

Contents

List of Symbols	xiii
List of TODOs	xv
1 Introduction	1
1.1 The case for standardization	2
1.2 Goal of the book	4
1.3 Structure of the book	5
1.4 Source code in this book	7
Part I Preliminaries	
2 Mathematical concepts	13
2.1 The finite element method	13
2.1.1 Weak Formulation	15
2.1.2 Discretisation by finite element methods	16
2.1.3 Computing the stiffness matrix	22
2.1.4 Dealing with Dirichlet boundary conditions	28
2.2 The finite volume method	31
2.2.1 Conservation laws	31
2.2.2 Second-order elliptic equations	35
2.3 Local grid adaptivity	36
2.3.1 Local adaptivity of grids and finite element spaces	38
2.3.2 h -refinement	39
2.3.3 Hierarchical grids and refinement trees	45
3 Getting started with Dune	47
3.1 Installation of DUNE	47
3.1.1 Installation from binary packages	47
3.1.2 Installation from source	48
3.2 A first DUNE application	49

3.2.1	Creating a new module	50
3.2.2	Testing the new module	51
3.3	Example: Solving the Poisson equation using finite elements	52
3.3.1	The main method	53
3.3.2	Assembling the stiffness matrix	60
3.4	Example: Solving the transport equation with a finite volume method	64
3.4.1	Discrete linear transport equation	64
3.4.2	The <code>main</code> method	67
3.4.3	The method <code>evolve</code>	70
4	The design and structure of Dune	75
4.1	Software functionality for finite element and finite volume methods	75
4.2	The structure of DUNE	78
4.3	The DUNE core modules	81
4.4	Designing efficient interfaces	83
4.4.1	Dynamic polymorphism	84
4.4.2	Duck typing	85
4.4.3	Wrappers and engines	87
4.5	Coding style	89
4.5.1	Rules regarding the code in a DUNE module	89
4.5.2	Compatibility with the C++ standard library	90
4.6	Interface stability and backward compatibility	91
Part II The Core Modules		
5	Grids and the Dune grid interface	97
5.1	Hierarchical grids and grid views	100
5.2	Iterating over vertices and elements	103
5.3	Entities and geometries	107
5.3.1	Entities	109
5.3.2	Geometries	117
5.4	Intersections between neighboring elements	122
5.4.1	Intersections	122
5.4.2	Iterating over intersections	125
5.4.3	The <code>Intersection</code> interface class	126
5.5	Reference elements	132
5.5.1	Using the DUNE reference elements	134
5.5.2	<code>GeometryType</code> and the topology id	138
5.6	Attaching data to grids	142
5.6.1	Index sets and the boundary segment index	144
5.6.2	The <code>MultipleCodimMultipleGeomTypeMapper</code> class	149
5.6.3	Persistent numberings	153
5.7	Creating grids	156

5.7.1	Creating structured grids	157
5.7.2	The grid construction interface	159
5.7.3	Reading unstructured grids from Gmsh files	165
5.8	Writing grids and data to VTK files	169
5.8.1	Writing grids and data	170
5.8.2	Writing time-dependent data	172
5.9	Local grid adaptivity	175
5.9.1	Local grid adaptivity without data transfer	176
5.9.2	Preserving data across grid changes	180
5.10	Some existing grid managers	186
5.10.1	External grid managers	187
5.10.2	Built-in standalone grid managers	192
5.10.3	Meta grids	194
6	Dune grids on parallel distributed machines	195
6.1	DUNE data decomposition model	195
6.2	Setting up a distributed grid	202
6.2.1	Distributed structured grids	203
6.2.2	Distributed unstructured grids	204
6.3	Dynamic load-balancing	206
6.4	Communication	213
6.4.1	Subdomain communication	214
6.4.2	Collective communication	218
6.5	MPI setup with the MPIHelper class	220
6.6	Writing distributed grids to VTK files	222
6.7	Example: The Poisson equation on a distributed grid	223
6.7.1	Setting up the distributed algebraic problem	224
6.7.2	The distributed preconditioned CG method	229
7	Linear algebra with dune-istl	237
7.1	Constructing matrix and vector types by nesting	238
7.2	Data structures for vectors	242
7.2.1	Abstract interface	242
7.2.2	Vector implementations	247
7.3	Data structures for matrices	254
7.3.1	Abstract interface	254
7.3.2	Matrix implementations	259
7.4	Solvers and preconditioners	275
7.4.1	Solvers	275
7.4.2	Preconditioners	279
7.4.3	Parallel Solvers	281
7.4.4	Example: Solving the Poisson equation with a preconditioned CG method	282
7.5	Algebraic multigrid	289
7.5.1	Sequential algebraic multigrid	291

8	Finite elements and the dune-localfunctions module	295
8.1	Finite elements and affine families	296
8.2	The static interface for finite elements defined on the reference element	298
8.2.1	Sets of shape functions and the <code>LocalBasis</code> classes	300
8.2.2	Degrees of freedom and the <code>LocalInterpolation</code> classes	303
8.2.3	The <code>LocalCoefficients</code> Classes	305
8.3	Implementations of the local finite element interface	307
8.3.1	Affine-equivalent finite elements	307
8.3.2	Elements that are not affine-equivalent	311
8.4	The dynamic interface	313
8.4.1	The abstract base classes	314
8.4.2	Obtaining implementations of the virtual interface	316
9	Quadrature Rules	317
9.1	Numerical Integration	317
9.1.1	One-Dimensional Integration	318
9.1.2	Multidimensional Integrals	319
9.2	The DUNE Quadrature Rule Interface	321
Part III Solving Partial Differential Equations		
10	Function Spaces and Discrete Functions	327
10.1	Function space bases	329
10.1.1	Trees of function spaces	329
10.1.2	Trees of function space bases	331
10.1.3	Indexing basis functions by multi-indices	332
10.1.4	Strategy-based construction of multi-indices	336
10.1.5	Localization to single grid elements	341
10.2	Programmer interface for function space bases	342
10.2.1	The interface for a global function space basis	344
10.2.2	The user interface for a localized basis	345
10.2.3	The user interface of the tree of local bases	347
10.2.4	Multi-indices	349
10.3	Constructing trees of function space bases	353
10.3.1	Basis implementations provided by <code>dune-functions</code>	354
10.3.2	Combining bases into trees	355
10.4	Treating subtrees as separate bases	359
10.5	Global functions and grid functions	361
10.6	Building blocks for function interfaces	361
10.6.1	Function objects and functions	361
10.6.2	Type erasure and <code>std::function</code>	362
10.7	Extended function interfaces	365
10.7.1	Differentiable functions	365

10.7.2	GridView functions and local functions	368
10.8	Combining global bases and coefficient vectors	371
10.8.1	Vector backends	371
10.8.2	Interpreting coefficient vectors as finite element functions	373
10.8.3	Interpolation	374
10.9	Writing functions to a file	377
10.10	Example: Solving the Stokes equation with <code>dune-functions</code>	379
10.10.1	The Stokes equation	379
10.10.2	The driven-cavity benchmark	380
10.10.3	Implementation	381
11	Discretizing partial differential equations with <code>dune-pdelab</code>	393
11.1	Example: Linear Reaction–Diffusion Equation	395
11.2	Implementing element assemblers	402
11.2.1	The residual formulation	402
11.2.2	Assembling element residuals and their derivatives	405
11.2.3	Implementing element assemblers: The <code>LocalOperator</code> interface	406
11.2.4	Example: The p -Laplace equation with a reaction term	412
11.2.5	Boundary and skeleton integrals	422
11.2.6	Example: Discontinuous Galerkin methods	426
11.3	Dirichlet boundary conditions	442
11.3.1	Dirichlet boundary conditions and the residual form	442
11.3.2	Example: Poisson equation with Dirichlet boundary conditions	445
11.4	Linear algebra backends	449
11.4.1	The ISTL backend	450
11.4.2	The Eigen backend	454
11.4.3	Working with the actual data structures	456
11.4.4	The Simple backend	459
11.5	Local grid adaptivity	462
11.5.1	Local adaptivity in <code>dune-pdelab</code>	463
11.5.2	Example: The Poisson equation with residual-based grid adaptation	466
11.6	Parallel PDELab	478
11.6.1	Parallel Solver Backends	478
11.6.2	Example: Solving the p -Laplace problem using a parallel AMG preconditioner	482

Part IV Appendix

A	Installation and the Dune Build System	489
A.1	Installing DUNE modules	489
A.1.1	The <code>dunecontrol</code> program	490
A.1.2	Specifying the module search path with the <code>DUNE_CONTROL_PATH</code> variable	493
A.1.3	Setting global build options	495
A.1.4	Building unit tests and DOXYGEN documentation	496
A.2	DUNE modules	497
A.2.1	The structure of DUNE modules	497
A.2.2	The <code>dune.module</code> file	499
A.2.3	The files <code>config.h.cmake</code> and <code>config.h</code>	501
A.2.4	Unit tests	501
A.2.5	Documentation	502
A.2.6	Adding new code to a DUNE module	504
B	Complete source codes of the example programs	507
B.1	Finite element method for the Poisson equation	507
B.2	Finite volume method for the linear transport equation	511
B.3	Local grid adaptivity without data transfer	514
B.4	Local grid adaptivity with data transfer	515
B.5	The Poisson equation on a distributed grid	518
B.6	The Poisson equation on a distributed grid using ISTL solvers	524
B.7	The sequential AMG preconditioner	531
B.8	The Stokes equation using Taylor–Hood elements	532
B.9	Linear reaction–diffusion problem using <code>pdelab</code> and finite elements	537
B.10	p -Laplace problem using <code>pdelab</code> and finite elements	539
B.11	Reaction-diffusion equation using a DG method	543
B.12	Linear reaction–diffusion problem using <code>pdelab</code> and finite elements with Dirichlet boundary conditions	551
B.13	Demonstrating the linear algebra backends	552
B.14	Adaptive grid refinement using <code>pdelab</code>	556
B.15	Solving the p -Laplace problem using a parallel AMG preconditioner	561
	GNU Free Documentation License	567
	1. APPLICABILITY AND DEFINITIONS	567
	2. VERBATIM COPYING	569
	3. COPYING IN QUANTITY	569
	4. MODIFICATIONS	570
	5. COMBINING DOCUMENTS	572
	6. COLLECTIONS OF DOCUMENTS	572
	7. AGGREGATION WITH INDEPENDENT WORKS	573
	8. TRANSLATION	573
	9. TERMINATION	573

<i>Contents</i>	xi
10. FUTURE REVISIONS OF THIS LICENSE	574
11. RELICENSING	574
ADDENDUM: How to use this License for your documents.....	575
References	577

Draft version – Do not distribute!