Computational Aspects of Meshfree Methods

Introduction: Meshfree methods are Galerkin methods for solving PDEs where the basis functions are not constructed on the basis of a given mesh or triangularization (or similar subdivision of the domain in question), but are obtained by using "particles" which have given centers and "radii" (or some other measure of size). These are particularly useful for dealing with a number of situations where current conventional Finite Element Methods (FEMs) have difficulties such as large-deformation problems in elasticity, problems involving discontinuities, and shape optimization problems where re-meshing is required in conventional FEMs. Meshfree methods are noted for their flexibility and robustness for a wide range of difficult problems.

However, there is a price to pay for this flexibility: • More attention has to be paid to the geometric issues and associated algorithms. • Matrix assembly must be done differently to conventional FEMs. • The linear systems are less structured than those obtained from conventional FEMs. • Essential boundary conditions are more difficult to satisfy.

In this talk we will discuss the tasks for meshfree solvers (assembly and linear solvers).

Assembly processes and computational geometry: Conventional finite element assembly algorithms typically look like this, assuming that we need to use a quadrature method (with N_Q weights $\omega_k$and points y_k) for N basis functions with $N_Q \gg N$ typically:

```
loop i
    loop j connected to i
        loop k: y_k ∈ S_i ∩ S_j
            K_ij ← K_ij + ω_k ∇Ψ_i(y_k) · ∇Ψ_j(y_k)
```

With a mesh available, given i the task of finding all j where "j is connected to i" is just a matter of finding the neighborhood of i in the graph of the mesh. (This is illustrated in Figure 1.) However, for meshfree methods this turns out to be more complicated.

Let $S_i = \overline{\{\, x \mid \Psi_i(x) \neq 0 \,\}}$ be the support of Ψ_i. If we used the above assembly algorithm to meshfree methods, for each i we would have to first find all the j's where $S_i \cap S_j \neq \emptyset$. (See Figure 1.) The second task is to find all quadrature points belonging to $S_i \cap S_j$. For the case of axis-aligned rectangular sets S_i, this is known as a range-finding task and takes $O(N_Q \log^{d-1} N_Q)$ memory, and $O(\log^{d-1} N_Q + \#\{k \mid y_k \in S_i \cap S_j\})$ time per query. The first geometric task seems at least to be a harder problem to do in near $O(\#\{j \mid S_j \cap S_i \neq \emptyset\})$ time.

Instead, we make the loop on k the outer-most loop:

```
loop k
    I ← { i | y_k ∈ S_i }
    loop i, j ∈ I
        K_ij ← K_ij + ω_k ∇Ψ_i(y_k) · ∇Ψ_j(y_k)
```

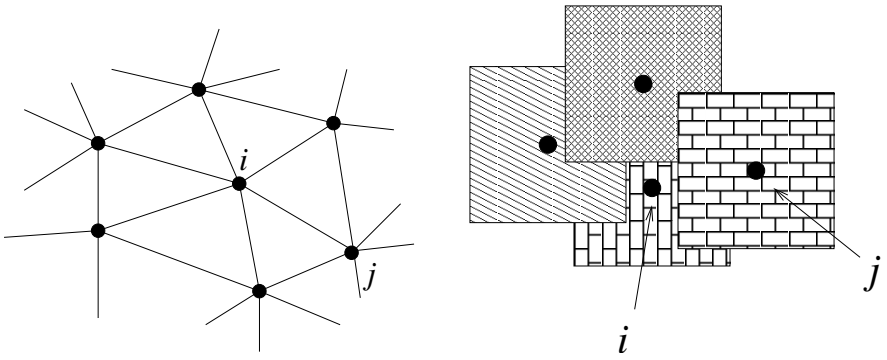

Figure 1: Identifying non-zeros of the stiffness matrix (FEM on left, Meshfree on right)

With this approach, the only geometric task for each quadrature point y_k is to determine the support sets S_i that contain y_k. Quadtrees are a suitable data structure for carrying out this task. Taking the quadrature points to be chosen "at random" we found that the expected time needed to do this is $O(a^{d-1}\,\#\{k \mid y_k \in S_i\} + (\log N)/d)$ where a is the worst-case aspect ratio.

Efficient sparse matrix storage: Conventional FEM assembly processes can assemble K row-by-row , which makes compressed row storage schemes very attractive. Modifying the assembly process to have the outer-most loop over the quadrature points makes this much less practicable: K is updated in a pseudo-random way. Instead we propose using hash tables.

Sufficient Covering: Another discrete issue for meshfree methods is that the support sets must cover the domain "sufficiently well" in some sense in order. If there is a point of the domain that is not covered by a support set then the meshfree basis functions are no longer defined there. Also, the convergence theory of Melenk assumes that there is a (modest) upper bound on number of support sets covering any point of the domain.

Clearly, then, it would be useful to be able to check the maximum and minimum covering numbers for a domain Ω: $\min / \max_{x \in \Omega} \#\{\, i \mid x \in S_i \,\}$. Note that the *covering number* of a point $x \in \Omega$ is simply the number of support sets containing x: $\#\{\, i \mid x \in S_i \,\}$.

Linear Solvers: Another main difficulty is that of solving the linear systems that arise from meshfree systems. While direct methods can always be used in principle, they tend to be slow and memory intensive, especially for three-dimensional problems. We are therefore looking at developing efficient linear systems solvers that are particularly adapted to meshfree methods.

As multigrid methods are the leading iterative methods for large-scale problems arising from PDEs, we are especially interested in developing multigrid or multilevel methods that can achieve for meshfree discretizations close to the performance obtained by multigrid methods for more structured FEM or similar

discretizations. We have been adapting Algebraic MultiGrid (AMG) methods to meshfree discretizations. There are a variety of approaches which use combinatorial and geometric information from the problem within the AMG approach, which can lead to efficient solvers for the systems.