Open-source CFD software: FeatFlow

The FEA(S)T groups

Institute of Applied Mathematics, LS III

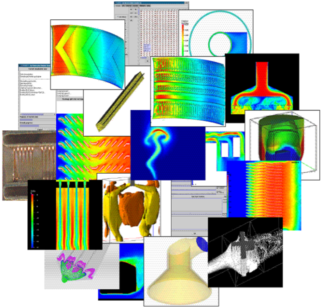
Dortmund University of Technology, Germany

matthias.moeller@math.tu-dortmund.de

Lake Tahoe, January 7, 2009

Overview

1. FeatFlow
   - FeatFlow 1.x
   - FeatFlow 2.0
2. Preprocessing
   - DeViSoR Grid3D
3. Example application
   - Poisson equation
4. Postprocessing
   - General Mesh Viewer
5. UnConventional HPC
   - FEAST



FeatFlow 1.x

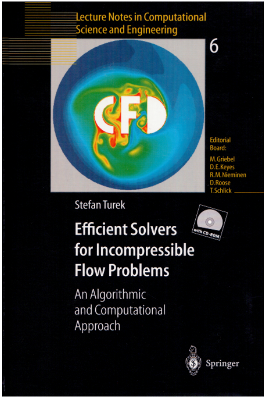
Finite **E**lement **A**nalysis **T**oolbox + Flow solver

High performance unstructured finite element package for the numerical solution of the incompressible Navier-Stokes equations

- based on the finite element packages Feat2D and Feat3D
- written in Fortran 77 (and some C routines) by Stefan Turek
- designed for education, scientific research and industrial applications
- full source-code and user manuals are available online
- many extensions are not included in the official release ☹

Visit the FeatFlow homepage

http://www.featflow.de



http://www.featflow.de/album

Theoretical background

Discretization techniques

Incompressible Navier-Stokes equations

$$\mathbf{u}_t - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{f}, \quad \nabla\cdot\mathbf{u} = 0, \qquad \text{in } \Omega\times(0,T]$$

- Spatial discretization techniques
  - nonconforming rotated multilinear finite elements for $\mathbf{u}$
  - piecewise constant pressure approximation for p
  - Samarskij upwind or streamline diffusion stabilization
- Temporal discretization techniques
  - implicit one-step-θ-scheme (Backward Euler, Crank-Nicolson)
  - implicit fractional-step-θ scheme (second-order accurate)
  - adaptive time-stepping based on local discretization error

Solution techniques

technische universität dortmund

Discretized incompressible Navier-Stokes equations

Given $\mathbf{u}^n$, $\mathbf{g}$ and k, solve for $\mathbf{u} = \mathbf{u}^{n+1}$ and $p = p^{n+1}$

$$[M + \theta k N(\mathbf{u})]\mathbf{u} + kBp = \mathbf{g}, \qquad B^T\mathbf{u} = 0$$

where $\mathbf{g} = [M - \theta_1 k N(\mathbf{u}^n)]\mathbf{u}^n + \theta_2 k \mathbf{f}^{n+1} + \theta_3 k \mathbf{f}^n$

- Nonlinear/linear solution strategies
  - coupled fixed point defect correction method `CC2D/CC3D`
  - nonlinear discrete projection scheme `PP2D/PP3D`
  - linear multigrid techniques with adaptive step-length control
  - ILU/SOR or Vanka-like block Gauß-Seidel smoother/solver

Finite **E**lement **A**nalysis **T**oolbox + Flow solver

The modern successor of FeatFlow 1.x for the numerical solution of flow problems by the finite element method

- modular object-oriented design by Michael Köster et al.
- written in Fortran 95 (kernel + applications)
- external libraries in F77/C (BLAS, UMFPACK, LAPACK)
- designed for education of students and scientific research
- detailed in-place documentation of the source-code

Official release not yet available; get the ALPHA snapshot

`http://www.featflow.de/download/Featflow2_2.0ALPHA.tar.gz`

Prerequisites

technische universität
dortmund

Unix/Linux and Mac OS X

- compatible `C` and `F95` compiler
  - `GCC` and `G95` version 0.91
  - Intel® C++/Fortran Compilers for Linux
  - Sun Studio C, C++ and Fortran Compilers
- GNU `make` utility
  - Makefiles are provided for all applications

Windows XP, Vista

- Microsoft® VisualStudio 2003, 2005 or 2008
  - Project files are provided for all applications
- Intel® C++/Fortran Compilers for Windows
- Cygwin™ environment
  - General Mesh Viewer (GMV)

Prerequisites

Unix/Linux and Mac OS X

- compatible C and F95 compiler
  - GCC and G95 version 0.91
  - Intel® C++/Fortran Compilers for Linux
  - Sun Studio C, C++ and Fortran Compilers
- GNU make utility
  - Makefiles are provided for all applications

Windows XP, Vista

- Microsoft® VisualStudio 2003, 2005 or 2008
  - Project files are provided for all applications
- Intel® C++/Fortran Compilers for Windows
- Cygwin™ environment
  - General Mesh Viewer (GMV)

Linux is used for this workshop

Getting FEATFLOW 2.0

- Unpack the downloaded archive file

```
$ tar xvzf Featflow2_2.0ALPHA.tar.gz
```

- Change into the base directory

```
$ cd Featflow2 ; ls
```

```
applications    Globals.power   object
bin             Globals.sparc   readme.txt
codefragments   Globals.x86     Rules_apps_f90.mk
feat2win.txt    Globals.x86_64  Rules_apps.mk
Globals.alpha   kernel          Rules_libs.mk
Globals.ia64    libraries       VERSIONS
Globals.mac     Makefile
Globals.mk      matlab
```

Getting FEATFLOW 2.0

- Unpack the downloaded archive file

```
$ tar xvzf Featflow2_2.0ALPHA.tar.gz
```

- Change into the base directory

```
$ cd Featflow2 ; ls
```

applications	Globals.power	object
bin	Globals.sparc	readme.txt
codefragments	Globals.x86	Rules_apps_f90.mk
feat2win.txt	Globals.x86_64	Rules_apps.mk
Globals.alpha	kernel	Rules_libs.mk
Globals.ia64	libraries	VERSIONS
Globals.mac	Makefile	
Globals.mk	matlab	

Getting FEATFLOW 2.0

- Unpack the downloaded archive file
  `$ tar xvzf Featflow2_2.0ALPHA.tar.gz`

- Change into the base directory
  `$ cd Featflow2 ; ls`

```
applications    Globals.power    object
bin             Globals.sparc    readme.txt
codefragments   Globals.x86      Rules_apps_f90.mk
feat2win.txt    Globals.x86_64   Rules_apps.mk
Globals.alpha   kernel           Rules_libs.mk
Globals.ia64    libraries        VERSIONS
Globals.mac     Makefile
Globals.mk      matlab
```

Top-level build options

```
$ make [ALT=xxx] [ID=yyy] <target>
```

- Some values for `target`

`help`	- print additional help and further option
`id`	- print out settings for current ID
`all`	- compile all libraries and application modules
`apps`	- compile all application modules
`libs`	- compile all libraries

Top-level build options

```
$ make [ALT=xxx] [ID=yyy] <target>
```

- Some values for target

`help`	- print additional help and further option
`id`	- print out settings for current ID
`all`	- compile all libraries and application modules
`apps`	- compile all application modules
`libs`	- compile all libraries

- Some available make modifiers

`ALT=xxx`	- specify alternative ID-xxx to use
`ID=yyy`	- override the autodetected architecture ID by yyy

Building FEATFLOW 2.0, cont'd.

technische universität
dortmund

Understanding the ALT/ID concept

$ make id   run on x86_64 GNU/Linux

```
Machine-ID (Barracuda) : pc64-core2-linux

Compilers to be used:
  C compiler:         /usr/bin/gcc
  C++ compiler:       /usr/bin/g++
  Fortran compiler:   /usr/local/g95/32bit_integers/0.91/bin/g95
  F-Library archiver: /usr/bin/ar
  C-Library archiver: /usr/bin/ar

Flags to be used:
  OPTFLAGS        = -O3 -m64 -ffast-math -fexpensive-optimizations -fprefetch-loop-arrays -mmmx -msse -msse2 -msse3
  OPTFLAGSC       =
  OPTFLAGSCPP     =
  OPTFLAGSF       =
  OPTFLAGSDEBUG   = -g
  OPTFLAGSCDEBUG  =
  OPTFLAGSCPPDEBUG=
  OPTFLAGSFDEBUG  = -O0 -g -Wall -fbounds-check -ftrace=full
  FCFLAGS         = -pipe -fmod= -march=nocona
  CCFLAGS         = -pipe -march=nocona
  CPPFLAGS        = -pipe -march=nocona
  BUILDLIB        = feat3d feat2d sysutils umfpack2 amd umfpack4 minisplib lapack blas
  BLASLIB         = (standard BLAS, included in installation package)
  LAPACKLIB       = (standard LAPACK, included in installation package)
  LDLIBS          =
  LDFLAGS         =
```

Building FEATFLOW 2.0, cont'd.

technische universität dortmund

Understanding the ALT/ID concept

$ make ID=pc-core2-linux id   run on x86_64 GNU/Linux

```
Machine-ID (Barracuda) : pc-core2-linux

Compilers to be used:
  C compiler:         /usr/bin/gcc
  C++ compiler:       /usr/bin/g++
  Fortran compiler:   /usr/local/g95/32bit_integers/0.91/bin/g95
  F-Library archiver: /usr/bin/ar
  C-Library archiver: /usr/bin/ar

Flags to be used:
  OPTFLAGS         = -O3 -m32 -ffast-math -fexpensive-optimizations -fprefetch-loop-arrays -mmmx -msse -msse2 -msse3
  OPTFLAGSC        =
  OPTFLAGSCPP      =
  OPTFLAGSF        =
  OPTFLAGSDEBUG    = -g
  OPTFLAGSCDEBUG   =
  OPTFLAGSCPPDEBUG=
  OPTFLAGSFDEBUG   = -O0 -g -Wall -fbounds-check -ftrace=full
  FCFLAGS          = -pipe -fmod= -march=nocona
  CCFLAGS          = -pipe -march=nocona
  CPPFLAGS         = -pipe -march=nocona
  BUILDLIB         = feat3d feat2d sysutils umfpack2 amd umfpack4 minisplib lapack blas
  BLASLIB          = (standard BLAS, included in installation package)
  LAPACKLIB        = (standard LAPACK, included in installation package)
  LDLIBS           =
  LDFLAGS          =
```

Building FeatFlow 2.0, cont'd.

Understanding the ALT/ID concept

`$ make ALT=ifc id` run on x86_64 GNU/Linux

```
Machine-ID (Barracuda) : pc64-core2-linux-ifc

Compilers to be used:
  C compiler:         /usr/local/intel/cce/10.1.021/bin/icc
  C++ compiler:       /usr/local/intel/cce/10.1.021/bin/icpc
  Fortran compiler:   /usr/local/intel/fce/10.1.021/bin/ifort
  F-Library archiver: /usr/local/intel/fce/10.1.021/bin/xiar
  C-Library archiver: /usr/local/intel/fce/10.1.021/bin/xiar

Flags to be used:
  OPTFLAGS         = -O3 -ipo -xT
  OPTFLAGSC        =
  OPTFLAGSCPP      =
  OPTFLAGSF        =
  OPTFLAGSDEBUG    = -g
  OPTFLAGSCDEBUG   = -traceback
  OPTFLAGSCPPDEBUG= -traceback
  OPTFLAGSFDEBUG   = -warn all -check all -traceback
  FCFLAGS          = -cm -fpe0 -vec-report0 -module
  CCFLAGS          = -vec-report0
  CPPFLAGS         = -vec-report0
  BUILDLIB         = feat3d feat2d sysutils umfpack2 amd umfpack4 minisplib lapack blas
  BLASLIB          = (standard BLAS, included in installation package)
  LAPACKLIB        = (standard LAPACK, included in installation package)
  LDLIBS           =
  LDFLAGS          = -lsvml
```

Building FeatFlow 2.0, cont'd.

technische universität
dortmund

Understanding the ALT/ID concept

```
$ make [ALT=xxx] [ID=yyy] <target>
```

- compiler settings are defined in the global configuration files `Global.[alpha,ia64,mac,power,sparc,x86,x86_64]`
- new compilers and/or architectures can be easily included
- special purpose settings, e.g. `pc64-opteron-linux-ifclarge`

Building FeatFlow 2.0, cont'd.

Understanding the `ALT/ID` concept

```
$ make [ALT=xxx] [ID=yyy] <target>
```

- compiler settings are defined in the global configuration files `Global.[alpha,ia64,mac,power,sparc,x86,x86_64]`
- new compilers and/or architectures can be easily included
- special purpose settings, e.g. `pc64-opteron-linux-ifclarge`

Building the Poisson example application

```
$ cd applications/poisson
$ make
```

... after some time ...

```
Done, poisson-pc64-core2-linux is ready.
```

Building FeatFlow 2.0, cont'd.

Understanding the ALT/ID concept

```
$ make [ALT=xxx] [ID=yyy] <target>
```

- compiler settings are defined in the global configuration files `Global.[alpha,ia64,mac,power,sparc,x86,x86_64]`
- new compilers and/or architectures can be easily included
- special purpose settings, e.g. `pc64-opteron-linux-ifclarge`

Building the Poisson example application

```
$ cd applications/poisson
$ make debug
```

turn on debugging facilities of the compiler

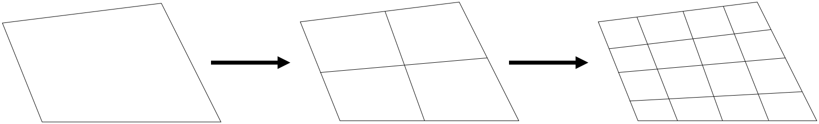
... after some time ...

```
Done, poisson-pc64-core2-linux is ready.
```

Preprocessing: *Grid generation*

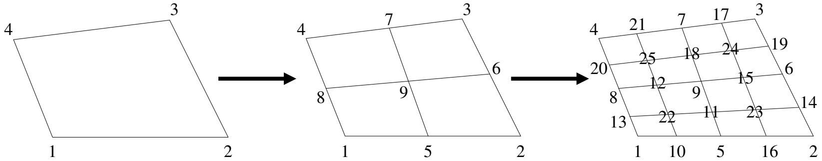
Geometric multigrid approach

1. Construct initial coarse grid in the external preprocessing step
2. Generate hierarchy of regularly refined meshes in the application



Geometric multigrid approach

1. Construct initial coarse grid in the external preprocessing step
2. Generate hierarchy of regularly refined meshes in the application



Two-level ordering strategy

- adopt all coordinates from coarser grid levels unchanged
- introduce new coordinates at edge/face/cell midpoints

Coarse grid description

- File format is described in the FEAT2D/FEAT3D manuals
- Supported element types: triangles, quads (2D) and hexahedra (3D)

Coarse grid description

- File format is described in the FEAT2D/FEAT3D manuals
- Supported element types: triangles, quads (2D) and hexahedra (3D)
- Domain triangulation is specified in `TRI` file (2D/3D)
  - coordinate values of vertices in the interior
  - parameter values of vertices at the boundary
  - elements in terms of their corner nodes
  - first two lines are treated as header/comments!

Coarse grid description

- File format is described in the FEAT2D/FEAT3D manuals
- Supported element types: triangles, quads (2D) and hexahedra (3D)
- Domain triangulation is specified in `TRI` file (2D/3D)
  - coordinate values of vertices in the interior
  - parameter values of vertices at the boundary
  - elements in terms of their corner nodes
  - first two lines are treated as header/comments!
- Boundary parametrization is specified in `PRM` file (only 2D)
  - each boundary component is described by $p \in [0, p_{\max}]$
  - the 'interior' is located left to the boundary
    $\rightarrow$ do not mix up (counter-)clockwise orientation
  - supported boundary types: lines, (arcs of) circles

DeViSoR Grid3D

Coarse grid generator for FEATFLOW and FEAST

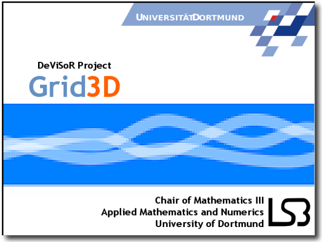
- written in Java + OpenGL and published under the GPL
- available at http://www.feast.uni-dortmund.de
- send requests, bug reports to devisor@featflow.de
- on-line help system and self-contained tutorial included

- Unpack the downloaded archive file

```
$ unzip grid-3.0.21.zip
```

- Start the application

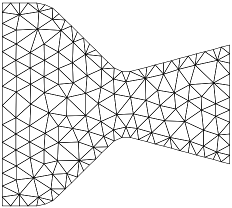
```
$ cd grid-3.0.21
$ java -jar grid3d.jar
```

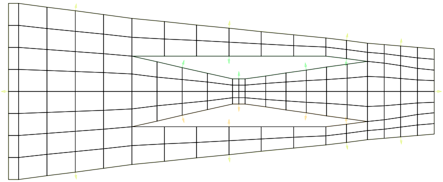


Alternative grid generators

- GiD – the personal pre- and post-processor
  - evaluation version is available at http://gid.cimne.upc.es
  - fully automatic structured and unstructured coarse grid generator
  - supports triangular, quadrilateral, tetrahedral, hexahedral elements
  - provides an effective easy-to-use and geometric user interface

- GiD2Feat – set of tools to convert GiD meshes to PRM/TRI files





Example application: *Poisson equation*

Possion equation

technische universität dortmund

- Change into the application source directory

```
$ cd applications/poisson/src; ls
```

```
poissonXd_callback.f90  poisson.f90
poissonXd_methodYYY.f90
```

- Open the application main source file

```
$ emacs poisson.f90
```

Possion equation

- Change into the application source directory
  ```
  $ cd applications/poisson/src; ls
  ```
  ```
  poissonXd_callback.f90  poisson.f90
  poissonXd_methodYYY.f90
  ```

- Open the application main source file
  ```
  $ emacs poisson.f90
  ```

Initialization and finalization

- `system_init()` initialize system-wide settings
- `storage_init(999, 100)` initialize storage management
- `storage_done()` finalize storage management

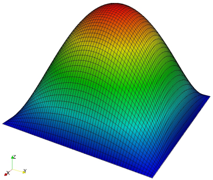
Possion equation, cont'd.

technische universität dortmund

Sample problem: $-\Delta u = f$ in $\Omega = (0,1)^2$, $u = 0$ on $\partial\Omega$

- right hand side $f(x,y) = 32(x(1-x) + y(1-y))$
- analytical solution $u(x,y) = 16x(1-x)y(1-y)$

- Open the demonstration module file
  `$ emacs poisson2d_method0_simple.f90`
  - contains the corresponding subroutine
  - includes all required kernel modules
  - provides detailed step-by-step tutorial
- Open the callback function module file
  `$ emacs poisson2d_callback.f90`



Possion equation, cont'd.

Grid generation

- `boundary_read_prm` read boundary parametrization
- `tria_readTriFile2D` read domain triangulation
- `tria_quickRefine2LevelOrdering` perform regular refinement
- `tria_initStandardMeshFromRaw` generate data structures

Spatial discretization

- `spdiscr_initBlockDiscr2D` prepare block discretization
- `spdiscr_initDiscr_simple` initialize spatial discretization
- `bilf_createMatrixStructure` create scalar matrix structure
- `bilf_buildMatrixScalar` discretize the bilinear form
- `linf_buildVectorScalar` discretize the linear form/r.h.s.

Possion equation, cont'd.

Dirichlet boundary conditions

- `boundary_createRegion` specify boundary segment
- `bcasm_newDirichletBConRealBD` calculate discrete b.c.'s
- `matfil_discreteBC` set b.c.'s in system matrix
- `vecfil_discreteBCrhs` set b.c.'s in right hand side
- `vecfil_discreteBCsol` set b.c.'s in solution vector

Linear BiCGStab solver

- `linsol_initBiCGStab` initialize linear solver
- `linsol_setMatrices` attach system matrix
- `linsol_initStructure`, `linsol_initData`
- `linsol_solveAdaptively` solve linear system

Possion equation, cont'd.

Solution output

- `ucd_startGMV` start export on GMV format
- `ucd_addVariableVertexBased` add vertex-based solution data
- `ucd_addVariableElementBased` add cell-based solution data
- `ucd_write` write solution data to file

Clean-up and finalization

- `XXX_release`, `XXX_releaseYYY` free allocated memory
- `XXX_done` stop sub-system

General naming convention of subroutines

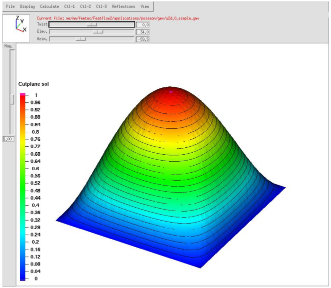
- `abbreviatedModulefile_NameOfSubroutine`

Postprocessing: *Visualization of the solution*

3D scientific visualization tool

- developed at Los Alamos National Lab by Frank Ortega
- available at http://www-xdiv.lanl.gov/XCM/gmv/
- supported OS: UNIX/Linux, Mac OS X, Windows (Cygwin)

- unstructured meshes in 2D/3D
- cutlines, cutplanes, cutspheres
- vertex-based, cell-based data sets
- contour, vector plots



Alternative: importer for ParaView based on development CVS-version

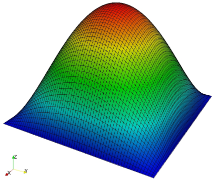
Advanced topics: *Multigrid, Mesh Adaptation, ...*

Possion equation, revisited

Sample problem: $-\Delta u = f \quad \text{in } \Omega = (0,1)^2, \quad u = 0 \quad \text{on } \partial\Omega$

- right hand side $\quad f(x,y) = 32(x(1-x) + y(1-y))$
- analytical solution $\quad u(x,y) = 16x(1-x)y(1-y)$

- Open the demonstration module file
  $ emacs poisson2d_method1_mg.f90
  - linear geometric multigrid solver
  - Jacobi or ILU(0) smoother
  - direct coarse grid solver (UMFPACK)



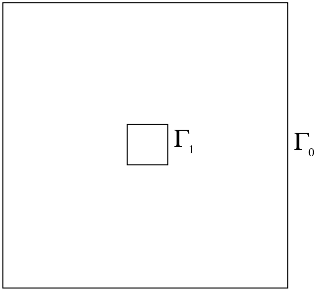
Anisotropic diffusion

Example: `applications/anisotropicdiffusion.f90`

$$\mathcal{D} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

$$\begin{cases} -\nabla \cdot (\mathcal{D}\nabla u) = 0 & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_0 \\ u = 2 & \text{on } \Gamma_1 \end{cases}$$

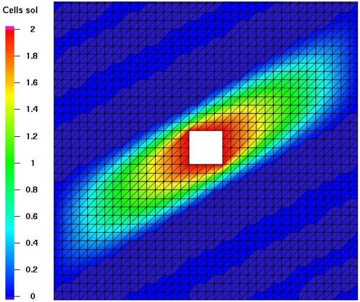
- $k_1 = 100, \quad k_2 = 1, \quad \theta = -\frac{\pi}{6}$
- linear finite elements, $h = 1/36$
- Galerkin fails: $u^{\min} = -0.0553$
- h-adaptation: $u^{\min} = -0.0068$

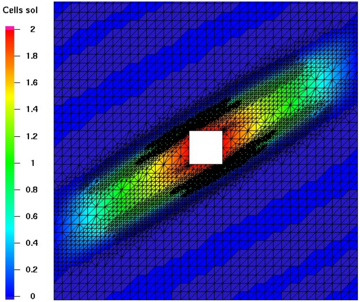


Anisotropic diffusion

Example: `applications/anisotropicdiffusion.f90`

$$\mathcal{D} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$





Mesh adaptation

- conformal mesh refinement based on red-green strategy
- vertex-locking algorithm for mesh re-coarsening procedure
- nodal generation function stores birth certificates
  - provides complete characterization of elements
  - youngest node corresponds to refinement level
  - is required for the vertex-locking algorithm
- state function for element characterization
- local two-level ordering strategy

Mesh adaptation

technische universität dortmund

- conformal mesh refinement based on red-green strategy
- vertex-locking algorithm for mesh re-coarsening procedure
- nodal generation function stores birth certificates
  - provides complete characterization of elements
  - youngest node corresponds to refinement level
  - is required for the vertex-locking algorithm
- state function for element characterization
- local two-level ordering strategy

This has been addressed in the talk on Monday

Mesh adaptation

- conformal mesh refinement based on red-green strategy
- vertex-locking algorithm for mesh re-coarsening procedure
- nodal generation function stores birth certificates
    - provides complete characterization of elements
    - youngest node corresponds to refinement level
    - is required for the vertex-locking algorithm
- state function for element characterization
- local two-level ordering strategy

Mesh genealogy, revisited

Triangulation $\mathcal{T}_m(\mathcal{E}_m, \mathcal{V}_m)$, $m = 0, 1, 2, \ldots$ consists of

$$\mathcal{E}_m = \{\Omega_k \,:\, k = 1, \ldots, N_{\mathrm{E}}\} \quad \text{and} \quad \mathcal{V}_m = \{\mathrm{v}_i \,:\, i = 1, \ldots, N_{\mathrm{V}}\}$$

- nodal **generation function** $g : \mathcal{V}_m \rightarrow \mathbb{N}_0$ is defined recursively

$$g(\mathrm{v}_i) := \begin{cases} 0 & \text{if} \quad \mathrm{v}_i \in \mathcal{V}_0 \\ \max\limits_{\mathrm{v}_j \in \Gamma_{kl}} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Gamma_{kl} := \bar{\Omega}_k \cap \bar{\Omega}_l \\ \max\limits_{\mathrm{v}_j \in \partial\Omega_k} g(\mathrm{v}_j) + 1 & \text{if} \quad \mathrm{v}_i \in \Omega_k \setminus \partial\Omega_k \end{cases}$$

Characterization of elements

technische universität dortmund

Idea I: State function $s : \mathcal{E}_m \to \mathbb{Z}$ (in MSB representation)

- Set `Bit[0]` to 1 for quadrilateral, otherwise set it to zero
- Set `Bit[k=1..4]` to 1 if both endpoints of edge k have same age
- If no two endpoints have same age, then find local position k of the youngest vertex, set `Bit[k]` to 1 and negate the state

Idea II: Define local ordering strategy within each element *a priori*

element characterization	element state
triangle/quadrilateral from $\mathcal{T}_0$	0/1
green quadrilateral	3, 5, 9, 11,17, 21
red quadrilateral	7, 13, 19, 25
inner red triangle	14
'other' triangle	2, 4, 8
green triangle	-8, -4, -2

Characterization of elements

technische universität dortmund

Idea I: State function $s : \mathcal{E}_m \to \mathbb{Z}$ (in MSB representation)

- Set `Bit[0]` to 1 for quadrilateral, otherwise set it to zero
- Set `Bit[k=1..4]` to 1 if both endpoints of edge k have same age
- If no two endpoints have same age, then find local position k of the youngest vertex, set `Bit[k]` to 1 and negate the state

Idea II: Define local ordering strategy within each element *a priori*

element characterization	element state
triangle/quadrilateral from $\mathcal{T}_0$	0/1
green quadrilateral	3, 5, 9, 11,17, 21
red quadrilateral	7, 13, 19, 25
inner red triangle	14
'other' triangle	2, 4, 8
green triangle	-8, -4, -2

Characterization of elements

technische universität dortmund

Idea I: State function $s : \mathcal{E}_m \to \mathbb{Z}$ (in MSB representation)

- Set `Bit[0]` to 1 for quadrilateral, otherwise set it to zero
- Set `Bit[k=1..4]` to 1 if both endpoints of edge k have same age
- If no two endpoints have same age, then find local position k of the youngest vertex, set `Bit[k]` to 1 and negate the state

Idea II: Define local ordering strategy within each element *a priori*

element characterization	element state
triangle/quadrilateral from $\mathcal{T}_0$	0/1
green quadrilateral	3, 5, 9, 11,17, 21
red quadrilateral	7, 13, 19, 25
inner red triangle	14
'other' triangle	2, 4, 8
green triangle	-8, -4, -2

Characterization of elements

technische universität dortmund

Idea I: State function $s : \mathcal{E}_m \to \mathbb{Z}$ (in MSB representation)

- Set `Bit[0]` to 1 for quadrilateral, otherwise set it to zero
- Set `Bit[k=1..4]` to 1 if both endpoints of edge k have same age
- If no two endpoints have same age, then find local position k of the youngest vertex, set `Bit[k]` to 1 and negate the state

Idea II: Define local ordering strategy within each element *a priori*

element characterization	element state
triangle/quadrilateral from $\mathcal{T}_0$	0/1
green quadrilateral	3, 5, 9, 11,17, 21
red quadrilateral	7, 13, 19, 25
inner red triangle	14
'other' triangle	2, 4, 8
green triangle	-8, -4, -2

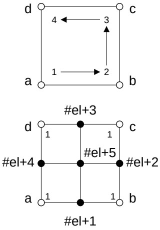
Characterization of elements

Idea I: State function $s : \mathcal{E}_m \to \mathbb{Z}$ (in MSB representation)

- Set `Bit[0]` to 1 for quadrilateral, otherwise set it to zero
- Set `Bit[k=1..4]` to 1 if both endpoints of edge k have same age
- If no two endpoints have same age, then find local position k of the youngest vertex, set `Bit[k]` to 1 and negate the state

Idea II: Define local ordering strategy within each element *a priori*

element characterization	element state
triangle/quadrilateral from $\mathcal{T}_0$	0/1
green quadrilateral	5, 21
red quadrilateral	13
inner red triangle	14
'other' triangle	4
green triangle	-8, -4, -2



Storage management

technische universität dortmund

Concept of dynamically allocatable arrays in Fortran 9x

- `integer, dimension(:), allocatable :: Iarray`
- `allocate(Iarray(n))` allocate array of size `n` dynamically
- `deallocate(Iarray)` deallocate dynamically allocated array

Storage management

technische universität dortmund

Concept of dynamically allocatable arrays in Fortran 9x

- `integer, dimension(:), allocatable :: Iarray`
- `allocate(Iarray(n))` — allocate array of size `n` dynamically
- `deallocate(Iarray)` — deallocate dynamically allocated array

Concept of dynamically allocatable arrays in FeatFlow

- `storage_init`, `storage_done` — initialize/finalize storage
- `storage_new` — allocate new memory storage
- `storage_realloc` — re-allocate memory storage
- `storage_free` — deallocate memory storage
- `storage_getbase_XXX` — access memory storage

Storage management, cont'd.

- Supported data: 8/16/32/64 integer, SP/DP real, logical, strings
- Memory storage is accessible via integer handle `ihandle`
- Pointer to the memory is assigned via `storage_getbase_XXX`

Implementation details

- memory storage handling is internally mapped to de/allocate
- different storage management can be implemented without changes
- handles can be easily passed to subroutines and functions
- derived types typically consist of a set of handles + scalar data

Storage management, cont'd.

technische universität dortmund

Impl

ges

Derived type for triangulation structure

```
type t_triangulation
   integer :: ndim = 0
   integer :: NVT = 0
   integer :: NMT = 0
   integer :: NAT = 0
   integer :: NEL = 0
   . . .
   integer :: h_DvertexCoords = ST_NOHANDLE
   integer :: h_IverticesAtElement = ST_NOHANDLE
   integer :: h_IneighboursAtElement = ST_NOHANDLE
end type
```

Example: *Using handles*

technische universität dortmund

Create new storage

- `storage_new (scall, sname, Isize, ctype, ihandle, &`
  `cinitNewBlock, <rheap>)`
- `ctype` ∈ {`ST_DOUBLE`,`ST_SINGLE`,`ST_INTx`,`ST_CHAR`,`ST_LOGICAL`}
- `cinitNewBlock` ∈ {`ST_NEWBLOCK_ZERO`,`ST_NEWBLOCK_NOINIT`}

Accessing storage, e.g. 2-dimensional double array

- `real(DP), dimension(:,:), pointer :: p_Darray`
- `storage_getbase_double2D (ihandle, p_Darray, <rheap>)`

Releasing storage

- `storage_free (ihandle)`

UnConventional High Performance Computing

Finite **E**lement **A**nalysis & **S**olution **T**ools

High performance finite element package for the efficient simulation of large scale problems on heterogeneous hardware

- written in Fortran 90 and C (MPI communication)
- CFD (Stokes, Navier-Stokes), CSM (Elasticity)
- macro-wise domain decomposition approach
- fast and robust parallel multigrid methods
- unconventional hardware as FEM co-processors

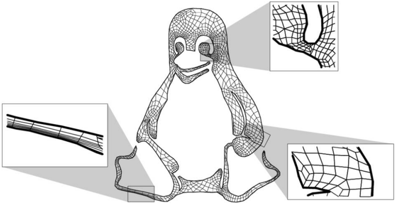
Visit the Feast homepage

http://www.feast.uni-dortmund.de

Scalable **R**ecursive **C**lustering (ScaRC) solver

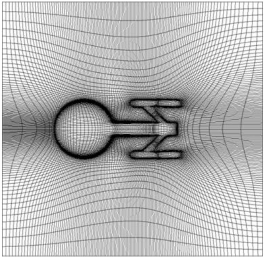
Combine domain decomposition and cascaded multigrid methods

Globally unstructured



hide anisotropies to improve robustness

Locally structured



UnConventional HPC

- Graphic processing unit (GPU)
  128 parallel scalar processors
  @ 1.35 GHz, $\approx$ 350 GFLOP/s
  GDDR3 memory @ 900 MHz

- Cell multi-core processor (PS3)
  7 synergistic processing units
  @ 3.2 GHz, 218 GFLOPS/s
  Memory @ 3.2 GHz

Future vision

technische universität dortmund

Unified FEAT+FEAST finite element package

- Decompose the domain into **globally** unstructured macro-cells
- Use generalized tensor product grid or unstructured mesh **locally**
- Reuse FEM co-processors in the FEATFLOW package
- Enable special features, (e.g. h-adaptation) per macro-cell

On-line resources and additional material

- FeatFlow project: `http://www.featflow.de`
- Feast project: `http://www.feast.uni-dortmund.de`