Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 314159

Development of a FEM code for fluid-structure coupling

Stephan Herb

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Miriam Mehl
Betreuer/in:	Dipl.-Ing. Florian Lindner
Beginn am:	2015-06-08
Beendet am:	2015-12-08
CR-Nummer:	

Contents

1 Introduction

Here comes the introduction. And before that the abstract (that needs to be put into LaTeX as special paragraph)

2 Framework Evaluation

Part of the thesis was to find several frameworks which ease the work with the finite element method. An evaluation of these frameworks was done to select a suitable one for the given task. The evaluation's criteria are presented in this chapter as well as a short description of the studied frameworks.

2.1 General Aspects

In preparation of evaluating the frameworks many criteria were created to objectify the search for the most suitable. The individual aspects were as follows:

- Open Source: All frameworks under consideration need to be published under the GNU Lesser General Public License or similar license that allows modification and/or redistribution.
- Parallelization: In order to accelerate the calculations the framework has to be able to support the widely used Message Passing Interface (MPI).
- The programming language was chosen to be C++. Therefore the framework has to be written in this language.
- Mesh file import: Common mesh files types like gmsh or xda/xdr must be able to load by the framework. Simultaneously the framework must support finite elements like triangles and quadrilaterals with three and four nodes respectively and be able to handle two dimensional elements defined within three dimensional space.
- The framework should handle different types of boundary conditions defined in a mesh file.
- Built-in solvers: In order so solve the matrix-vector-system the framework must provide a variety of different iterative solvers.
- Convenience functions: To optimize the calculations the framework should make use of functions to get matrix-vector and matrix-matrix products, transpose matrix or sparse-matrices.
- Accessible and detailed documentation: In order to guarantee maintainability and expandability the framework has to have a good documentation itself.
- Up-to-date: The framework should be well maintained and actively supported by its developers to ensure a long term compatibility with possible new features of the thesis' code.
- The framework should be used by at least a few projects. This shows the framework's importance and usability.
- Easy-to-learn syntax and structure: A rather subjective aspect but an important one. The limited time for the thesis does not allow to study highly complicated structures or semantics. This accompanies the documentation aspect.

2.2 Frameworks Overview

The following list contains FEM libraries and frameworks which were evaluated.

2.2.1 Feel++

- "Feel++ is a unified C++ implementation of Galerkin methods (finite and spectral element methods) in 1D, 2D and 3D to solve partial differential equations."[Con]
- creation of versatile mathematical kernels allow testing and comparing different techniques and methods in solving problems
- focus on close mathematical abstractions regarding partial differential equations (PDE)
- [PCD$^+$12] - imports e.g. gmsh mesh files
- seamlessly parallel with mpi
- currently used in projects at Cemosis (Center for Modeling and Simulation in Strasbourg, France) including fluid structure interactions, high field magnets simulation, or, optical tomography
- actively developed, last major release were on February 2015

2.2.2 OOFEM

[Pat09] - Object Oriented Finite Element Solver (OOFEM) - actively developed with latest release from February 2014
- object oriented architecture; extensible in terms of new element types, boundary conditions or numerical algorithms
- modules for structural mechanics, transport problems and fluid dynamics
- focuses on efficient and robust solution to mechanical, transport and fluid problems
- written in C++ with focus on portability
- interfaces to various external software libraries like PETSc, ParMETIS, or, ParaView
- is used in several publications [Pat]

2.2.3 GetFEM++

- latest release from July 2015 - framework for solving potentially coupled systems of linear and nonlinear PDE
- written in C++ but provides interfaces to languages like Python and Matlab
- model description that gather the variables, data and terms of a problem and some predefined bricks representing classical models
- easy switching from one method to another due to separation of geometric transformation, integration methods, and, finite element method
- can be used to construct generic finite element codes, where methods and the problem's dimension can be changed very easily
- uses MPI for parallelization, though it is stated that "a certain number of procedures are still remaining sequential" [SPR] - imports e.g. gmsh mesh files
- used in project like IceTools [Jar] (open source model for glaciers), EChem++ [BLSS]

(Problem Solving Environment for Electochemistry) and SimNIBS [Thi] (software for Simulation of Non-invasive Brain Stimulation)

2.2.4 MFEM

[mfeb] - The Modular Finite Element Method (MFEM) library acts as a toolbox that provides the building blocks for developing finite element algorithms
- it has a wide range of mesh types, e.g. triangular and quadrilateral 2D elements, curved boundary elements or topologically periodic meshes
- supports MPI-based parallelism throughout the library
- variety of built-in solvers
- written in highly portable C++ and extensible due to separation of mesh, finite element and linear algebra abstractions
- hypre library is tightly integrated within MFEM, for example the use of high-performance preconditioners - The object oriented design of the library as well as the separation of the different parts of the library like the mesh functions, the finite elements, and, the linear algebra, focusing on adapt the code to a variety of applications - use in several publications [mfea]

2.2.5 libMesh

[KPSC06] - actively developed and active user community
- wide variety of mesh file formats to import from (e.g. gmsh, vtk, xda,)
- seamlessly integrated parallel functionality with MPI
- seamlessly interfaces optional external libraries like PETSc or ParMETIS
- complete documentation and documented source code available
- "framework for the numerical simulation of PDE using arbitrary unstructured discretizations on serial and parallel platforms".
- "provide support for adaptive mesh refinement (AMR) computations in parallel"
- supports a variety of 1D, 2D, and 3D geometric and finite element types
- created at The University of Texas at Austin in the CFDLab in March 2002. Contributions have come from developers at the Technische Universität Hamburg-Harburg Institute of Modelling and Computation, CFDLab associates at the PECOS Center at UT-Austin, the Computational Frameworks Group at Idaho National Laboratory, NASA Lyndon B. Johnson Space Center, and MIT.

3 Shell Elements

Mathematical fundamentals of shell elements divided into the two parts and the coordinate transformation

3.1 Introduction to Linear Elasticity Problems

- [Ste15] ch3.1 (S.61)
- Einführung von Vektoren und Matrizen (Verschiebungsvektor, Tensoren bzw. Vektoren der Dehnungen und Spannungen)
- Verknüpfung der Versch. mit den Dehnungen + Stoffgesetz (kinematische Beziehung)
- ???

3.2 Plane Element

First part of shell element: plane part. derivation of this part with two exemplary finite element types

3.2.1 Problem Definition

- Bild ähnlich ch7.1 mit xy-Ausdehnung, Dicke t, Mittelfläche, Rand, KoSys; Streckenlast q0 und Volumenkraft g aber weglassen
- Bed. für ebenen Spannungszustand (eb. Dehnungszustand erwähnen und auf Ref (z.B. Zienkiewicz) verweisen)
- Verschiebungen, Dehnungen und Spannungen beschreiben + kinematische Beziehung + Stoffgleichung
- Randbedingungen: q0 Streckenlast beschreiben, aber angeben, dass sie hier nicht beachtet werden, punktf. Ladungen werden hier benutzt
- resultierendes Gesamtpotential ohne b und q

3.2.2 Tri-3 Plane Element

mathematical derivation of three node triangular plane element discretization
see Steinke [Ste15] page 215-221
- Bild von trianguliertem Körper
- ansatzfunktion (abgewandelt statt phi, u und v separat)
- interpolationsbedingunen führen auf Formfunktionen
- Dehnungs-Verschiebungs-Beziehung -> führt zu B + Spannungs-Verschiebungs-Beziehung
- Einsetzen in Gesamtpotential + Variation
- Bestimmung von K kürzen: wichtig ist nur s.221 K = t*int(H*dA,A) mit dV = t*dA

3.2.3 Quad-4 Plane Element

- mathematical derivation of four node quadrilateral plane element
- see Steinke [Ste15] page 237-250 + Cook [CMPW02] page 202-208
- isoparametrisches viereckselement (7.5.1) + bild

- bild von original- und bildebene
- eigenschaften der ansatzfunktion + formfunktionen
- verschiebungen von uv mit formfunktionen und u darstellen
- jacobi-matrix aufstellen, inverse und determinante nicht so genau (bzw. einfach, weil J eh nur 2x2 ist)
- dehnungs-verschiebungs-beziehung AUS ANDERER QUELLE, DA IN [Ste15] UNZUFRIEDENSTELLEND BESCHRIEBEN
- stefigkeitsmatrix $K = int_V(B^T DBdV) = t*int_V(B^T DBdA) = t*int\_1^1 int\_1^1(B^T DB|J|dsdr)$
+ erwähnen, dass |J| Flächeninhalt angibt (ref finden)

3.3 Plate Bending Element

Second part of shell element: plate part. derivation of this part with two exemplary finite element types

3.3.1 Problem Definition

[Ste15] ch8.1
- bild wie bei 3.2.1
- hier wird die kirchhoff-platten-theorie verwendet
- voraussetzungen der kirchhoff-platte
- größen der platte: durchbiegung w + verdrehungen dw/dx bzw. dw/dy
- dehnungs-verschiebungs-beziehung
- stoffgleichung (krümmungs-momenten-beziehung) -> führt auf Dp und Querkräfte Qx,Qy
- gleichgewichtsbeziehung der platte (p ist flächenlast; näher untersuchen)
- auf verschiedene randbedingungen der platte eingehen
- funktional der platte
- forderungen an plattenelement

3.3.2 Tri-3 Plate Element

[Ste15][Spe88] - mathematical derivation of three node triangular plane element
- see Steinke [Ste15] page 275-282 + [Spe88] for reference of element
- 8.4.4 beschreibt probleme mit einigen dreiecksplattenelementen, deshalb für meine arbeit element von specht [Spe88]
- ansatzfunktion nach specht
- interpolationsbedingungen
- entwicklung der formfunktionen
- krümmungs-verschiebungs-beziehung
- steifigkeitsmatrix

3.3.3 Quad-4 Plate Element

- mathematical derivation of four node quadrilateral plane element
- see Zienkiewicz [ZT00] page 174-177,219-222 [ZTZT77] ??? **[BT82] page 1658-1662**
- formulierung als DKQ element
- **!! formatierung an meine anpassen, da sehr unterschiedlich zu anderen referenzen !!**
- dkq basiert auf diskretisierung der stress-energie, unter vernachlässigung der transverse shear strain energy (gl.(1) s.4)
- krümmungsvektor, Db angeben
- considerations für dkq formulierung
- formfunktionen, koeffizienten
- jacobi-matrix, -inverse, determinante
- krümmungs-verschiebungs-beziehung
- steifigkeitsmatrix

3.4 Coordinate Transformation

erster Entwurf
see [NTRNXB08]
The nodes and elements in the mesh are defined in global three dimensional space. The elements need to be transformed into local two dimensional space in order to be able to calculate their local stiffness matrix. This local stiffness matrix must then be transformed back into the global system. Transform arbitrary 3D triangle onto xy-Plane:
- given triangle with vertices $A = (a_x, a_y, a_z), B = (b_x, b_y, b_z)$ and $C = (c_x, c_y, c_z)$ ordered in counter-clockwise direction
- let U be the vector from node A to B: $U = B - A = (b_x - a_x, b_y - a_y, b_z - a_z)$ and let V be the vector from node A to C: $V = C - A = (c_x - a_x, c_y - a_y, c_z - a_z)$
- First local unit vector $\tilde{x} = \frac{1}{|U|}U$
- Second local unit vector $\tilde{z} = U \times V \longrightarrow \tilde{z} = \frac{1}{\tilde{z}}\tilde{z}$
- Third local unit vector $\tilde{y} = \tilde{z} \times \tilde{x}$
- Define transformation matrix T as follows:

$$T = \begin{pmatrix} \tilde{x}^T \\ \tilde{y}^T \\ \tilde{z}^T \end{pmatrix} = \begin{pmatrix} \tilde{x}_x & \tilde{x}_y & \tilde{x}_z \\ \tilde{y}_x & \tilde{y}_y & \tilde{y}_z \\ \tilde{z}_x & \tilde{z}_y & \tilde{z}_z \end{pmatrix}$$

- Assembly of element's stiffness needs derivatives. Therefore every triangle can be translated in such a way, that node A lies in the global origin.
- It follows: $\tilde{A} = (0\ 0\ 0)^T, \tilde{B} = \left(\tilde{b}_x\ 0\ 0\right)^T, \tilde{C} = (\tilde{c}_x\ \tilde{c}_y\ 0)^T$
- Node A will not be changed by the transformation with T, B will be projected onto the local x-axis due to the definition of it as the vector between A and B and C will be projected onto the local xy-plane.
- One can see that the z component vanishes by transforming into local space

- given quadrilateral with vertices $A = (a_x, a_y, a_z), B = (b_x, b_y, b_z), C = (c_x, c_y, c_z), D = (d_x, d_y, d_z)$ ordered in counter-clockwise direction
- let I be the midpoint of the edge AB as follows: $I = A + \frac{1}{2}(B - A)$. Analogously let J,K and L be the midpoints of the edges BC, CD and DA: $J = B + \frac{1}{2}(C - B), K = C + \frac{1}{2}(D - C), L = D + \frac{1}{2}(A - D)$
- let U be the vector from node L to J: $U = J - L = (j_x - l_x, j_y - l_y, j_z - l_z)$ and let V be the vector from node I to K: $V = K - I = (k_x - i_x, k_y - i_y, k_z - i_z)$
- First local unit vector $\tilde{x} = \frac{1}{\|U|}U$
- Second local unit vector $\tilde{z} = U \times V \longrightarrow \tilde{z} = \frac{1}{|\tilde{z}|}\tilde{z}$
- Third local unit vector $\tilde{y} = \tilde{z} \times \tilde{x}$
- Define transformation matrix T as follows:

$$T = \begin{pmatrix} \tilde{x}^T \\ \tilde{y}^T \\ \tilde{z}^T \end{pmatrix} = \begin{pmatrix} \tilde{x}_x & \tilde{x}_y & \tilde{x}_z \\ \tilde{y}_x & \tilde{y}_y & \tilde{y}_z \\ \tilde{z}_x & \tilde{z}_y & \tilde{z}_z \end{pmatrix}$$

3.5 Shell Element

The combination of the two previous parts and the transformations results in the final shell element
- bild wie bei 8.7 von scheibe und platte und kombination zu schale + erklärungen, welche unbekannten und kräfte man bei welchem teil hat
- erklärung, warum man hier einfach Plane und Plate unabhängig voneinander berechnen und dann zusammenwerfen darf
- gesamtsteifigkeitsmatrix besteht aus blöcken (3x3 bei tri3, 4x4 bei quad4), Ku=F (gleichung 718 bei [Ste15]) - Sei K_m die lokale Steifigkeitsmatrix vom Membran/Plane-Teil und K_p die vom Plate-bending-Teil
- Kij weisst in der Spalte theta_z und Zeile theta_z $(k_i j)_6 6$ eine null auf -> erklärung und warum schlecht. ANDERE REFERENZ ERKLÄRT, WAS WIR DESHALB MACHEN (1/1000 der diagonalwerte)
- Dann muss die (Rück-)Transformationsmatrix T erstellt werden, da SKM im lokalen KoSys definiert ist, aber in die globale Systemmatrix einsortiert werden muss
- Je nachdem ob 3 oder 4 Knotenelement (Tri-3/Quad-4) sieht K und T natürlich anders aus
- Wir transformieren blockweise von lokal nach global zurück $(K_{ij}, 1 \leq i, j \leq 3(4))$

4 FEM Code Implementation

contains development of the program code with focus on the assembly of the system and its solving, the process of parallelization and the coupling step with preCISE

4.1 Introduction to libMesh

was kann libMesh eigentlich alles; wo unterstützt es einen, was muss man selbst machen

4.2 libMesh FEM

details about the implementation with the libmesh FEM framework:
- initialization: loading of parameters, setting up libmesh (evtl. uninteressant und es kann weg, oder es muss noch mehr hier rein)
- mesh loading/import: wie sieht mesh file aus bzw. welche typen werden akzeptiert, welche ids für bcs müssen verwendet werden
- set up of system: erstellen des linearimplicitsystems, erstellen der variablen, der bcs, des solvers usw.
- assembly of system matrix and RHS: größter teil; hier wird auf die erstellung der lokalen und globalen stiffnessmatrix eingegangen (integral mit gauss-quadratur lösen z.B. [Ste15] s.248), das auslesen der forces und der entsprechende eintrag in der rhs gesetzt; das mitverfolgen der bereits bearbeiteten knoten mittels unordered_set usw.
- boundary conditions: eventuell bereits unter setup; grundsätzlich auf die beiden bc-typen eingehen, wie das in libmesh gelöst wurde
- solving and getting the result vector: das lösen an sich ist eine code-zeile. hier kann man aber schreiben, mit was libmesh umgehen kann an lösern, welche einstellmöglichkeiten es gibt (error-eps, #iters). und es geht drum, wie man an die tatsächlichen werte für die displacements kommt und was daraus am ende wird
- für die standalone-version noch ein absatz zur ausgabe in exodus2-file

4.3 Parallelization with MPI

additional steps to make the code ready for multi process execution with MPI
- viel ist es nicht, was man tun muss, damit libmesh mit mpi läuft
- grundsätzlich ist zum lösen des gleichungssystem mit mehreren prozessen petsc als externe lib notwendig
- am mesh muss nichts verändert werden, da libmesh automatisch eine partitionierung des meshes vornimmt (kann aber verbessert werden)
- damit rhs korrekt gesetzt wird muss über die prozessgrenzen hinweg klar sein, ob knoten bereits bearbeitet wurde oder nicht. wie das gelöst wurde kommt hier rein

5 Coupling with preCICE

still under construction; just a rough idea what this section will contain

5.1 Coupling

short introduction what coupling means (in this case)

5.1.1 Introduction to coupling

...

5.1.2 Coupling methods

...

5.2 preCICE

short introduction what preCICE is

5.3 Implementation

"modification" of the code to work with preCICE

6 Validation

The code was tested with several problems to validate its correctness and state where and why there are differing results to the existing (commercial) FEM codes

6.1 Test A: Membrane Displacement with Tri-3

Sprungbrett bestehend aus 8 Elementen; links fest eingespannt, rechts Kraft in y-Richtung an beiden Randknoten
Sprungbrett bestehend aus 32 Elementen in Fischgrätmuster angeordnet; selbe BCs aber andere Kraftwerte
test_c.xda, test_d.xda - beides korrekt

6.2 Test B: Membrane Displacement with Quad-4

Sprungbrett bestehend aus 3 Elementen mit BC wie in Test A aber einzelne Kraft auf oberen rechten Knoten in neg. y-Richtung
selbes Mesh wie in test_d.xda nur eben mit 16 Elementen. Selbe BCs, selbe Kraftwerte
test_e.xda, test_f.xda - beides korrekt

6.3 Test C: Plate Displacement with Tri-3

Platte an allen 4 Seiten eingespannt. Einzelne Kraft im Zentrum in neg. z-Richtung
Alternativ mit anderen Parametern test_g
test_a_triN.xda, test_g_triAB_N.xda - korrekt, noch nicht getestet

6.4 Test D: Plate Displacement with Quad-4

Selbes mesh wie Test C nur eben mit Quadelementen
test_a_quadN.xda, test_g_quad_N.xda - korrekt, noch nicht getestet

6.5 Test E: Shell Displacement with Tri-3

Ein H-Trägerbalken. Am einen Ende fest eingespannt. Am anderen Ende wird oben eine Kraft am äußeren Knoten in den Balken hinein in flacher Ebene gegeben, gleichzeitig wird unten an der gegenüberliegenden Seite eine Kraft in entgegengesetzter Richtung gegeben
test_j_tri.xda - korrekt

6.6 Test F: Shell Displacement with Quad-4

Gleich wie Test E nur eben Quadelemente
test_j_quad.xda - korrekt

6.7 Test G: Convergence (increasing number of elements)

??? theoretisch mit Test C/D bereits durchführbar mit N=2,4,8,16,32,64,128

6.8 Test H: MPI (increasing number of processes)

??? theoretisch alle Tests, z.B. E/F mit Prozessoranzahl = 1,2,4,8,16. In dem Fall ist natürlich die Zeit interessant und ob die Ergebnisse jeweils alle gleich sind

6.9 Test I: Coupling with preCICE

???

7 Conclusion

What does my code do, what problems arose, what problems persist, what does my code cannot do, where are opportunities for extensions, etc.

References

[BLSS] Martin Bogdan, Kai Ludwig, Elena Sapozhnikova, and Bernd Speiser. Echem++ - a problem solving environment for electrochemistry. http://www.echem.uni-tuebingen.de/echem/software/EChem++/echem++.shtml/.

[BT82] Jean-Louis Batoz and Mabrouk Ben Tahar. Evaluation of a new quadrilateral thin plate bending element. *International Journal for Numerical Methods in Engineering*, 18(11):1655–1677, 1982.

[CMPW02] Robert D. Cook, David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis*. J. Wiley, 2002.

[Con] Feel++ Consortium. Feel++ - finite element embedded library in c++. http://www.feelpp.org.

[Jar] Alexander H. Jarosch. icetools: a numerical ice flow model download. http://sourceforge.net/projects/icetools/.

[KPSC06] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006. http://dx.doi.org/10.1007/s00366-006-0049-3.

[mfea] Mfem - finite element discretization library. http://mfem.org/publications/.

[mfeb] MFEM: Modular finite element methods. http://www.mfem.org.

[NTRNXB08] N Nguyen-Thanh, Timon Rabczuk, H Nguyen-Xuan, and Stéphane PA Bordas. A smoothed finite element method for shell analysis. *Computer Methods in Applied Mechanics and Engineering*, 198(2):165–177, 2008.

[Pat] Bořek Patzák. publications [oofem wiki]. http://www.oofem.org/wiki/doku.php?id=publications.

[Pat09] Bořek Patzák. Oofem project home page, 2009. http://www.oofem.org.

[PCD+12] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samaké, and Gonçalo Pena. Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences, 2012.

[Spe88] Bernhard Specht. Modified shape functions for the three-node plate bending element passing the patch test. *International Journal for Numerical Methods in Engineering*, 26(3):705–715, 1988.

[SPR] Luis Saavedra, Julien Pommier, and Yves Renard. Mpi parallelization of getfem++ - getfem++. http://download.gna.org/getfem/html/homepage/userdoc/parallel.html.

[Ste15] Peter Steinke. *Finite-Elemente-Methode: Rechnergestützte Einführung*. Springer-Verlag, 2015.

[Thi] Axel Thielscher. free software package for the simulation of non-invasive brain stimulation. http://simnibs.de/.

[ZT00] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method: Solid mechanics*, volume 2. Butterworth-heinemann, 2000.

[ZTZT77] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift