Fast Integration Techniques in 3D Boundary Elements *

I.G. Graham† W. Hackbusch‡ S.A. Sauter§

1 Introduction

Boundary element methods are popular in numerical engineering, especially for solving classical PDEs on complicated and or/infinite 3D regions, where (for example in scattering problems) they are often preferred to finite elements. More generally, they can be combined with finite elements in order to efficiently handle problems in which different PDEs on different parts of a domain are coupled together. Boundary elements can be used on the boundaries of linear homogeneous regions, with (domain) finite element approximation in nonlinear or inhomogeneous regions. There is a large literature on this topic in which fast solution strategies based on this type of decomposition have been widely reported (e.g., [5]).

The inclusion of boundary elements in such a coupled solution strategy poses two problems not found in the stand-alone finite element method: the calculation of complicated (sometimes singular) integrals to form the stiffness matrix, and the solution of full systems. Much recent work has been done on system solution, yielding almost optimal solvers with close to $O(N)$ complexity, where N is the number of degrees of freedom. Less work has been done on optimising the assembly of the stiffness matrix which (if it is fully assembled) costs $C_1N^2 + O(N)$ operations, where C_1 is moderately large and driven by the number of kernel evaluations required by the integrator. In modern boundary element codes the cost of stiffness matrix assembly can be the principle bottleneck. In this paper we describe a new procedure ([2, 3]) for computing approximations to Galerkin stiffness matrices using only $N^2+O(N)$ kernel evaluations (i.e. $C_1 = 1$). The approximate stiffness matrix is guaranteed to be accurate enough so that the corresponding numerical solution inherits the same stability and convergence properties as are enjoyed by conventional Galerkin methods.

A key point in our method is the decomposition of the (closed) surface $\Gamma \subset \mathbb{R}^3$ on which the integral equation is posed into a number of smooth closed components $\{\Gamma_\ell : \ell = 1, \ldots L\}$ (with the interiors of the Γ_ℓ assumed pairwise disjoint). As is typical in the description of 2D manifolds in 3D space, each Γ_ℓ is here assumed to be parametrised by a polygonal planar chart $\tilde{\Gamma}_\ell$. We shall devise fast integration techniques on the $\tilde{\Gamma}_\ell$, which then induce corresponding fast methods on Γ. One of the

*this work was supported by British Council / DAAD ARC grant 869.

†(igg@maths.bath.ac.uk) Mathematical Sciences, University of Bath, Bath, BA2 7AY, U.K.

‡(wh@numerik.uni-kiel.de), Praktische Mathematik, Universität Kiel, 24098 Kiel, Germany

§(sas@mathematik.uni-leipzig.de), Mathematisches Institut, Universität Leipzig, 04109 Leipzig, Germany

novel contributions of this paper is to provide a more efficient way of implementing the new algorithm when several parametrised pieces of a smooth surface have to be stitched together. In order to explain the basic principles of the method and to present the novel contribution in the least technical context we assume here that Γ is a $\mathcal{C}^\infty$ surface. Note that this is by no means essential and the technical details for general piecewise smooth surfaces are given in ([2, 3, 4]). The new implementation described here can also be used for integration over smooth portions of such piecewise smooth surfaces.

We shall consider general boundary integral equations of the form

$$(\lambda I + \mathcal{K})u(\mathbf{x}) := \lambda u(\mathbf{x}) + \int_\Gamma k(\mathbf{x},\mathbf{y})u(\mathbf{y})dy = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma \ , \tag{1.1}$$

where the real scalar λ and function f are given and u is to be found. For convenience we assume that $\tilde{\Gamma} := \cup_\ell \tilde{\Gamma}_\ell$ forms the surface of a polyhedron (although, in principle any standard chart system would also do) and we assume that the parametrisation is via a bi-Lipschitz bijectiion $\boldsymbol{\eta} : \tilde{\Gamma} \to \Gamma$. (Only action of the mapping $\boldsymbol{\eta}$ is explicitly required in practice.)

A standard boundary element method for (1.1) then begins by triangulating $\tilde{\Gamma}$ using a family of meshes $\tilde{\mathcal{T}}_h$, with parameter $h \to 0$ denoting the mesh diameter. As the meshes are refined we assume that they remain *shape-regular* (see, e.g. [2]). We denote the mesh nodes by $\{\tilde{\mathbf{x}}_p : \ p \in \mathcal{N}\}$ (where $\mathcal{N}$ is an index set), and we denote the corresponding nodal basis for the continuous piecewise linear functions with respect to $\tilde{\mathcal{T}}_h$ on $\tilde{\Gamma}$ by $\{\tilde{\phi}_p : \ p \in \mathcal{N}\}$. The induced functions $\phi_p := \tilde{\phi}_p \circ \boldsymbol{\eta}^{-1} : \Gamma \to \mathbb{R}$ are used to approximate (1.1) using Galerkin's method. The ϕ_p are piecewise smooth with respect to the curvilinear triangular mesh on Γ with triangles $\boldsymbol{\eta}(\tau) : \ \tau \in \tilde{\mathcal{T}}_h$ and nodes $\mathbf{x}_p := \boldsymbol{\eta}(\tilde{\mathbf{x}}_p) : \ p \in \mathcal{N}$.

The chief computational task in assembling the resulting stiffness matrix is computing the N^2 entries (where $N = |\mathcal{N}|$):

$$K_{p,q} = \int_\Gamma \int_\Gamma k(\mathbf{x},\mathbf{y})\phi_q(\mathbf{y})\phi_p(\mathbf{x})dydx \ , \quad p,q \in \mathcal{N}. \tag{1.2}$$

Our new method adaptively partitions K into a "conventional Galerkin" part in which $K_{p,q}$ is computed by conventional quadrature and another part which is done by very cheap unconventional rules (using typically between 10 and 100 times fewer kernel evaluations). The complexity theory of the resulting "hybrid" algorithm (see §3) shows that under appropriate conditions the cheap part dominates. In §4 we give a new set of numerical results (for computation of the 3D harmonic Neumann-Dirichlet map) which show that the algorithm can compute more than 90% of the matrix K using the cheap method, without damaging the underlying convergence properties of the numerical solution.

To motivate the quadrature techniques for (1.2), let us first of all ignore the fact that the "kernel function" k is normally singular at $\mathbf{x} = \mathbf{y}$ and imagine that k is globally smooth. Then both conventional and unconventional approaches to computing (1.2) can be obtained as two applications of rules which compute, for general smooth F, the "fundamental integral":

$$\int_\Gamma F(\mathbf{x})\phi_p(\mathbf{x})dx = \int_{\tilde{\Gamma}} \tilde{F}(\tilde{\mathbf{x}})\tilde{\phi}_p(\tilde{\mathbf{x}})d\tilde{x} \ . \tag{1.3}$$

Here $\tilde{F}(\tilde{\mathbf{x}}) := F(\boldsymbol{\eta}(\tilde{\mathbf{x}}))g(\tilde{\mathbf{x}})$, g is the "Gram determinant" of $\boldsymbol{\eta}$ appearing in the area element transformation $dx = g(\tilde{\mathbf{x}})d\tilde{x}$ and we have used the fact that $\phi_p(\boldsymbol{\eta}(\tilde{\mathbf{x}})) = \tilde{\phi}_p(\tilde{\mathbf{x}})$. Note our assumptions ensure that g is a smooth and uniformly positive function on each $\tilde{\Gamma}_\ell$ (although g may not be smooth globally).

The conventional approach to (1.3) separates the right-hand side into a sum of integrals over each of the planar triangles in supp $\tilde{\phi}_p$, on which standard rules (for example from [8]) can be applied. This approach results in a fairly high complexity in terms of kernel evaluations. For example, a popular rule which will be exact for (1.3) when $\tilde{F}$ is a bivariate polynomial on each triangle requires evaluations of $\tilde{F}$ at the nodes, edge mid-points and centroids of a triangle. Recalling that N is the number of nodes in the mesh, there are about $2N$ triangles and $3N$ edges and so use of this rule to compute (1.3) for all p requires about $6N$ evaluations of F. Computing all elements of the matrix K by iterating this rule would then require about $36N^2$ kernel evaluations. Higher order conventional rules will typically require CN^2 evaluations with C higher still.

However we can obtain the same degree of precision in $N^2+O(N)$ kernel evaluations using the following more unconventional approach. Note that the main reason for separating (1.3) into a sum over triangles is the fact that $\tilde{\phi}_p$ is only triangle-wise smooth. However $\tilde{\phi}_p$ is simple enough so that it can be integrated easily against any polynomial over all of its support. Since k is (temporarily assumed) globally smooth, this suggests that we should devise rules for (1.3) which work well for smooth $\tilde{F}$, and in which $\tilde{\phi}_p$ is treated as a weight-function. This gives us the freeedom to use evaluations of F at any points, and if we use only *nodal* evaluations then we can compute (1.3) for all p using only N evaluations of F. Iterating this means that K can be computed using exactly N^2 evaluations, namely $k(\mathbf{x}_p, \mathbf{y}_q)$, $p, q \in \mathcal{N}$. Note also that, in contrast to conventional rules, the N^2 evaluation count remains the same irrespective of the degree of precision required, and each nodal evaluation of k may be used in the computation of $K_{p,q}$ for several p, q.

Our task then is to devise quadrature approximations of the right-hand side of (1.3) which take some subset of the nodes $\{\tilde{\mathbf{x}}_p : p \in \mathcal{N}\}$ as quadrature points and which are exact for polynomials of some specified degree. However, since we are dealing here with general meshes, and therefore quadrature rules with general abscissae (in 2D regions), the weights of these quadrature rules are not known *a priori* and have to be computed as part of the algorithm. For example, suppose supp $\tilde{\phi}_p$ is entirely contained inside one of the charts $\tilde{\Gamma}_\ell$ and suppose we wish to construct a quadrature rule which is exact for (1.3) when $\tilde{F}$ is a bivariate polynomial on $\tilde{\Gamma}_\ell$. Then we would choose at least 6 nodes $\{\tilde{\mathbf{x}}_j : j \in J_p\}$ in $\tilde{\Gamma}_\ell$ (with J_p denoting a suitable index set) and (using the method of undetermined coefficients), seek weights $\{\tilde{w}_j : j \in J_p\}$ such that the equation

$$\sum_{j \in J_p} \tilde{w}_j \tilde{\Pi}_i(\tilde{\mathbf{x}}_j) = \int_{\tilde{\Gamma}_\ell} \tilde{\Pi}_i(\tilde{\mathbf{x}}) \tilde{\phi}_p(\tilde{\mathbf{x}}) d\tilde{x} \ , \tag{1.4}$$

where $\{\tilde{\Pi}_i : i = 1, \ldots 6\}$ are a suitable basis for the bivariate quadratic polynomials (e.g. $1, \tilde{x}_1, \ldots \tilde{x}_2^2$, in local coordiantes on $\tilde{\Gamma}_\ell$). This yields 6 equations to solve for the weights $\{\tilde{w}_j : j \in J_p\}$.

More generally, if supp $\tilde{\phi}_p$ overlaps with two charts $\tilde{\Gamma}_\ell$ and $\tilde{\Gamma}_{\ell'}$ say, then $\tilde{\Gamma}_\ell \cup \tilde{\Gamma}_{\ell'}$ may not be a smooth subset of $\tilde{\Gamma}$, even though Γ itself is smooth. In [2, 3] this situation

was dealt with by separating supp $\tilde{\phi}_p$ into smooth components, each from one $\tilde{\Gamma}_\ell$, then selecting at least 6 nodal quadrature points in each relevant $\tilde{\Gamma}_\ell$, and solving systems of form (1.4) in each component for the weights. This approach is natural when an edge of parameter space $\tilde{\Gamma}$ corresponds to a true edge in Γ. However when a smooth (part of) Γ is parametrised by a piecewise smooth (part of) $\tilde{\Gamma}$ it is more natural to use an alternative approach, based on overlapping subdivisions of Γ, which is described in §2. In any case it should be clear that we can in principle construct node-based rules of precision 2 (or indeed any precision) for (1.3) and that the computation of the corresponding weights requires the solution of (possibly underdetermined) systems such as (1.4). The theory of solvability of these systems is an interesting problem in its own right and in [3] stability is demonstrated in a wide range of practical situations for rules with degree of precision 1 and 2. Computing one (or several) such node-based rules for all $p \in \mathcal{N}$ is an $O(N)$ process.

Now recall that the "real" kernel function k in (1.1) is a fundamental solution of a PDE (or a derivative of a fundamental solution) and so it can be expected to blow up as $\mathbf{x} \to \mathbf{y}$. To allow for this we shall assume here that $k(\mathbf{x}, \mathbf{y})$ is $\mathcal{C}^\infty$ for $\mathbf{x} \neq \mathbf{y}$ (more general kernels are allowable [2, 3]), and also that there exists $\alpha > 0$ such that for all integers $m \geq 0$, there exists a constant $B_m > 0$ such that

$$|D^m\{k(\boldsymbol{\eta}(\tilde{\mathbf{x}}), \boldsymbol{\eta}(\tilde{\mathbf{y}}))\}| \leq B_m |\tilde{\mathbf{x}} - \tilde{\mathbf{y}}|^{-\alpha - m} \ , \quad \text{for } \tilde{\mathbf{x}} \neq \tilde{\mathbf{y}} \ , \quad \tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \tilde{\Gamma} \ . \tag{1.5}$$

Here D^m denotes any mth order partial differential operator with respect to $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$. Most kernels in practice satisfy this and the general theory in [2] shows that, in order for the quadrature error in computing K not to damage the overall convergence rate of the Galerkin scheme, the degree of precision of the quadrature rules needed to compute $K_{p,q}$ must be higher if supp $\tilde{\phi}_p$ is close to supp $\tilde{\phi}_q$ compared to when they are well-separated. Moreover special regularising transforms (e.g. [1]) are needed when supp $\tilde{\phi}_p \cap$ supp $\tilde{\phi}_q \neq \emptyset$. But (most importantly for this paper), only low degree of precision rules (e.g. 1 or 2) are typically needed when the distance between supp $\tilde{\phi}_p$ and supp $\tilde{\phi}_q$ is $O(1)$ (as $h \to 0$). This motivates the hybrid algorithm given in §3, which uses cheap low order node-based rules when supp $\tilde{\phi}_p$ and supp $\tilde{\phi}_q$ are far enough apart, and uses the conventional approach elsewhere. In the experiments in §4 we see that in standard applications the cheap method strongly dominates the computation, yielding a complexity of exactly N^2 kernel evaluations (at pairs of node-points) together with a additional $O(N)$ evaluations due to the small part of the matrix done by conventional methods. The solutions computed by the cheap method exhibit the same rate of convergence as those computed by the conventional method. To the cost of the cheap method we should also add $O(N)$ operations needed to compute the weights of the node-based rules in the first phase of the algorithm.

2 Overlapping Decompositions

In this section we describe an efficient method for computing node-based rules when $\tilde{\Gamma}$ contains edges which do not correspond to edges of Γ. We assume that each $\tilde{\Gamma}_\ell$ is extended to a slightly larger planar polygon $\tilde{\Gamma}^e_\ell$, with the distance between the boundaries of $\tilde{\Gamma}_\ell$ and $\tilde{\Gamma}^e_\ell$ being bounded below by $\epsilon > 0$. Assume also that each parametrisation map $\boldsymbol{\eta}_\ell := \boldsymbol{\eta}|_{\tilde{\Gamma}_\ell}$ can be extended to a smooth map $\boldsymbol{\eta}^e_\ell$ which takes $\tilde{\Gamma}^e_\ell$ into a neighbourhood Γ^e_ℓ of Γ_ℓ. (Note that such an extended chart system would normally be part

of the geometric description of a smooth manifold.) In order to use these extended charts in the computation of (1.3), we need also to assume that the inverse maps $(\boldsymbol{\eta}_\ell^e)^{-1} : \Gamma_\ell^e \to \tilde{\Gamma}_\ell^e$ are known, and that, for each $p \in \mathcal{N}$, there exists ℓ (depending on p) such that supp $\phi_p \subset \Gamma_\ell^e$. Clearly the latter assumption holds if the overlap ϵ is sufficiently large compared to h (e.g. $\epsilon = Ch$ for a suitably large constant C would suffice). Then we can write, analogously to (1.3),

$$\int_\Gamma F(\mathbf{x})\phi_p(\mathbf{x})dx = \int_{\tilde{\Gamma}_\ell^e} \tilde{F}^e(\tilde{\mathbf{x}}^e)\phi_p(\boldsymbol{\eta}_\ell^e(\tilde{\mathbf{x}}^e))d\tilde{x}^e \ , \tag{2.1}$$

with $\tilde{F}^e(\tilde{\mathbf{x}}^e) := F(\boldsymbol{\eta}_\ell^e(\tilde{\mathbf{x}}^e))g_\ell^e(\tilde{\mathbf{x}}^e)$, and g_ℓ^e denoting the Gram determinant of $\boldsymbol{\eta}_\ell^e$.

Now since $\tilde{\Gamma}_\ell^e$ is planar, we can apply the procedure introduced in the previous section to generate node-based rules for (2.1). To compute a rule with degree of precision d, select at least $(d+1)(d+2)/2$ nodes $\{\mathbf{x}_j : j \in J_p\}$ on Γ_ℓ^e, pull them back to the set $\{\tilde{\mathbf{x}}_j^e = (\boldsymbol{\eta}_\ell^e)^{-1}(\mathbf{x}_j) : j \in J_p\}$ of nodes on $\tilde{\Gamma}_\ell^e$, and then seek weights $\tilde{w}_j$ so that

$$\sum_{j\in J_p} \tilde{w}_j \tilde{\Pi}_i(\tilde{\mathbf{x}}_j^e) = \int_{\tilde{\Gamma}_\ell^e} \tilde{\Pi}_i(\tilde{\mathbf{x}}^e)\phi_p(\boldsymbol{\eta}_\ell^e(\tilde{\mathbf{x}}^e))d\tilde{x}^e \ , \quad i = 1, \ldots, (d+1)(d+2)/2 \ , \tag{2.2}$$

where the $\tilde{\Pi}_i$ are a basis for the polynomials of degree d on the planar domain $\tilde{\Gamma}_\ell^e$. This procedure yields a (possibly underdetermined) linear system for $\tilde{\boldsymbol{w}}$:

$$M\tilde{\boldsymbol{w}} = \tilde{\boldsymbol{b}} \ , \tag{2.3}$$

where M is the trivially computed Vandermonde matrix containing the values of the basis functions evaluated at the quadrature points. From (2.2) and the definition of ϕ_p, each element of the right-hand side $\tilde{\boldsymbol{b}}$ can be written

$$\tilde{b}_i = \int_{\tilde{\Gamma}_\ell^e} \tilde{\Pi}_i(\tilde{\mathbf{x}}^e)\tilde{\phi}_p((\boldsymbol{\eta}^{-1} \circ \boldsymbol{\eta}_\ell^e)(\tilde{\mathbf{x}}^e))d\tilde{x}^e \ .$$

In the case when supp $\tilde{\phi}_p \subset \tilde{\Gamma}_\ell$, the map $\boldsymbol{\eta}^{-1} \circ \boldsymbol{\eta}_\ell^e$ is the identity on supp $\tilde{\phi}_p$, and so $\tilde{b}_i$ is simply the integral of a piecewise polynomial over a set of planar triangles which can be done exactly using triangle-based quadrature rules of appropriate order. By contrast, if supp $\tilde{\phi}_p$ contains triangles which are in other charts besides $\tilde{\Gamma}_\ell$, then we have to write

$$\tilde{b}_i = \int_{\tilde{\Gamma}} \tilde{\Pi}_i((\boldsymbol{\eta}_\ell^e)^{-1} \circ \boldsymbol{\eta})(\mathbf{y}))G(\mathbf{y})\tilde{\phi}_p(\mathbf{y})dy \ , \tag{2.4}$$

where G is the Gram determinant of the map $\mathbf{y} \mapsto ((\boldsymbol{\eta}_\ell^e)^{-1} \circ \boldsymbol{\eta})(\mathbf{y})$. The first two terms in the integrand in (2.4) are smooth on the intersection of supp $\tilde{\phi}_p$ with each of the $\tilde{\Gamma}_\ell$ and so $\tilde{b}_i$ can be approximated to arbitrary accuracy by appropriate triangle-based rules on $\tilde{\Gamma}$. We emphasise that the computation of $\tilde{b}_i$ is part of the process of computing the weights for the node-based rules and it does not require any kernel evaluations. We shall see in §4 that the computation of the weights (since it is an $O(N)$ process) is a small part of the overall computation time and the extra work that has to be done is small compared to that saved in using the node-based rules for the assembly of K instead of conventional rules.

In the following section we review the hybrid Galerkin algorithm and we also extend its theory ([2, 3]) to the case where the weights of the node-based rules are taken to be the solution $\hat{\boldsymbol{w}}$ of the approximation to (2.2):

$$M\hat{\boldsymbol{w}} = \hat{\boldsymbol{b}} \ , \tag{2.5}$$

where $\hat{\boldsymbol{b}}$ is an appropriately accurate approximation of $\tilde{\boldsymbol{b}}$. This extension of the theory is used in the computations in §4. In §3 we need following generalisation of [2, Lemma 2.7] which can be proved by reworking the proof in this more general situation. To state this lemma we let h_τ denote the diameter of any (curvilinear) triangle $\tau \in \mathcal{T}_h$ and we define the mesh diameter "local" to the node $\mathbf{x}_p$ by $h_p = \max\{h_\tau : \tau \in \mathcal{T}_h, \tau \subset \text{supp } \phi_p\}$.

Lemma 1 *Suppose that the solution $\hat{\boldsymbol{w}}$ of (2.5) is stable in the sense that $\sum_{j\in J_p} |\hat{w}_j| \leq \sigma \sum_{j\in J_p} \hat{w}_j$, and that the points $\{\mathbf{x}_j : j \in J_p\}$ are not too far from $\mathbf{x}_p$, i.e., $\max\{|\mathbf{x}_j - \mathbf{x}_p| : j \in J_p\} \leq \delta h_p$. Suppose also that $\hat{\boldsymbol{b}}$ is a sufficiently accurate approximation to $\tilde{\boldsymbol{b}}$ in the sense that*

$$\|\hat{\boldsymbol{b}} - \tilde{\boldsymbol{b}}\|_\infty \leq \beta h_p^{d+3} \ , \tag{2.6}$$

where $\sigma, \delta, \beta > 0$ are bounded independently of h. Then the resulting node-based quadrature rule for (2.1) satisfies:

$$\Big| \int_{\tilde{\Gamma}_\ell^e} \tilde{F}^e \phi_p \circ \boldsymbol{\eta}_\ell^e - \sum_{j\in J_p} \hat{w}_j \tilde{F}^e(\tilde{\mathbf{x}}_j^e) \Big| \leq C h_p^{d+1} \int_\Gamma \phi_p \ ,$$

where C depends on σ, δ, β and the $d+1$st derivatives of $\tilde{F}^e$ on $\tilde{\Gamma}_\ell^e$.

Remark. Conditions sufficient to ensure the above stability requirement follow from the analysis in [3, §3].

3 Hybrid Galerkin Algorithm and Complexity

In this section we give a more precise description of the hybrid Galerkin algorithm outlined in §1, together with an extension of its theory to the case of the overlapping decompositions introduced in §2. Letting H^μ, $\mu \in [-1, 1]$, denote the usual the Sobolev space on Γ, we shall assume that the operator in (1.1) satisfies $\lambda I + \mathcal{K} : H^\mu \to H^{-\mu}$, and that the corresponding bilinear form $a(u, v) = ((\lambda I + \mathcal{K})u, v)$ is $H^\mu-$elliptic (where $(\cdot, \cdot)$ denotes the dual pairing between $H^{-\mu}$ and H^μ). This assumption is widely satisfied in practice (see, e.g. [2]). For the Laplace differential operator the standard boundary integral equations are the single-layer potential and hypersingular equations (where $\lambda = 0$ in (1.1) and $\mu = -1/2, 1/2$ respectively) and the classical second-kind equation with the double-layer potential operator (where $\lambda = \pm 1/2$ and $\mu = 0$).

The usual Galerkin method seeks $U \in S_h := \text{span}\{\phi_p : p \in \mathcal{N}\}$ such that $a(U, V) = (f, V)$ for all $V \in \mathcal{S}$. This is equivalent to a linear system with matrix $\lambda M + K$, with M an easy to compute sparse mass matrix and K given in (1.2) above. It is well-known that U satisfies the quasi-optimal error estimate: $\|u - U\|_\mu \leq C h^{2-\mu}$ as $h \to 0$. The algorithm presented below provides an automatic procedure for computing an approximation $\tilde{K}$ to K in such a way that the corresponding discrete Galerkin solution $\tilde{U}$ also enjoys this convergence rate. The algorithm has a first phase in which the

weights of (relatively low-order) node-based rules are computed on the charts described above and a second phase where these are used (where possible) to compute $K_{p,q}$, with conventional rules used otherwise. In determining what degree of precision should be employed the following quantities become important: $h_{p,q} := \max\{h_p, h_q\}$, and

$$\rho_{p,q} = \min\{|\mathbf{x}-\mathbf{y}| : \mathbf{x} \in \text{supp } \phi_p \cup \{\mathbf{x}_j : j \in J_p\},\ \mathbf{y} \in \text{supp } \phi_q \cup \{\mathbf{x}_{j'} : j' \in J_q\}\} \ .$$

The algorithm requires the user to choose the following parameters: real numbers $\delta, \sigma, C^* \geq 1$ and positive integers $d_{\min} \leq d_{\max}$. Also $\chi = 1 - \mu - \min\{\mu, 0\}$.

Theorem 2 provides a typical convergence result for the algorithm described below. For simplicity we assume here that the mesh diameter $h < 1$ and that the mesh refinement is such that Nh^2 is bounded above and below as $h \to 0$. (Quasiuniformity is sufficient but not necessary for this.) More general versions, including extensions to anisotropically refined meshes are contained in [2, 3, 4]. The proof of Theorem 2 is obtained by extending the arguments in [3, 4], using Lemma 1 above to deal with errors in approximate weights of node-based rules.

Theorem 2 *If the stiffness matrix $\tilde{K}$ is computed by the algorithm below, and if the weights $\hat{\boldsymbol{w}}$ are computed from (2.5), where $\hat{\boldsymbol{b}}$ satisfies (2.6), then the corresponding discrete Galerkin solution $\tilde{U} \in \mathcal{S}_h$ satisfies the optimal error estimate:*

$$\|u - \tilde{U}\|_{H^\mu} \leq C(u, C^*, \sigma, \delta, \beta) h^{2-\mu} \quad \text{as} \quad h \to 0 \ . \tag{3.1}$$

The algorithm is called a "hybrid" Galerkin algorithm because it can be viewed as a blend of two traditional integral equation discretisations: the Galerkin and Nyström methods (for more details see [3]).

Hybrid Galerkin Algorithm

procedure generate_node_based_quadrature_rules;
begin
  for all $p \in \mathcal{N}$ **do**
    for all integers $d \in [d_{\min}, d_{\max}]$ **do**
    (*i*) Select at least $(d+1)(d+2)/2$ nodes $\{\mathbf{x}_j : j \in J_p(d)\}$ on Γ
      with the property $\max\{|\mathbf{x}_p - \mathbf{x}_j| : j \in J_p\} \leq \delta h_p$.
    (*ii*) Find $\{\hat{w}_j : j \in J_p(d)\}$ by solving (2.5) with $\hat{\boldsymbol{b}}$ satisfying (2.6).
    (*iii*) If the $\hat{w}_j$ satisfy $\sum_j |\hat{w}_j| \leq \sigma \sum_j \hat{w}_j$, then
      set $B_p(d)$ = "admissible" and store the $J_p(d)$ and $\hat{w}_j$.
      Otherwise set $B_p(d)$ = "inadmissible"
    end of loop over d
  end of loop over p
end;

The second phase of the algorithm then follows:

procedure generate_hybrid_system_matrix;
begin
for all $p, q \in \mathcal{N}$ **do**
begin

- Compute d_{pq}, the smallest positive integer satisfying

$$d_{p,q} \geq \frac{\chi + (1+\alpha)\log(C^*\rho_{p,q})/\log h_{p,q}}{1 - \log(C^*\rho_{p,q})/\log h_{p,q}} \tag{3.2}$$

if $(d_{p,q} \in [d_{\min}, d_{\max}])$ and $(B_p(d_{pq}) =$ "admissible") and $(B_q(d_{pq}) =$ "admissible")
then

- Compute $\tilde{K}_{pq}$ using the node-based rules obtained above.

else

- Compute $\tilde{K}_{pq}$ using conventional triangle-based rules, and ensure that

$$|K_{p,q} - \tilde{K}_{p,q}| \leq C h_{p,q}^{\chi+1} \int_\Gamma \phi_p \int_\Gamma \phi_q \ . \tag{3.3}$$

end
end;

Note that (3.3) is always possible, with C independent of h, using triangle-based rules (see, e.g. [1, 6]). In [3] we analysed (in the case of quasi-uniform meshes) the complexity gains in using the hybrid algorithm compared with the conventional Galerkin method in the case of the three standard examples from harmonic potential theory: The single-layer, double-layer and hypersingular equations. For these three equations we computed the ratio of the number of entries of the matrix which will be computed by triangle-based rules to the number which will be computed by node-based rules. The asymptotic behaviour of this ratio is given in the following table

	$d_{\max} = 1$	$d_{\max} = 2$	$d_{\max} = 3$
single − layer	−	$const$	$h^{4/5}$
double − layer	$const$	$h^{4/5}$	$h^{4/3}$
hypersingular	$h^{2/5}$	h	$h^{10/7}$

.

Table 1

Thus, provided we choose $d_{\max} \geq 3$ in the hybrid algorithm, then, for all three model equations, the number of matrix entries computed by triangle-based quadrature becomes negligible compared to the number computed by node-based quadrature as $h \to 0$ and then the number of kernel evaluations required by the whole algorithm approaches N^2. Only $d_{\max} = 2$ is needed to ensure this property in the case of the double layer and $d_{\max} = 1$ in the case of the hypersingular equation. Since using triangle-based rules alone requires CN^2 kernel evaluations with moderately large C (see §1), we would expect that the hybrid algorithm would improve on conventional algorithms by a factor approaching C as $h \to 0$. The results in §4 show that the estimates Table 1 are realised in practice, even for moderately coarse meshes.

4 Numerical results

In this section we report on some numerical experiments for the the hybrid algorithm applied to the computation of the harmonic Dirichlet-Neumann map corresponding to a smooth bounded domain $\Omega \subset \mathbb{R}^3$ with boundary Γ. If Φ is harmonic in Ω then it is well-known that Φ satisfies *Green's third identity* on Γ:

$$\mathcal{V}\frac{\partial \Phi}{\partial n} - \mathcal{W}\Phi = \frac{1}{2}\Phi \ , \tag{4.1}$$

where $\partial/\partial n$ denotes differentiation in the outward normal direction from Ω and $\mathcal{V}$, $\mathcal{W}$ are, respectively, the single and double-layer potentials:

$$\mathcal{V}\phi(\mathbf{x}) = \frac{1}{4\pi}\int_\Gamma \frac{1}{|\mathbf{x}-\mathbf{y}|}\phi(\mathbf{y})dy \ , \ \ \mathcal{W}\phi(\mathbf{x}) = \frac{1}{4\pi}\int_\Gamma \frac{\partial}{\partial n}\left(\frac{1}{|\mathbf{x}-\mathbf{y}|}\right)\phi(\mathbf{y})dy \ . \tag{4.2}$$

If the Neumann data v of Φ is given on Γ, the Neumann-Dirichlet map can be computed by solving the integral equation

$$(\frac{1}{2}I + \mathcal{W})u = \mathcal{V}v \tag{4.3}$$

for the Dirichlet data u. Conversely if u is given, computation of the Dirichlet-Neumann map requires the solution of (4.3) for v. Both these maps are commonly used in domain decomposition procedures. These two equations constitute two different instances of problem (1.1), with $\lambda = 1/2$ in the first case and $\lambda = 0$ in the second. It is well known that the appropriate energy spaces for these equations is H^μ with (for the first equation) $\mu = 0$, and (for the second) $\mu = -1/2$. Since the surface Γ is here assumed smooth, the parameter α introduced in §2 is $\alpha = 1$ for both equations. Detailed experiments on using the hybrid Galerkin method for approximating and inverting the operator $\mathcal{V}$ (and hence computing the Dirichlet-Neumann map) which verify the theory above are given in [3]. In this paper we concentrate on the Neumann-Dirichlet map.

For given v, (4.3) has a unique solution u only up to an arbitrary additive constant. To avoid this difficulty we add the easily discretised term $\int_\Gamma u$ to the right-hand side of (4.3). The resulting equation has a unique solution which corresponds to one of the solutions of (4.3).

In our experiments we consider the specific case where Γ is the unit sphere $\{\mathbf{x} \in \mathbb{R}^3 : |\mathbf{x}| = 1\}$ and Φ is the harmonic function $\Phi(\mathbf{x}) = 1/|\mathbf{x} - (2,0,0)^T|$. Using the Neumann boundary data of this function as v we compute an approximation $\tilde{U}$ to the corresponding Dirichlet data u by solving (4.3) with Galerkin's method, using the hybrid algorithm to compute the stiffness matrix. In the tables below the L_2 norm of the error is approximated by the L_2 norm of the interpolant of the error, with the necessary integrals done using 4 point conical Gauss rules in each triangle.

To parametrise Γ we use 8 charts comprising the triangular faces of a double pyramid $\tilde{\Gamma}$ with nodes $(\pm 1, \pm 1, \pm 1)$, inscribed in Γ. The mapping $\boldsymbol{\eta} : \tilde{\Gamma} \to \Gamma$ is given (globally in this case) by $\boldsymbol{\eta}(\mathbf{x}) = \mathbf{x}/|\mathbf{x}|$. For each chart $\tilde{\Gamma}_\ell$, the extension map $\boldsymbol{\eta}_\ell^e$ is again defined by this same formula. Meshes are constructed on Γ by starting with the triangles comprising $\tilde{\Gamma}$ itself (Level 1) and performing 4 levels of hierarchical refinement, with each new level obtained from the previous by standard quadrasection of each triangle. This yields a sequence of five meshes on $\tilde{\Gamma}$ with 6, 18, 66, 258 and 1026

nodes, respectively. The finest level discretisations on $\tilde{\Gamma}$ and Γ are depicted in Fig. 1. There are many nodes on chart edges and so the overlapping procedure in §2 is very important to obtain optimal complexity here.

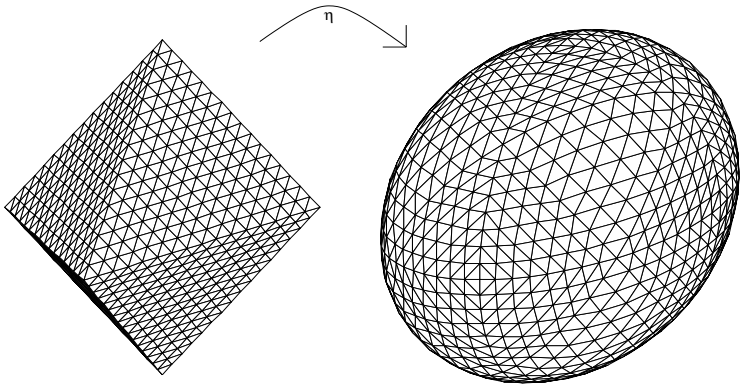

Figure 1: The unit sphere Γ and its 8 charts.

In our implementation of the hybrid algorithm we set $d_{\max} = d_{\min} = 2$ and we computed node-based rules of degree of precision 2 for $p \in \mathcal{N}$. To simplify the computation we omitted this computation for the 6 nodes which correspond to the vertices of the level 1 mesh, but we did it for all other nodes. This omission will have minimal effect on the complexity results reported below. For each p the quadrature points for the rule of precision 2 are chosen as the node $\mathbf{x}_p$ itself together with the 6 nodes connected to it. The resulting underdetermined systems were solved using the minimal norm algorithm in [3]. Thus $\delta = 1$ in Lemma 1. Rather than specify σ in advance we computed all quadrature rules and then recorded the observed values of σ below. In the implementation of (3.2) we set $h_{p,q} = h := \sqrt{(}2) * (2^{levels-1})$ with $levels = 1, 2, 3, 4, 5$, and $\rho_{p,q} = \|\mathbf{x}_p - \mathbf{x}_q\|_2 - 2 * h$. When triangle-based rules are required we used conical Gauss rules of appropriate order and regularising transforms for the singular integrals (see [3]). For this problem $\alpha = 1$ and the energy norm is H^μ with $\mu = 0$ and so $\chi = 1$. We ran the algorithm for various choices of C^*. As C^* increases, the number of entries of the matrix computed by node-based rules increases. Nevertheless for fixed C^*, as the meshes are refined, the optimal error estimate in Theorem 2 will be realised.

In the tables below % is the percentage of the matrix actually computed with node-based rules and $\%_{\max}$ is the maximum possible percentage which could be computed in this way (i.e., the percentage of the matrix entries for which the supports of the basis functions do not intersect). In all the tables, columns headed "EOC" contain estimated orders of convergence (computed by extrapolation) for the numbers in the column immediately to its left.

			$C^* = 1/4$			$C^* = 1/2$		
N	σ	$\%_{\max}$	$\|u - \tilde{U}\|_{L_2}$	EOC	%	$\|u - \tilde{U}\|_{L_2}$	EOC	%
66	3.7	73	3.38(-3)		0	3.97(-3)		59
258	2.6	93	7.31(-4)	2.2	0	1.69(-3)	1.2	68
1026	3.3	98	1.87(-4)	2.0	0	2.25(-4)	2.9	70

Table 2

			$C^* = 1$			$C^* = 3$		
N	σ	$\%_{\max}$	$\|u - \tilde{U}\|_{L_2}$	EOC	%	$\|u - \tilde{U}\|_{L_2}$	EOC	%
66	3.7	59	3.97(-3)		59	3.97(-3)		59
258	2.6	88	1.60(-3)	1.3	87	2.00(-3)	1.0	88
1026	3.3	97	2.55(-4)	2.6	93	8.43(-4)	1.3	97

Table 3

The results clearly confirm the complexity predictions in Table 1. For small enough C^* (namely when $1/C^* > \mathrm{diam}(\Gamma)$) no entries of the matrix are computed with node-based rules, the method is then the conventional Galerkin method and it converges with $O(h^2)$. As C^* increases more entries are computed with the node-based rules. For each fixed C^* the order of convergence approaches $O(h^2)$ but with an asymptotic constant which increases with C^*. For the largest value of C^*, the rate of convergence has not yet achieved h^2 but 97% of the matrix is done using the cheap method. The moderate choice $C^* = 1$ achieves $O(h^2)$ but still computes 93% of the matrix cheaply. Table 4 compares the flops for the computation of the weights in the unconventional node-based rules with those needed to compute the "Nyström" matrix $k(\mathbf{x}_p, \mathbf{x}_q)$, for $p \neq q$, $p, q \in \mathcal{N}$. The first is growing with $O(N)$ and the second with $O(N^2)$. The number of flops for creating the stiffness matrix by triangle-based rules will on the other hand be typically 10-100 times the second column. Thus the cost of solving the small systems in phase 1 of the hybrid algorithm has a big pay-off in overall computation times.

N	weights	Nyström matrix
66	2.3(+5)	9.6(+4)
258	7.5(+5)	1.5(+6)
1026	2.5(+6)	2.3(+7)

Table 4: Comparison of flops for computing weights and Nyström matrix

References

[1] S. Erichsen and S. A. Sauter. Efficient automatic quadrature in 3D Galerkin BEM, *Comp. Meth. Appl. Mech. Engrg.*, 157:215–224 (1998).

[2] I. G. Graham, W. Hackbusch and S. A. Sauter. Discrete boundary element methods on general meshes in 3D. Mathematics Preprint 97/19, University of Bath.

[3] I.G. Graham, W. Hackbusch and S.A. Sauter, Hybrid boundary elements: Theory and implementation. Mathematics Preprint 98-6, University of Kiel.

[4] I. G. Graham, W. Hackbusch and S. A. Sauter. The Hybrid Galerkin Method on Degenerate Meshes University of Bath Mathematics Preprint Number 98/22.

[5] U. Langer, Parallel Iterative solution of symmetric coupled FE/BE equations via domain decomposition, In Contemporary Mathematics 157, AMS, Providence, 1994.

[6] S. A. Sauter. Cubature techniques for 3D Galerkin BEM, in W. Hackbusch and G. Wittum, editors, *BEM: Implementation and Analysis of Advanced Algorithms*, Vieweg Verlag, 1996.

[7] S. A. Sauter and A. Krapp On the effect of numerical integration in the Galerkin boundary element method. *Numer. Math.* 74: 337-359, 1997

[8] A.H. Stroud. *Approximate Calculation of Multiple Integrals* Prentice Hall, Englewood Cliffs, 1973.