INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2014; **100**:620–632
Published online 17 September 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4769

On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers

N. Collier[1,5,*,†], L. Dalcin[1,3,4] and V. M. Calo[1,2]

[1]*Center for Numerical Porous Media, King Abdullah University of Science and Technology, Saudi Arabia*
[2]*Applied Mathematics & Computational Science and Earth Science & Engineering, King Abdullah University of Science and Technology, Saudi Arabia*
[3]*Consejo Nacional de Investigaciones Científicas y Técnicas, Santa Fe, Argentina*
[4]*Universidad Nacional del Litoral, Santa Fe, Argentina*
[5]*Oak Ridge National Laboratory, Oak Ridge, TN, USA*

SUMMARY

We compare the computational efficiency of isogeometric Galerkin and collocation methods for partial differential equations in the asymptotic regime. We define a metric to identify when numerical experiments have reached this regime. We then apply these ideas to analyze the performance of different isogeometric discretizations, which encompass C^0 finite element spaces and higher-continuous spaces. We derive convergence and cost estimates in terms of the total number of degrees of freedom and then perform an asymptotic numerical comparison of the efficiency of these methods applied to an elliptic problem. These estimates are derived assuming that the underlying solution is smooth, the full Gauss quadrature is used in each non-zero knot span and the numerical solution of the discrete system is found using a direct multi-frontal solver. We conclude that under the assumptions detailed in this paper, higher-continuous basis functions provide marginal benefits. Copyright © 2014 John Wiley & Sons, Ltd.

Received 28 March 2014; Revised 14 July 2014; Accepted 19 July 2014

KEY WORDS: computational efficiency; finite elements; collocation; isogeometric; asymptotic analysis

1. INTRODUCTION

When studying properties of numerical methods, we frequently examine convergence – the evolution of the solution error under discretization refinement. Convergence is important to understand for at least two reasons. First, it demonstrates that a method applied to a problem and subject to refinement will approach the correct solution. Second, we prefer methods with large rates of convergence in the expectation that such a feature is an indicator of efficiency.

However, a method's convergence rate is at best an incomplete picture of its computational efficiency. First, in equating convergence to efficiency, we commit the crime of assuming that the solution cost per degree of freedom is equivalent for different methods. As we show in [1, 2] and is verified in [3], it is possible to see orders of magnitude difference in per-degree-of-freedom costs across different methods. Second, while it is reasonable to believe that methods with high convergence rates would be preferable, these methods may be more expensive than lower-order methods when engineering accuracy is required [4]. That is, the benefits of higher convergence rates may not become apparent in the accuracy ranges relevant in practice. However, we acknowledge research [5–8] that demonstrates that higher order methods perform better when high accuracy is

*Correspondence to: N. Collier, Oak Ridge National Laboratory, PO Box 2008 MS6301, Oak Ridge, TN 37831-6301, USA.
†E-mail: nathaniel.collier@gmail.com

desired. This is particularly true for the class of smooth problems conventionally used in academic benchmarks. Last, a method's convergence rate provides no metric for comparing methods with the same rate.

In order to distinguish among methods with the same convergence rate, we usually initiate more in-depth studies about how costs of methods increase as the number of degrees of freedom (N) increases. While we are primarily concerned with the exponent α on N in the leading term of the cost estimates, the constants in front of N^{α} are also of interest. While these constants are certainly dependent on many factors (hardware, libraries, underlying data structures, optimization capabilities of the compiler, etc.), it is possible to sharpen their estimation in terms of discretization-specific parameters to better understand relative costs of methods.

The most common technique of estimating costs of a method is to enumerate the number of floating point operations (FLOPs) as a function of discretization parameters. We emphasize that using such a model implicitly assumes that the overall computation time is dominated by the FLOPs on the processor. Another way to state this is that we assume that an algorithm's *arithmetic intensity*, the ratio of floating operations to memory accesses, is sufficiently high. A matrix LU-factorization is an example of an algorithm with high arithmetic intensity where the FLOPs model of costs makes sense. However, there are algorithms where the computing time is dominated by memory access and subsequently have a low arithmetic intensity. A sparse matrix-vector product is an example of such an algorithm where the ratio of operations to accesses is roughly one (or 0.25 FLOPs per byte as in [9]). For these types of algorithms, we must model costs by counting memory accesses. The real difficulty occurs when estimating the computational time for algorithms with medium arithmetic intensity as both models fail to accurately account for the effects of memory hierarchies in current processor architectures.

So while modeling the computational cost by counting either FLOPs or memory accesses may be educational, the end estimates may not accurately reflect the computational time. What this suggests is that highly detailed estimates may not be useful when making decisions about methods. In the end, it is inevitable that in understanding the cost of methods, the most direct measure of cost is to utilize timing data. However, timing data apply for such a narrow scope: a specific problem and (perhaps poor) implementation of the numerical algorithm. Despite that timing data are as truthful as you can possibly be, conclusions drawn from its use are highly specialized. This situation sows doubt in the generality and scope of these conclusions. For this reason, these kinds of studies are more common in engineering practice or when a specific problem will be solved many times (as in optimization, inverse problems, or uncertainty quantification) and the optimal method is important. We are left with the question of how general a conclusion can be made from these kinds of numerical experiments. This paper makes precise the types of conclusions that can be drawn from numerical experiments.

Consider the fictitious (yet realistic) scenario depicted in Figure 1 where we are comparing three methods applied to some target application. We plot the error $\mathcal{E}(N)$ versus the cost $\mathcal{C}(N)$ as the number of degrees of freedom N increases. While the optimal method is ambiguous and N is small, by increasing N, we eventually observe an asymptotic behavior and a firm conclusion can be made. We see that asymptotically, the curves representing methods 1 and 2 are parallel lines yet the slope of method 3 is not as steep. This could be for two reasons: either method 3 converges at a lower rate or its cost complexity is higher than the other methods. Either way, we must conclude that method 3 is not competitive despite the fact that for part of its pre-asymptotic behavior, it seemed to be the optimal method. We also observe that despite the fact that method 2 was the most expensive in the pre-asymptotic regime, it is the most efficient method in the limit as N becomes large. Furthermore, the difference between the method 1 and method 2 curves at some fixed cost represents how much more efficient method 2 is relative to method 1.

We propose that more general conclusions can be drawn from numerical data if those data are provably in the asymptotic regime. Additionally, we can detect when we are in the asymptotic regime by deriving what we call the *efficiency rate*, the slope of the accuracy versus cost functions expressed on a logarithmic scale for large number of degrees of freedom,

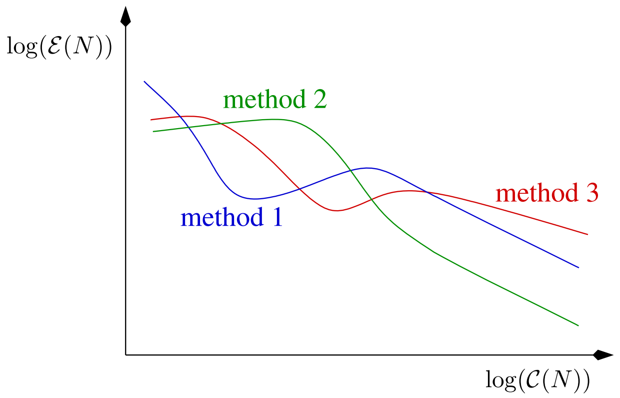


Figure 1. A comparison of the error ($\mathcal{E}(N)$) versus the cost ($\mathcal{C}(N)$) of three fictitious methods as the number of degrees of freedom (N) increases.

$$r = \lim_{N\to\infty}\left(\frac{\frac{\partial}{\partial N}\left(\log(\mathcal{E}(N))\right)}{\frac{\partial}{\partial N}\left(\log(\mathcal{C}(N))\right)}\right). \tag{1}$$

Relevant conclusions can only be drawn once cost versus accuracy comparisons from numerical experiments match that theoretical estimate. That is not to say that the efficiency of methods in the pre-asymptotic regime is not of interest. We are only pointing out that making general conclusions about the efficiency of methods before this limit is reached is premature.

In what remains of this paper, we illustrate the use of these ideas by comparing several methods applied to a specific problem. First, we specify the target problem as well as the methods we study and their corresponding error $\mathcal{E}(N)$ and cost $\mathcal{C}(N)$ models under the following assumptions:

- the best approximation estimates for smooth solutions on an elliptic problem,
- uniformly refined discrete spaces defined on a hypercube,[‡]
- weak form integration using full Gauss quadrature at each element (non-zero span),
- sum-factorization is not used to exploit the tensor-product structure of the basis,
- algebraic solution of the discrete system is obtained using a direct multi-frontal solver, and
- implementation of the method relies on the open-source, high-performance isogeometric solution framework we developed [10].

Then we derive the efficiency rates for each method and a range of discretization parameters. Finally, we show numerical results and draw conclusions, which demonstrate how we envision efficiency rates being used to assess relative efficiency of different methods.

2. BACKGROUND

We compare traditional finite element methods (FEA-G, [11]), B-spline-based finite element spaces commonly referred to as isogeometric analysis (IGA-G, [12]), and B-spline-based collocation methods (IGA-C, [13]). This test is not meant to be exhaustive, but rather a sampling of isogeometric methods, which have been in the spotlight in the past few years. The comparison is interesting because it sheds some light on the potential usefulness of function spaces with high inter-element continuity. The FEA-G methods represent the minimum continuity of a continuous Galerkin method and the IGA-G/IGA-C methods represent the maximum continuity, C^{p-1}. While B-spline spaces allow for regularity C^k, $0 \leqslant k \leqslant p-1$, we compare only the maximum and minimum regularities

[‡]Meshes that significantly differ from a logical hypercube lead to computational estimates that correspond to lower dimensional objects.

Table I. Summary of convergence rates and models for $\mathcal{E}(N)$ for our target methods in different norms.

	Error approximation		$\mathcal{E}(N)$	
Method	$\|\|\cdot\|\|_{L_2(\Omega)}$	$\|\|\cdot\|\|_{H_1(\Omega)}$	$\|\|\cdot\|\|_{L_2(\Omega)}$	$\|\|\cdot\|\|_{H_1(\Omega)}$
FEA-G	h^{p+1}	h^p	$p^{p+1}N^{-(p+1)/d}$	$p^pN^{-p/d}$
IGA-G	h^{p+1}	h^p	$N^{-(p+1)/d}$	$N^{-p/d}$
IGA-C (p even)	h^p	h^p	$N^{-p/d}$	$N^{-p/d}$
IGA-C (p odd)	h^{p-1}	h^{p-1}	$N^{-(p-1)/d}$	$N^{-(p-1)/d}$

and treat the conclusions as limiting cases. There are large differences in the constants/rates of the convergence/cost of these three methods, and thus, they provide a good testing ground for the key ideas presented in this paper.

We use a multi-frontal direct solver [14, 15] to solve the linear systems resulting from the application of our target methods to an elliptic PDE. We choose to use multi-frontal direct methods for this comparison for two reasons. First, it is simpler to analyze and develop full cost estimates. Iterative solvers are more computationally efficient for larger systems, provided that a good preconditioner can be obtained. However, their cost estimates involve an additional complication. The number of iterations required for convergence must be modeled and is problem-dependent. Second, multi-frontal direct solvers constitute the backend linear solver of popular scientific software packages. For example, UMFPACK [16] is the backend sparse solver for MATLAB[§] [17], octave [18], scilab [19], and scipy [20]. Therefore, it is important to understand the effect that the choice of discretization has on the cost of these commonly used solvers. In what remains of this section, we derive estimates for the error as well as costs in terms of the number of degrees of freedom for each method.

2.1. Convergence

With respect to the element size, IGA-G methods approximate smooth solutions at the same rate as FEA-G methods, but with better constants [21]. For fixed N, the constants can be orders of magnitude different, an effect that increases with the polynomial order. In the left two columns of Table I, we summarize convergence rates for elliptic problems[¶] in terms of the element size h. We can use these rates to develop expressions for $\mathcal{E}(N)$. For the traditional finite element function spaces (FEA-G) with N_e uniform elements in a structured mesh, of polynomial degree p, and spatial dimension d, we know that the number of degrees of freedom is

$$N \approx N_e p^d \tag{2}$$

as illustrated in Figure 2(a). Assuming a unit domain, we relate the number of elements to the mesh size h by $N_e = 1/h^d$. We combine these expressions to relate the number of degrees of freedom to the mesh size, $N = (p/h)^d$ or $h = pN^{-1/d}$. For the maximum continuity B-spline spaces, the expression is similar with one change: the number of degrees of freedom roughly equals the number of elements

$$N \approx N_e \tag{3}$$

as illustrated in Figure 2(b). This means that for IGA-G and IGA-C, $h = N^{-1/d}$. The two right columns of Table I summarize the convergence estimates of each method in terms of N.

[§]MATLAB interfaces to UMFPACK through the built-in ‘backslash’ command if the input linear system is sparse.

[¶]These estimates are for bounded polynomial orders, $p \leqslant 20$, and are relevant because most practical implementations to date use polynomial orders within this range. However, they do not account for the exponential behavior observed in spectral-like discretizations, such as the p-method or spectral elements.

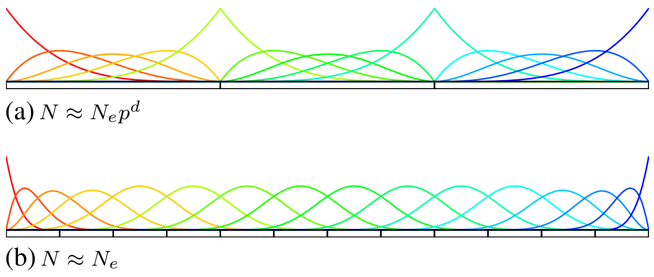


Figure 2. Higher-continuous B-spline spaces have roughly one degree of freedom per element. (a) $N \approx N_e p^d$ and (b) $N \approx N_e$.

2.2. *Assembly costs*

Here we develop assembly cost estimates for our test methods, which are based on FLOPs counts required to numerically integrate the weak form. The matrix assembly process for Galerkin methods (FEA-G and IGA-G) is

$$\begin{aligned}\text{Galerkin cost} &\propto \text{(loop elements)(loop quadrature)(double loop local basis)}\\ &\approx N_e N_q N_\ell^2\\ &\approx N_e (p+1)^d (p+1)^{2d}\\ &\approx N_e (p+1)^{3d}\\ &\approx N_e p^{3d}\end{aligned} \tag{4}$$

where the number of quadrature points per element is N_q and the number of local basis functions in each element is N_ℓ. We have assumed that full Gauss quadrature is used, $N_q = (p+1)^d$. The authors are aware of near-optimal quadrature rules [22, 23] for particular discretizations of IGA-G, which could be used to greatly reduce the assembly costs. Nevertheless, presently there are no general quadrature rules available for smooth functions of arbitrary order in general geometries. Additionally, as these estimates will show, in the asymptotic regime, the dominant computational cost is the solution of the linear system. Thus, more efficient quadrature for IGA-G would improve the lower-order terms in the overall cost. If we apply Equations (2) and (3) to Equation (4), then we conclude that the assembly cost for FEA-G is Np^{2d} and for IGA-G is Np^{3d}.

In contrast, the B-spline-based collocation method is far simpler,

$$\begin{aligned}\text{Collocation cost} &\propto \text{(loop collocation points)(loop local basis)}\\ &\approx N N_\ell\\ &\approx N(p+1)^d\\ &\approx Np^d\end{aligned} \tag{5}$$

We summarize the three estimates in the assembly column of Table II and plot timing data for the assembly of a stiffness matrix for our target methods and a range of polynomial orders in Figure 3. While all are linear in N, the assembly of IGA-G is greater than FEA-G by a constant on the order of p^d. This is also seen in the timing data. The estimates also suggest that FEA-G should be p^d times more expensive to assemble than IGA-C. This effect we see to lesser degree in the timing data. This is because the IGA-C estimates are coarse approximations where we neglect operations. For example, we are not considering that for our model problem, IGA-C requires second derivative computations while FEA-G and IGA-G only need first derivatives. Adding these effects produces more accurate yet specialized and complex estimates [3]. We keep the coarse approximations in favor of producing simple estimates, which can be used to develop intuition on the cost of the methods.

Table II. Summary of cost models $\mathcal{C}(N)$ for our target methods.

Method	$\mathcal{C}_A(N)$	$\mathcal{C}_S(N)$
FEA-G	Np^{2d}	$Np^{2d} + N^{3(d-1)/d}$
IGA-G	Np^{3d}	$N^{3(d-1)/d}p^3$
IGA-C	Np^d	$N^{3(d-1)/d}(p/2)^3$

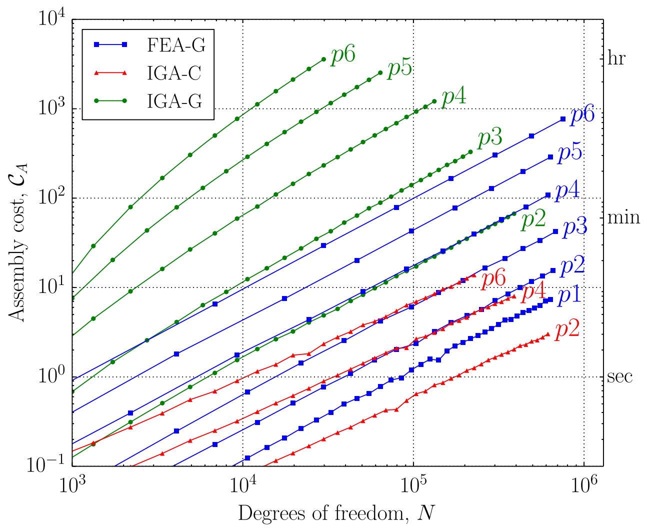


Figure 3. The scaling of the assembly cost of our three target methods with respect to the number of degrees of freedom.

2.3. Solve costs

We use a multi-frontal direct solver to solve the linear systems. In [2], we developed estimates for the computational complexity of multi-frontal direct solvers applied to linear systems, which result from the use of FEA-G and IGA-G. Here, we cite these estimates but also explain their derivation in an alternative and more intuitive manner. Then we extend them to include linear systems resulting from IGA-C.

The multi-frontal direct solver algorithm is an extension of the frontal algorithm, where multiple frontal matrices are used. This essentially means that the LU-factorization can be started in multiple places at once. This reduces the serial computational complexity of a direct solver while being amenable to parallelization. One such way to locate the starting points for the multiple fronts is to use the nested-dissection algorithm [24]. The basic operation in the nested-dissection algorithm is to select a minimal set of basis functions, called a *separator*, which, upon removal of the corresponding rows and columns in the system matrix, leaves two linear systems that are independent of each other and approximately the same size.

Figure 4 depicts the application of one level of nested-dissection to a 1D system matrix resulting from each of our target methods. The Galerkin methods have their corresponding function spaces super imposed to highlight their relationship to the separator, represented by the darkened region. For FEA-G, the separator width is one, $s = 1$. This is related to the fact that at element interfaces, there is a single non-zero basis function, making it an ideal choice for a separator. In contrast, IGA-G methods have a separator width $s = p$ due to increased overlap in basis functions. The IGA-C methods also have a p-dependent separator width, but to a lesser extent, $s = p/2$.

The selection of the separator is recursively applied on each submatrix, producing a binary tree structure of subproblems to be solved. The leaves of this tree are small linear systems. Then the algorithm proceeds by statically condensing the fully assembled part of the linear systems in the

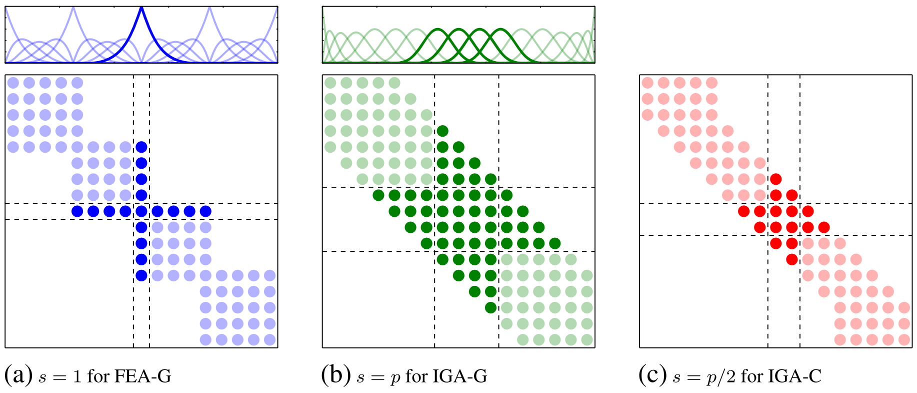


Figure 4. Illustration of the separator s each of our target methods. We depict sample 1D systems using a $p = 4$ basis and with the same number of degrees of freedom. The Galerkin methods are superimposed with their corresponding B-spline spaces to emphasize the correspondence of the separator to the basis function interactions. (a) $s = 1$ for FEA-G, (b) $s = p$ for IGA-G, and (c) $s = p/2$ for IGA-C.

leaves. Children with common parents then combine their Schur complements and the process repeats going up towards the root of the tree. In [2], we developed estimates by summing the cost of computing the Schur complements at each level of the elimination tree, where the costs at each level are directly related to the number of basis functions within the separator. Finally, we neglected the lower-order terms.

Here, we repeat the 3D, FEA-G estimate of computational time complexity

$$\mathcal{C}_S(N) = Np^6 + N^2. \tag{6}$$

We reason that the largest cost of the multi-frontal algorithm is due to the Schur complement computation at the root of the tree. The computation on the remaining subtrees is of similar order. The work to be carried out at the root of the elimination tree involves the solution of a dense linear system whose number of unknowns is equal to the size of the separator. For FEA-G on a structured mesh, the size of the separator is the number of degrees of freedom on a line in 2D or a face in 3D, which divides the mesh into two parts, that is, $(\sqrt[d]{N})^{d-1}$. We estimate the work by cubing the separator size,[‖] that is, $(N^{(d-1)/d})^3$. This explains the second term, N^2, of our 3D estimates. A potentially relevant source of additional work comes from statically condensing the fully assembled degrees of freedom (corresponding to bubble basis functions) on the leaves of the elimination tree, that is, $N_e((p-1)^d)^3 \approx N_e p^{3d} \approx Np^{2d}$. This explains the first term, Np^6, of our 3D FEA-G estimates. If the polynomial order is large, $p^6 \gg N$, this would constitute a nontrivial amount of work to that of the root of the tree.

The multi-dimensional separator for IGA-G is the number of degrees of freedom on a line in 2D or a face in 3D, which divides the mesh into two parts multiplied by the 1D separator width p, that is, $p(\sqrt[d]{N})^{d-1}$. This leads to a work estimate for the root of the tree of $(p(N^{(d-1)/d}))^3$, or as we report in 3D, N^2p^3. There is no term analogous to the first term in the FEA-G estimates, because the leaves of the elimination tree do not contain any fully assembled degrees of freedom. Linear systems resulting from IGA-C share a similar difficulty in the performance of the direct solver. However, the separators here are of width $p/2$. This makes the work estimate for the root of the tree $N^{3(d-1)/d}(p/2)^3$.

We emphasize that the additional cost of solving the linear systems resulting from IGA-G and IGA-C is not a consequence of higher exponents of N but rather potentially large p-dependent constants. In Table II, we summarize both assembly and solve cost estimates, and in Figure 5,

[‖]The complexity of a dense linear solve is $\mathcal{O}(n^3)$ for a matrix of size $n \times n$ [25].

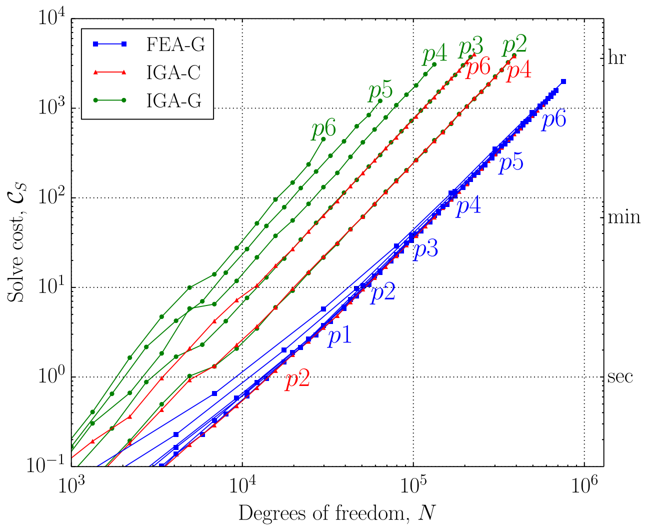


Figure 5. The scaling of solution cost measured in seconds (multi-frontal direct solver) with respect to the number of degrees of freedom of our three target methods applied to a 3D problem.

we plot timing results for the solution of a stiffness matrix using the PETSc [26–28] interface to MUMPS [29, 30]. The solve cost for all the methods in 3D scales quadratically with N, as our estimates predict. Remarkably, all FEA-G methods require the same solve time for sufficiently large but fixed N, regardless of the polynomial order p. This is due to the p-independence of the size of the separator at every level (and particularly at the root) of the elimination tree. As our estimates predict, the IGA-G/IGA-C methods are more expensive than FEA-G by a constant because of their separator's p-dependence. Of particular interest is that IGA-C ($p = 6$) has the same solve cost as IGA-G ($p = 3$). We see the same effect for IGA-C ($p = 4$) and IGA-G ($p = 2$). This effect is expected from an understanding that while both separators depend on p, the IGA-C separator is half as large.

3. EFFICIENCY RATES

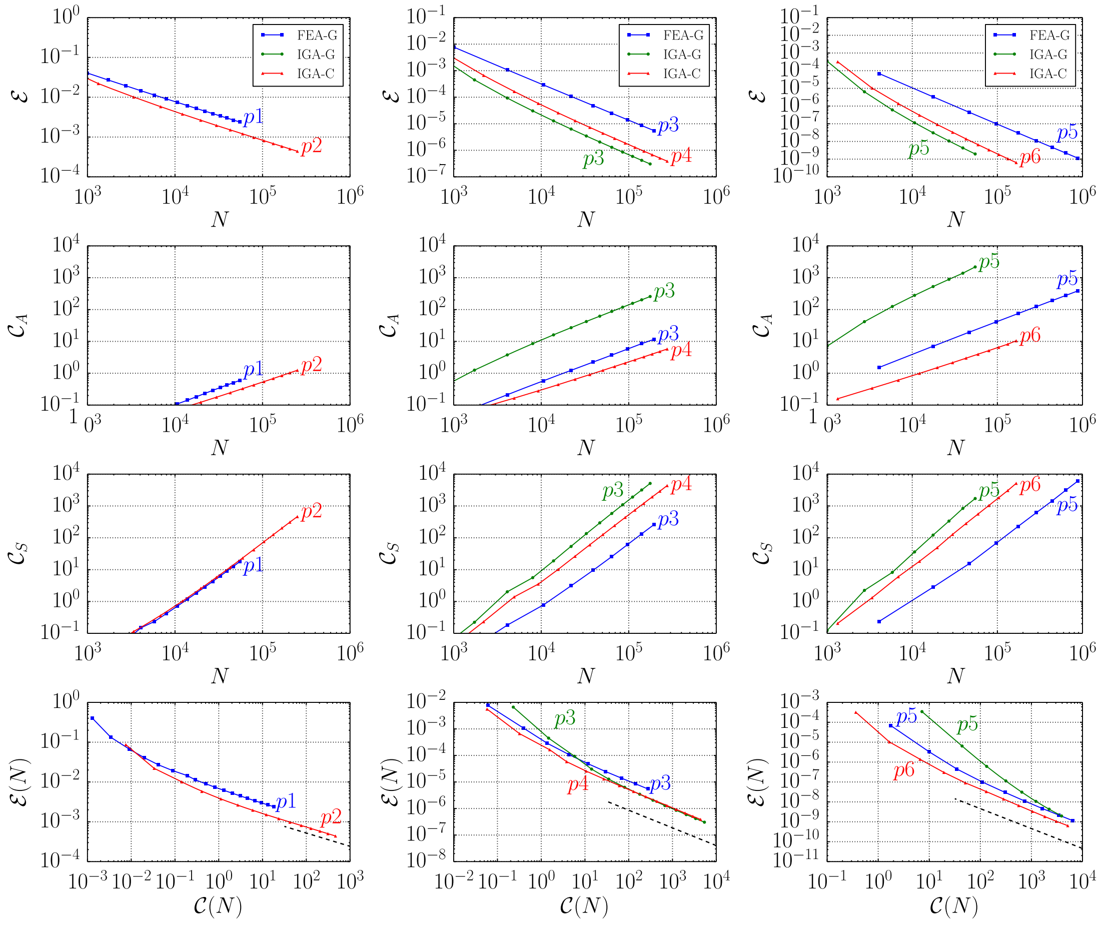
We use Equation (1) and the information in Tables I and II to determine the efficiency rates of the target methods. Here we show the derivation of the rate for FEA-G in the L_2-norm when $d = 3$:

$$\begin{aligned}
\mathcal{E}(N) &= \mathcal{O}(p^{p+1}N^{-(p+1)/3}) \\
\mathcal{C}(N) &= \mathcal{C}_A(N) + \mathcal{C}_S(N) \\
&= \mathcal{O}(Np^6) + \mathcal{O}(Np^6 + N^2) \\
r &= \lim_{N\to\infty}\left(\frac{\frac{\partial}{\partial N}(\log(\mathcal{E}(N)))}{\frac{\partial}{\partial N}(\log(\mathcal{C}(N)))}\right) \\
&= \lim_{N\to\infty}\left(-\frac{(Np + N + 2p^7 + 2p^6)}{6(N + p^6)}\right) \\
&= -\frac{p+1}{6}
\end{aligned}$$

The rates for the remaining methods and error measures are derived in similar fashion and summarized in Table III.

Table III. Efficiency rates for our targets methods applied to the model problem for $d = 3$.

	Efficiency rate, r	
Method	$\|\|\cdot\|\|_{L_2(\Omega)}$	$\|\|\cdot\|\|_{H_1(\Omega)}$
FEA-G	$-(p+1)/6$	$-p/6$
IGA-G	$-(p+1)/6$	$-p/6$
IGA-C (even p)	$-p/6$	$-p/6$
IGA-C (odd p)	$-(p-1)/6$	$-(p-1)/6$



(a) $r = 1/3$ (b) $r = 2/3$ (c) $r = 1$

Figure 6. Comparison of methods for the low frequency problem ($\omega = 1$). The first three rows of plots constitute the error $\mathcal{E}$, assembly cost $\mathcal{C}_A$, and solve cost $\mathcal{C}_S$, in terms of degrees of freedom N. The efficiency rate r is constant in each column. The fourth row is the composition of the three rows, $\mathcal{E}(N)$ (in the L_2 norm) versus $\mathcal{C}(N)$ (in seconds). The slope of the black dashed line corresponds to the expected efficiency rate. (a) $r = 1/3$, (b) $r = 2/3$, and (c) $r = 1$.

4. NUMERICAL EXPERIMENTS

To illustrate the relationship of error to cost, we solve an elliptic problem on the unit domain of the form

$$\begin{cases} -\nabla \cdot (\nabla u) + u = f & \text{on } \Omega \\ u = 0 & \text{on } \Gamma \end{cases} \tag{7}$$

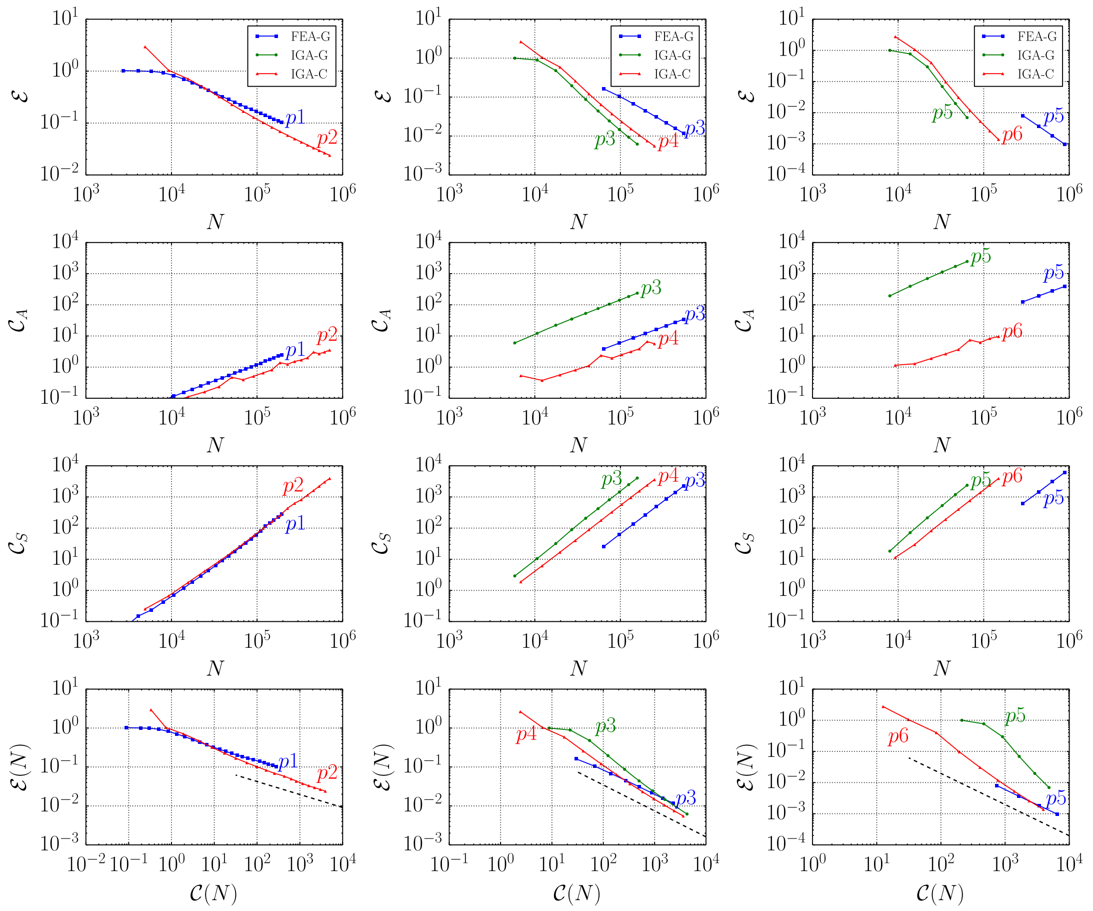
where $\mathbf{x} \in \Omega = [0,1]^d$ and Γ is the boundary of Ω. We choose a solution

$$u = \prod_{i=1}^{d} \sin(2\pi\omega x_i) \tag{8}$$

where $x_i = [\mathbf{x}]_i$ and ω refers to some spatial frequency. Then we use the PDE to compute the associated forcing

$$f = (1 + d(2\pi\omega)^2) \prod_{i=1}^{d} \sin(2\pi\omega x_i) \tag{9}$$

The motivation behind such a problem is to have a non-polynomial exact solution that is dimension independent and for which we can control the spatial frequency of the solution. The test problem is also the best-case scenario for the use of higher-continuous basis functions because of an infinite number of continuous derivatives in the exact solution. We perform a numerical study of the error and costs for each of our target methods for $d = 3$ by varying the number of elements in each discretization. We measure error in a relative L_2 norm and cost in seconds. The studies consist of



(a) $r = 1/3$ (b) $r = 2/3$ (c) $r = 1$

Figure 7. Comparison of methods for the high-frequency problem ($\omega = 10$). The first three rows of plots constitute the error $\mathcal{E}$, assembly cost $\mathcal{C}_A$, and solve cost $\mathcal{C}_S$, in terms of degrees of freedom N. The efficiency rate r is constant in each column. The fourth row is the composition of the three rows, $\mathcal{E}(N)$ (in the L_2 norm) versus $\mathcal{C}(N)$ (in seconds). The slope of the black dashed line corresponds to the expected efficiency rate. (a) $r = 1/3$, (b) $r = 2/3$, and (c) $r = 1$.

h-refinements where we only use odd numbers of elements to reduce serendipitous benefits due to alignment of the mesh to features of the solution. Finally, we consider two cases, one where the spatial frequency of the solution is low ($\omega = 1$) and one where it is high ($\omega = 10$). The experiments were conducted on a desktop machine with a 3.07 GHz Intel Core i7 950 processor, 8 megabytes of L3 cache and a cache line of 64 bytes, a 4.8 GT/s Intel QPI memory interconnect, and 12 gigabytes of 1066 MHz DDR3 memory.

In Figure 6, we present low frequency results for a sequence of efficiency rates, $r = \{1/3, 2/3, 1\}$. Each method is distinctly colored and annotated with its corresponding polynomial order. The IGA-C results are a polynomial order higher because they converge in the L_2 norm one order lower for equal polynomial order. The first plot in the sequence is the classical convergence plot, however, in terms of the number of degrees of freedom N. As p increases, IGA-G and IGA-C have increasingly better approximability while asymptotically converging at the same rate as FEA-G. The second plot in the sequence is the assembly cost per degree of freedom. For equal order schemes, the assembly benefit of IGA-C is not seen until $p = 6$. The third plot is the solve cost per degree of freedom. In this plot, we see that both IGA-G and IGA-C have higher solve costs, an effect that grows with p. The final plot in the sequence is the composition of the previous three, depicting the error as a function of the cost. The black dashed line represents the asymptotic efficiency rate, which we derived in Section 3 and label in the subcaption of each figure. While the pre-asymptotic behavior varies, one can observe that the final outcome is that there is no large (order of magnitude) difference between methods. This is because despite favorable constants in the convergence of IGA-G and IGA-C, this return is diminished by unfavorable constants in the costs.

In the high-frequency case (Figure 7), we draw similar conclusions with one main difference. The IGA-G results appear to have a higher efficiency rate than expected. This is particularly evident in the bottom plot of Figure 7(c). While the assembly costs are linear in N and the solve costs are quadratic, the enormous constants on assembly costs dwarf the extra power of N in the solve costs. Thus, the costs of IGA-G appear linear in N and the overall efficiency rate appears better than it should. The correct efficiency rate will only be achieved for much larger problems once the solve time dominates the overall computation. In contrast, the FEA-G results reach the asymptotic regime faster than those of IGA-G and IGA-C.

5. CONCLUSIONS

In this paper, we compare the computational efficiency of alternative methods. The main advice is that numerical methods have multiple components (convergence, assembly costs, and solve costs) where each scales differently as the number of degrees of freedom increases. We derive a metric that can be used to determine when a method has reached the final asymptote, and we reason that this is the point at which it is safe to draw more general conclusions about a method and its implementation.

To illustrate the point, we have compared three methods for finding smooth solutions to elliptic PDEs, which explore the usefulness of higher inter-element continuity in the choice of basis functions. The implementation uses full Gauss quadrature in each element and uses a multi-frontal direct solver algorithm to solve the discrete algebraic system. We see that despite favorable constants in the error per degree of freedom, the use of higher-continuous basis functions incurs additional costs. Asymptotically, we see that there is not a large difference between methods, suggesting that C^k spaces may not be worth the restrictions, which come with their use. This is, of course, a conclusion that is specific to the test problem solved as well as the open-source implementation. While we have carried out diligence in producing an efficient software, PetIGA [10, 31] is general and not thoroughly optimized and tuned for each of the methods. Additional comparisons could be performed with a tuned or alternative implementation. We feel that given how closely the methods perform, this is unwarranted.

This is not, however, the final word on the subject. Comparisons also could include convergence measured in the H_1 norm, which is more favorable for IGA-C. Furthermore, we have only considered elliptic problems, and ignored other aspects of numerical methods such as stability. However, we conclude from this research that higher inter-element continuity as in B-splines and Hermite polynomials is a property that should be used sparingly.

ACKNOWLEDGEMENTS

We would like to thank Rio Yokota and Jack Poulson for their suggestions, which improved the quality of this work.

REFERENCES

1. Collier N, Dalcin L, Pardo D, Calo VM. The cost of continuity: performance of iterative solvers on isogeometric finite elements. *SIAM Journal on Scientific Computing* 2013; **35**(2):A767–A784.
2. Collier N, Pardo D, Dalcin L, Paszynski M, Calo VM. The cost of continuity: a study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering* 2012; **213–216**(0):353–361.
3. Schillinger D, Evans JA, Reali A, Scott MA, Hughes TJR. Isogeometric collocation: cost comparison with galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Computer Methods in Applied Mechanics and Engineering* 2013; **267**:170–232.
4. Löhner R. Improved error and work estimates for high-order elements. *International Journal for Numerical Methods in Fluids* 2013; **72**(11):1207–1218.
5. Brown J. Efficient nonlinear solvers for nodal high-order finite elements in 3d. *Journal of Scientific Computing* 2010; **45**(1–3):48–63.
6. Demkowicz L, Oden JT, Rachowicz W, Hardy O. Toward a universal $h - p$ adaptive finite element strategy, part 1. Constrained approximation and data structure. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**(12):79 – 112.
7. Oden JT, Demkowicz L, Rachowicz W, Westermann TA. Toward a universal $h - p$ adaptive finite element strategy, part 2. A posteriori error estimation. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**(12): 113–180.
8. Rachowicz W, Oden JT, Demkowicz L. Toward a universal $h - p$ adaptive finite element strategy part 3. Design of h-p meshes. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**(12):181–212.
9. Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM* 2009; **52**(4):65–76.
10. Dalcin L, Collier N. Petiga: high performance isogeometric analysis, 2012. (Available from: https://bitbucket.org/dalcinl/petiga) [Accessed on August 2014].
11. Hughes TJR. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.
12. Cottrell JA, Hughes TJR, Bazilevs Y. *Isogeometric Analysis: Toward Unification of CAD and FEA*. John Wiley and Sons: West Sussex, United Kingdom, 2009.
13. Auricchio F, Beirão da Veiga L, Hughes TJR, Reali A, Sangalli G. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences* 2012; **20**(11):2075–2107.
14. Duff IS, Reid JK. The multifrontal solution of indefinite sparse symmetric linear. *ACM Transactions on Mathematical Software* 1983; **9**:302–325.
15. Duff IS, Reid JK. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal on Scientific and Statistical Computing* 1984; **5**(3):633–641.
16. Davis TA. Algorithm 832: Umfpack v4.3 – an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software* 2004; **30**(2):196–199.
17. MATLAB. The MathWorks Inc.: Natick, Massachusetts, 2014.
18. Eaton JW. *GNU Octave Manual*. Network Theory Limited, 2002. (Available from: https://www.gnu.org/software/octave/octave.pdf) [Accessed on August 2014].
19. Scilab Enterprises. *Scilab: Free and Open Source Software for Numerical Computation*. Scilab Enterprises: Orsay, France, 2012.
20. Jones E, Oliphant T, Peterson P, *et al.* SciPy: open source scientific tools for Python, 2001.
21. Sangalli G, Hughes TJR, Beirão da Veiga L, Cottrell JA, Bazilevs Y. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences* 2006; **16**(07): 1031–1090.
22. Auricchio F, Calabrò F, Hughes TJR, Reali A, Sangalli G. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 2012; **249–252**(0):15–27.
23. Hughes TJR, Reali A, Sangalli G. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(58):301–313.
24. George A. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis* 1973; **10**(2): 345–363.
25. Golub GH, Van Loan CF. *Matrix Computations* (3rd edn). The Johns Hopkins University Press: Baltimore, Maryland, 1996.
26. Balay S, Brown J, Buschelman K, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Smith BF, Zhang H. PETSc users manual. *Technical Report ANL-95/11 - Revision 3.4*, Argonne National Laboratory, 2013.
27. Balay S, Brown J, Buschelman K, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Smith BF, Zhang H. PETSc Web page, 2013. (Available from: http://www.mcs.anl.gov/petsc) [Accessed on August 2014].

28. Balay S, Gropp WD, McInnes LC, Smith BF. Efficient management of parallelism in object oriented numerical software libraries. In *Modern Software Tools in Scientific Computing*, Arge E, Bruaset AM, Langtangen HP (eds). Birkhäuser Press: Boston, 1997; 163–202.
29. Amestoy PR, Duff IS, Koster J, L'Excellent J-Y. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal of Matrix Analysis and Applications* 2001; **23**(1):15–41.
30. Amestoy PR, Guermouche A, L'Excellent J-Y, Pralet S. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing* 2006; **32**(2):136–156.
31. Calo VM, Collier N, Dalcin L. PetIGA: high-performance isogeometric analysis. *arxiv,* (1305.4452), 2013. (Available from: http://arxiv.org/abs/1305.4452) [Accessed on August 2014].