I.1.(d) Algebraic Substructuring Method
Background

- Substructuring dates back to the 1960s, e.g., CMS
- Connection with domain decomposition methods
- Substructuring holds great promise for solving extremely large scale problems, e.g., commercial AMLS
- Open questions as the techniques extended for broader applications:
  - arbitrary eigenmodes
  - accuracy
  - high frequency response analysis
  - performance optimization (memory, out-of-core, ...)


Eigenvalue problem and multi-level substructure

- Eigenvalue problem: \( Kq = \lambda Mq, \quad K^T = K, \quad M^T = M > 0 \)
- Substructure partition
  Single-level

\[
K = \begin{bmatrix}
K_{11} & K_{13} \\
K_{22} & K_{23} \\
K_{31} & K_{32} & K_{33}
\end{bmatrix}, \quad M = \begin{bmatrix}
M_{11} & M_{13} \\
M_{22} & M_{23} \\
M_{31} & M_{32} & M_{33}
\end{bmatrix}
\]

Multi-level: nested dissection (ND)

- Can use METIS to record any sparse matrix to the ND form.
Algebra substructuring algorithm

1. Perform a congruence transformation such that
\[
L^{-T}(K, M)L^{-1} = \begin{bmatrix}
K_{11} & K_{22} \\
K_{22} & \hat{K}_{33}
\end{bmatrix}, \begin{bmatrix}
M_{11} & \hat{M}_{13} \\
\hat{M}_{31} & M_{22} & \hat{M}_{23} \\
\hat{M}_{32} & \hat{M}_{33}
\end{bmatrix} \equiv (\hat{K}, \hat{M})
\]
this is so-called the Craig-Bampton form in structure dynamics.

2. Compute \textbf{partial} local modes by local cutoff values:
\[
K_{11}S_1 = M_{11}S_1\Theta_1 \\
K_{22}S_2 = M_{22}S_2\Theta_2 \quad \rightarrow \quad S_m = \begin{bmatrix}
S_1 \\
S_2 \\
I
\end{bmatrix}
\]

3. Project onto the \textbf{AS subspace} \{\(L^{-1}S_m\)\}:
\[
(L^{-1}S_m)^T(K, M)(L^{-1}S_m) = (K_m, M_m)
\]

4. Solve the reduced eigenvalue problem:
\[
K_m\Phi = M_m\Phi\Theta,
\]
5. Compute the global modes $\Phi = [\Phi_l \quad \Phi_n \quad \Phi_r ]$

$\Phi_n$ are retained modes determined by cutoff values

$\Phi_t = [\Phi_l \quad \Phi_r ]$: truncated modes

6. Return approximate eigenpairs

$(\theta^\sigma + \sigma, L^{-1} S_m \phi) \approx (\lambda, q)$ of $(K, M)$
Why (when) AS works?

• Consider the **full** eigendecomposition of \((K_{11}, M_{11})\) and 
\((K_{22}, M_{22})\)

\[
K_{11}V_1 = M_{11}V_1 \Lambda_1, \quad K_{22}V_2 = M_{22}V_2 \Lambda_2.
\]

• An eigenvector \(\hat{x}\) of \((\hat{K}, \hat{M})\) can be expressed as the follows

\[
\hat{x} = Vy = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ I \end{bmatrix}
\]

• If \(\hat{x}\) can be well approximated by a linear combination of

\[
S = \begin{bmatrix} S_1 \\ S_2 \\ I \end{bmatrix},
\]

then the vectors \(y_1\) and \(y_2\) must contain only a few large entries, and all components of \(y_1\) and \(y_2\) are likely to be small and negligible – **key observation/requirement**
• An example of the magnitude of $y_1$ and $y_2$:

(a finite element model corresponding to a five-cell traveling wave accelerating structure)

• For a formal analysis to derive an *a priori* error bound of the smallest eigenpair in terms of the small components of $y_1$ and $y_2$, see [Yang *et al*’05, Voss *et al*].
Implementation and performance evaluation

- Major operations:
  1. block elimination and projection
  2. projected eigenvalue problem

- Costs
  1. flops: more than a single sparse Cholesky factorization
  2. storage: block Cholesky factors + projected matrices + ...

But no triangular solvers, no (re)-orthogonalization

- ASEIG package ([Gao et al, ACM/TOMS 2008])
  1. interleave different steps
  2. recompute some of intermediate matrix blocks instead of storing. *50% of memory saving with about extra 15% recompute time*
Case study: accelerator cavity design (SLAC)

- Electromagnetic modeling of a 6-cell DDS structure for the design of next generation accelerator (SLAC)

- Computing small eigenvalues (tightly clustered) out of a large-eigenvalue dominated eigen spectrum

- $N = 65730$

- 4-level AS, $n_{proj} \approx 3000$

- Many eigenvalues are wanted (up to 8%)

- SIL requires multiple shifts (factorizations)
An industrial case study: NVH analysis

Modal analysis for noise, vibration and harshness (NVH)

- DOFs = 1,584,622
- 1073 eigenmodes
- SI Lanczos method
- NX NASTRAN

<table>
<thead>
<tr>
<th></th>
<th>CPU(min)</th>
<th>I/O(GB)</th>
<th>Wallclock(min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequential</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>283</td>
<td>2476</td>
<td>721</td>
</tr>
<tr>
<td>AS</td>
<td>103</td>
<td>1402</td>
<td>167</td>
</tr>
<tr>
<td>parallel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>45</td>
<td>325</td>
<td>85</td>
</tr>
<tr>
<td>AS</td>
<td>9</td>
<td>98</td>
<td>47</td>
</tr>
</tbody>
</table>

16-node AMD Opteron (1.8GHz), 4GB RAM, 73GB per node
Further reading

The study of substructuring methods from an linear algebra point of view can be found in


An implementation of the AS method is described in the following paper