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Nodes</th>
<th>D</th>
<th>[k : \lambda]</th>
<th>N_F</th>
<th>N_P</th>
<th>Runtime [hours]</th>
<th>Sust. perf. [TFl ops/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>top1</td>
<td>32</td>
<td>6.71e7</td>
<td>7.27e-3</td>
<td>148</td>
<td>2159</td>
<td>3.2 (83%)</td>
<td>2.96</td>
</tr>
<tr>
<td>top2</td>
<td>128</td>
<td>3.64e-3</td>
<td>3.62e-3</td>
<td>148</td>
<td>4319</td>
<td>4.9 (88%)</td>
<td>11.5</td>
</tr>
<tr>
<td>top3</td>
<td>512</td>
<td>1.07e9</td>
<td>4.84e-4</td>
<td>104</td>
<td>32603</td>
<td>10.8 (99%)</td>
<td>4.6</td>
</tr>
<tr>
<td>graphene1</td>
<td>128</td>
<td>1.07e9</td>
<td>2.42e-4</td>
<td>104</td>
<td>64926</td>
<td>16.4 (99%)</td>
<td>18.2</td>
</tr>
<tr>
<td>graphene2</td>
<td>512</td>
<td>1.07e9</td>
<td>2.42e-4</td>
<td>104</td>
<td>64926</td>
<td>16.4 (99%)</td>
<td>18.2</td>
</tr>
</tbody>
</table>
SPMD/OK Programming Model

- SPMD (‘BSP’) vs. task parallelism
- Heterogenous cluster: distribute problem according to limiting resource (e.g. memory bandwidth)
- Optimized Kernels make sure each component runs as fast as possible
- User sees a simple functional interface (no general-purpose looping constructs etc.)

A success story: Chebyshev methods on Piz Daint

Only needs sparse matrix times multiple vector (spMMV) products and an occasional vector operation
PHIST Software Architecture

a Pipelined Hybrid-parallel Iterative Solver Toolkit

- facilitate algorithm development using **GHST**
- holistic performance engineering
- portability and interoperability
PHIST Software Architecture

a Pipelined Hybrid-parallel Iterative Solver Toolkit

- facilitate algorithm development using **GHIST**
- holistic performance engineering
- portability and interoperability
Useful Abstraction: Kernel Interface

Choose from several ‘backends’ at compile time, to

- easily use **PHIST** in existing applications
- perform the same run with different kernel libraries
- compare numerical accuracy and performance
- exploit unique features of a kernel library (e.g. preconditioners)
What’s new in PHIST?

New features

- allow general preconditioners in Jacobi-Davidson solver
- solve generalized EVP (with hpd B): $A\mathbf{x} = \lambda B\mathbf{x}$
- new blocked Krylov solvers BiCGStab and TFQMR
- full GPU and hybrid support with GHOST
- automatically generated Fortran 2003 and C++ bindings for C API
- new kernel libraries: PETSc, Eigen

New applications

- stability analysis of flows (ongoing, with U. Groningen)
- rational Krylov method for model order reduction (ongoing work @ MPI Magdeburg)
- material science (structure optimization, plans with U. Erlangen and DLR material physics)
Outlook

- integrate **GHDel** kernels into Trilinos/Kokkos
- **PHIST** to enter the xSDK (https://xsdk.info/)
- implementation of kernels with data dependencies using RACE (Gauß-Seidel, Kaczmarz and (multi-level) ILU)
Questions?

Contact

Dr.-Ing. Achim Basermann  
Department Head  
*High Performance Computing*  
Simulation and Software Technology  
German Aerospace Center (DLR)

Achim.Basermann@DLR.de  
Phone +49 (0)2203 / 601 33 26  
http://www.DLR.de/sc

The ESSEX II team