Parallel solution of large sparse eigenproblems using a Block-Jacobi-Davidson method

Melven Röhrig-Zöllner

Simulation and Software Technology
German Aerospace Center (DLR)

April 25, 2014

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

RWTH AACHEN UNIVERSITY

Background

Aim: Calculate a set of outer eigenpairs (λ_i, v_i) of a large sparse matrix:

$$Av_i = \lambda_i v_i$$

Project: Equipping Sparse Solvers for Exascale (ESSEX) of the DFG SPPEXA programme

Outline

Outline

Jacobi-Davidson QR method

Background

- Introduced 1998 by Fokkema et. al.
- Motivated by inexact Newton / Rayleigh quotient iteration (RQI)

Jacobi-Davidson QR method

Background

- Introduced 1998 by Fokkema et. al.
- Motivated by inexact Newton / Rayleigh quotient iteration (RQI)

Sketch of the algorithm

while not converged **do** ▷ Outer iteration
    Project problem to small subspace
    Solve small eigenvalue problem
    Calculate approximation and residual
    Approximately solve correction equation ▷ Inner iteration
    Orthogonalize result
    Enlarge subspace
end while

Outer iteration

Subspace iteration

Deflation

Outer iteration

Subspace iteration

- Galerkin projection onto small subspace $\mathcal{W} \subset \mathbb{R}^n$:

$$
\begin{aligned}
& Av - \lambda v \perp \mathcal{W}, && v \in \mathcal{W} \\
\Leftrightarrow \quad & (W^T A W)s - \tilde{\lambda} s = 0, && \text{for orth. basis } W
\end{aligned}
$$

Deflation

Outer iteration

Subspace iteration

- Galerkin projection onto small subspace $\mathcal{W} \subset \mathbb{R}^n$:

$$
\begin{aligned}
& Av - \lambda v \perp \mathcal{W}, && v \in \mathcal{W} \\
\Leftrightarrow \quad & (W^T A W)s - \tilde{\lambda} s = 0, && \text{for orth. basis } W
\end{aligned}
$$

- Approximation $\tilde{\lambda}$ with $\tilde{v} = Ws$

Deflation

Outer iteration

Subspace iteration

- Galerkin projection onto small subspace $\mathcal{W} \subset \mathbb{R}^n$:

$$\begin{aligned} & Av - \lambda v \perp \mathcal{W}, && v \in \mathcal{W} \\ \Leftrightarrow \quad & (W^T A W)s - \tilde{\lambda} s = 0, && \text{for orth. basis } W \end{aligned}$$

- Approximation $\tilde{\lambda}$ with $\tilde{v} = Ws$

Deflation

- Project out already known eigenvector space Q:

$$\bar{A} := (I - QQ^T)A(I - QQ^T)$$

Jacobi-Davidson correction equation

Basic approach

- Based on current approximation $A\tilde{v} - \tilde{\lambda}\tilde{v} = r$

Jacobi-Davidson correction equation

Basic approach

- Based on current approximation $A\tilde{v} - \tilde{\lambda}\tilde{v} = r$
- Avoid (near) singularity of $(A - \tilde{\lambda}I)^{-1}$ in RQI/Newton through a projection with $\tilde{Q}^{\perp} = \begin{pmatrix} Q & \tilde{v} \end{pmatrix}^{\perp}$

Jacobi-Davidson correction equation

Basic approach

- Based on current approximation $A\tilde{v} - \tilde{\lambda}\tilde{v} = r$
- Avoid (near) singularity of $(A - \tilde{\lambda}I)^{-1}$ in RQI/Newton through a projection with $\tilde{Q}^{\perp} = \begin{pmatrix} Q & \tilde{v} \end{pmatrix}^{\perp}$

→ Solve approximately:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}I)(I - \tilde{Q}\tilde{Q}^T)w_{k+1} = -r$$

Jacobi-Davidson correction equation

Basic approach

- Based on current approximation $A\tilde{v} - \tilde{\lambda}\tilde{v} = r$
- Avoid (near) singularity of $(A - \tilde{\lambda}I)^{-1}$ in RQI/Newton through a projection with $\tilde{Q}^{\perp} = \begin{pmatrix} Q & \tilde{v} \end{pmatrix}^{\perp}$

$\rightarrow$ Solve approximately:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}I)(I - \tilde{Q}\tilde{Q}^T)w_{k+1} = -r$$

- Use some steps of iterative solver (GMRES, BiCGStab, ...)
  $\rightarrow$ inner iteration

Jacobi-Davidson correction equation

Basic approach

- Based on current approximation $A\tilde{v} - \tilde{\lambda}\tilde{v} = r$
- Avoid (near) singularity of $(A - \tilde{\lambda} I)^{-1}$ in RQI/Newton through a projection with $\tilde{Q}^{\perp} = \begin{pmatrix} Q & \tilde{v} \end{pmatrix}^{\perp}$

$\rightarrow$ Solve approximately:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda} I)(I - \tilde{Q}\tilde{Q}^T) w_{k+1} = -r$$

- Use some steps of iterative solver (GMRES, BiCGStab, ...) $\rightarrow$ inner iteration
- Provides new direction w_{k+1} for the subspace iteration

Block JDQR method

Idea

- Calculate corrections for n_b eigenvalues at once
  $\rightarrow$ hopefully improves performance

Block JDQR method

Idea

- Calculate corrections for n_b eigenvalues at once
  $\rightarrow$ hopefully improves performance
- Block correction equation with $\tilde{Q} = \begin{pmatrix} Q & \tilde{v}_1 & \dots & \tilde{v}_{n_b} \end{pmatrix}$:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}_i I)(I - \tilde{Q}\tilde{Q}^T) w_{k+i} = -r_i \qquad i = 1, \dots, n_b$$

Block JDQR method

Idea

- Calculate corrections for n_b eigenvalues at once
  $\rightarrow$ hopefully improves performance
- Block correction equation with $\tilde{Q} = \begin{pmatrix} Q & \tilde{v}_1 & \dots & \tilde{v}_{n_b} \end{pmatrix}$:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}_i I)(I - \tilde{Q}\tilde{Q}^T) w_{k+i} = -r_i \qquad i = 1, \dots, n_b$$

$\rightarrow$ Approximately solve n_b linear systems at once

Block JDQR method

Idea

- Calculate corrections for n_b eigenvalues at once
  $\rightarrow$ hopefully improves performance
- Block correction equation with $\tilde{Q} = \begin{pmatrix} Q & \tilde{v}_1 & \dots & \tilde{v}_{n_b} \end{pmatrix}$:

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}_i I)(I - \tilde{Q}\tilde{Q}^T) w_{k+i} = -r_i \qquad i = 1, \dots, n_b$$

$\rightarrow$ Approximately solve n_b linear systems at once

- Provides n_b new directions w_{k+i} for the subspace iteration

Outline

Required linear algebra operations

Sparse matrix-multiple-vector multiplication (spMMVM)

Block vector operations

Required linear algebra operations

Sparse matrix-multiple-vector multiplication (spMMVM)

- distributed CRS format (stores non-zero entries row-wise)
- avoid loading A several times (memory bounded)

Block vector operations

Deutsches Zentrum für Luft- und Raumfahrt
DLR German Aerospace Center

RWTH AACHEN UNIVERSITY

Required linear algebra operations

Sparse matrix-multiple-vector multiplication (spMMVM)

- distributed CRS format (stores non-zero entries row-wise)
- avoid loading A several times (memory bounded)

Block vector operations

- different types of operations: $(V, W \in \mathbb{R}^{n \times n_b})$

	local	all-reduction
BLAS 1	$V + W$	$\|v_i\|$
BLAS 3	VM	$V^T W$

Required linear algebra operations

Sparse matrix-multiple-vector multiplication (spMMVM)

- distributed CRS format (stores non-zero entries row-wise)
- avoid loading A several times (memory bounded)

Block vector operations

- different types of operations: $(V, W \in \mathbb{R}^{n \times n_b})$

	local	all-reduction
BLAS 1	$V + W$	$\|v_i\|$
BLAS 3	VM	$V^T W$

- message aggregation for all-reductions
- better code balance for BLAS 3 operations (memory bounded)

spMMVM single node

Setup

- Matrix dimensions: $n \approx 10^6$ with $n_{nzr} \approx 10$ non-zeros per row
- CRS format
- 6-core Intel Westmere CPU

spMMVM single node

Setup

- Matrix dimensions: $n \approx 10^6$ with $n_{nzr} \approx 10$ non-zeros per row
- CRS format
- 6-core Intel Westmere CPU
- Roofline performance model:

$$P_{ideal} = \min\left(P_{peak}, \frac{b_{data}}{B_{code}}\right)$$

spMMVM single node

Setup

- Matrix dimensions: $n \approx 10^6$ with $n_{nzr} \approx 10$ non-zeros per row
- CRS format
- 6-core Intel Westmere CPU
- Roofline performance model:

$$P_{ideal} = \min\left(P_{peak}, \frac{b_{data}}{B_{code}}\right)$$

- Code balance:

$$B_{CRS,NT} = \frac{6}{n_b} + \frac{8}{n_{nzr}} + \frac{\kappa}{2}\left[\frac{bytes}{flops}\right]$$

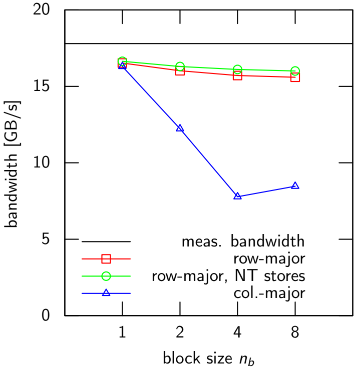
spMMVM single node (2)

Bandwidth

Performance

spMMVM single node (2)

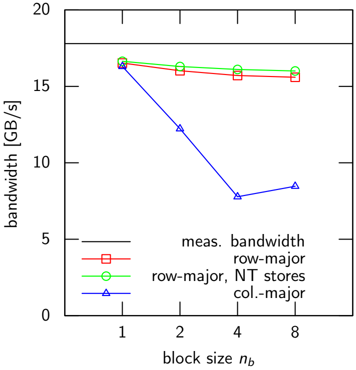
Bandwidth



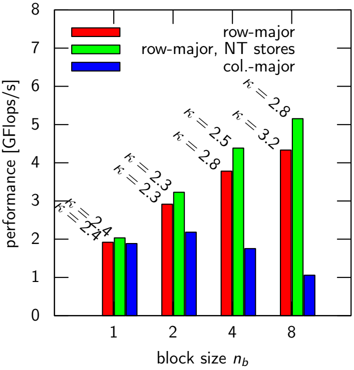
Performance

spMMVM single node (2)

Bandwidth



Performance



Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

Block vector operations

Jacobi-Davidson Operator

- spMMVM + $2\times$ GEMM:

$$y_i \leftarrow (I - QQ^T)(A - \sigma_i I)x_i$$

with $Q \in \mathbb{R}^{n\times 10}$, $i = 1, \ldots, n_b$

Block vector operations

Jacobi-Davidson Operator

- spMMVM + 2× GEMM:

$$y_i \leftarrow (I - QQ^T)(A - \sigma_i I)x_i$$

with $Q \in \mathbb{R}^{n\times 10}$, $i = 1, \ldots, n_b$

- For GEMM here: performance model is accurate!

Block vector operations

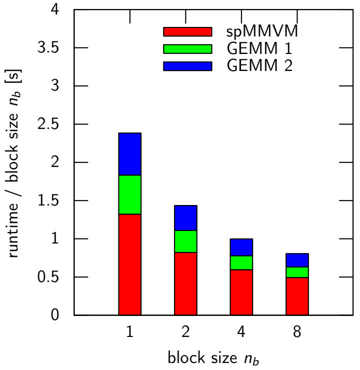
Jacobi-Davidson Operator

- spMMVM + 2× GEMM:

$$y_i \leftarrow (I - QQ^T)(A - \sigma_i I)x_i$$

with $Q \in \mathbb{R}^{n \times 10}$, $i = 1, \ldots, n_b$

- For GEMM here: performance model is accurate!



Block vector operations

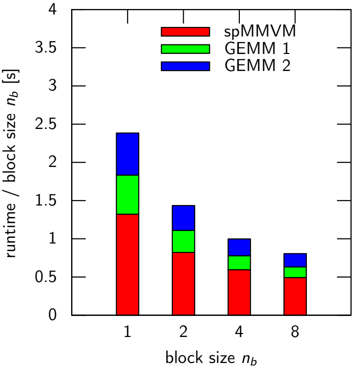
Jacobi-Davidson Operator

- spMMVM + 2× GEMM:

$$y_i \leftarrow (I - QQ^T)(A - \sigma_i I)x_i$$

with $Q \in \mathbb{R}^{n \times 10}$, $i = 1, \ldots, n_b$

- For GEMM here: performance model is accurate!
- 1.6 times faster for $n_b = 2$
  2.5 times faster for $n_b = 4$



spMMVM inter-node

Setup

- strong scaling
  $(n \approx 10^7, n_{nzr} \approx 15)$

spMMVM inter-node

Setup

- strong scaling
  $(n \approx 10^7, n_{nzr} \approx 15)$
- Distributed CRS
  $\rightarrow$ communication overhead

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

RWTH AACHEN UNIVERSITY

spMMVM inter-node

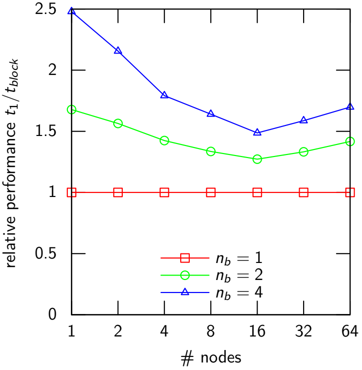
Setup

- strong scaling
  $(n \approx 10^7, n_{nzr} \approx 15)$
- Distributed CRS
  $\rightarrow$ communication overhead
- Each node:
  2× 6-core Intel Westmere

spMMVM inter-node

Setup

- strong scaling
  $(n \approx 10^7, n_{nzr} \approx 15)$
- Distributed CRS
  $\rightarrow$ communication overhead
- Each node:
  2× 6-core Intel Westmere



Outline

Block-Jacobi-Davidson algorithm

Performance analysis

Implementation

- Frameworks from the ESSEX project
- Block pipelined GMRES algorithm

Results

Conclusion

Frameworks from the ESSEX project

phist (Pipelined Hybrid-parallel Linear Solver Toolkit)

GHOST (General Hybrid Optimized Sparse Toolkit)

Frameworks from the ESSEX project

phist (Pipelined Hybrid-parallel Linear Solver Toolkit)

- General C-interface to linear algebra libraries:
  - Trilinos (C++, `http://trilinos.sandia.gov`)
  - GHOST
  - self-written (row-major storage, Fortran + C99, NT-stores)

GHOST (General Hybrid Optimized Sparse Toolkit)

Frameworks from the ESSEX project

phist (Pipelined Hybrid-parallel Linear Solver Toolkit)

- General C-interface to linear algebra libraries:
  - Trilinos (C++, `http://trilinos.sandia.gov`)
  - GHOST
  - self-written (row-major storage, Fortran + C99, NT-stores)
- Iterative solvers

GHOST (General Hybrid Optimized Sparse Toolkit)

Frameworks from the ESSEX project

phist (Pipelined Hybrid-parallel Linear Solver Toolkit)

- General C-interface to linear algebra libraries:
  - Trilinos (C++, `http://trilinos.sandia.gov`)
  - GHOST
  - self-written (row-major storage, Fortran + C99, NT-stores)
- Iterative solvers
- Large test framework

GHOST (General Hybrid Optimized Sparse Toolkit)

Frameworks from the ESSEX project

phist (Pipelined Hybrid-parallel Linear Solver Toolkit)

- General C-interface to linear algebra libraries:
  - Trilinos (C++, `http://trilinos.sandia.gov`)
  - GHOST
  - self-written (row-major storage, Fortran + C99, NT-stores)
- Iterative solvers
- Large test framework

GHOST (General Hybrid Optimized Sparse Toolkit)

- Developed at the RRZE in Erlangen (ESSEX project partner)
- Hybrid-parallel MPI+OpenMP+CUDA

Block pipelined GMRES algorithm

Idea

- Solve n_b linear systems $Ax_i = b_i$ at once

Block pipelined GMRES algorithm

Idea

- Solve n_b linear systems $Ax_i = b_i$ at once
- Remove converged systems and add new ones (pipelining)

Block pipelined GMRES algorithm

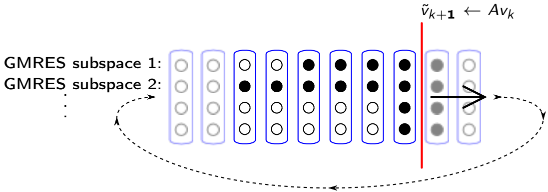
Idea

- Solve n_b linear systems $Ax_i = b_i$ at once
- Remove converged systems and add new ones (pipelining)
- Standard GMRES method (for each system)

Block pipelined GMRES algorithm

Idea

- Solve n_b linear systems $Ax_i = b_i$ at once
- Remove converged systems and add new ones (pipelining)
- Standard GMRES method (for each system)



Outline

Block-Jacobi-Davidson algorithm

Performance analysis

Implementation

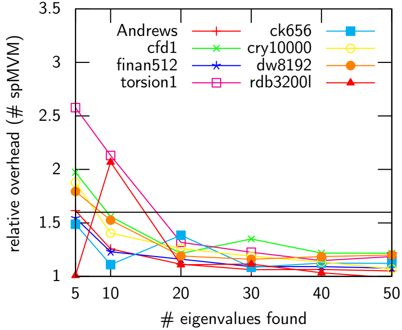
Results

- Numerical behavior
- Performance of the complete algorithm

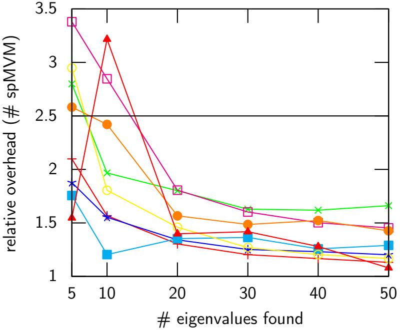
Conclusion

Numerical behavior

Block size 2



Block size 4

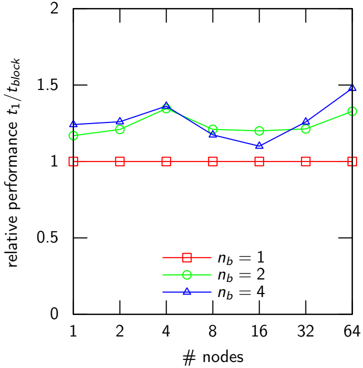


Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

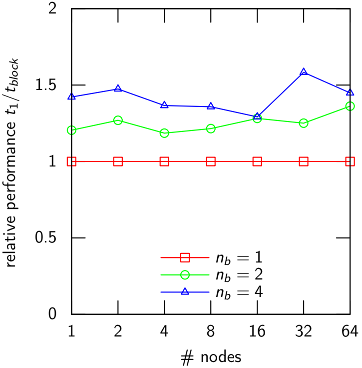
RWTH AACHEN UNIVERSITY

Performance of the complete algorithm

10 eigenvalues



20 eigenvalues



Outline

Conclusion

Block Jacobi-Davidson algorithm

- Working algorithm for outer eigenvalues
  - All basic operations reach almost the modeled performance limit

Conclusion

Block Jacobi-Davidson algorithm

- Working algorithm for outer eigenvalues
  - All basic operations reach almost the modeled performance limit
- Numerical experiments:
  - Overhead small for > 10 desired eigenvalues

Conclusion

Block Jacobi-Davidson algorithm

- Working algorithm for outer eigenvalues
  - All basic operations reach almost the modeled performance limit
- Numerical experiments:
  - Overhead small for > 10 desired eigenvalues
- Performance analysis:
  - Great impact of memory layout (row- instead of col.-major)
  - Improves code balance (node-level)
  - Message aggregation (inter-node)

Conclusion

Block Jacobi-Davidson algorithm

- Working algorithm for outer eigenvalues
  - All basic operations reach almost the modeled performance limit
- Numerical experiments:
  - Overhead small for > 10 desired eigenvalues
- Performance analysis:
  - Great impact of memory layout (row- instead of col.-major)
  - Improves code balance (node-level)
  - Message aggregation (inter-node)

$\Rightarrow$ Improved performance by factor 1.2–1.6

Conclusion (2)

Outlook

- Integrate row-major approach in ESSEX (GHOST library from Erlangen) (partly done)

Conclusion (2)

Outlook

- Integrate row-major approach in ESSEX (GHOST library from Erlangen) (partly done)
- Need more asynchronous algorithms for exascale

Conclusion (2)

Outlook

- Integrate row-major approach in ESSEX (GHOST library from Erlangen) (partly done)
- Need more asynchronous algorithms for exascale
- Need fast orthogonalization kernel (TSQR)

Conclusion (2)

Outlook

- Integrate row-major approach in ESSEX (GHOST library from Erlangen) (partly done)
- Need more asynchronous algorithms for exascale
- Need fast orthogonalization kernel (TSQR)
- Parallel preconditioning of the linear problems (Jonas Thies work)

Conclusion (2)

Outlook

- Integrate row-major approach in ESSEX (GHOST library from Erlangen) (partly done)
- Need more asynchronous algorithms for exascale
- Need fast orthogonalization kernel (TSQR)
- Parallel preconditioning of the linear problems (Jonas Thies work)
- Block Pipelined GMRES also interesting for other applications (FEAST in ESSEX)