COMPARISON OF EIGENVALUE SOLVERS FOR LARGE SPARSE MATRIX PENCILS

Fatih Yaman, Wolfgang Ackermann and Thomas Weiland

August 24, 2012, Warnemünde

Contents

- **Introduction**
  - Problem, Method, Solvers, Tools
  - FEM formulation for Generalized Eigenvalue Problem
  - Jacobi-Davidson Method
- **Simulations for Solver Comparisons**
  - Spherical Resonator
  - 9-cell TESLA Cavity
  - Billiard Resonator
- **Extracting large amount of eigenvalues with Matlab**
- **Conclusions**

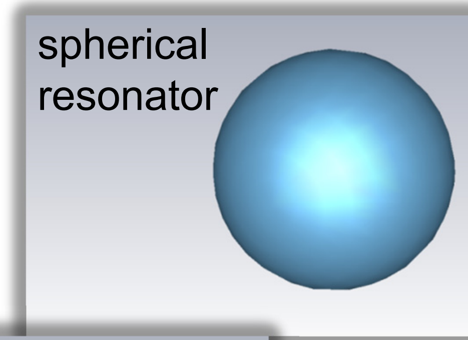
Introduction

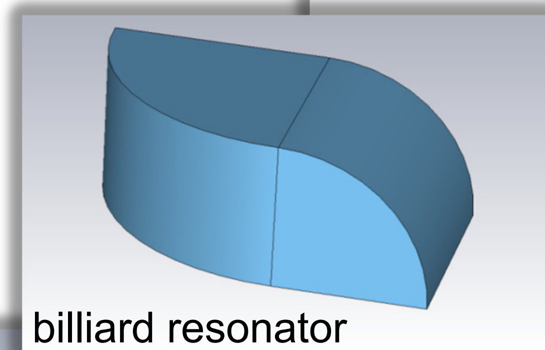
Aim : to investigate **performance of available eigensolvers**
- time and memory consumption
- applicability, efficiency and robustness

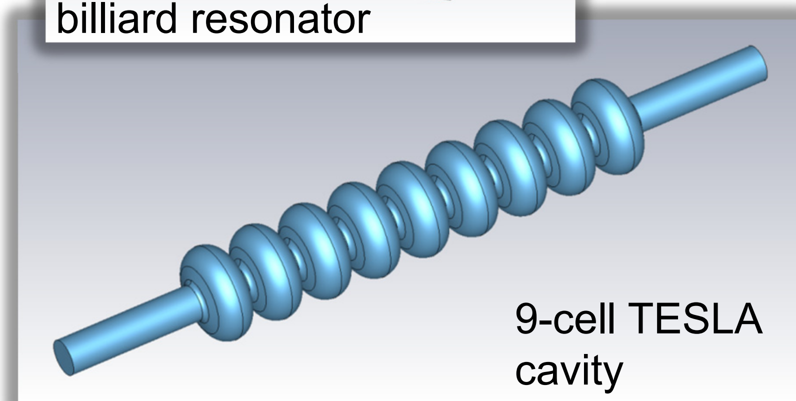
Method : three setups **for large sparse matrix pencils**

up to 10^6 DOF with 10^8 nonzero elements
- Spherical resonator
- Billiard resonator
- 9-cell TESLA cavity

Software Tools: recent versions of solvers
- CST 2012
- Matlab R2011
- SLEPc 3.2
- Pysparse 1.1.1
- CEM3D







Hardware Tools

- Standard personal computer:
  - 2 processors 8 cores total, 2.27 GHz *(Intel Xeon E5520)*
  - 24 GB RAM and 64 – bit operating system

- Cluster computer:
  - 172 nodes
  - 172 x 2 processors = 344
    (*Intel Xeon X5650 processors)*
  - 172 x 12 cores = 2,064 cores
  - 172 x 24 GB = 4,03 TB
  - Infiniband (QDR) Gigabit

Wakefield Cluster, TEMF, TU Darmstadt

FEM formulation for Generalized Eigenvalue Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem formulation
  - Local Ritz approach

$$\vec{E} = \vec{E}(\vec{r}) = \sum_{i=1}^{n} c_i \, \vec{w}_i(\vec{r})$$

$\vec{w}$ vectorial function

c_i scalar coefficient

i global index

n number of DOFs

$$\mathrm{curl}\ 1/\mu_\mathrm{r}\ \mathrm{curl}\vec{E} = \left(\frac{\omega}{c_0}\right)^2 \varepsilon_\mathrm{r}\, \vec{E}\ \Big|_{\vec{r}\in\Omega}$$

$$\vec{n} \times \vec{E}\ \Big|_{\vec{r}\in\partial\Omega} = 0 \qquad \mathrm{div}(\varepsilon\vec{E}) = 0$$

continuous eigenvalue problem

Galerkin testing of the fundamental equation

$$a_{ij} = \iiint_\Omega 1/\mu_\mathrm{r}\, \mathrm{curl}\vec{w}_i \cdot \mathrm{curl}\vec{w}_j \ \mathrm{d}\Omega$$

$$b_{ij} = \iiint_\Omega \varepsilon_\mathrm{r}\, \vec{w}_i \cdot \vec{w}_j \ \mathrm{d}\Omega$$

$$A \cdot \vec{c} = \lambda\, B \cdot \vec{c},\ \lambda = \left(\frac{\omega}{c_0}\right)^2,\ \vec{c} = \{c_i\}$$

discrete eigenvalue problem

Methods in Literature

TECHNISCHE UNIVERSITÄT DARMSTADT

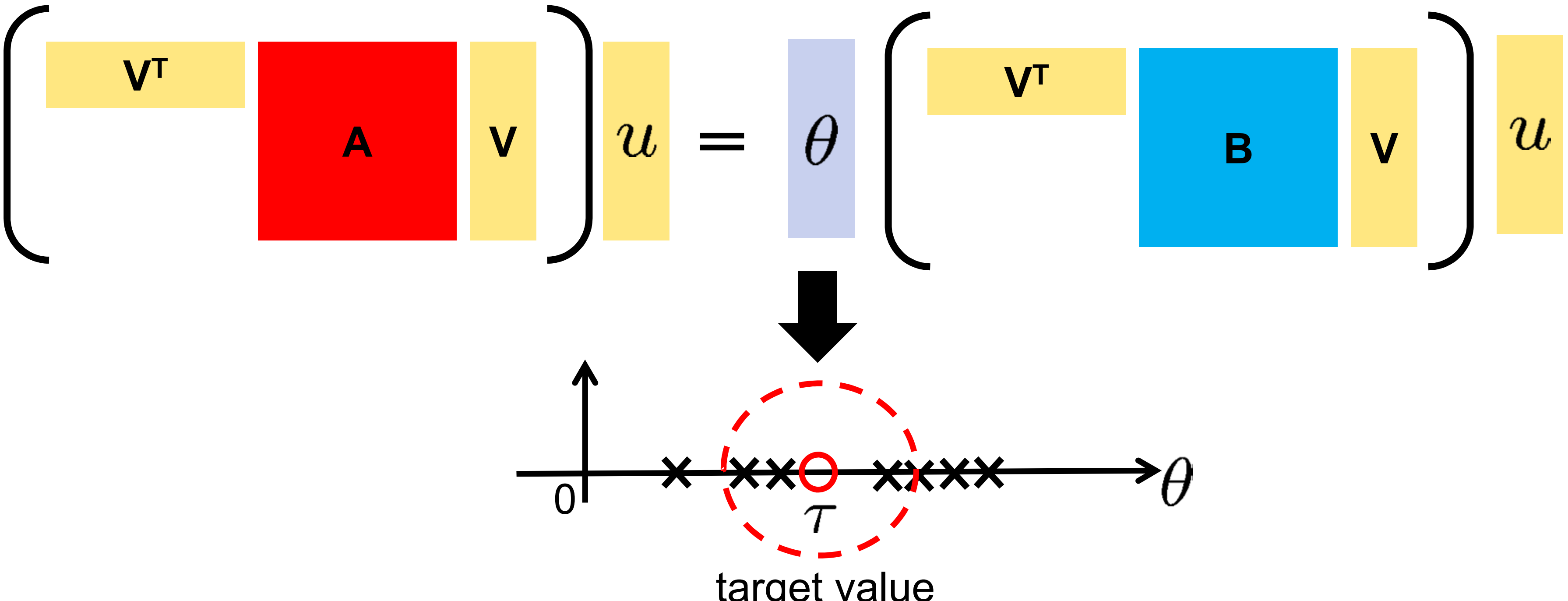
- Arnoldi
- Lanczos
- Krylov-Schur
- Generalized Davidson
- Jacobi-Davidson
    - very efficient compared with the rest of the methods when computing interior eigenvalues *
    - Jacobi-Davidson algorithm performed best for the largest cases of our matrix eigenvalue problems having orders above 30.000 **

*E. Romero and J.E. Roman, '*A parallel implementation of Davidson methods for large-scale eigenvalue problems in SLEPc*' , ACM Trans. on Math. Software, preprint, 2012

**P. Arbenz and R. Geus, '*A comparison of solvers for large scale eigenvalue problems occuring in the design of resonant cavities*' , Numer. Linear Algebra Appl. 6, 3-16 ,1999.

Jacobi-Davidson Algorithm

- $A\,x = \lambda\,B\,x$ for $B > 0$
- $V_k = \mathrm{span}\{v_1, \cdots, v_k\}$ be a subspace where $v_k^T B v_j = \delta_{kj}$
- Obtain solution $(\theta,\ u)$ from projected problem



Jacobi-Davidson Algorithm

- Calculate Ritz vector $x = u^T V$
- Check convergence $\| r \|_2 := \| (A - \theta B)x \|_2 < \epsilon$
- Solve the so-called correction equation

$$(I - BVV^T)(A - \theta B)(I - VV^T B)z = -r$$

for the unknown z iteratively with

- Transpose-Free Quasi-Minimal Residual *(tfqmrs)* in CEM3D
- Biconjugate gradient method stabilized (bcgsl) in SLEPc
- Quasi-Minimal Residual (qmrs) in Pysparse

Jacobi-Davidson Algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- orthonormalize z against V_k using Gram-Schmidt to obtain v_{k+1}

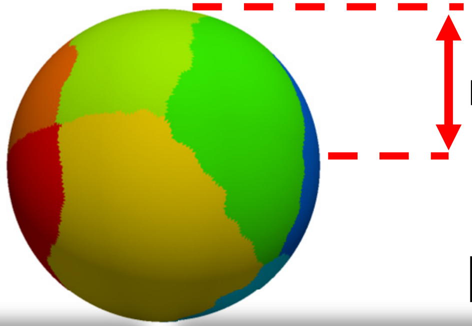
- expand the search subspace $V_{k+1} = \text{span}\{v_1, \cdots, v_{k+1}\}$

- goto obtain a solution the projected problem for updated subspace V_{k+1}

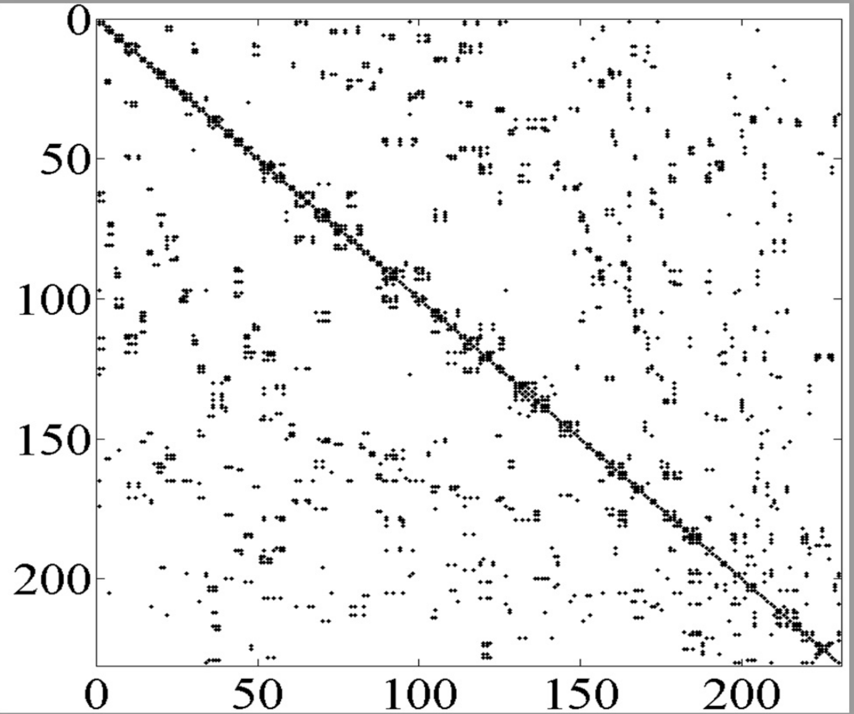
Spherical Resonator Simulations

TECHNISCHE UNIVERSITÄT DARMSTADT

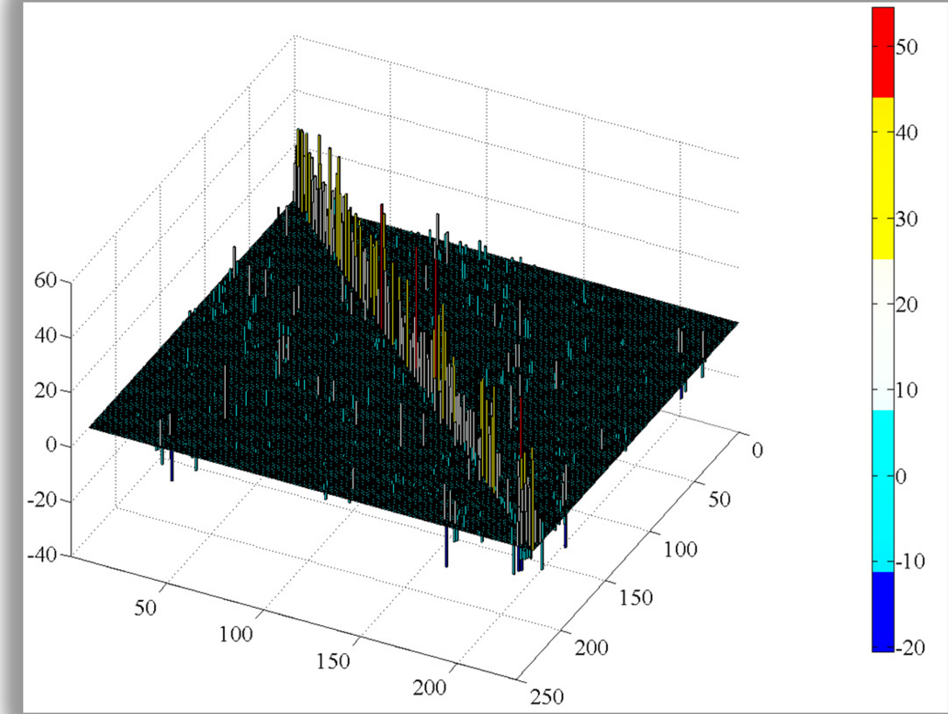
distribution on 10 nodes



Matrix A



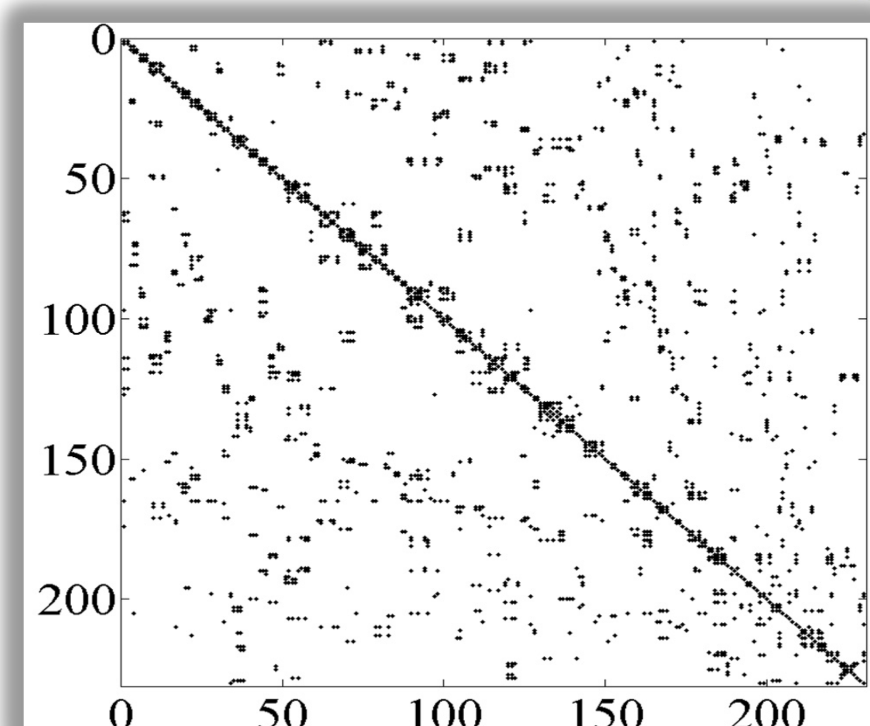
Matrix A (city plot)



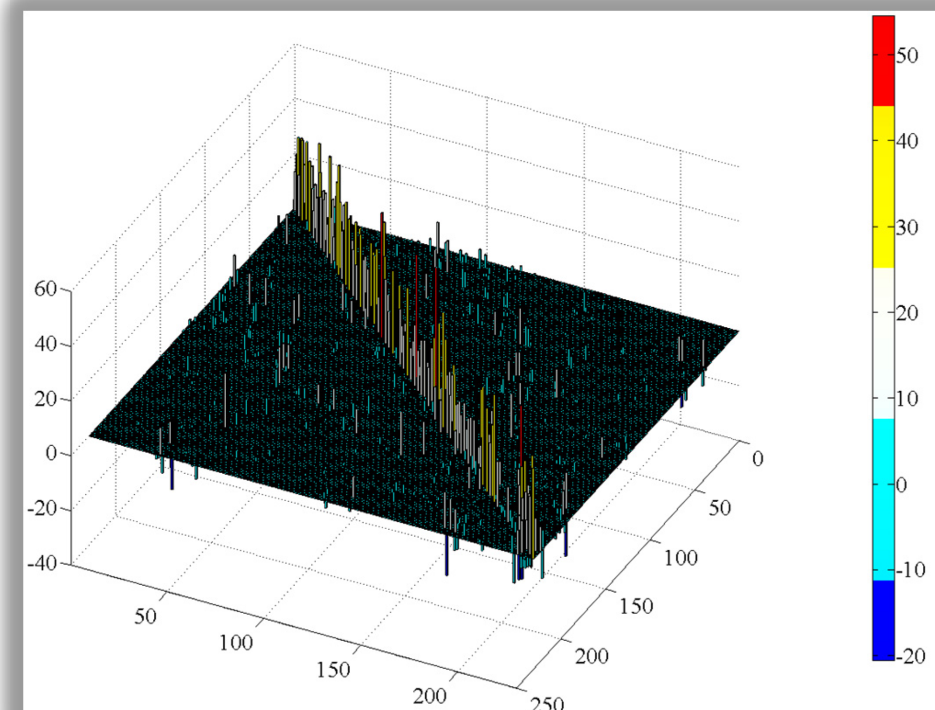
Spherica

- Number of mesh cells: **387**
- Number of DOF: **230**
- Number of Nonzero elements: **2042**
- Sparsity % : **3.86**

Matrix A



Matrix A (city plot)



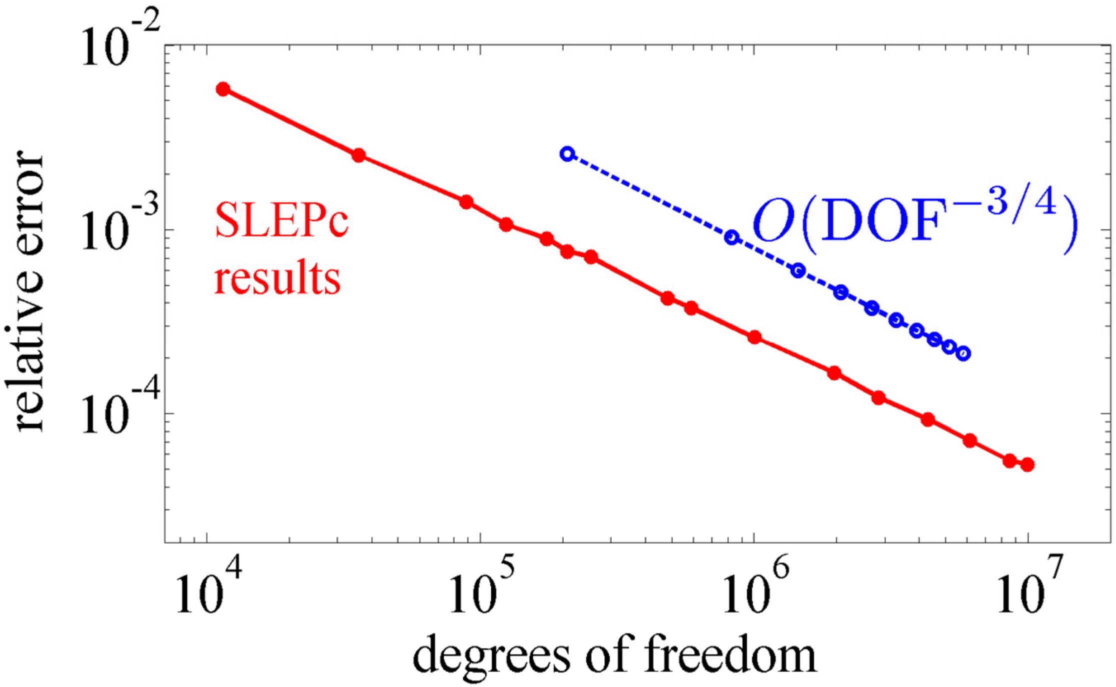
Convergence for SLEPc eigencomputations

TECHNISCHE UNIVERSITÄT DARMSTADT

- analytical expression : $\frac{d}{dx}\{\sqrt{x}\, J_{m+\frac{1}{2}}(x)\} = 0$

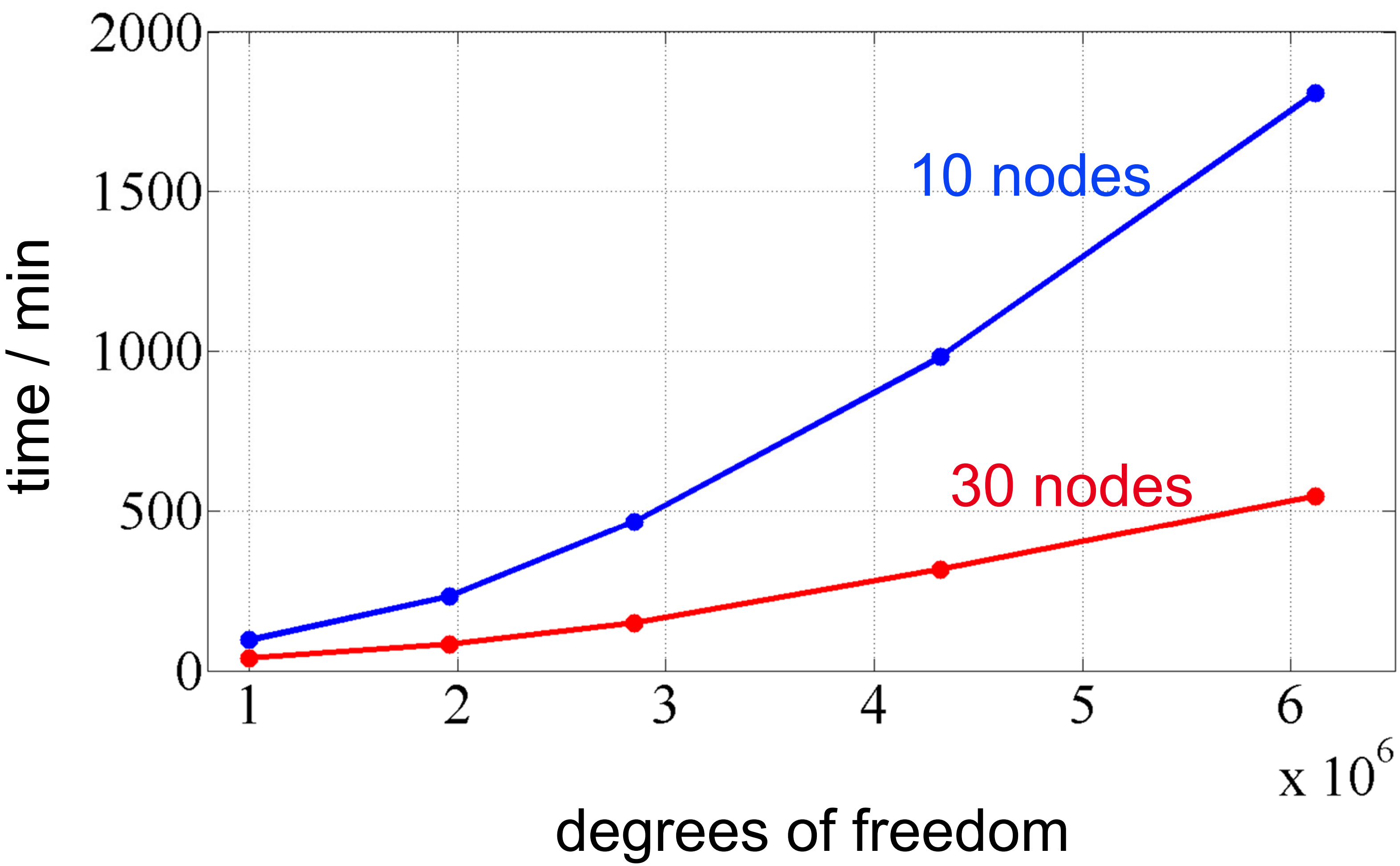
S. Gallagher and W.J. Gallagher, ``The Spherical Resonator,'' IEEE Trans.on Nuclear Sci. (1985).

- $\text{relative error} = \max_{i \in \text{DOF}} \frac{|\lambda^{\text{analytical}} - \lambda_i^{\text{numerical}}|}{\lambda^{\text{analytical}}}$

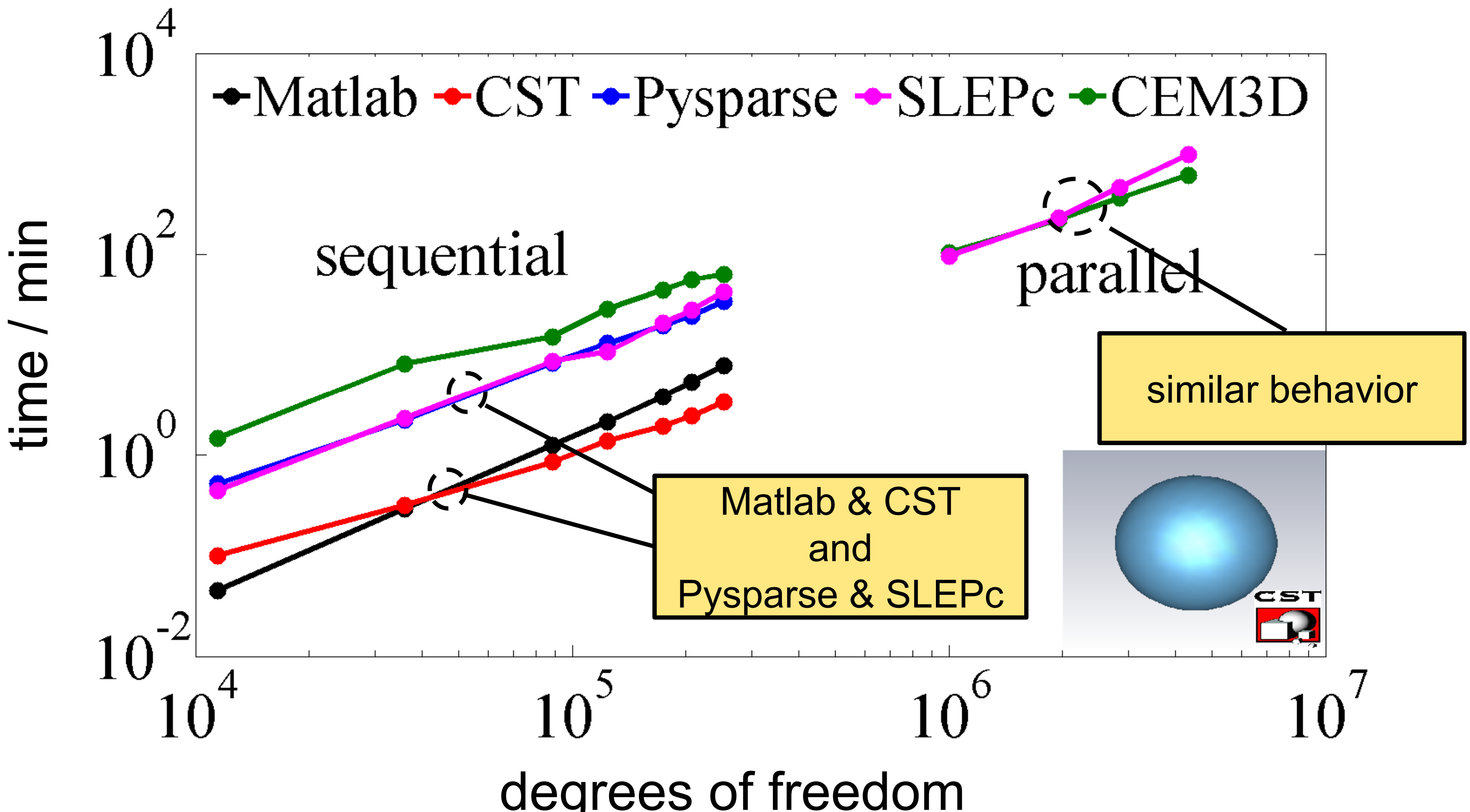


SLEPc time consumption for different number nodes for Spherical Resonator

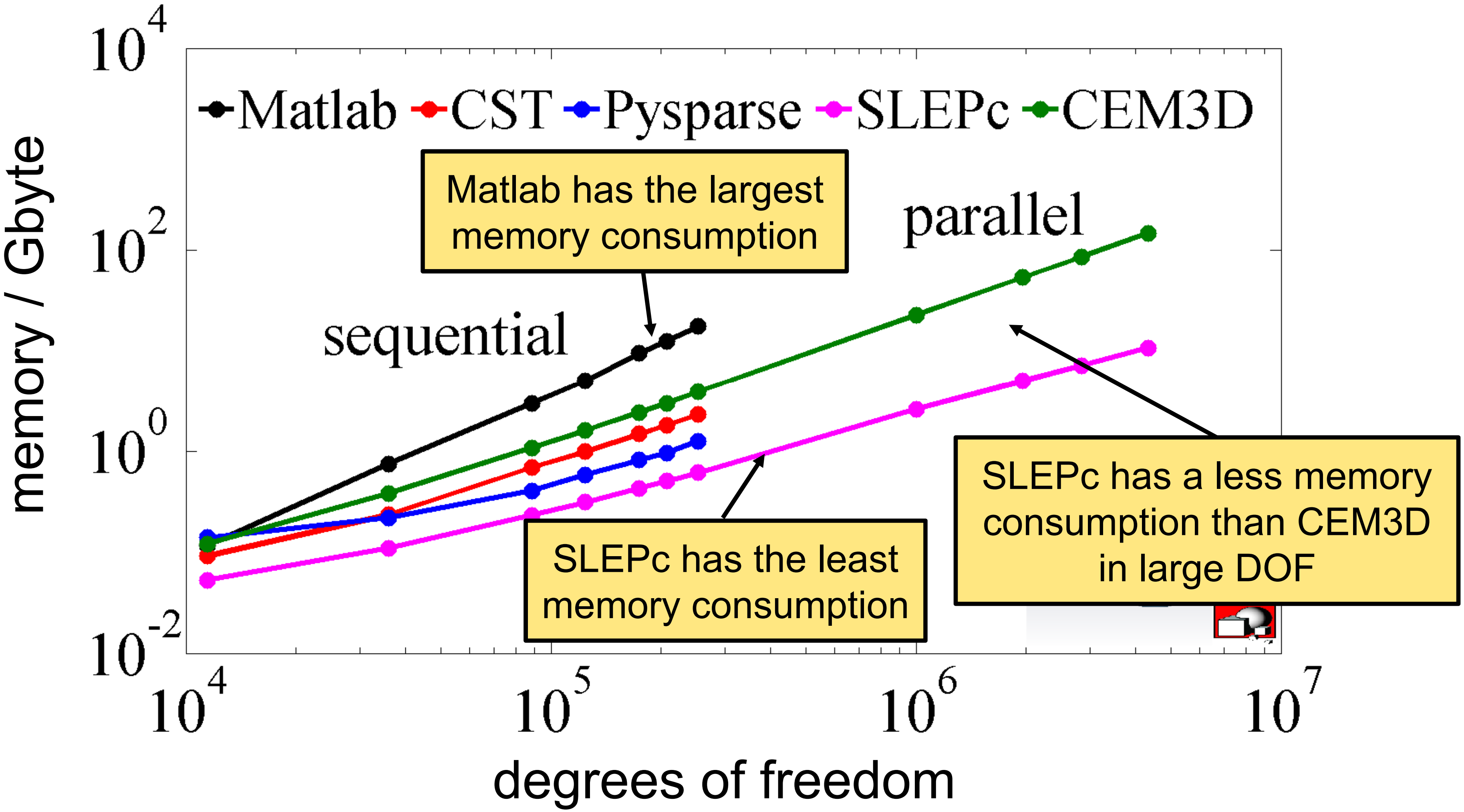
TECHNISCHE UNIVERSITÄT DARMSTADT



Spherical Resonator Time Consumption

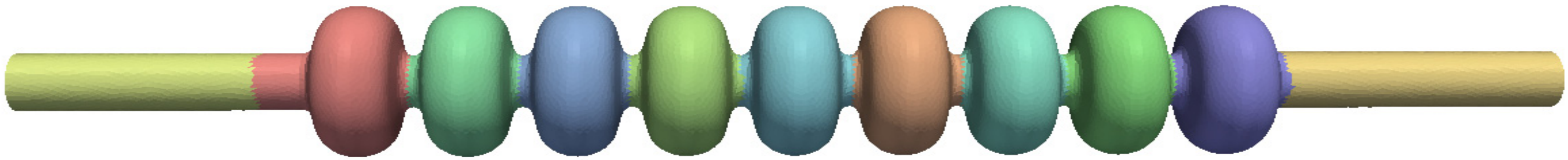


Spherical Resonator Memory Consumption

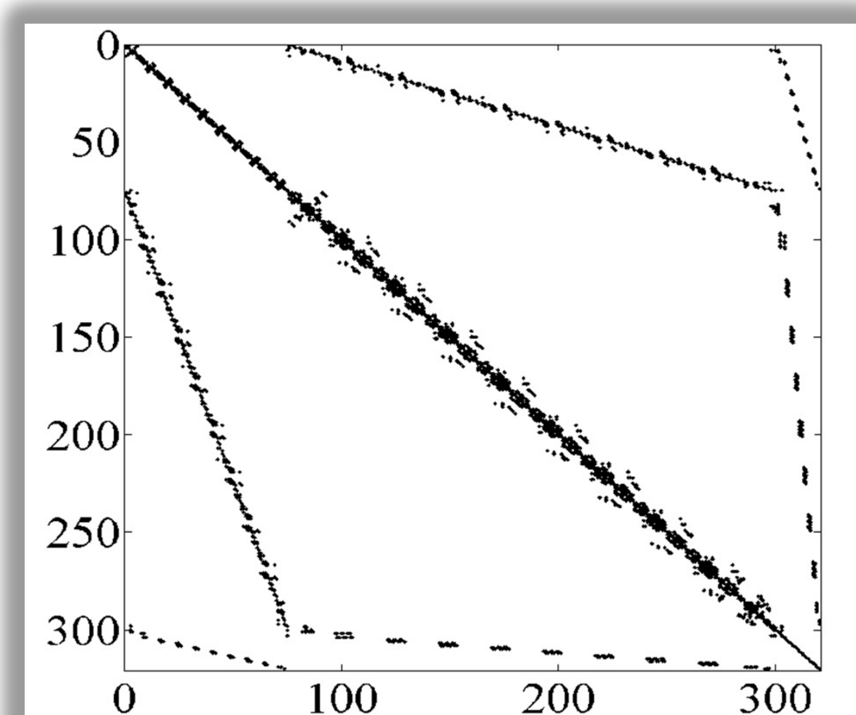


TESLA Cavity Simulations

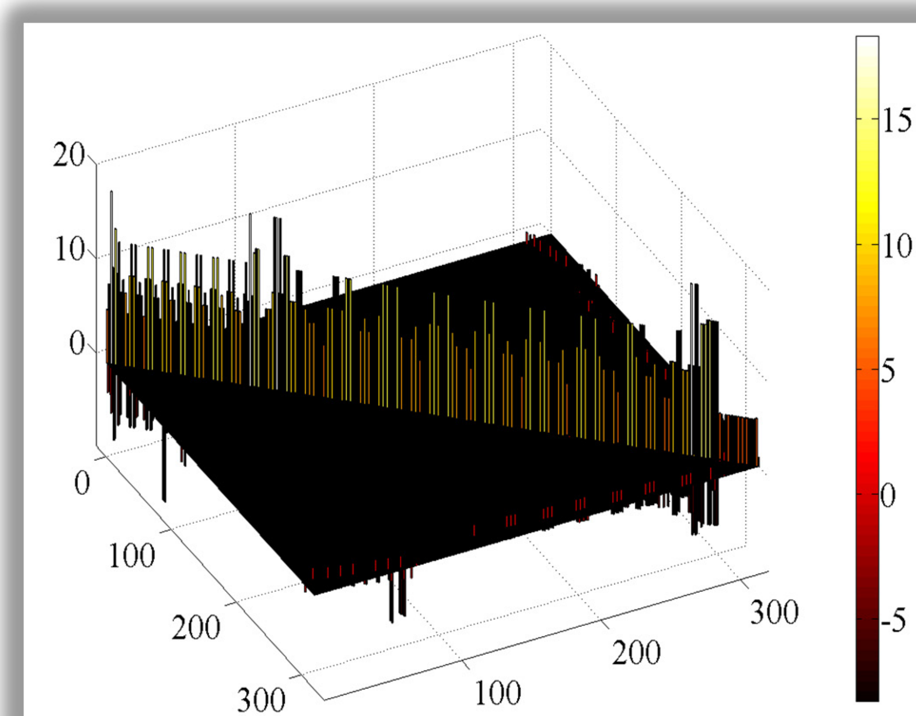
distribution on 10 nodes



Matrix A



Matrix A (city plot)

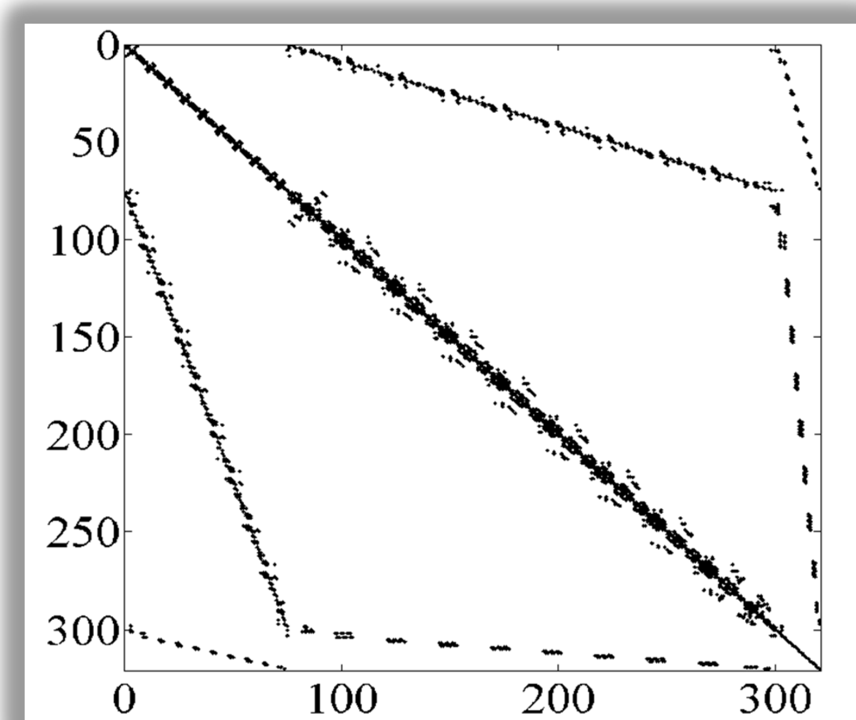


TESLA Ca

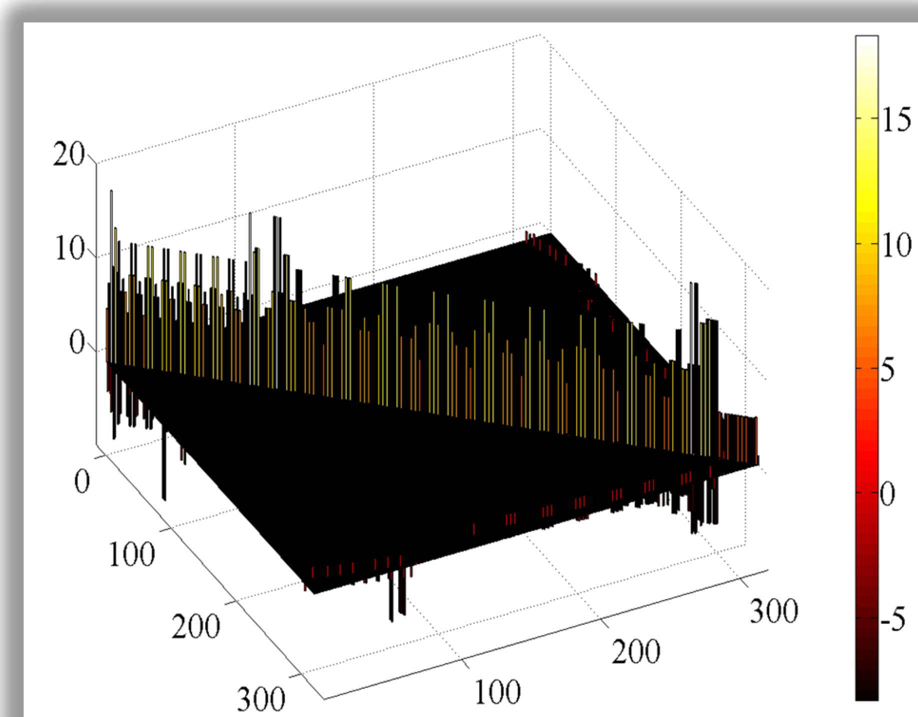
distribution c

- Number of mesh cells: **528**
- Number of DOF: **320**
- Number of Nonzero elements: **2398**
- Sparsity % : **2.34**

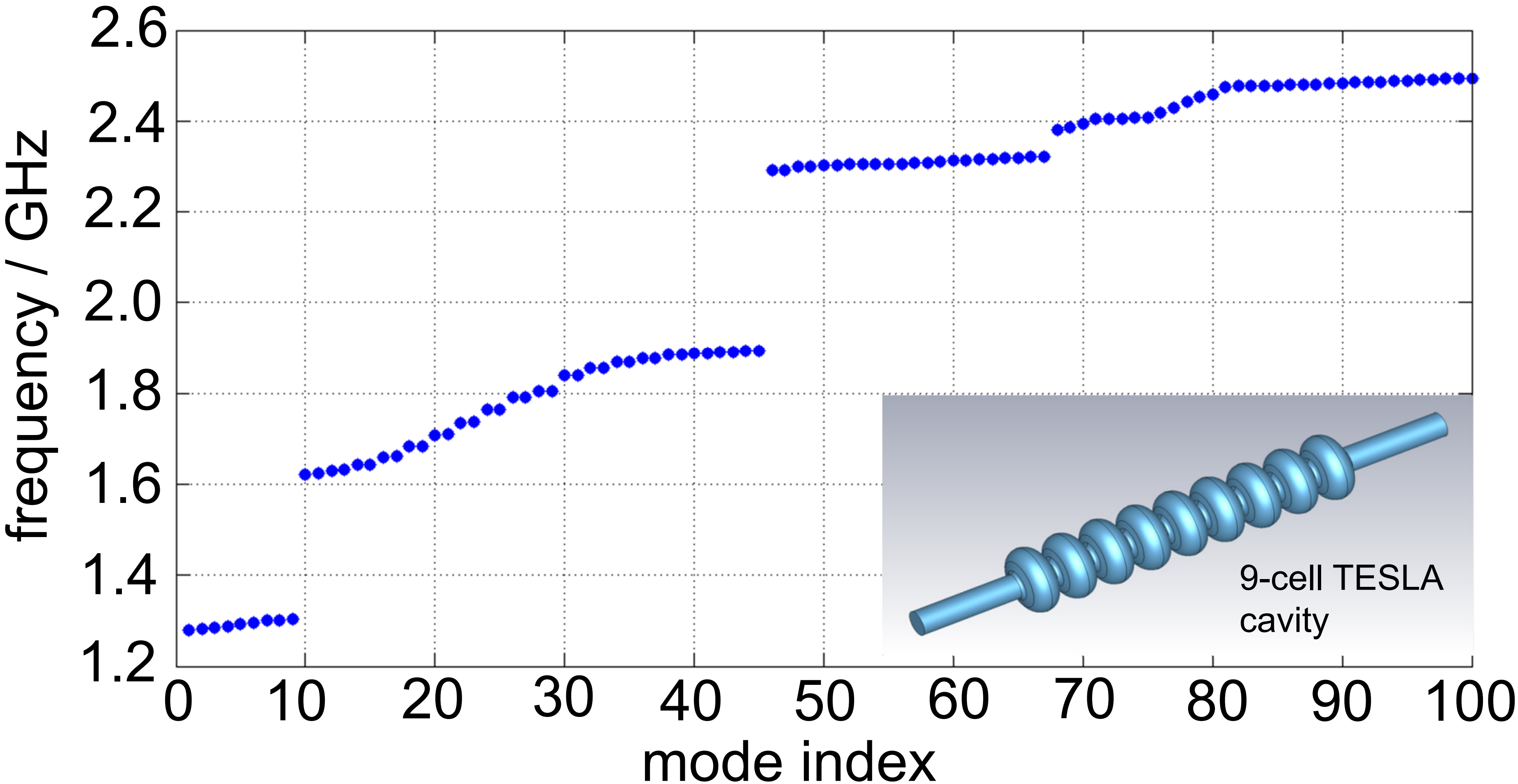
Matrix A



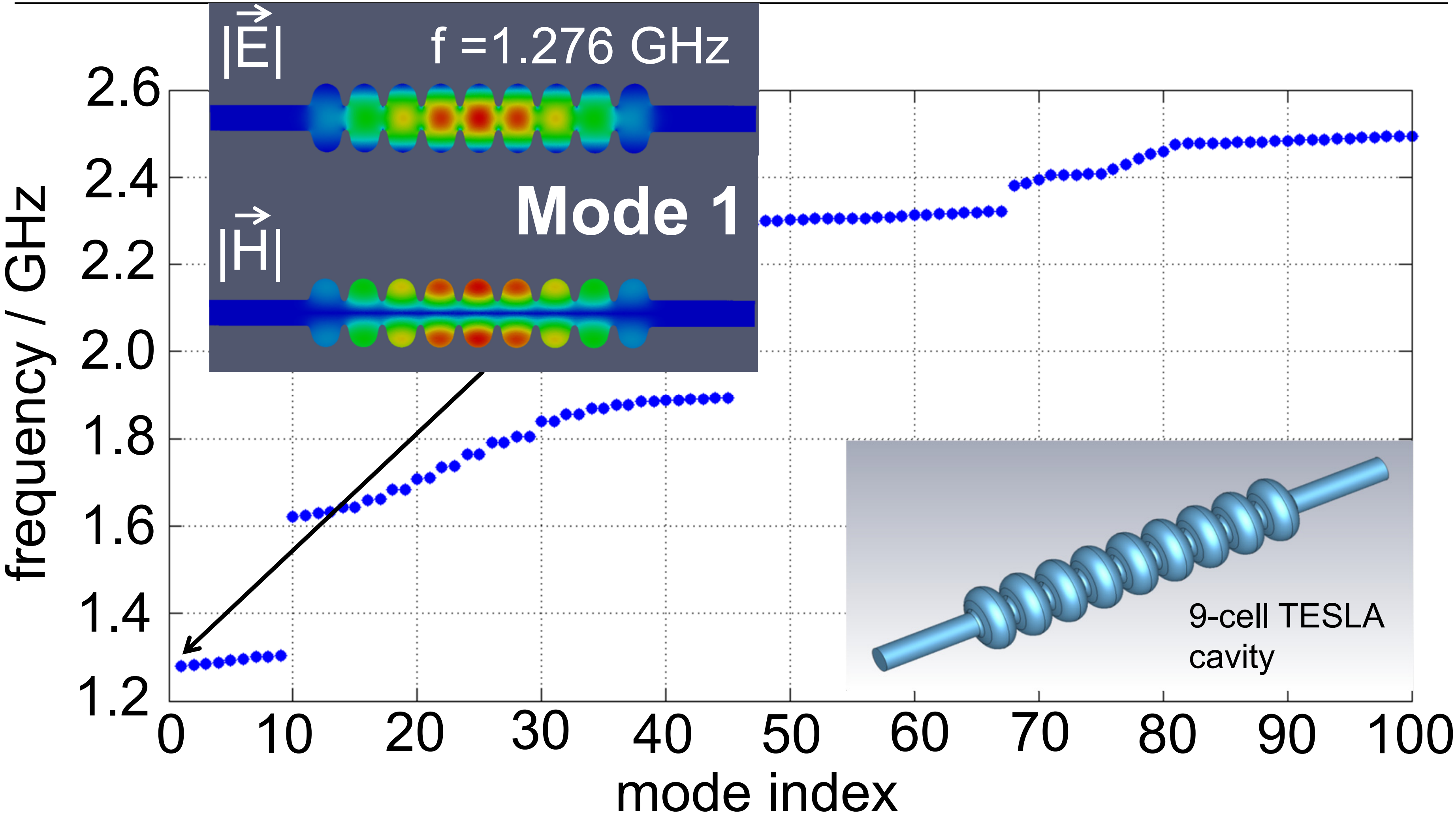
Matrix A (city plot)



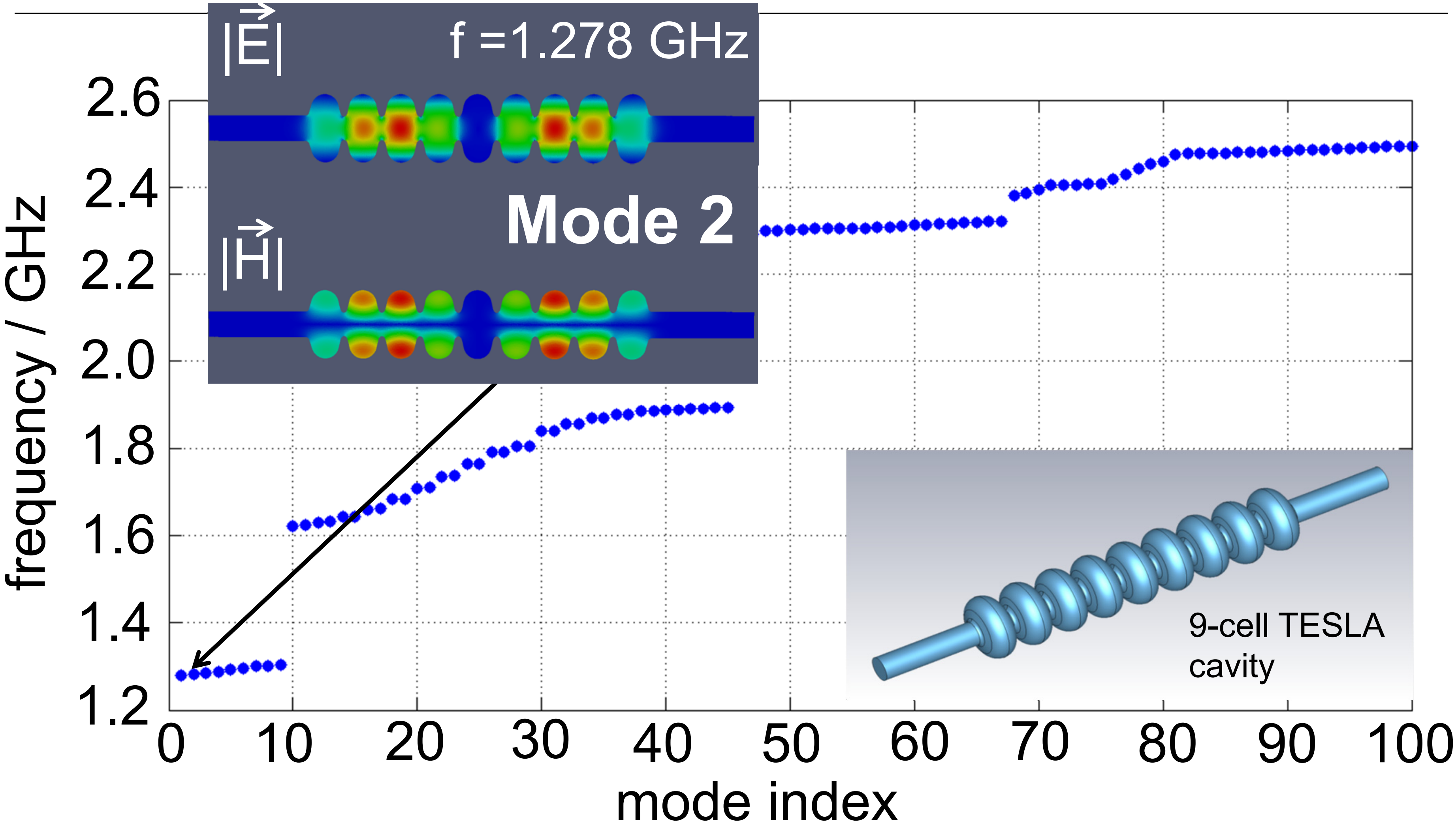
TESLA Cavity Eigenfrequency Spectrum



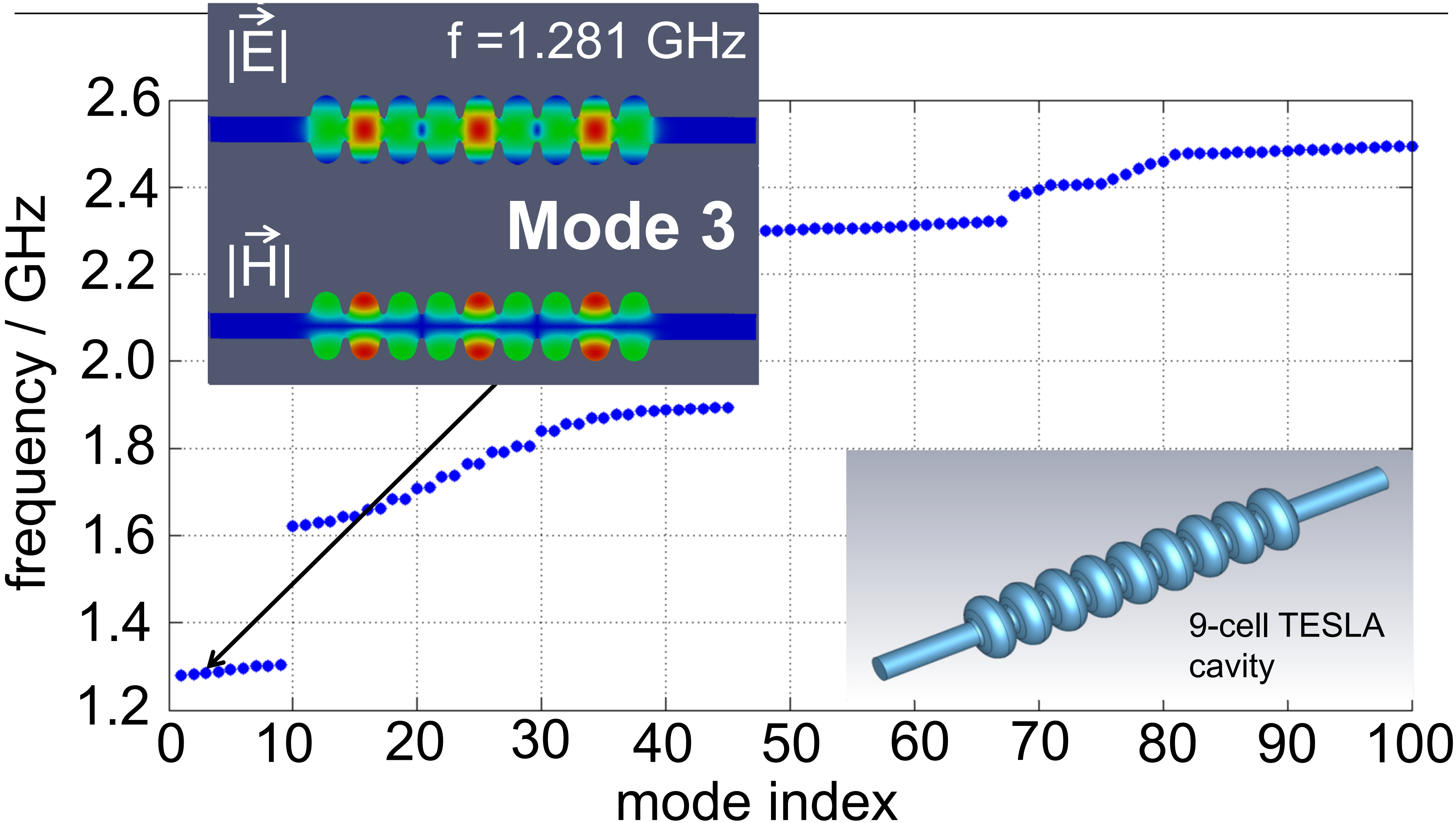
TESLA Cavity Eigenfrequency Spectrum



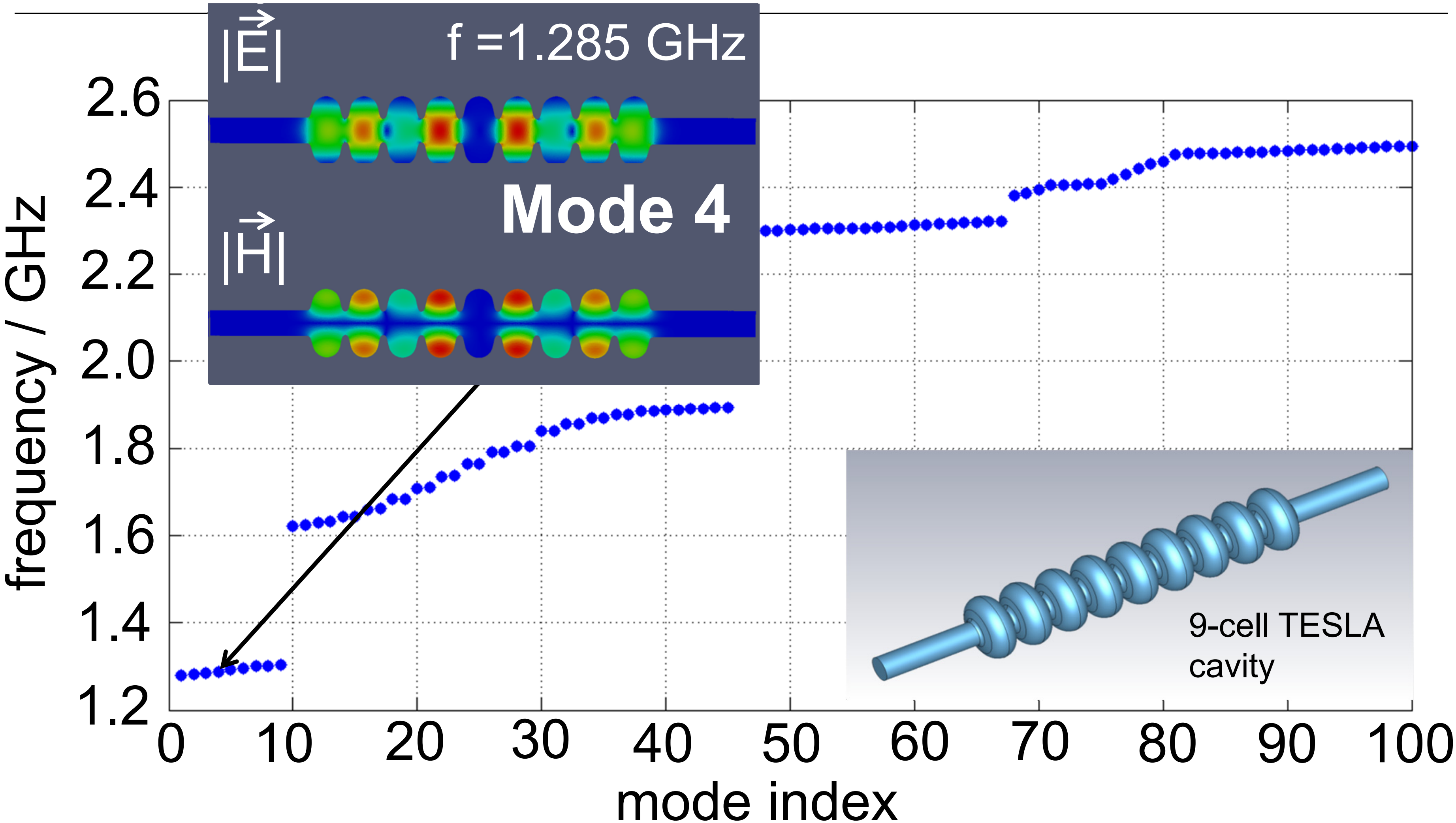
TESLA Cavity Eigenfrequency Spectrum



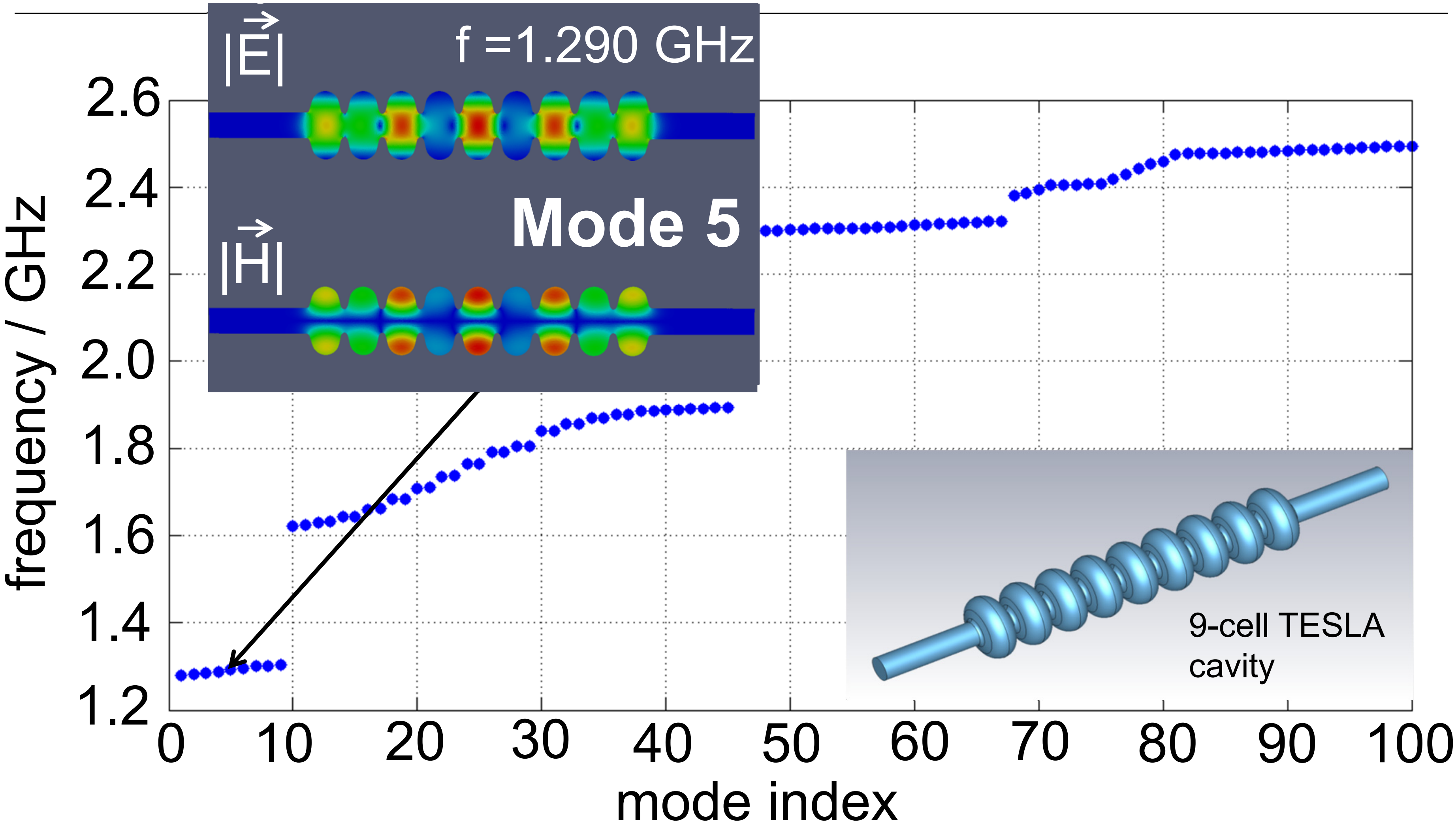
TESLA Cavity Eigenfrequency Spectrum



TESLA Cavity Eigenfrequency Spectrum

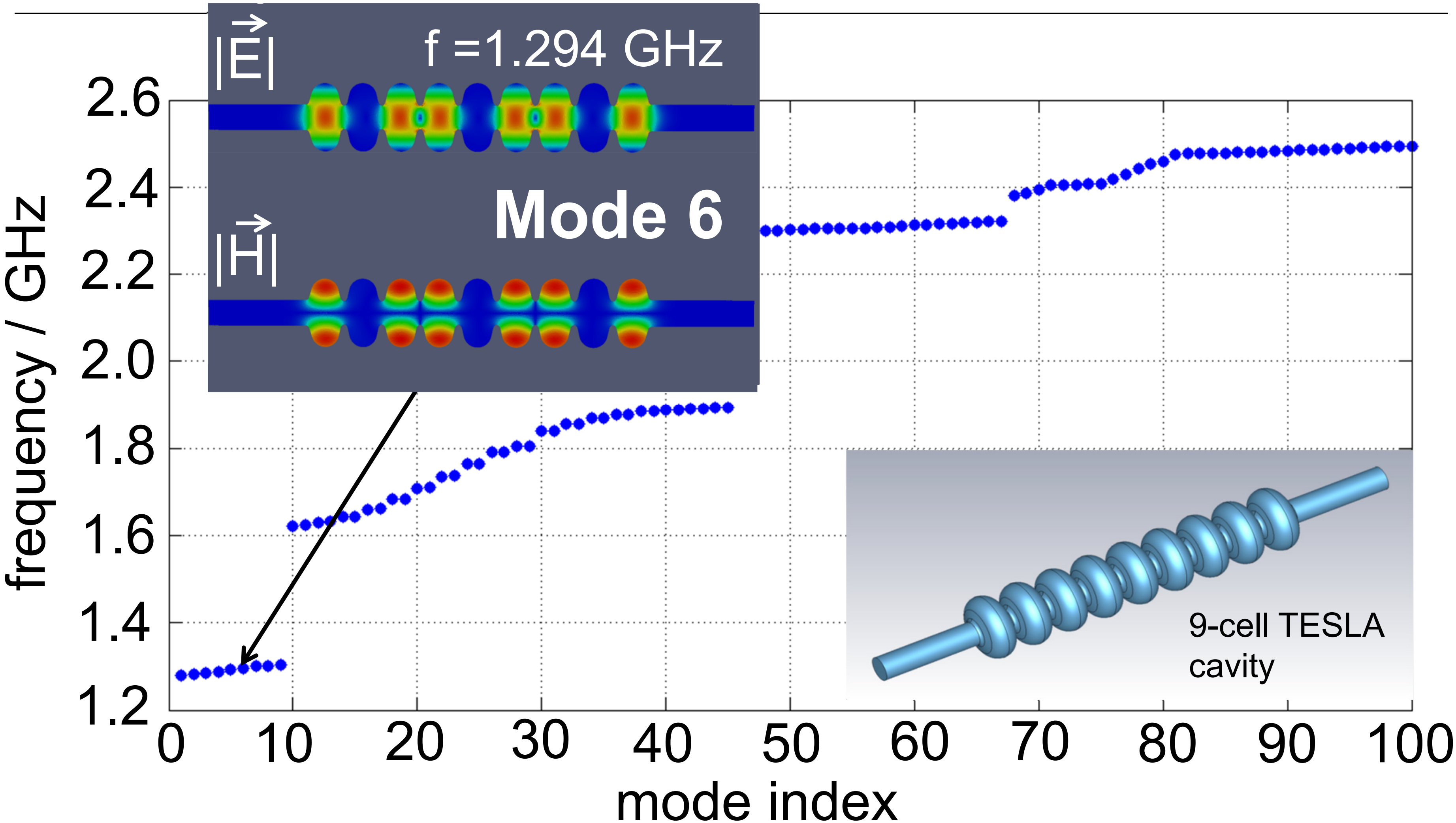


TESLA Cavity Eigenfrequency Spectrum

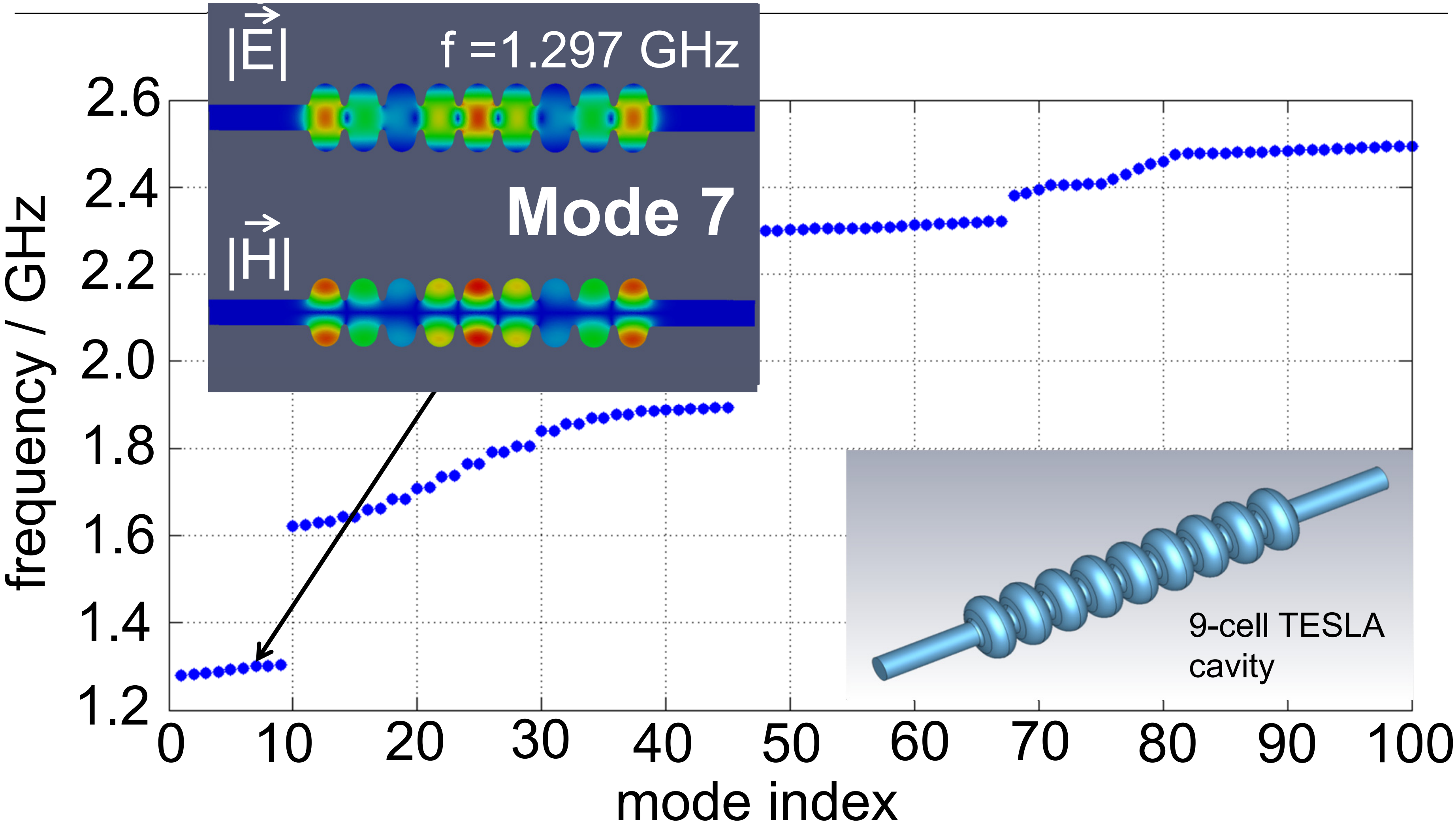


TESLA Cavity Eigenfrequency Spectrum

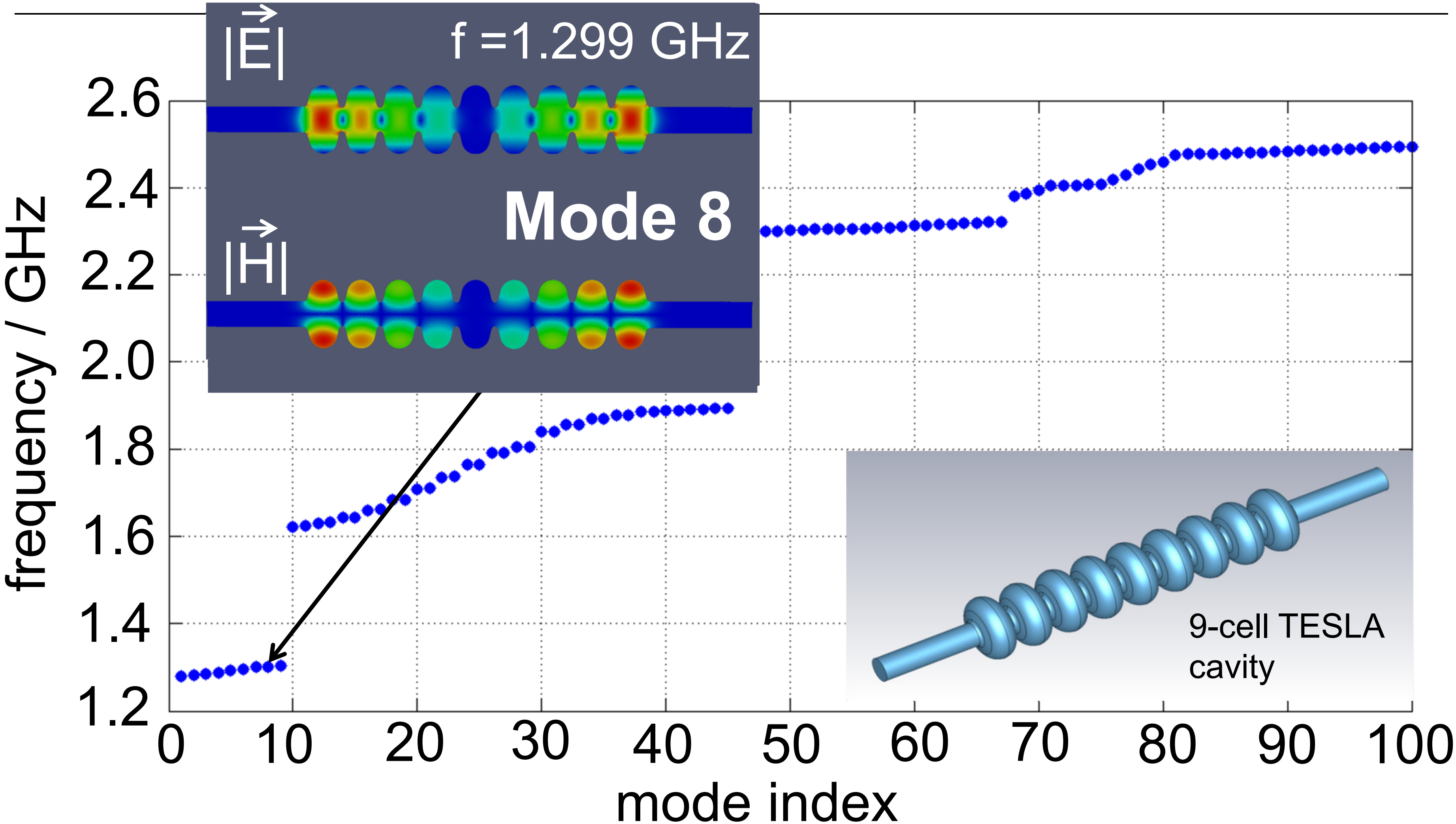
TECHNISCHE UNIVERSITÄT DARMSTADT



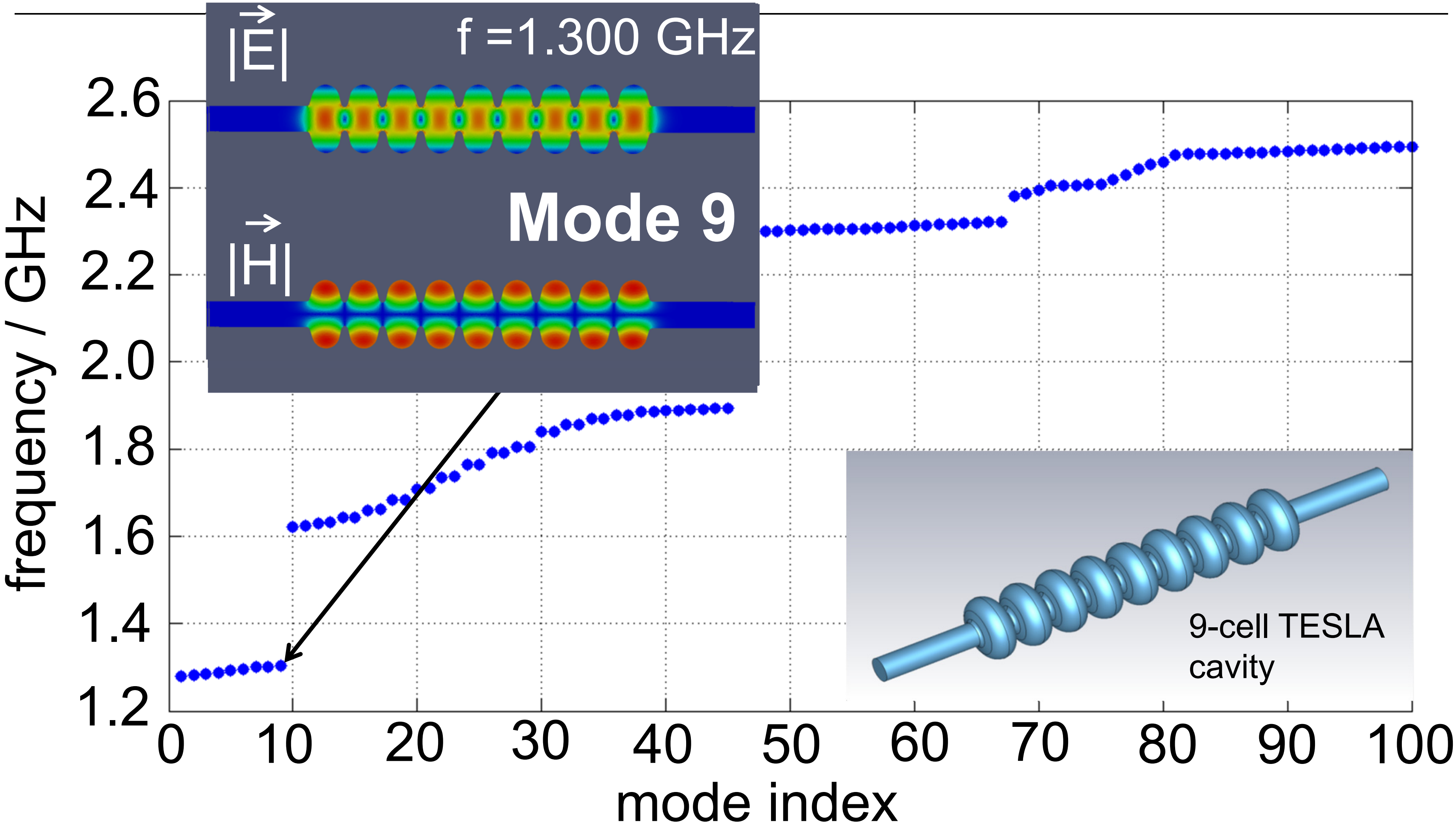
TESLA Cavity Eigenfrequency Spectrum



TESLA Cavity Eigenfrequency Spectrum



TESLA Cavity Eigenfrequency Spectrum



Eigenfrequency Convergence

TECHNISCHE UNIVERSITÄT DARMSTADT

DOF	MATLAB	CST	Pysparse	SLEPc	CEM3D
11,423	1.32883647670	1.32883647673	1.3288364767	1.32883647670	1.32883647670447
51,655	1.30814468722	1.30814468725	1.30814468722	1.30814468722	1.30814468722384
80,965	1.30657183563	1.30657183566	1.30657183563	1.30657183563	1.30657183563176
126,026	1.30407993559	1.30407993561	1.30407993559	1.30407993559	1.30407993558613
167,045	1.30380711815	1.30380711818	1.30380711815	1.30380711815	1.30380711815453
228,118	1.30346087939	1.30346087941	1.30346087939	1.30346087939	1.30346087938640
291,124	1.30300542277	1.30300542279	1.30300542277	1.30300542277	1.30300542276642
995,538	---	---	---	1.30139746557	1.30139746557195
1,789,655	---	---	---	1.30077368230	1.30077368229511
2,509,211	---	---	---	1.30064937316	1.30064937315972
4,182,153	---	---	---	1.30046785768	1.30046785768403
5,981,980	---	---	---	1.30036553228	1.30036553227884

Eigenfrequency Convergence

TECHNISCHE UNIVERSITÄT DARMSTADT

DOF	MATLAB	CST	Pysparse	SLEPc	CEM3D
11,423	1.32883647670	1.32883647673	1.3288364767	1.32883647670	1.32883647670 4473
51,655	1.30814468722	1.30814468725	1.30814468722	1.30814468722	1.308144687223849
80,965	1.30657183563	1.30657183566	1.30657183563	1.30657183 [illegible]	[illegible]
126,026	1.30407993559	1.30407993561	1.30407993559	1.30407993 [illegible]	[illegible]
167,045	1.30380711815	1.30380711818	1.30380711815	1.3038071 [illegible]	[illegible]
228,118	1.30346087939	1.30346087941	1.30346087939	1.30346087939	1.30346087938 6402
291,124	1.30300542277	1.30300542279	1.30300542277	1.30300542277	1.30300542276 6423
995,538	---	---	---	1.30139746557	1.30139746557 1954
1,789,655	---	---	---	1.30077368230	1.30077368229 5119
2,509,211	---	---	---	1.30064937316	1.30064937315 9727
4,182,153	---	---	---	1.30046785768	1.30046785768 4032
5,981,980	---	---	---	1.30036553228	1.30036553227 8849

specified solver accuracy 10^{-9}

Eigenfrequency Convergence

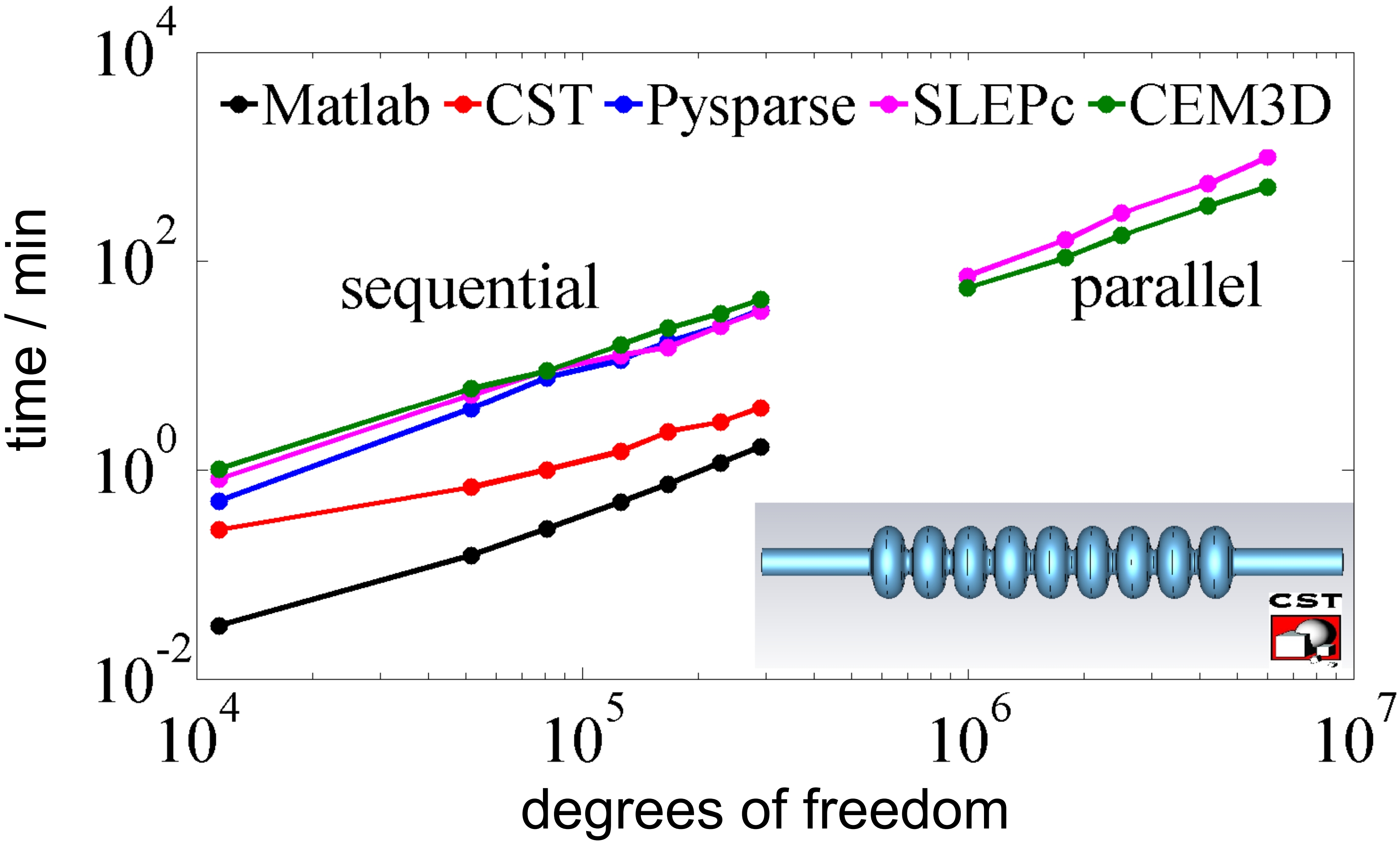
TECHNISCHE UNIVERSITÄT DARMSTADT

DOF	MATLAB	CST	Pysparse	SLEPc	CEM3D
11,423	1.32883647670	1.32883647673	1.3288364767	1.32883647670	1.32883647704473
51,655	1.30814468722	1.30814468725	1.30814468722	1.30814468722	1.308144687223849
80,965	1.30657	66	1.30657183563	1.30657183	[illegible]
126,026	1.30407	61	1.30407993559	1.3040799	[illegible]
167,045	1.30380	18	1.30380711815	1.3038071	[illegible]
228,118	1.30346087939	1.30346087941	1.30346087939	1.30346087939	1.30346087938 6402
291,124	1.30300542277	1.30300542279	1.30300542277	1.30300542277	1.30300542276 6423
995,538	---	---	---	1.30139746557	1.30139746557 1954
1,789,655	---	---	---	1.30077368230	1.30077368229 5119
2,509,211	---	---	---	1.30064937316	1.30064937315 9727
4,182,153	---	---	---	1.30046785768	1.30046785768 4032
5,981,980	---	---	---	1.30036553228	1.30036553227 8849

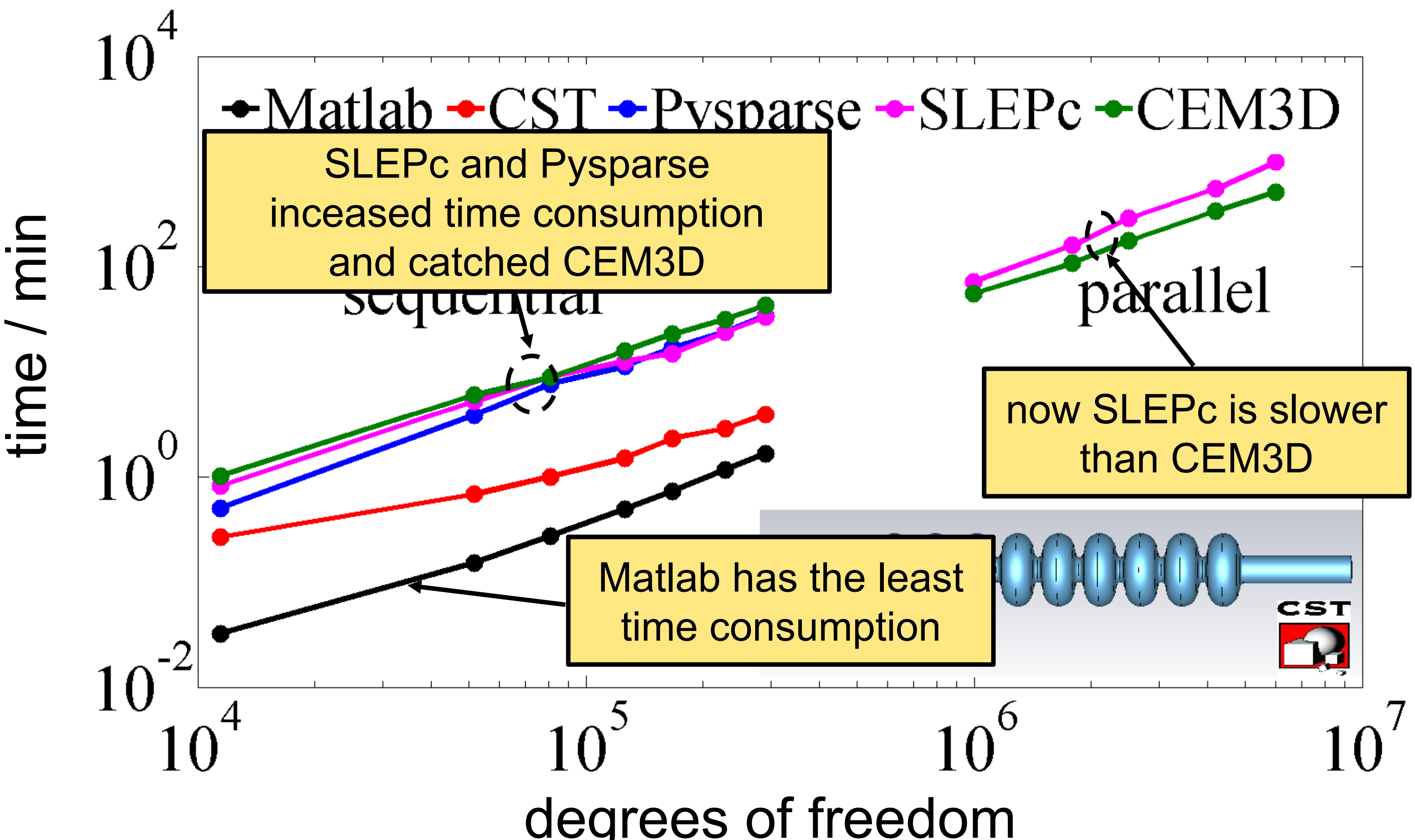
converges to 1.300 GHz

specified solver accuracy 10^{-9}

TESLA Cavity Time Consumption

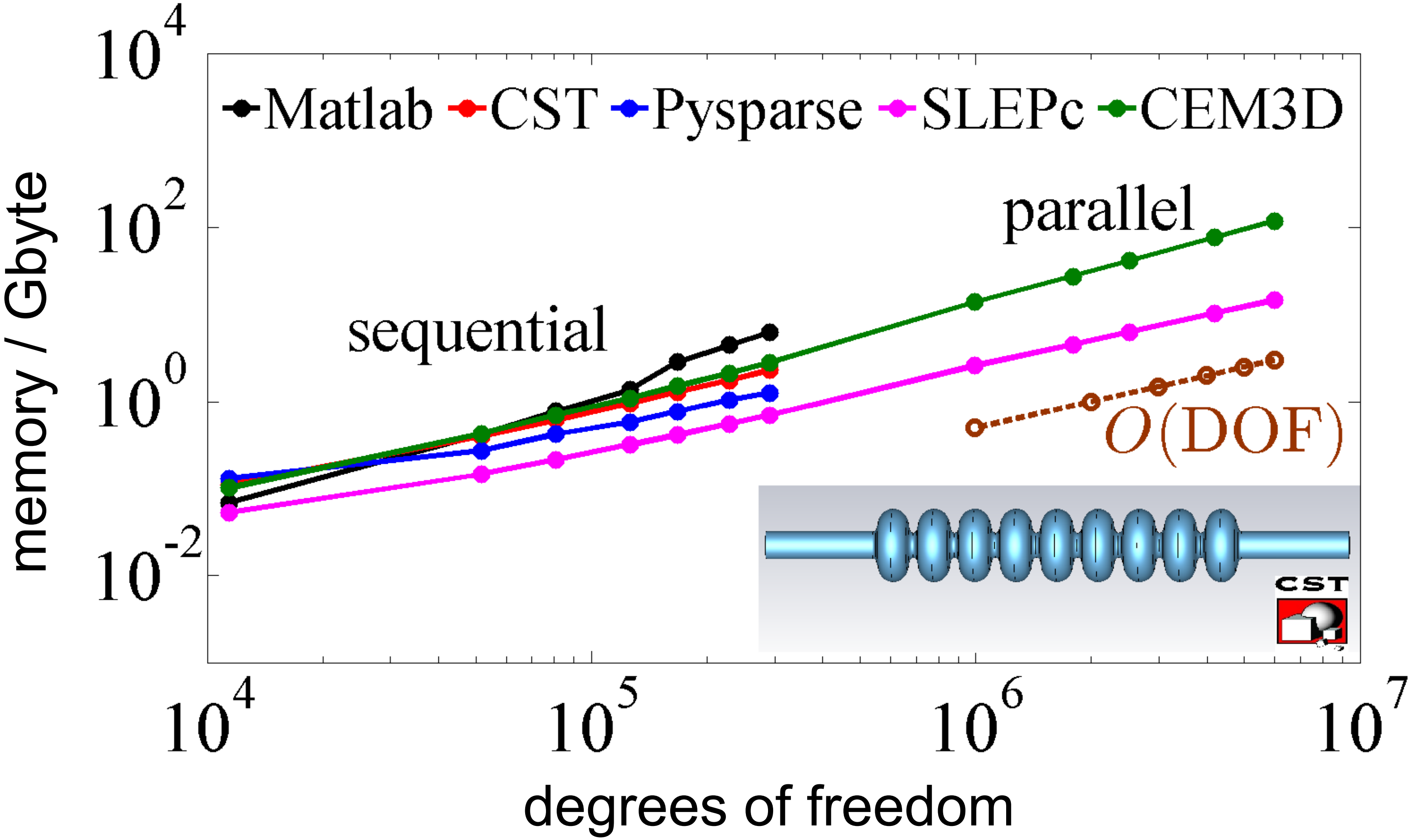


TESLA Cavity Time Consumption

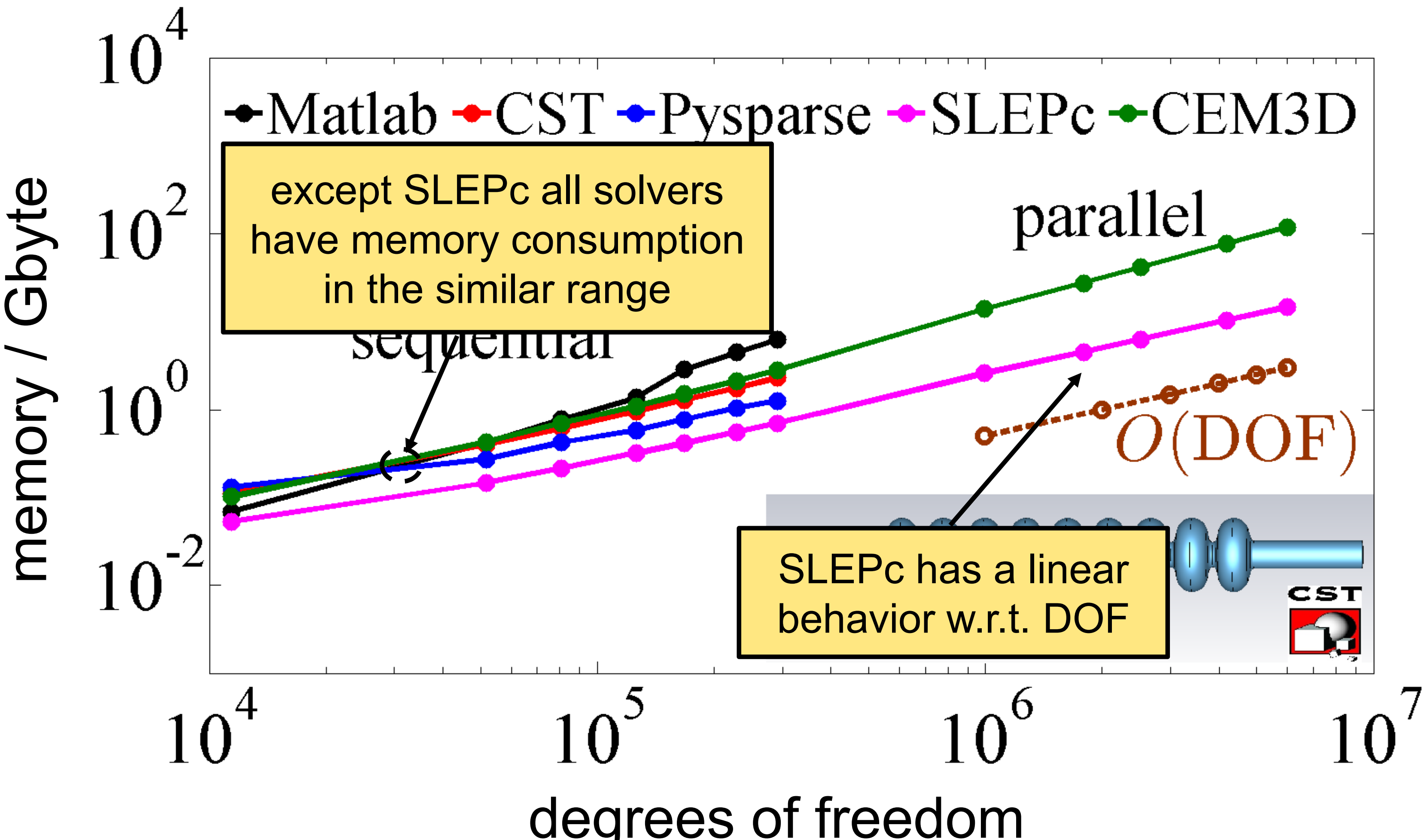


TESLA Cavity Memory Consumption

TECHNISCHE UNIVERSITÄT DARMSTADT

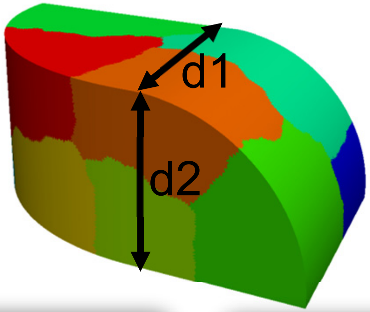


TESLA Cavity Memory Consumption



Billiard Resonator Simulations

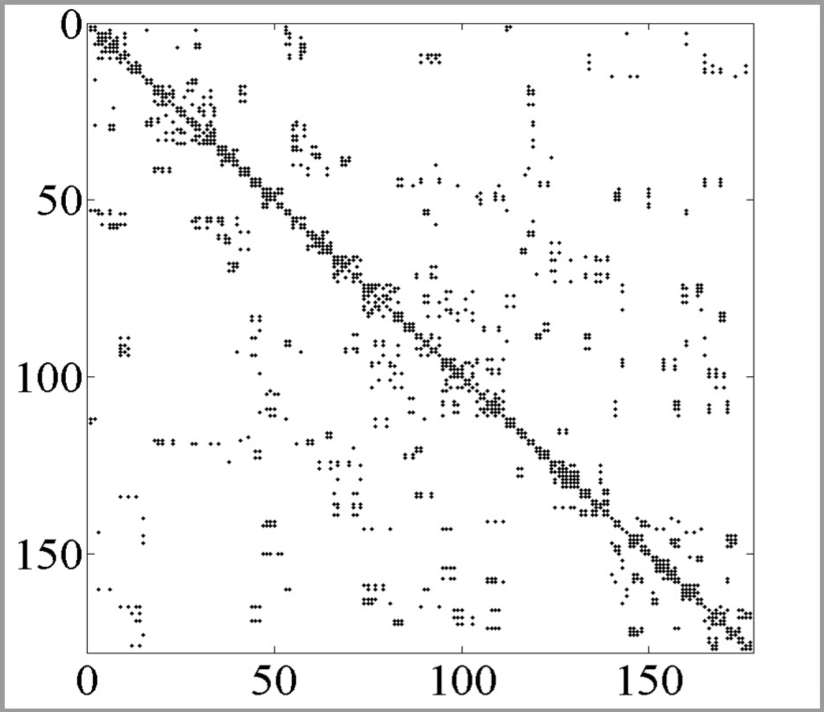
distribution on 10 nodes



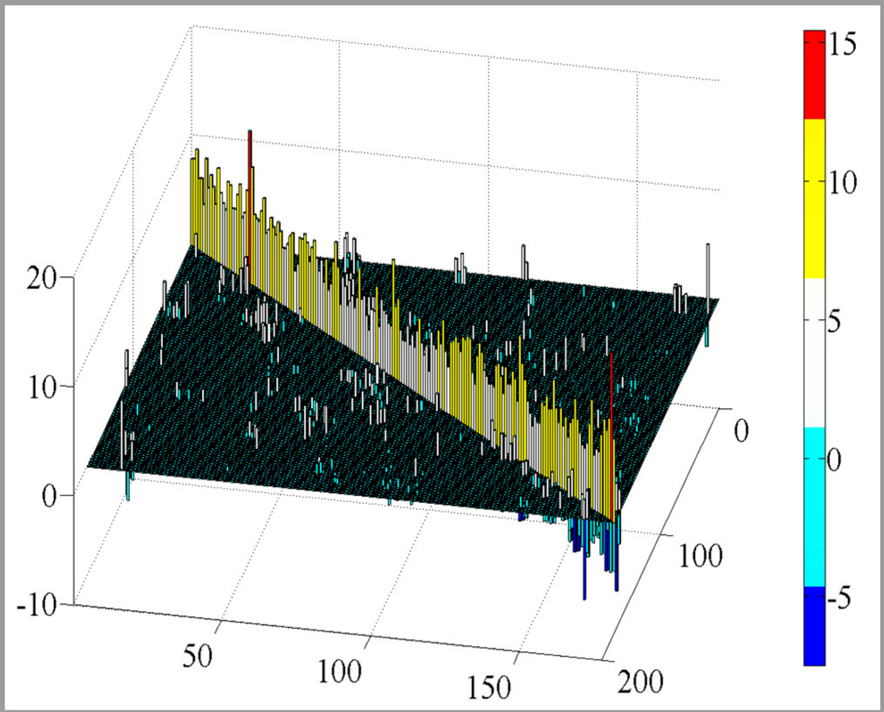
d1 =19.82cm

d2 =14.14cm

Matrix A



Matrix A (city plot)

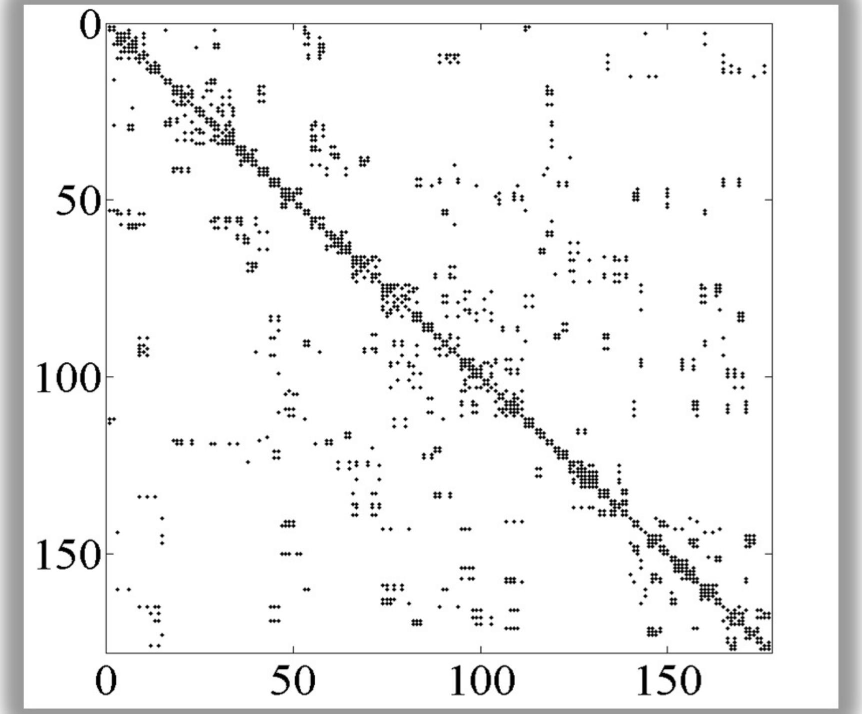


Billiard

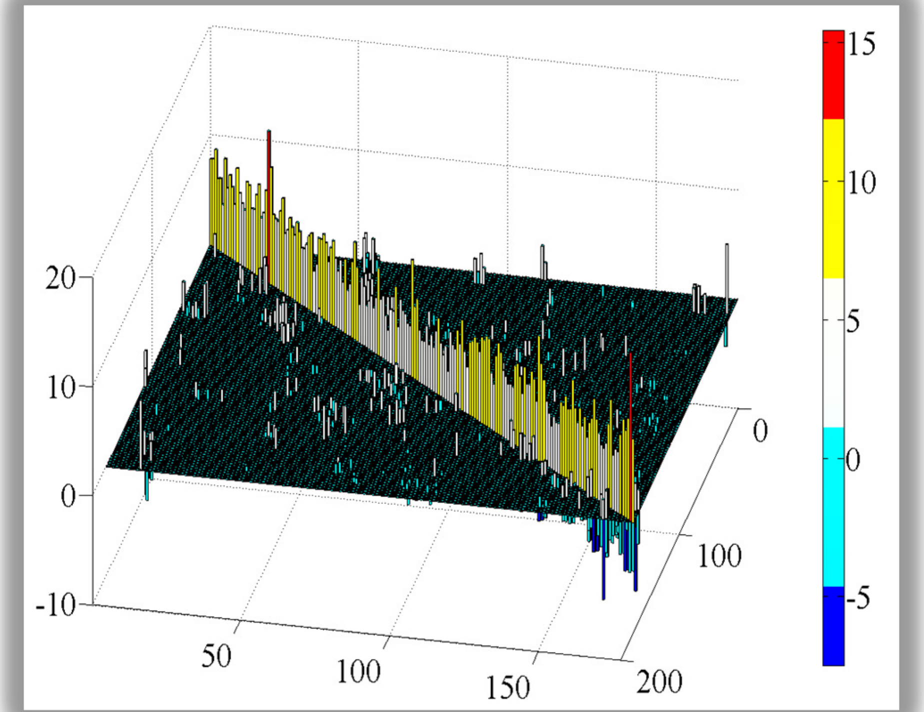
- Number of mesh cells: **301**
- Number of DOF: **177**
- Number of Nonzero elements: **1477**
- Sparsity % = **4.71**

distribution

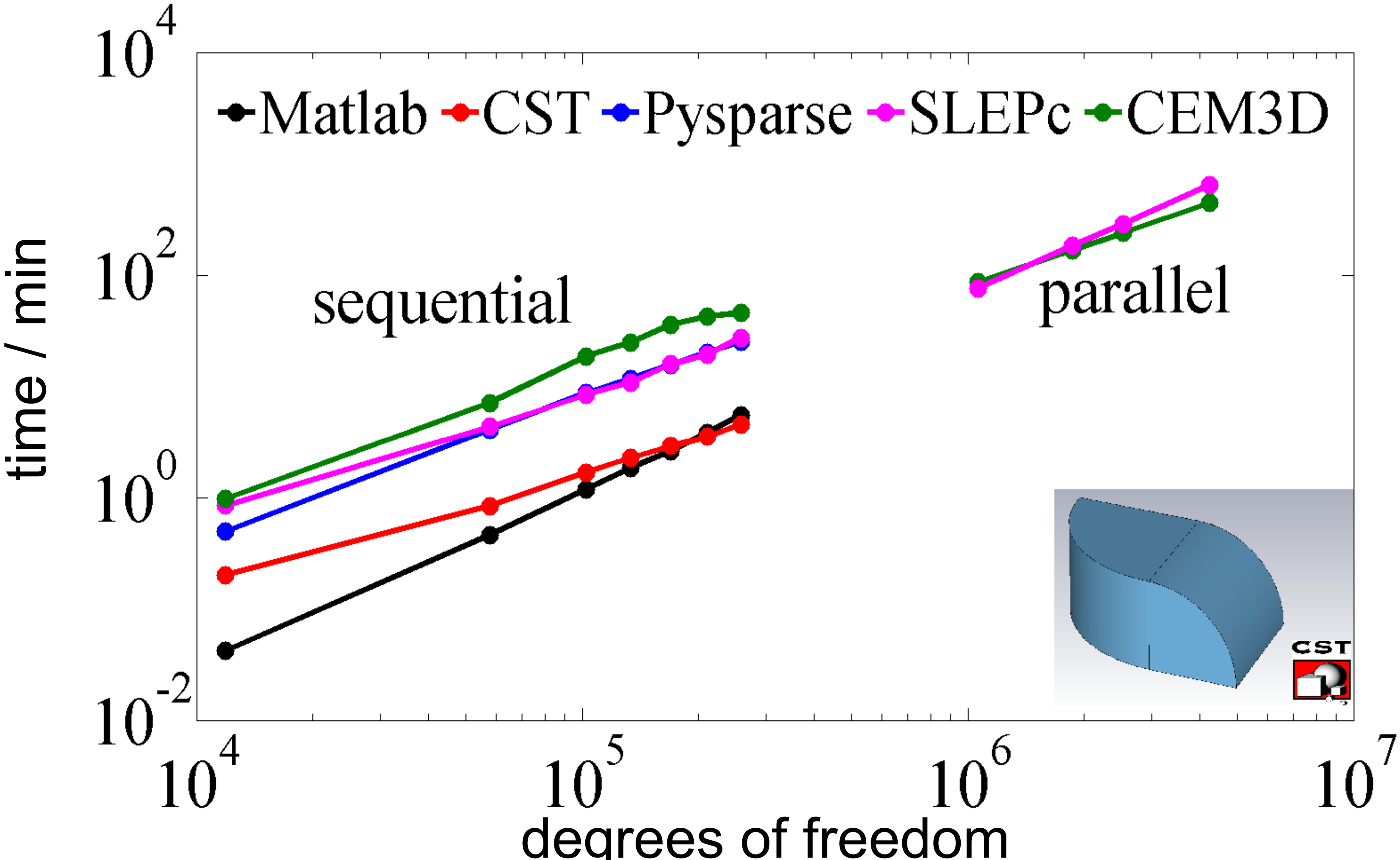
Matrix A



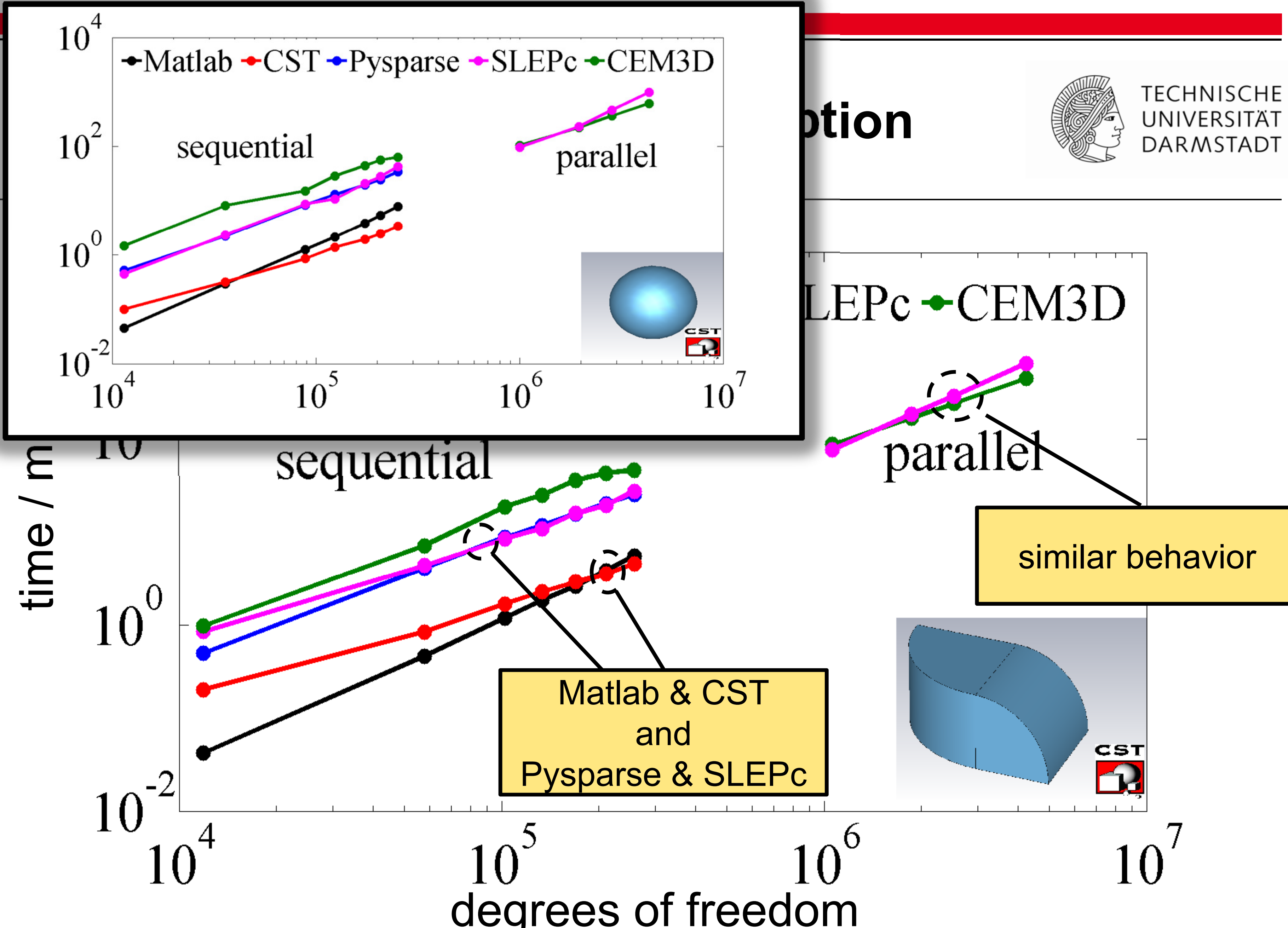
Matrix A (city plot)



Billiard Resonator time Consumption

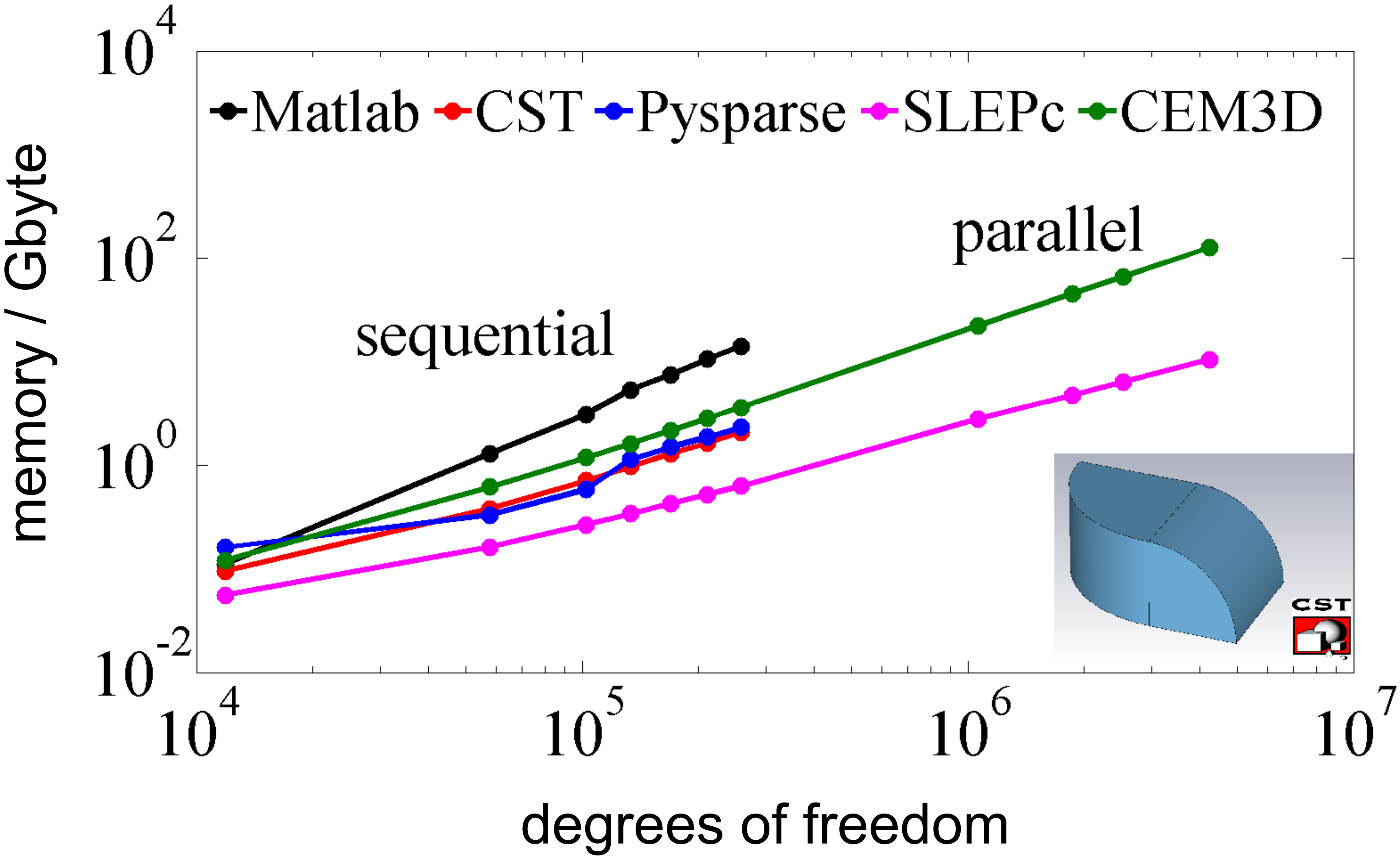


ption



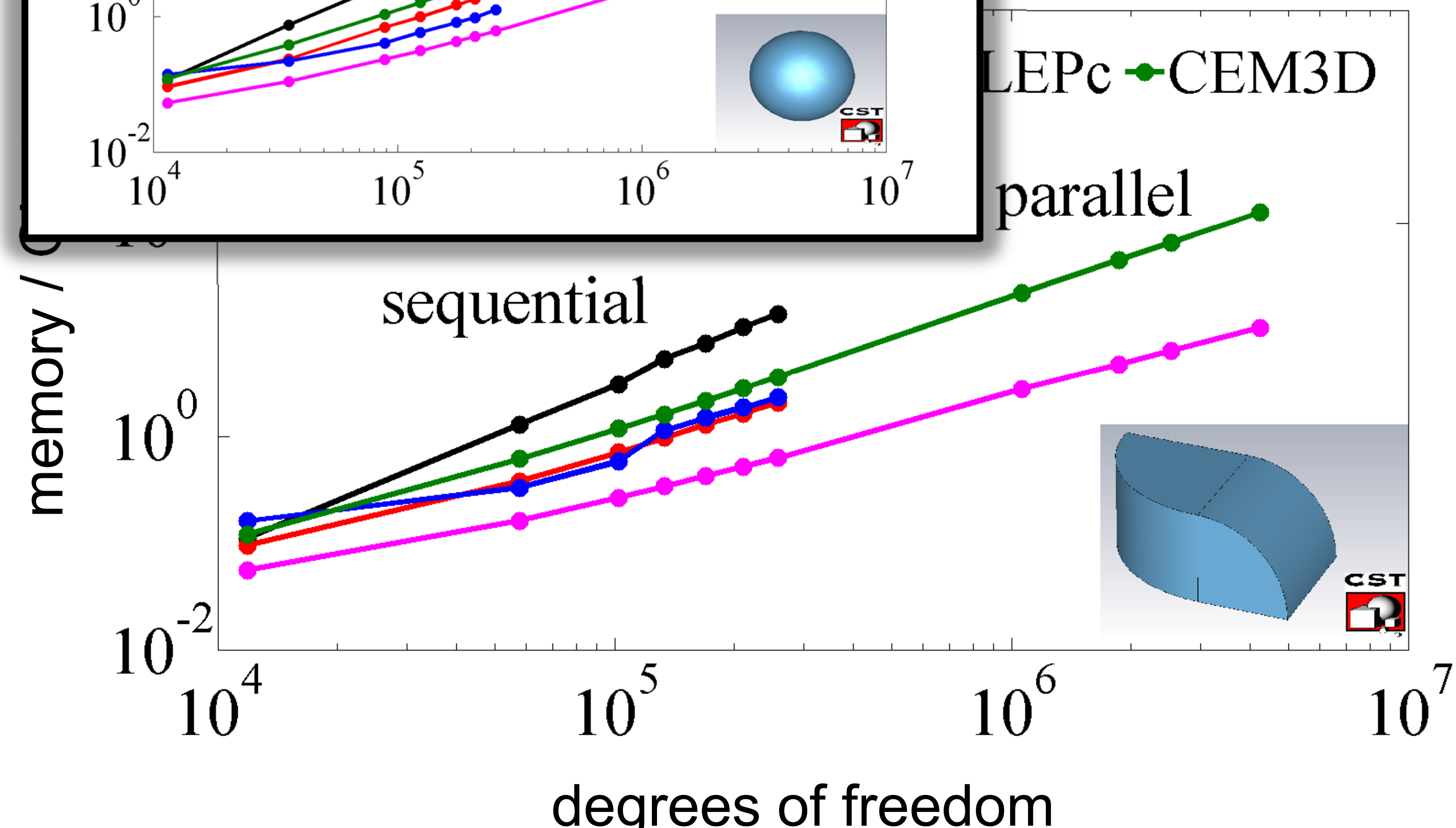
Matlab
CST
Pysparse
SLEPc
CEM3D
sequential
parallel
10^{4}
10^{2}
10^{0}
10^{-2}
10^{4}
10^{5}
10^{6}
10^{7}
CST
time / m
degrees of freedom
similar behavior
Matlab & CST
and
Pysparse & SLEPc

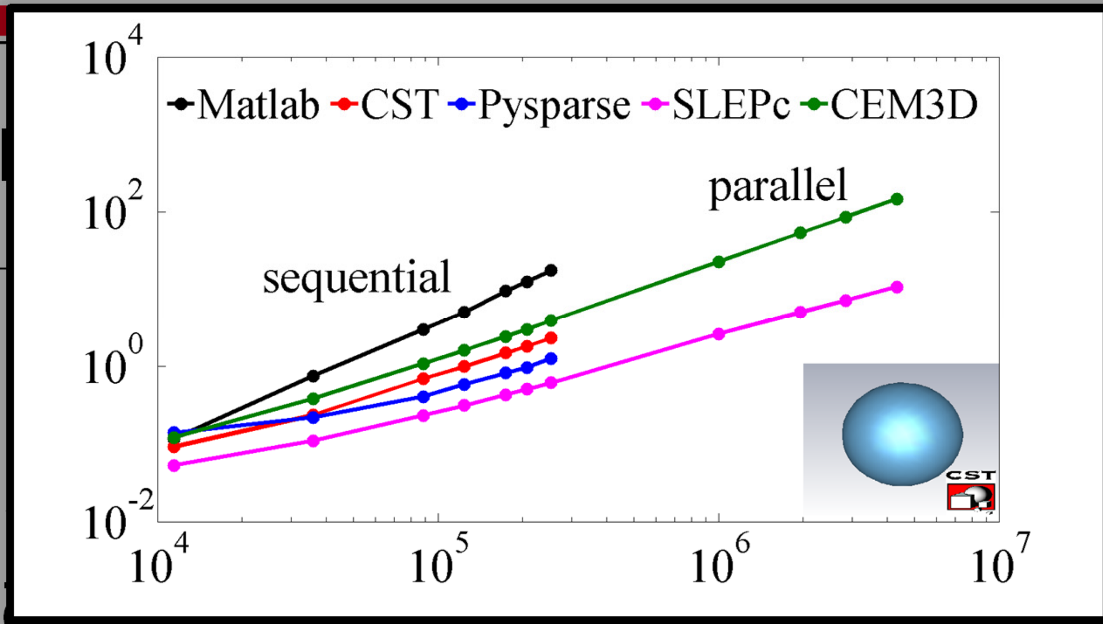
Billiard Resonator Memory Consumption



umption

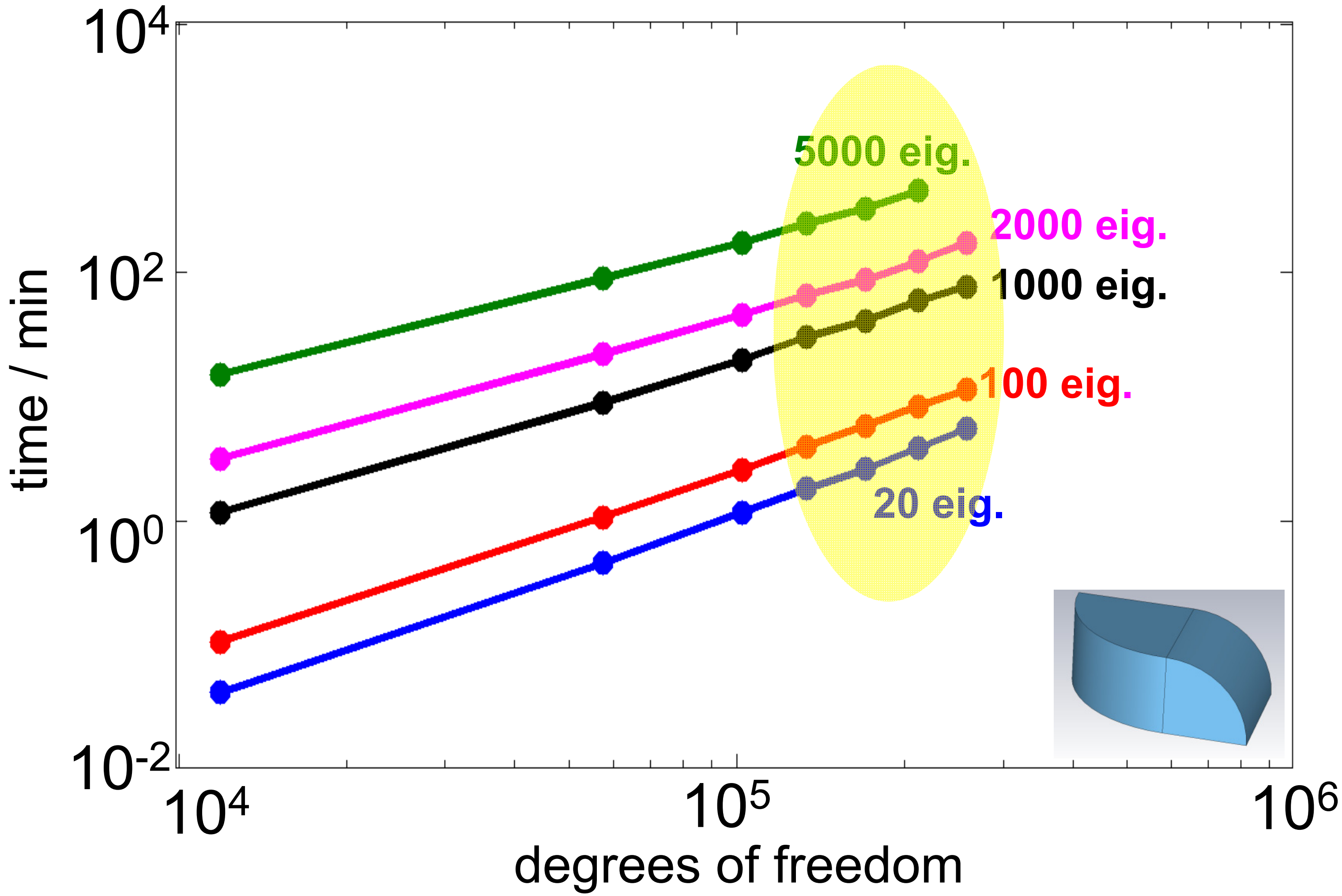
TECHNISCHE UNIVERSITÄT DARMSTADT



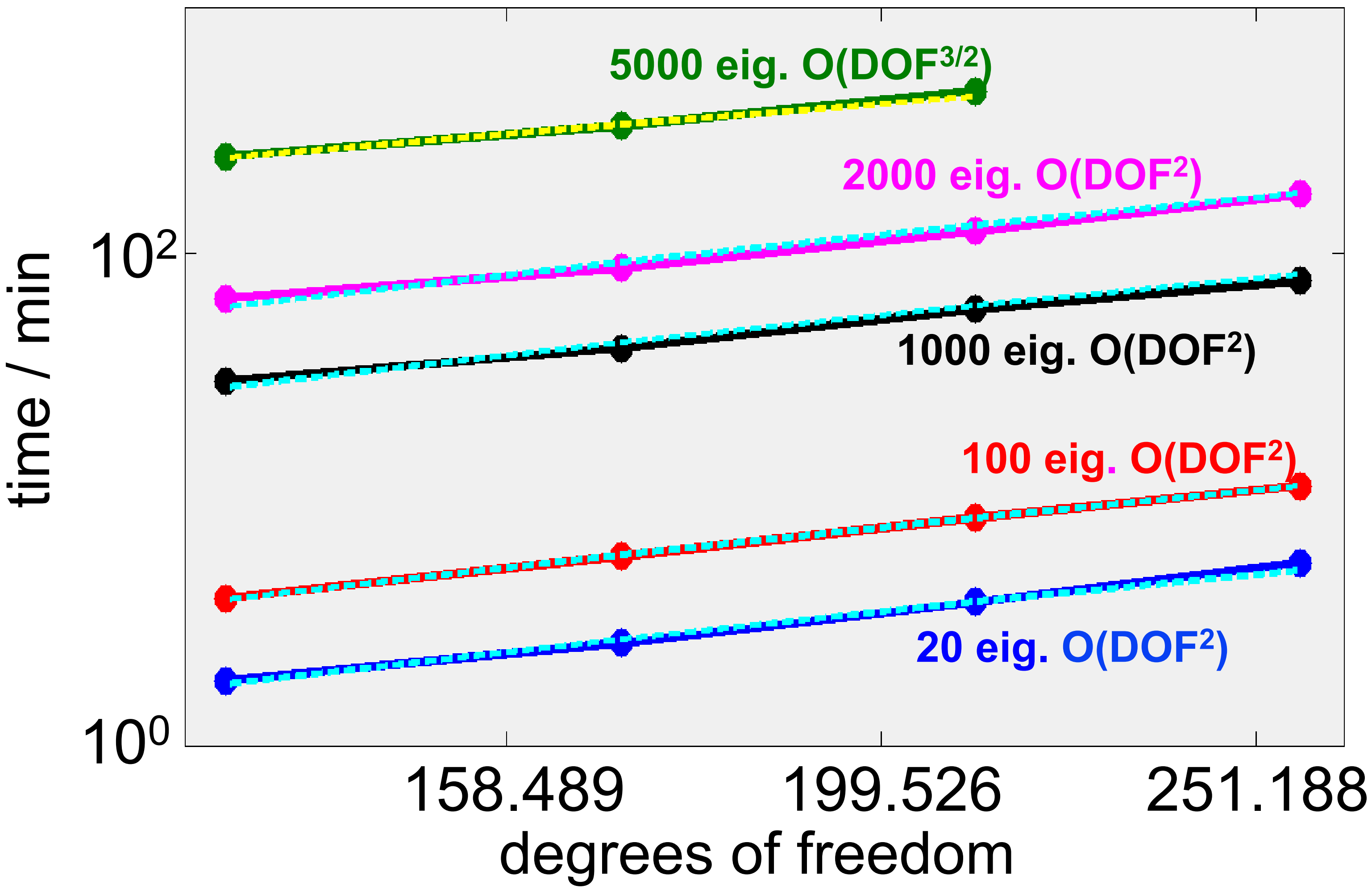


Matlab time consumption for different number of eigenvalue computations

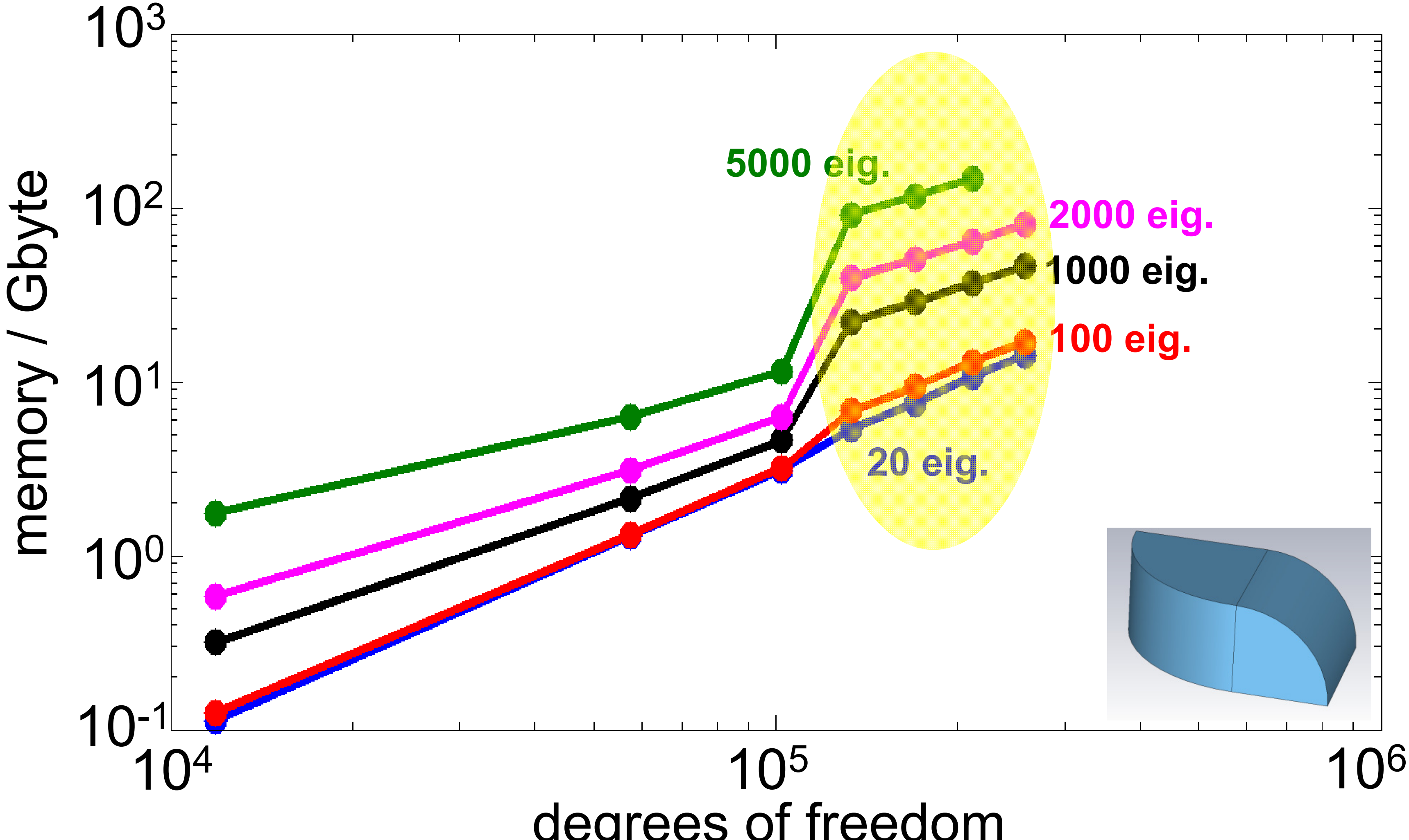
TECHNISCHE UNIVERSITÄT DARMSTADT



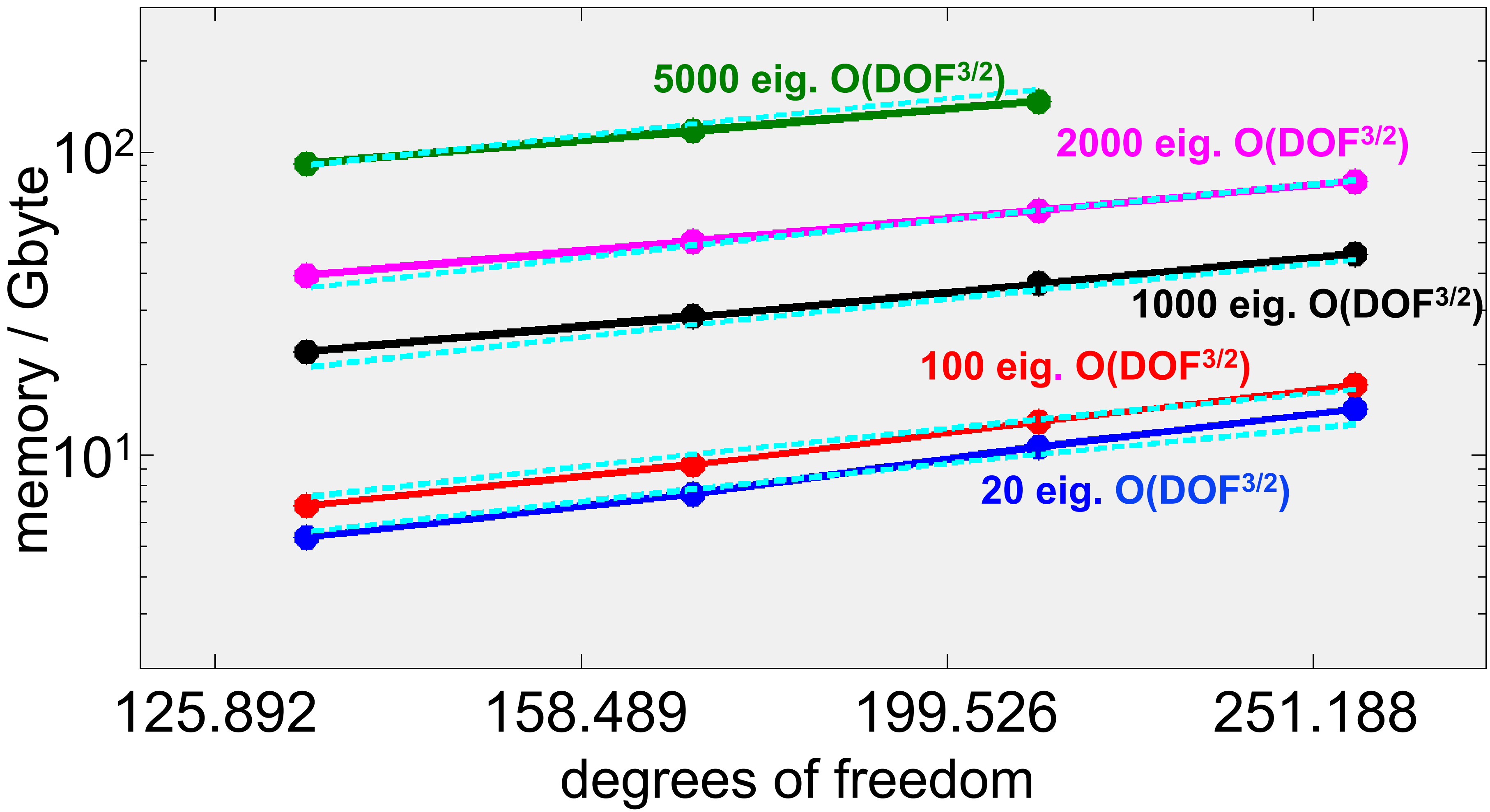
Matlab time convergence rate



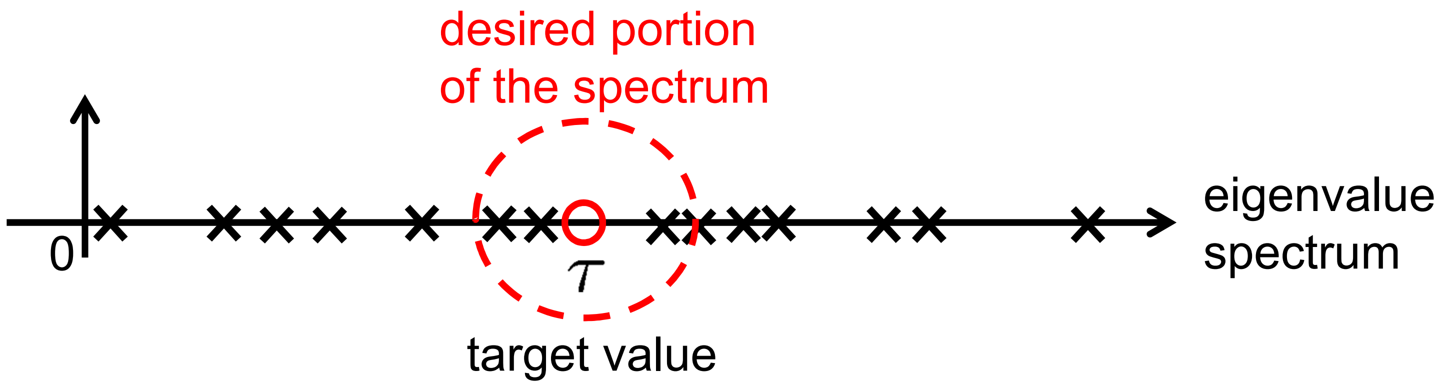
Matlab memory consumption for different number of eigenvalue computations



Matlab memory convergence rate

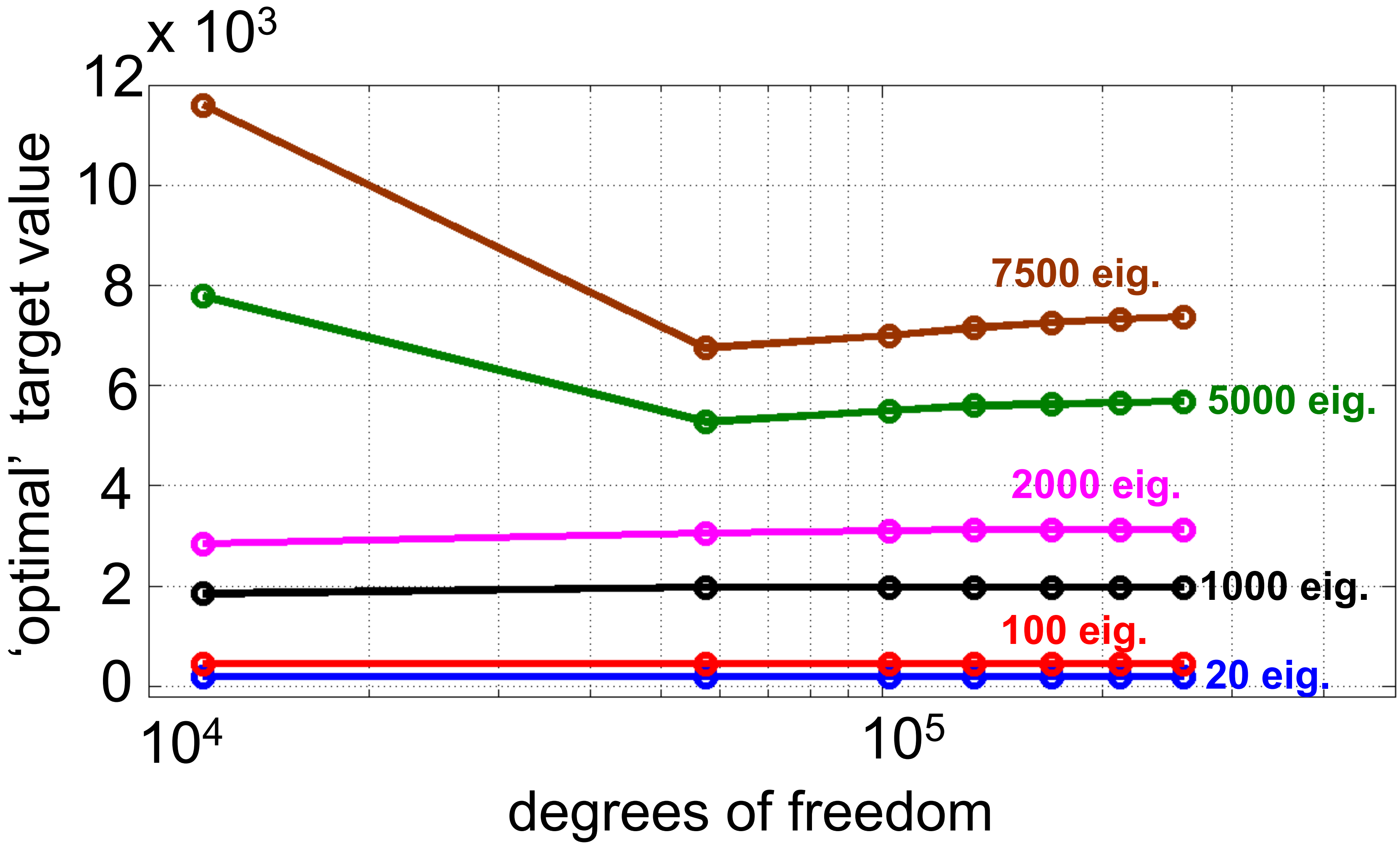


Matlab target value vs DOF



for the same amount of requested eigenvalues target value should not change w.r.t. to size of the problem

Matlab target value vs DOF



Conclusions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Solvers change their behavior depending on the geometry of the problem
- (Matlab and CST) can be considered as a group and (Pysparse, SLEPc and CEM3D) as another group which behaves similar from time consumption point of view
- As a very important key point in large problem sizes: the robustness of chosen target value in CST and CEM3D increases the applicability of the solver
- SLEPc uses the least memory in all experiments and can be used on a single computer or on a cluster for very large amount of DOF
- Matlab is a fast solver but needs large amount of memory

References for Softwares

TECHNISCHE UNIVERSITÄT DARMSTADT

V. Hernandez, J. E. Roman and V.Vidal, ``SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems,'' ACM Trans. on Math. Software 31 (2005) 3.
http://www.grycap.upv.es/slepc/download/download.htm

R. Geus, ``The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems with application to the design of accelerator cavities, ''
ETH Zurich, PhD Thesis, 2002.
http://sourceforge.net/projects/pysparse/

MATLAB R2011a, The MathWorks Inc., Natick, MA, 2011.

CST AG, CST 2012, Darmstadt, Germany. MICROWAVE STUDIO

W. Ackermann and T. Weiland, ``High Precision Cavity Simulations,'' , MOADI1

A. Henderson, ParaView Guide, A Parallel Visualization Application.,
Kitware Inc., 2007.

Thank you for your attention

F. Yaman