More than 25 Years of CalculiX: Sustaining an Industrial Open-Source FEM Code

22.07.2026 | L. Bruder, C. Wölfle, G. Dhondt

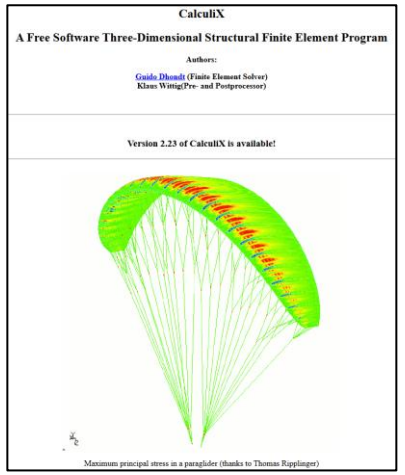
CalculiX – a Multidisciplinary 3D Finite Element Software

Started in 1998 in spare time

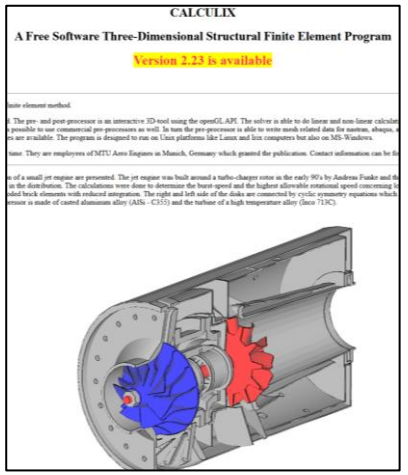
- CalculiX CrunchiX → Solver (Guido Dhondt) → **this talk**
- CalculiX GraphiX → Pre- and Postprocessor (Klaus Wittig)
- ABAQUS-like .inp input format (supports a limited set of keywords)
- .frd / .dat output format

Since 2000

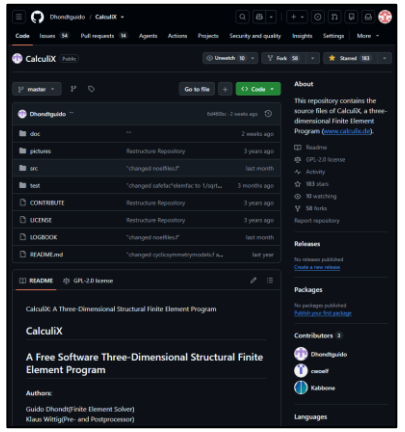
- Distributed under the GPL-2.0 license
- Continuous feature enrichment
- Co-developers (mostly students & university partners) for specific implementations
- Increasing usage at MTU (especially structural mechanics, heat transfer and secondary air system)
  - → Reduced license costs (massive job parallelization / important for multi-query tasks such as optimization)
  - → Custom feature development
  - → Strengthening in-house FEM know-how
- Increasing user community (mostly structural mechanics & heat transfer)



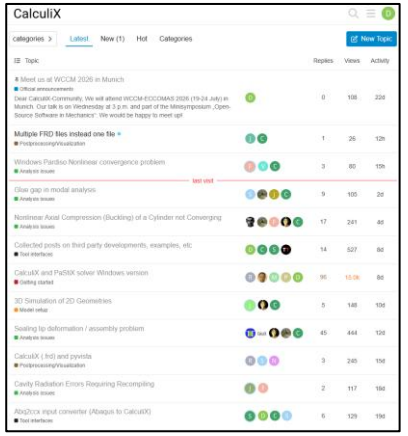
dhondt.de



calculix.de



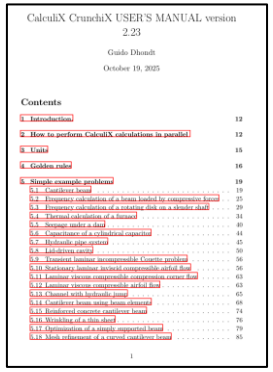
github.com/Dhondtguido/CalculiX



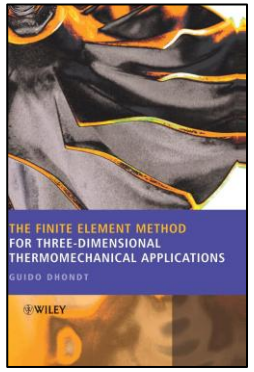
https://calculix.discourse.group

CalculiX – a Multidisciplinary 3D Finite Element Software

- CalculiX User's Manual (> 800 pages)
- Dhondt, G. The Finite Element Method for Three-Dimensional Thermomechanical Applications, Wiley, 2004. (→ new issue in preparation)
- Several publications (journal articles, student thesis, etc.)
- Test suite with > 600 minimal examples



User's Manual

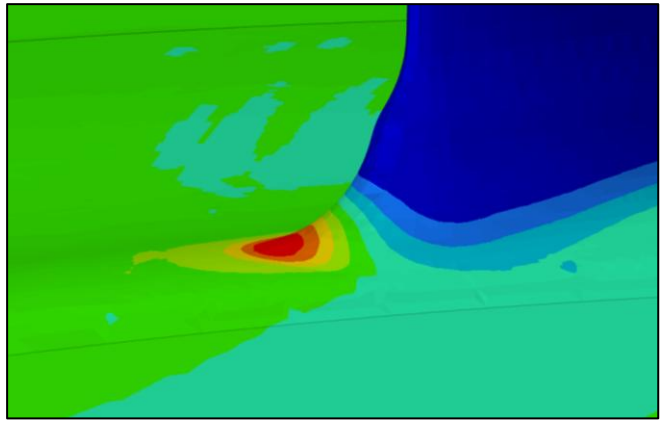


Book

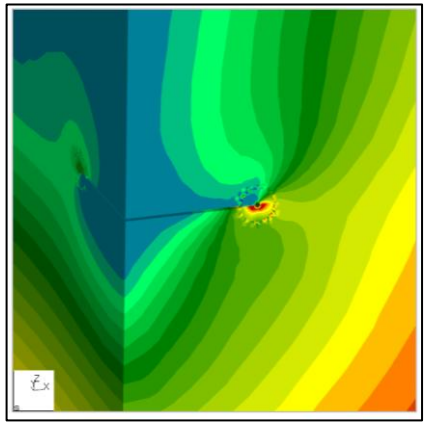
Supported types of analysis

Statics
(Complex) Frequency
Implicit & explicit dynamics
Steady state dynamics
Modal dynamics
Crack propagation
Substructure generation
Buckling
Sensitivity
Feasible Direction
Robust Design
Heat transfer (conduction, convection, radiation)
Aerodynamic & hydraulic networks
Diffusion mass transfer in a stationary medium
Irrotational incompressible inviscid flow
Hydrodynamic lubrication
Shallow water motion
Shallow water calculations
Stationary groundwater flow
Turbulent flow in open channels
Acoustics
Electrostatics
CFD
Electromagnetism
Green functions

Custom proprietary developments based on the CalculiX solver



Shape optimization via vertex morphing

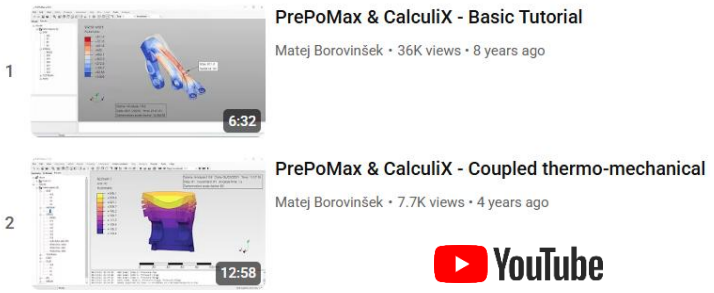


Cracktracer3D

CalculiX Community

Several initiatives (mostly decentralized / independent)

- Dedicated pre- and postprocessors: CalculiX GraphiX, PrePoMax
- Software interfacing with CalculiX: preCICE, FreeCAD, Mecway
- GitHub projects: ccx2paraview, Cubit-CalculiX, abq2ccx, CalculiX-Examples, …
- Packaged by Linux distributions
- Tutorials & discussions
- Validated against ISO 10211-2017 (Thermal bridges in building construction)
- Used by other companies



FreeCAD

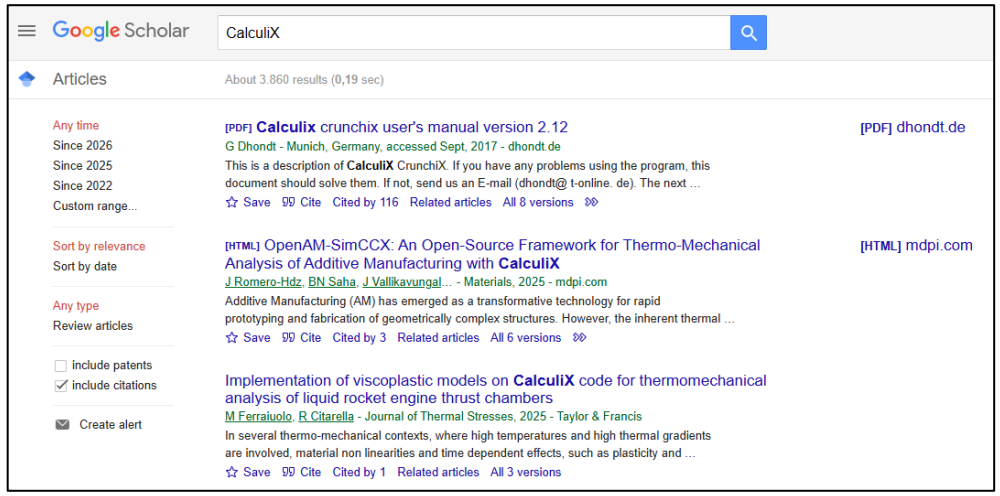
PrePoMax

Mecway

preCICE

Statistics on community activity

- Site downloads per month
  - → ~1000 source code (latest version)
  - → ~300 Linux executable (latest version)
  - → ~400 Windows executable (latest version)
- ~800 registered users on Discourse with ~100 posts per month (+ a lot more anonymous readers)
- ~3800 hits on Google Scholar



Development model

Status quo for a long time

- Single author development
- Code distribution via website
- Bug reporting via email
- Contributions via email

Pros

- Mature and stable software
- Consistent coding style and documentation
- Most-used functionality well tested and optimized

Cons

- Very few community developers
- Redistribution of marginally different custom binaries / sources

Transitioning to a team-based development

- Development team at MTU
- Allocating responsibilities
- Establishing a git strategy
- Issue tracking
- CI/CD

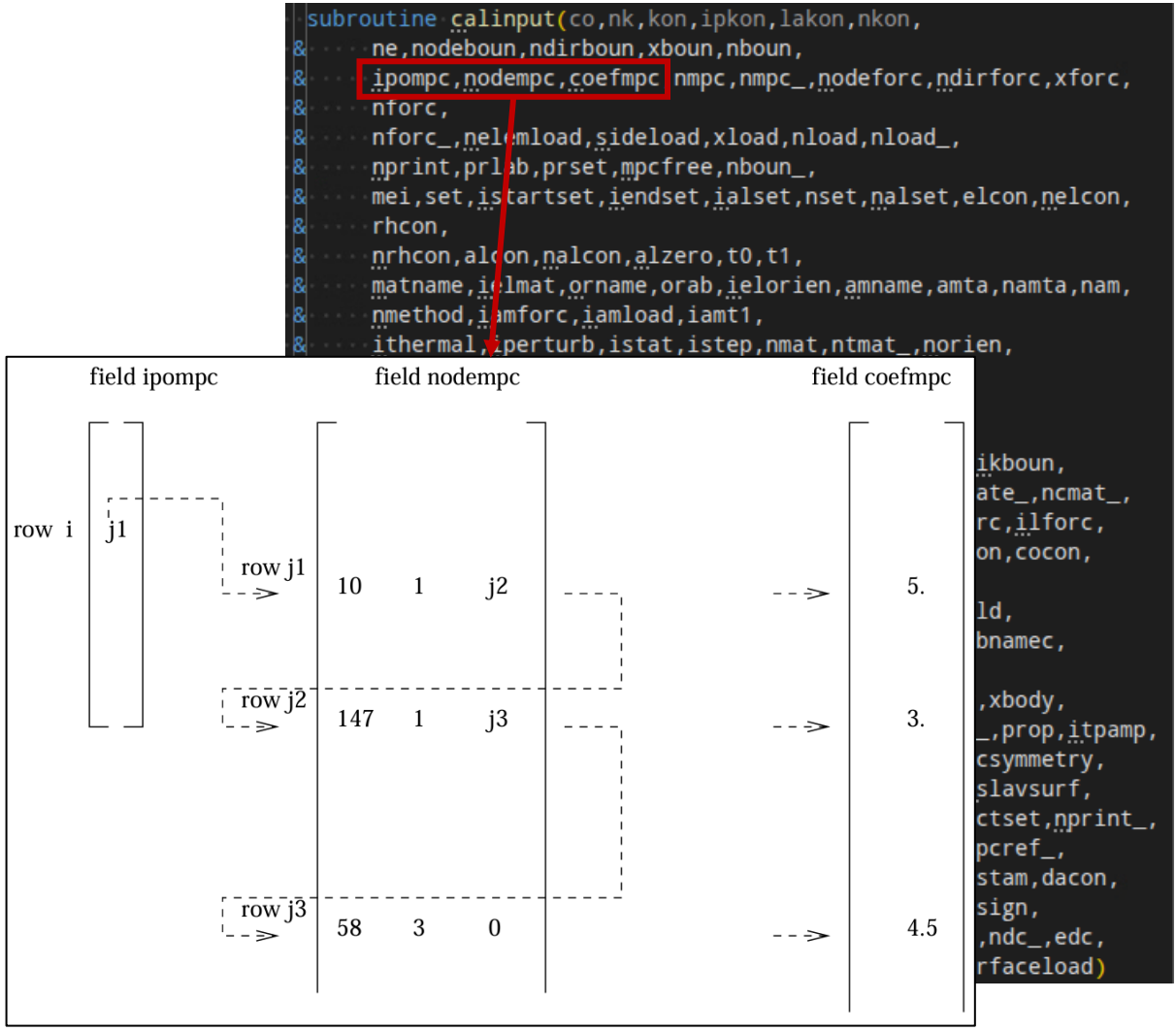
Encouraging community contributions

- Centralizing activity via GitHub and Discourse
- Enabling multi-contributor workflows
  - → Contributing guidelines
  - → Coding standards
  - → Code reviews

Code Base Properties & Challenges

- Large, traditional, procedural mixed C/Fortran (fixed format) code (~250k SLOC)
  - → C mainly as driver and memory management layer → requires careful manual management of dynamic memory
  - → Virtually no abstractions / encapsulation → unwieldy function signatures (>100 arguments in extreme cases)
  - → Implicit interfaces on Fortran side → modern Fortran features / checks unavailable
  - → Interoperability through calling of raw symbols → some platform dependency (mitigated by extensive macro use)

Example: internal representation of MPCs



$$5 * u_1^{(10)} + 3 * u_1^{(147)} + 4.5 * u_3^{(58)} = 0$$

Code Base Properties & Challenges

- Support for legacy architectures / compilers / solvers → test automation for building & testing all combinations desirable
- Few external dependencies
  - → Mostly handwritten algorithms also for standard functionality (e.g. geometric and sparse matrix operations)
  - → Solvers and standard linear algebra packages (mainly SPOOLES, PARDISO, PaStiX, BLAS/LAPACK, ARPACK)
- Legacy output format (.frd) → large company-internal toolchain dependency but fewer open source tools (notably CalculiX GraphiX)
- Bare-bones Makefile → fragmentation of 3rd party build scripts / documentation across platforms
- Shared memory parallelization based on “parallel-region”-local POSIX threads

Example: y=A*x for real sparse symmetric matrices

```
      subroutine op(x,y,ad,au,jq,irow,na,nb)
!
      implicit none
!
      integer irow(*),na,nb,j,l,i,jq(*)
!
      real*8 y(*),x(*),au(*),ad(*)
!
!     diagonal terms
!
      do i=na,nb
        y(i)=ad(i)*x(i)
      enddo
!
!     off-diagonal terms
!
      do j=na,nb
        do l=jq(j),jq(j+1)-1
          i=irow(l)
          y(i)=y(i)+au(l)*x(j)
          y(j)=y(j)+au(l)*x(i)
        enddo
      enddo
!
      return
      end
```

Code Modernization

Goal 1: Reducing technical debt

- Replacing/updating legacy dependencies (e.g. hard-copied LAPACK routines, ARPACK, divergent forked PaStiX)
- Transitioning to OpenMP (see example)
- Introducing standardized C/Fortran interoperability using ISO_C_BINDING

Goal 2: Improving maintainability

- Gradually introducing abstractions and modularization → reducing code redundancies & function sizes
- Gradually building (in-code, parseable) developer documentation
- Introducing/enabling assistive tooling (e.g. linters, asan)

Goal 3: Improving utility

- Harmonization of build systems
- (Unified) interfacing to modern solver & linear algebra libraries
- Device offloading for special tasks

Replaceable by **OpenMP** loop in applybounfem

```
// --- GLOBALS: all args duplicated as _1 pointers ---
int *nodeboun1, *nboun1, *nmpc1, *nfreestream1, *nsolidsurf1, ...;
double *xbounact1, *vold1, *v1, ...;

// --- C CALLER (compfluidfem.c) ---
void compfluidfem(...) {
   int num_cpus, ithread[num_cpus];
   pthread_t tid[num_cpus];
   // 1. Stash all args into globals
   nodeboun1 = nodeboun; nboun1 = nboun;
   // ... (25+ assignments)
   // 2. Launch threads
   for (i = 0; i < num_cpus; i++) {
      ithread[i] = i;
      pthread_create(&tid[i], NULL, applybounfemmt, &ithread[i]);
   }
   for (i = 0; i < num_cpus; i++)
      pthread_join(tid[i], NULL);
}

// --- C SCHEDULER ---
void *applybounfemmt(int *id) {
   // helper function for calculating the threads' loop bounds
   int nbouna = *id * ceil(*nboun1/num_cpus) + 1;
   int nbounb = min(nbouna + ceil(*nboun1/num_cpus), *nboun1);
   int nmpca  = ...;  // same pattern for each array
   int nmpcb  = ...;
   int nfreestreama, nfreestreamb, ...;
   FORTRAN(applybounfem, (nodeboun1, ..., &nbouna, &nbounb,
                          &nmpca, &nmpcb,
                          &nfreestreama, &nfreestreamb,
                          &nsolidsurfa, &nsolidsurfb));
}
      interop
      macro

// --- Fortran WORKER (applybounfem.f) ---
subroutine applybounfem(..., nbouna, nbounb, nmpca, nmpcb,
                        nfreestreama, nfreestreamb, nsolidsurfa, nsolidsurfb) {
   do j = nbouna, nbounb      // SPCs
   do j = nmpca, nmpcb        // MPCs (×3)
   do j = nfreestreama, nfreestreamb  // freestream
   do j = nsolidsurfa, nsolidsurfb  // solid surf
}
```

Summary

→ Development and maintenance of CalculiX will continue ☺

Key points:

- Relatively large user base in comparison to number of developers
- Transitioning to a team-based development
- Improving accessibility for community contributions
- Maintaining stability and consistency

Other points on our current agenda:

- Implementation of the Riks solver (arc-length method)
- Improved linear solver interfaces
- User conference planned for Q4/2026 in Munich (will be announced via Discourse)

Development team

L. Bruder
C. Wölfle
G. Dhondt
M. Haßler
F.-J. Ertl
S. Teichtmeister
S. Erhard
T. Petzke