BIT Numerical Mathematics (2008)
DOI:10.1007/s10543-008-0178-8

AUTOMATED FEM DISCRETIZATIONS FOR THE STOKES EQUATION *

ANDY R. TERREL[1], L. RIDGWAY SCOTT[2], MATTHEW G. KNEPLEY [3] and ROBERT C. KIRBY[4]

[1] *Department of Computer Science, University of Chicago, 1100 E 58th St, Chicago, IL 60637, USA. email: aterrel@cs.uchicago.edu*

[2] *Department of Computer Science, University of Chicago, 1100 E 58th St, Chicago, IL 60637, USA. email: ridg@cs.uchicago.edu*

[3] *Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Building 221, Argonne, IL 60439-4844, USA. email: knepley@mcs.anl.gov*

[4] *Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 79409-1042, USA. email: robert.c.kirby@ttu.edu*

Abstract.

Current FEM software projects have made significant advances in various automated modeling techniques. We present some of the mathematical abstractions employed by these projects that allow a user to switch between finite elements, linear solvers, mesh refinement and geometry, and weak forms with very few modifications to the code. To evaluate the modularity provided by one of these abstractions, namely switching finite elements, we provide a numerical study based upon the many different discretizations of the Stokes equations.

AMS subject classification (2000): 74S05,65Y99,35Q30

Key words: Numerical methods, Finite Element Methods, Stokes Equations

1 Introduction

The power and complexity of both computer hardware and algorithms has continued to increase for many decades, and yet the demand for ever larger and more complex simulations to model natural phenomena is unabated. With each of these advances, the resulting software complexity is compounded. This too often leads to unacceptable compromises regarding the choice of algorithm or computer hardware in an attempt to control software development costs.

Many people have adopted rapid code development environments, such as Matlab, that enable simple ideas to be tested very quickly before creating special purpose code for a specific application. These approaches to 'rapid prototyping' have often been plagued by poor runtime performance. However, in several domains [2, 6, 8, 9, 25, 27, 42, 44] researchers are attempting to automate code development without compromising performance. One such area is numerical PDE solvers where several groups are automating code for the Finite Element Method (FEM) [29, 31, 32, 37, 41].

*Received February 13 2008. Accepted May 30 2008. Communicated by Rolf Stenberg.

Models are often chosen quite early in the development of large scale simulations, and changing parts of that model may involve a nearly complete rewrite of the code. FEM automation seeks a 'plug and play' paradigm for different discretizations and model equations, just as vendors have adopted for computer hardware. This automation allows a user to test many models and discretizations quickly by performing large simulations with all of them, as we will demonstrate in this paper.

Although the idea of FEM automation is not new, no code has yet been able to achieve it fully, due in part to a lack of fully developed, mathematically grounded interfaces between the different parts of FEM codes. To elaborate on this point, in Section 2, we discuss the canonical parts of an FEM code and propose interfaces motivated by the mathematical constructs from which they were developed. In Section 3, we review codes which partially resolve the automation problem. Finally, in Sections 4 and 5, we use the new automation techniques in a numerical study of discretizations for the Stokes equations. It should be noted that this comparison is useful on its own because such a comparison between methods, without automated codes, is quite difficult and rarely found in the literature.

2 Features of Fully Automated FEM Code

FEM can be thought of as a black box for simulation, taking a domain, function space, equation, and boundary conditions as input, and then returning a solution field. Unfortunately, it seems that all attempts at completely automating FEM in this fashion have failed in one aspect or another. Dependencies appear in surprising places and often the design of the software makes assumptions that limit applicable discretizations. This begs the question, what features are necessary for full automation of FEM software?

2.1 Pluggable Parts

Many projects provide an input language for equations but often at the expense of flexibility in the internal representation. For instance an arbitrary equation cannot be combined with an arbitrary choice of discretization and mesh. However, having a symbolic representation of the weak form allows a user to quickly change models without a lengthy coding and debugging cycle. More advanced transformations are also possible, such as automatic differentiation or construction of error estimators and optimization loops. Both automatically generated and handwritten codes generally represent the weak form directly as imperative code using numerical interpolants, which greatly limits automatic manipulation.

Function spaces are severely limited in most software, often only allowing a single element. With the development of FIAT [28] and other approaches [14, 41], we now have a simple, declarative representation for most finite element function spaces. This flexibility allows a user to access many modern elements. A nice example is given by the divergence-free velocity space tailored to incompressible material (e.g., Stokes') equations. Unfortunately, specification of the function spaces is not sufficient for modularity as we still must consider the interface to the mesh.

Mesh representations have been well studied [7, 10, 53], however various data structures and interfaces still abound. The mesh interface itself must be sufficiently rich in order to support the chosen element. For example, if your mesh

has no notion of orientation, the face normals necessary for the Raviart-Thomas element are unavailable or not efficiently calculable. Most mesh libraries have concentrated on features such as automated refinement or mesh movement, ignoring interface requirements for discretization and field construction, often limiting the code to linear Lagrange elements [22, 24, 48, 53, 54], although there is some work with the richer mesh data [33, 51]. Moreover, most libraries are limited to a single element shape, such as boxes or simplicies.

The algebraic solver is a clear success story for this modular approach to FEM code. For over a decade there have been efficient, parallel software packages providing a wide range of linear and nonlinear solvers [3, 21, 26]. The successful the development of these algorithms and related software is a wonderful example to which the FEM community could aspire.

2.2 Mathematical Interfaces

Automation requires well designed interfaces that are both robust and suitably abstract so as not to limit the algorithm choice. A fully automated FEM code with all the modular features described above requires interfaces developed from the underlying mathematical abstractions originally used to study the PDE. FEM codes have roughly four aspects: an algebraic representation of the PDE, a functional analytic approximation, a topological region specification, and an algebraic system solution.

FIAT allows a user to easily construct general families of finite elements. It also provides the ability to tabulate and obtain topological connectivity of these elements. Again, linear algebra provides a language for describing the components. FIAT can automatically produce not only quadrature rules for the element, but also the action of each basis element in the dual space, commonly called degrees of freedom. By reducing the element interface to these two pieces, along with simple size calculations, a code can switch elements with the same ease as linear solvers. We demonstrate the power of this approach for simulation of the Stokes equation.

A key insight into the interface design is the division of FEM operations into local and global sets [36]. Operations on a single element are purely local, and demand only knowledge of the geometry and function space of the element. Specification of fields over the mesh, or sections, uses only topological information from the mesh and simple size information from FIAT. A more sophisticated interaction is the definition of a unique global basis when local element basis vectors disagree, such as face normal integrals in Raviart-Thomas elements. However, with local and global information strictly separated, we can write generic routines to accomplish such tasks. This separation is essential for automation, and also provides opportunities for optimization since we can ignore unimportant details. By using a topological representation of the domain, this separation becomes quite natural, enabling the easy use of many different elements.

As alluded to earlier, the most successful modular interface applies to the algebraic solver. The fundamental abstraction in this case is linear algebra. All groups working were able to agree on roughly the same interfaces for Vector and Matrix classes. Moreover, in most cases, preconditioning can be treated with the same interface. This abstraction is able to handle an incredible range of algebraic solvers without modification to the other parts of the code.

3 Description of Current Automated FEM Packages

Automation seems to be very natural and valuable, but why then has a fully automated code not yet been produced? A cynical answer that one often encounters is the limited vision of the researchers who use the methods. While there are certainly quite different goals among FEM code writers, perhaps the fundamental reason for the lack of automation is the lack of mathematical understanding of the necessary interfaces. Nonetheless, a number of different approaches have emerged to further the development of such a code, each using different mathematical abstractions previously discussed.

Four open source FEM software projects, in different stages of development, that substantially automate many FEM algorithms discussed in this paper are the FEniCS project [20, 38], Sundance [39, 40], DEAL.II [5, 4], and PETSc-Sieve [34]. These four projects are presented in order to highlight different parts of the automation process but are only a sampling of the numerous projects of this type [19, 23, 43, 45, 47].

These projects, not surprisingly, share dependencies on a few middle-ware projects. Each of these FEM software libraries are able to use a parallel algebraic solver, such as those in PETSc [3] or Trilinos [26]. FIAT [28], the finite element generator, is also used by multiple packages. FIAT allows for the generation of quadrature rules for many different elements which are easily included.

The DEAL.II project is a set of libraries for FEM simulation on structured adaptive grids. It does not attempt to automate the entire FEM process, but uses templates to allow a wide range of elements to be used in two or three dimensions. For example, it gives no abstractions for the variational form. While DEAL.II does not provide a scripting style interface for the programming, it does bundle important data types and operations in a clear API. The most notable feature of DEAL.II is its ability to handle numerous elements and methods, but a user must still manually code a specific weak form and boundary conditions. Finally, it only handles one specific mesh type.

Sundance is a code designed to provide a rapid development environment and the ability to solve PDE constrained optimization problems. The user space is separated from the backend with simple interface calls, which give the feel of programming in a scripting environment with clever use of C++ overloading. Its symbolic representation of the input equations allow for optimizations whose efficiency rivals special purpose codes. It cannot currently switch between different elements, and has some limitations on the mesh types.

FEniCS is a collection of projects to develop simple modules that allow for research into FEM automation, such as FIAT and SyFi [41]. A key component is the FEniCS Form Compiler, FFC [30], which is used in the simulation engine, DOLFIN. FFC autogenerates optimized code for precomputing the discretized weak form for a given finite element space. DOLFIN provides the glue for pulling the rest of the simulation together, including solvers and visualization. This project lies between the previous two projects in that it provides a simpler interface than Deal.II but requires more coding to be as robust as Sundance. This lack of robustness is shown in the experiments when the $C^0 P_i C^{-1} P_{i-1}$ element produces an unconverged answer instead of giving a proper error. Whereas the previous two projects were built to solve engineering problems, this project focuses much more on automation research itself. For the purposes of this paper when we refer to experiments with FEniCS, we are referring to the use of FIAT, FFC, and DOLFIN codes.

Table 3.1: Snapshot of modularity in Automated FEM Packages. (*Dolfin is currently being parallelized [57].)

Software Packages	Function Spaces	Meshes	Matrix Solvers
Dolfin	uses FIAT	simplicies	not parallel*
Sundance	limited FIAT	simplicies	parallel
DEAL.II	menu	quads and simplicies	almost parallel
PETSc-Sieve	uses FIAT	quads and simplicies	parallel

The Sieve library handles both topology representation and definition of fields over such as mesh. It is inspired by the simple abstract framework of Grothendieck topologies [18], and treats meshes in a shape and dimension independent way [35]. This representation allows a simple interaction between the mesh topology (global) and the element function space (local). FIAT is used to define the element, augmented by a small code generator for the action of dual basis functionals. Moreover, a new domain specific language generator, Mython [46], is being incorporated to allow a scripting interface for equation specification. We summarize the observations about these various packages in Table 3.1.

4 Discretization of Stokes Equation

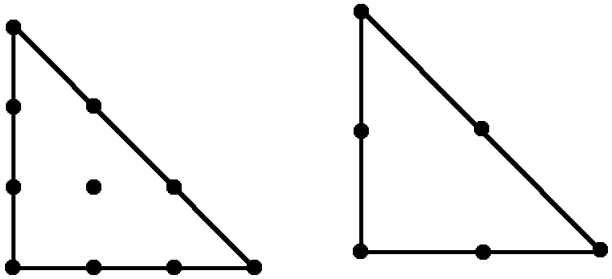
The Stokes equation is the standard equilibrium equation for low Reynolds number, incompressible flow.

$$\begin{aligned} -\Delta \mathbf{u} + \nabla \mathbf{p} &= \mathbf{f} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{4.1}$$

The problem has been well studied and is documented in several books [12, 13]. It is a challenging problem due to the coupling between the velocity and pressure resulting in a saddle point in the variational formulation and the divergence free criteria of the velocity field. The matrix equations will be indefinite which proves to be quite taxing on linear solvers. These difficulties are also two of the main challenges of the more general Navier-Stokes equations that are used in numerous fluid simulations.

This case was chosen for our study because there are many stable discretization methods but very few numerical studies including more than a single method. Typically coding one method is so difficult that the cost outweighs the value of coding another method, especially if a large legacy simulation is already using that method. For our software described above, we only need to make use of one mathematical abstraction to automatically generate numerous methods, that is the finite element. By calling FIAT from several different codes, different discretizations can be readily employed, allowing meaningful comparisons with few code changes. By studying the numerical discretization schemes of the Stoke's equation, any observations will affect the results of higher level simulations using Navier-Stokes.

Below is a brief explanation of the different discretization methods included in this study. It is followed by a description of our testing framework and results



(a) P_3 for V (b) P_2 for Π

Figure 4.1: Taylor-Hood Elements

from the different methods. While there have been many studies on efficient linear solvers for a particular discretization, this study uses only a direct linear solver so that the contribution of each discretization is not obscured.

4.1 Mixed method Formulation

One of major challenges posed by the Stokes equation is handling the coupling between the velocity and pressure. One natural way to solve the system is to create a mixed system, with blocks corresponding to fields, and each field using a different element. The variational form of this mixed system is as follows:

> Let $V = H^1(\Omega)^n$ and $\Pi = \{q \in L^2(\Omega) : \int_\Omega q dx = 0\}$. Given $F \in V'$, find functions $\mathbf{u} \in V$ and $p \in \Pi$ such that

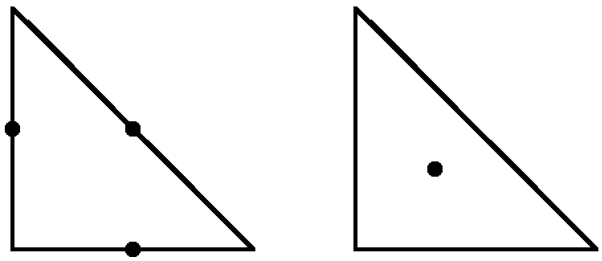
$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= F(\mathbf{v}) \quad \forall v \in V \\ b(\mathbf{u}, q) &= 0 \quad \forall q \in \Pi, \end{aligned} \tag{4.2}$$

where

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &:= \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} dx, \\ b(\mathbf{v}, q) &:= \int_\Omega (\nabla \cdot \mathbf{v}) q dx. \end{aligned}$$

This mixed method formulation uses two discrete spaces V and Π. Developing different finite element spaces for this system is quite challenging. Using the same continuous element spaces for both the pressure and velocity leads to an over-determined system of equations. Also, it is very attractive to use a discontinuous pressure space which provides a better solution for the divergence of the velocity, but can lead to singularities in the solution which depend on mesh. This study presents elements that have both continuous and discontinuous pressure spaces, and evaluates the numerical consequences of both.

The Taylor-Hood element [11, 55] is one of the most widely used elements for solving Stokes flow. It consists of a P_k element for the velocity space and P_{k-1} for the pressure space (see Figure 4.1). Because of the simplicity of using



(a) Crouzeix-Raviart for V (b) P_0 for Π

Figure 4.2: Crouzeix-Raviart Elements

Lagrangian elements, it can easily be extended to higher orders. This element produces a continuous pressure space, but the order of the pressure convergence is lower than that for the velocity.

The Crouzeix-Raviart Element [15] is a non-conforming element that uses integral moments over the element edges as a basis for the velocity and a discontinuous pressure space, P_0 (see Figure 4.2). For the low order case, the velocity edge moments are equivalent to evaluating the basis functions at the center of each edge.

Another possibility is just to use the high degree Lagrange element for velocity but use a discontinuous element (of a lower order) in the pressure space, what we loosely call $C^0P_iC^{-1}P_{i-1}$. It is important to note that stable convergence of this element is dependent upon the mesh. The formulation may not satisfy the inf sup condition due to singularities in the pressure space [49]. However, if one is able to eliminate certain problem points in the mesh, as in Fig. 5.1, the method becomes useful.

4.2 Iterated Penalty Method

In order to avoid the problems with the saddle point and the discontinuous pressure space, the Uzawa iteration method and penalty methods were developed. A combination of these two ingredients results in the iterated penalty method. Let $r, \rho \in \mathbb{R}$ and $\rho > 0$ and define $\mathbf{u}^n$ and $\mathbf{w}^n$ by

$$a(\mathbf{u}^n, \mathbf{v}) + r(\nabla \cdot \mathbf{u^n}, \nabla \cdot \mathbf{v}) = F(\mathbf{v}) - (\nabla \cdot \mathbf{v}, \nabla \cdot \mathbf{w^n})$$
$$\mathbf{w}^{n+1} = \mathbf{w}^n + \rho \mathbf{u}^n$$

The pressure may be recovered from the auxiliary $\mathbf{w}$ field, $p = \nabla \cdot \mathbf{w}$. This method uses only one space, but requires a higher order continuous element [50]. We use Lagrangian elements with degree greater than three. For stopping criterion, we use the constraint violation, or incompressibility error, $||\nabla \cdot \mathbf{u^n}|| < \epsilon$. The iteration count and accuracy is dependent upon the penalty coefficients ρ and r. For our experiments we use $\rho = -r = 1.0e-3$. The Iterated Penalty method is essentially a formulation of the $C^0P_iC^{-1}P_{i-1}$ element using a single space. Thus we have two models for the same mathematical process, but using quite different algorithms.

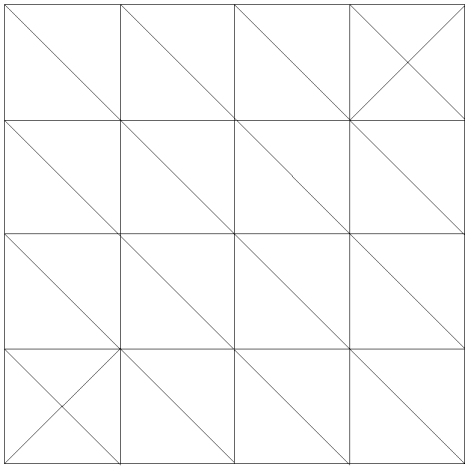


Figure 5.1: The uniform mesh with $n \times n$ rectangles with refined corners to reduce singularities in $C^0 P_i C^{-1} P_{i-1}$ pressure space used in the tests.

5 Results

In order to evaluate these methods, we will compare mesh sizes and element orders in a series of increasingly difficult problems. The number of degrees of freedom is a good measure of the total work the method will require, and will equate them throughout. The number of the degrees of freedom as a function of mesh size and element is shown in Table 5.1.

For our tests, we use a $n \times n$ uniform mesh for the square domain [0,1]×[0,1], and solve the following three problems:

- Case 0:

$$\begin{aligned} \mathbf{f} &= \begin{bmatrix} -2y+1 \\ 2x+1 \end{bmatrix}, \\ \mathbf{u} &= \begin{bmatrix} x^2 y \\ -xy^2 \end{bmatrix}, \text{ and} \\ p &= x + y - 1.0 \end{aligned}$$

- Case 1:

$$\begin{aligned} \mathbf{f} &= \begin{bmatrix} 8\pi^2 \sin(2\pi x)\cos(2\pi y) \\ -8\pi^2 \cos(2\pi x)\sin(2\pi y) \end{bmatrix}, \\ \mathbf{u} &= \begin{bmatrix} \sin(2\pi x)\cos(2\pi y) \\ -\cos(2\pi x)\sin(2\pi y) \end{bmatrix}, \text{ and} \\ p &= \sin(2\pi x)\sin(2\pi y) \end{aligned}$$

Table 5.1: A comparison of the degrees of freedom for each element organized by velocity order (p) and mesh element size (h). A '-' indicate that the order for that particular element is not stable, and a 'x' indicates the element is undefined.

p	h	Crouzeix-Raviart	$C^0P_iC^{-1}P_{i-1}$	Taylor-Hood	Iterated Penalty
1	.25	160	-	-	-
	.125	560	-	-	-
	.0625	2128	-	-	-
	.03125	8336	-	-	-
2	.25	x	286	205	-
	.125	x	990	677	-
	.0625	x	3742	2485	-
	.03125	x	14622	9557	-
3	.25	x	590	463	-
	.125	x	2078	1583	-
	.0625	x	7934	5935	-
	.03125	x	31166	23087	-
4	.25	x	1002	829	642
	.125	x	3562	2885	2242
	.0625	x	13674	10933	8514
	.03125	x	53866	42772	33346
5	.25	x	1522	1303	982
	.125	x	5442	4583	3462
	.0625	x	20962	17479	13222
	.03125	x	82722	68615	51942

- Case 2:

$$\mathbf{f} = \begin{bmatrix} 18\pi^2 \sin(3\pi x)\cos(3\pi y) \\ -18\pi^2 \cos(3\pi x)\sin(3\pi y) \end{bmatrix},$$
$$\mathbf{u} = \begin{bmatrix} \sin(3\pi x)\cos(3\pi y) \\ -\cos(3\pi x)\sin(3\pi y) \end{bmatrix}, \text{ and}$$
$$p = \sin(3\pi x)\sin(3\pi y)$$

The solution for case 0 is a low order polynomial that will be in most of the approximating spaces, and thus they should produce the exact answer to machine precision. For this reason we exclude case 0 from the presentation of results and only discuss it here to show our quick validation problem for our simulations. Complete analysis of all case can be found in Terrel [56]. The next two cases have more complicated solution which are harder to approximate in the given spaces, and therefore give a better idea of the convergence rates of the methods.

For each simulation, we use a basis tabulated by FIAT, and a simple routine generates the $n \times n$ uniform mesh with refined corners to avoid boundary elements that may cause singular points in the pressure space for the $C^0P_iC^{-1}P_{i-1}$ method, see Figure 5.1. For each linear solve, we used the UMFPACK [16] LU direct solver. The different simulation codes used were FEniCS (FFC version 0.2, DOLFIN version 0.6, and FIAT version 0.3.0) and Sundance version 2.2.0. At the time of the study, Sundance did not support discontinuous elements and thus reduced our methods tested in that system to only Taylor-Hood and Iterated Penalty (see Table 5.2).

In evaluating the method performance, a few features were notable. As the mesh was refined, where h is $1/n$, the methods maintained a consistent rate of error decrease from case 1 to case 2, see Table 5.3. This provides an indication

Table 5.2: The different simulation engines used for each method

	Sundance	FEniCS
Taylor-Hood	X	X
Crouzeix-Raviart	-	X
$C^0 P_i C^{-1} P_{i-1}$	-	X
Iterated Penalty	X	X

that the theoretical errors were being achieved by the separate simulations. The difference in mass balance between the divergence free elements and the continuous elements is clearly demonstrated. An interesting feature displayed in this charts is the relation between Iterated Penalty and the $C^0 P_i C^{-1} P_{i-1}$ element. For the $C^0 P_i C^{-1} P_{i-1}$ element with degree less than four, the method is not converges at the L^2 order (h^p) consistent with the theoretical predictions of de Boor and Höllig [1, 17]. Both of these methods achieve the same result, which is expected since they are mathematically equivalent, but of course use quite different algorithms. Runtimes more or less paralleled the number of degrees of freedom, however there are deviations, perhaps due to effective cache use. Further study using profiling tools, such as TAU [52], and taking into account the condition number of the resulting linear systems is required in order to make any substantial claims regarding computational complexity of the methods in question.

To give a sense of the actual error, Figure 5.2 gives a slice of the data for all fourth order methods with the FEniCS and Sundance. Crouziex-Raviart, with an equivalent number of DOFs, is included in order to compare with a low order method. Taylor Hood is the most accurate method in terms of L^2 errors for pressure and velocity, but, as the theory suggests, has a much larger divergence residual. This is partially due to the fact that our case study includes only very smooth functions, and thus the results could be different in a non-smooth context. An interesting difference in these results is Sundance's ability to achieve errors of 10^{-12} while with FEniCS does not progress beyond 10^{-8}.

Finally Figure 5.3 is included to show a blemish in testing of this nature. From Table 5.3 it appears that Sundance and FEniCS always produce very similar results, but Figure 5.3 for the largest mesh produced an uncoverged result and reported this solution as solved. While this may be a rare error, it is one that questions the robustness of the FEniCS package. In our interactions with both Sundance and FEniCS, we found that a significant effort was required to catch such errors in FEniCS whereas Sundance handled such errors "out of the box".

6 Conclusions

In our numerical study, FIAT's finite element interface allowed the simulations to easily change between different discretization schemes. This particular study is quite useful for other applications involving fluid dynamics since the Stokes equation is repeatedly solved when solving the Navier-Stokes equations. Small effects in the code, such as a divergence of the velocity, can have large impacts on other parts of simulations that depend on this feature, and thus FIAT's ability

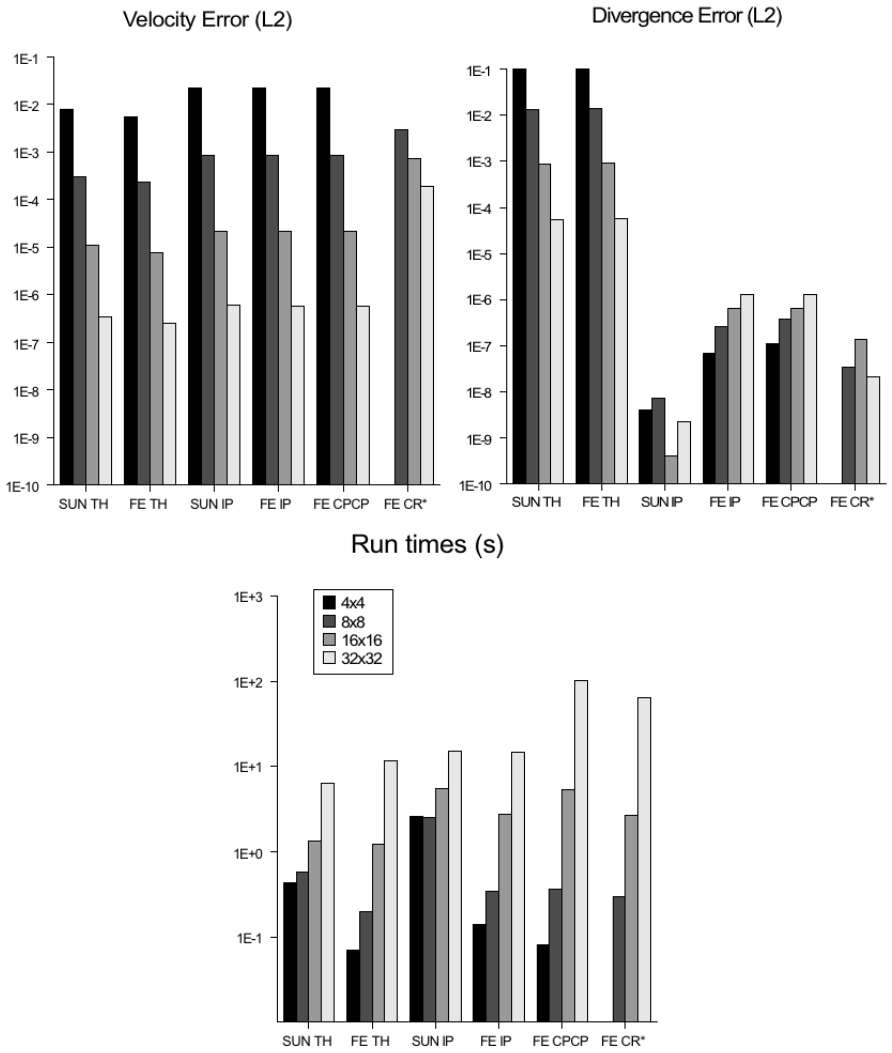


Figure 5.2: Comparison of 4th Order methods on Case 2 (* with Crouzeix-Raviart on a finer mesh to have equivalent number of DOFs), where SUN is Sundance, FE is Fenics, TH is Taylor-Hood, IP is Iterated Penalty, and CPCP is $C^0 P_i C^{-1} P_{i-1}$

Table 5.3: Convergence rates of the velocity L^2 error for the different elements and software, where SUN is Sundance and FE is FEniCS

p	Crouzeix-Raviart		$C^0P_iC^{-1}P_{i-1}$	
	FE	SUN	FE	SUN
1	$3.9h^{2.08\pm.1}$	X	-	-
2	-	-	$9.1h^{1.97\pm.07}$	X
3	-	-	$7.3h^{3.05\pm.08}$	X
4	-	-	$28.5h^{5.08\pm.22}$	X
5	-	-	$11.3h^{5.79\pm.09}$	X

p	Taylor-Hood		Iterated Penalty	
	FE	SUN	FE	SUN
1	-	-	-	-
2	$15.0h^{3.52\pm.41}$	$9.1h^{2.86\pm.05}$	-	-
3	$6.8h^{4.07\pm.06}$	$12.6h^{4.01\pm.13}$	-	-
4	$4.6h^{4.81\pm.15}$	$6.8h^{4.83\pm.1}$	$28.5h^{5.08\pm.22}$	$27.7h^{5.07\pm.2}$
5	$9.2h^{6.15\pm.32}$	$6.2h^{5.94\pm.06}$	$10.9h^{5.77\pm.06}$	$10.1h^{5.77\pm.08}$

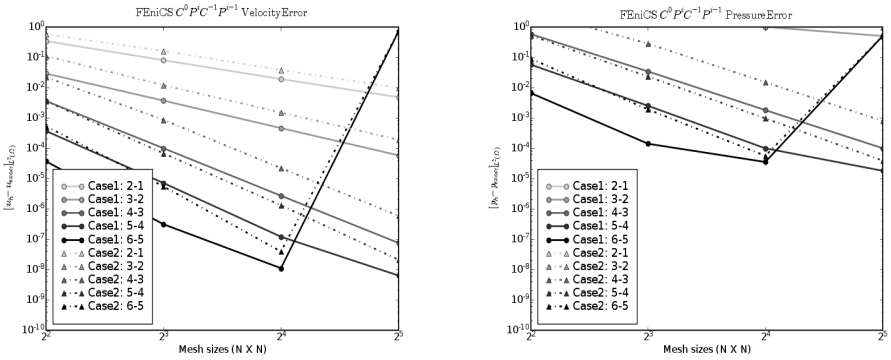


Figure 5.3: Anomalous $C^0P_iC^{-1}P_{i-1}$ Results

to change finite elements can allow a scientist to choose a model which gives the fidelity required by the specific application.

While these four FEM packages are moving forward, allowing for faster development on large scale scientific codes, further development of mathematical interfaces is necessary. In addition to allowing plug and play features to FEM codes, they also provide important ways to optimize the process automatically. Perhaps the nicest feature provided by these software packages is the scripting interface for the scientist to input various models. Often the hurdle of coding in lower level languages becomes the primary barrier for scientists in implementing new theoretical models.

REFERENCES

1. D. N. Arnold and J. Qin, *Quadratic velocity/linear pressure Stokes elements*, in Advances in Computer Methods for Partial Differential Equations-VII, R. Vichnevetsky, D. Knight, and G. Richter, eds., IMACS, New Brunswick, NJ, 1992, pp. 28–34.
2. A. A. Auer, G. Baumgartner, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Krishnamoorthy, S. Krishnan, C.-C. Lam, Q. Lu, M. Nooijen, R. Pitzer, J. Ramanujam, P. Sadayappan, and A. Sibiryakov, *Automatic code generation for many-body electronic structure methods: the Tensor Contraction Engine*, Molecular Physics, 104 (2006), pp. 211–228.
3. S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Web page*, 2001.
4. W. Bangerth, R. Hartmann, and G. Kanschat, *deal.II — a general-purpose object-oriented finite element library*, ACM Trans. Math. Softw., 33 (2007), p. 24.
5. ———, *deal.II project webpage*, 2007.
6. G. Baumgartner, A. Auer, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. J. Harrison, S. Hirata, S. Krishanmoorthy, S. Krishnan, C.-C. Lam, Q. Lu, M. Nooijen, R. M. Pitzer, J. Ramanujam, P. Sadayappan, and A. Sibiryakov, *Synthesis of high-performance parallel programs for a class of ab initio quantum chemistry models*, Proceedings of the IEEE, 93 (2005), pp. 276–292. special issue on "Program Generation, Optimization, and Adaptation".
7. G. Berti, *Generic Software Components for Scientific Computing*, PhD thesis, TU Cottbus, 2000. http://www.math.tu-cottbus.de/∼berti/diss.
8. P. Bientinesi, J. A. Gunnels, M. E. Myers, E. S. Quintana-Ortí, and R. A. van de Geijn, *The science of deriving dense linear algebra algorithms*, ACM Trans. Math. Softw., 31 (2005), pp. 1–26.
9. P. Bientinesi, E. S. Quintana-Ortí, and R. A. van de Geijn, *Representing linear algebra algorithms in code: the FLAME application program interfaces*, ACM Trans. Math. Softw., 31 (2005), pp. 27–59.
10. D. K. Blandford, G. E. Blelloch, D. E. Cardoze, and C. Kadow, *Compact representations of simplicial meshes in two and three dimensions*, International Journal of Computational Geometry and Applications, 15 (2005), pp. 2–24.
11. D. Boffi, *Three-dimensional finite element methods for the Stokes problem*, SIAM Journal on Numerical Analysis, 34 (1997), pp. 664–670.

12. S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 2nd ed., 2002.
13. F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, vol. 15 of Springer Series in Computational Mathematics, Springer-Verlag, New York, 1991.
14. P. Castillo, R. Rieben, and D. White, *FEMSTER: An object-oriented class library of high-order discrete differential forms*, ACM Trans. Math. Softw., 31 (2005), pp. 425–457.
15. M. Crouzeix and P.-A. Raviart, *Conforming and nonconforming finite element methods for solving the stationary Stokes equations*, RAIRO Anal. Numér., 7 (1973), pp. 33–75.
16. T. A. Davis, *Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method*, ACM Transactions on Mathematical Software, 30 (2004), pp. 196 – 199.
17. C. de Boor and K. Höllig, *Approximation order from bivariate C^1-cubics: A counterexample*, Proceedings of the American Mathematical Society, 87 (1983), pp. 649–655.
18. M. Demazure and A. Grothendieck, eds., *Séminaire de Géométrie Algébrique du Bois Marie – 1962-64 – Schémas en groupes*, vol. 1, Springer-Verlag, Berlin-New York, 1970, Springer-Verlag. Lecture notes in mathematics 151, in French.
19. P. Dular and C. Geuzaine, *GetDP: A general environment for the treatment of discrete problems*, 2005. http://www.geuz.org/getdp/.
20. T. Dupont, J. Hoffman, C. Johnson, R. C. Kirby, M. G. Larson, A. Logg, and L. R. Scott, *The FEniCS project*, Tech. Rep. 2003–21, Chalmers Finite Element Center Preprint Series, 2003.
21. R. D. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners*, in International Conference on Computational Science (3), P. M. A. Sloot, C. J. K. Tan, J. Dongarra, and A. G. Hoekstra, eds., vol. 2331 of Lecture Notes in Computer Science, Amsterdam, The Netherlands, 2002, Springer, pp. 632–641.
22. L. Feitag, C. Ollivier-Gooch, M. Jones, and P. Plassman, *SUMMA3D*, 1997. http://www-unix.mcs.anl.gov/ freitag/SC94demo/project/project.html, now defunct.
23. L. Formaggia, J. Gerbeau, and C. Prud'homme, *LifeV developer manual*, 2007. http://www.lifev.org/documentation/book/book.pdf.
24. C. Geuzaine and J.-F. Remacle, *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*, 2007. http://www.geuz.org/gmsh/.
25. A. Griewank, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
26. M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Softw., 31 (2005), pp. 397–423.
27. K. Kennedy, B. Broom, A. Chauhan, R. Fowler, J. Garvin, C. Koelbel, C. McCosh, and J. Mellor-Crummey, *Telescoping languages: A system for automatic generation of domain languages*, Proceedings of the IEEE, 93 (2005), pp. 387–408. special issue on "Program Generation, Optimization, and Adaptation".

28. R. C. Kirby, *Algorithm 839: FIAT, a new paradigm for computing finite element basis functions*, ACM Transactins on Mathematical Software, 30 (2004), pp. 502–516.
29. ———, *Optimizing FIAT with level 3 BLAS*, ACM Trans. Math. Softw., 32 (2006), pp. 223–235.
30. R. C. Kirby and A. Logg, *A compiler for variational forms*, ACM Trans. Math. Softw., 32 (2006), pp. 417–444.
31. ———, *Efficient compilation of a class of variational forms*, ACM Trans. Math. Softw., 33 (2007), p. 17.
32. R. C. Kirby, A. Logg, L. R. Scott, and A. R. Terrel, *Topological optimization of the evaluation of finite element matrices*, SIAM J. Sci. Computing, 28 (2006), pp. 224–240.
33. B. 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 (2006), pp. 237–254. http://dx.doi.org/10.1007/s00366-006-0049-3.
34. M. G. Knepley and D. A. Karpeev, *Flexible representation of computational meshes*, Technical Report ANL/MCS-P1295-1005, Argonne National Laboratory, October 2005.
35. ———, *Mesh algorithms for PDE with Sieve I: Mesh distribution*, 2008. To appear in Scientific Programming.
36. M. G. Knepley, A. R. Terrel, and L. R. Scott, *Finite element assembly on arbitrary meshes*. In preparation for publication in 2008.
37. J. Korelc, *Multi-language and multi-environment generation of nonlinear finite element codes*, Engineering with Computers, 18 (2002), pp. 312–327.
38. A. Logg, J. Hoffman, J. Jansson, R. C. Kirby, and G. N. Wells, *The FEniCS project webpage*, 2007.
39. K. R. Long, *Sundance: a rapid prototyping toolkit for parallel pde simulation and optimization*, in Large-Scale PDE-Constrained Optimization, Lecture Notes in Computational Science and Engineering, Heidelberg, 2003, Springer, pp. 331–342.
40. ———, *Sundance 2.0 tutorial*, Technical Report SAND2004-4793, Sandia National Laboratory, 2004.
41. K. A. Mardal, O. Skavhaug, G. Lines, G. Staff, and A. Odegard, *Using Python to solve partial differential equations*, Computing in Science and Engineering, 9 (2007), pp. 48–51.
42. G. L. Miller, S. Teng, W. Thurston, and S. A. Vavasis, *Automatic mesh partitioning*, tech. rep., Cornell University, Ithaca, NY, USA, 1992.
43. O. Pironneau, F. Hecht, and A. L. Hyaric, *freeFEM++*, 2007. http://www.freefem.org/ff++/ftp/freefem++doc.pdf.
44. M. Püschel, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gačić, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, *SPIRAL: Code generation for DSP transforms*, Proceedings of the IEEE, 93 (2005), pp. 232–275. special issue on "Program Generation, Optimization, and Adaptation".
45. Y. Renard and J. Pommier, *Getfem++. an open source generic C++ library for finite element methods*, 2007. http://home.gna.org/getfem/.

46. J. Riehl, *Mython project*, 2007. A new project found at http://code.google.com/p/basil/wiki/Mython.
47. A. Schmidt and K. G. Siebert, *ALBERT - software for scientific computations and applications*, Acta Mathematica Universitatis Comenianae, 70 (2001), pp. 105–122.
48. J. Schöberl, *NETGEN an advancing front 2d/3d-mesh generator based on abstract rules*, Computing and Visualization in Science, 1 (1997), pp. 41–52.
49. L. R. Scott and M. Vogelius, *Conforming finite element methods for incompressible and nearly incompressible continua*, in Large Scale Computations in Fluid Mechanics, B. E. Engquist, *et al.*, eds., vol. 22 (Part 2), Providence: AMS, 1985, pp. 221–244.
50. ———, *Norm estimates for a maximal right inverse of the divergence operator in spaces of piecewise polynomials*, M^2AN (formerly R.A.I.R.O. Analyse Numérique), 19 (1985), pp. 111–143.
51. E. S. Seol and M. S. Shephard, *Efficient distributed mesh data structure for parallel automated adaptive analysis*, Eng. with Comput., 22 (2006), pp. 197–213.
52. S. Shende and A. D. Malony, *TAU: The TAU parallel performance system*, International Journal of High Performance Computing Applications, 20 (2006), pp. 287–331.
53. J. R. Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, in Applied Computational Geometry: Towards Geometric Engineering, M. C. Lin and D. Manocha, eds., vol. 1148 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.
54. H. Si, *TetGen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator*, 2007. http://tetgen.berlios.de/.
55. C. Taylor and P. Hood, *A numerical solution of the Navier-Stokes equations using the finite element technique*, Computers and Fluids, 1 (1973).
56. A. R. Terrel, *FEM optimization with a case study of the Stokes equations*, Tech. Rep. TR-2008-02, Department of Computer Science, University of Chicago, 2008.
57. G. M. Vikstrøm, *Parallelization strategies for DOLFIN*, Master's thesis, University of Oslo, Department of Informatics, 2008.