FEM for the Poisson Problem in $\mathbb{R}^2$

April 14 & 16, 2003

1 Model Problem

1.1 Formulations

1.1.1 Strong Formulation

SLIDE 1

Find u such that

$$\begin{aligned} -\nabla^2 u &= f \qquad \text{in } \Omega \\ u &= 0 \qquad \text{on } \Gamma \end{aligned}$$

for Ω a *polygonal domain.* $\boxed{\text{N1}}$

Note 1 ***Generalization***

We look here at a particularly simple but nevertheless illustrative problem. First, we require our domain to be polygonal. More general domains demand — at least for accurate treatment — isoparametric (non-affine) mappings that result in elements with curved edges. Although the implementation of such elements is not particularly difficult, it requires more machinery that we can develop in this short series of lectures.

Second, we consider the homogeneous Dirichlet problem. In one space dimension, the difference between homogeneous and inhomogeneous Dirichlet conditions was quite small as regards formulation, analysis, and implemention. In two space dimensions, unless the inhomogeneous Dirichlet data u^D is piecewise polynomial, in our case here piecewise linear, we will not longer be able to exactly represent u^D within our finite element approximation space. In practice, one typically replaces u^D with its interpolant, but this clearly leads to new complications as regards the theory (a new "variational crime").

Treatment of Neumann and Robin conditions is a relatively straight-forward extension of what we present here. It is particularly in higher space dimensions that the convenience of natural boundary conditions is most beneficial, since calculation of normals and gradients at the boundary can be complicated (and in many cases, ambiguous). As regards inhomogeneous Neumann conditions, the same issues that arise in higher space dimensions for the inhomogeneous Dirichlet problem also arise for the Neumann problem.

Third, we restrict ourselves to two space dimensions. In fact, three dimensions is quite similar, though obviously the implementation (in particular mesh generation) is more complex, and computational cost potentially much higher.

1.1.2 Minimization/Weak Formulations

SLIDE 2

Find $u = \arg\min\limits_{w \in X} \underbrace{\tfrac{1}{2}a(w,w) - \ell(w)}_{J(w)}$;

or find $u \in X$ such that

$$a(u,v) = \ell(v), \forall v \in X \ ;$$

SLIDE 3

where

$$\begin{aligned} X = \{v \in H^1(\Omega) \mid v|_\Gamma = 0\} &\equiv H_0^1(\Omega) \ , \\ a(w,v) &= \int_\Omega \nabla w \cdot \nabla v \, dA \qquad \text{SPD} \ , \\ \ell(v) &= \text{`}\int_\Omega f \, v \, dA\text{'} \qquad \text{bounded} \ . \end{aligned}$$

Recall that f need not be in $L^2(\Omega)$: we can consider any $\ell \in H^{-1}(\Omega)$, for example a "line source" $\ell(v) = 1$ on Γ_ℓ, 0 elsewhere, where Γ_ℓ is some line of "finite extent" in Ω. (Note, however, that the delta distribution is not in $H^{-1}(\Omega)$ for $\Omega \in \mathbb{R}^2$.)

1.2 Regularity

SLIDE 4

In general, $\|u\|_{H^1(\Omega)} \leq C\|\ell\|_{H^{-1}(\Omega)}$.

If $f \in L^2(\Omega)$ *and* Ω is convex,

$$\|u\|_{H^2(\Omega)} \leq C\|f\|_{L^2(\Omega)}; \qquad \boxed{\text{N2}}$$

important for *convergence rate*.

Recall that $\|u\|^2_{H^1(\Omega)} = \int_\Omega |\nabla u|^2 + u^2 \, dA$, and that

$$\|\ell\|_{H^{-1}(\Omega)} = \sup_{v \in H^1(\Omega)} \frac{\ell(v)}{\|v\|_{H^1(\Omega)}}.$$

Note also that in $\mathbb{R}^2$ the H^2 norm includes the square of the cross derivatives as well.

Note 2 ***Regularity in $\mathbb{R}^2$***

In one space dimension it was sufficient that f be suitably smooth to ensure that u would be in $H^2(\Omega)$. In two space dimensions that is no longer true, due to the potential conflict between what the boundary data tells the derivatives to do and what the equation tells the derivatives to do. However, in the case in which the domain Ω is convex, then $f \in L^2(\Omega)$ *is* sufficient to ensure that u is in $H^2(\Omega)$.

It is certainly still possible that u is in $H^2(\Omega)$ even when Ω is not convex, but it is not typically the case. In particular, with a non-convex domain we introduce "re-entrant" corners and associated singularities. The worst case is a crack, where the re-entrant corner has an included angle of 2π; but even in that case the solution remains in $H^1(\Omega)$ (and in fact is more regular than $H^1(\Omega)$, but not as regular as $H^2(\Omega)$). The effect on the finite element convergence rate will be discussed briefly subsequently.

---

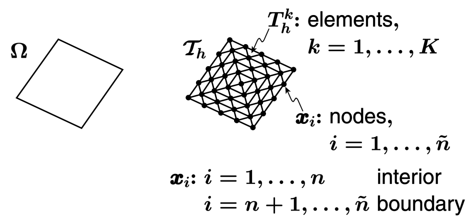
2 Finite Element Discretization

2.1 Triangulation

SLIDE 5

$$\overline{\Omega} = \bigcup_{T_h \in \mathcal{T}_h} \overline{T}_h$$

$\boxed{\text{N3}}$



Ω
$\mathcal{T}_h$
T_h^k: elements,
$k = 1, \dots, K$
x_i: nodes,
$i = 1, \dots, \tilde{n}$
x_i: $i = 1, \dots, n$ interior
$i = n+1, \dots, \tilde{n}$ boundary

Our numbering of nodes is purely for convenience of exposition; it will greatly simplify our definitions of spaces and bases, and the description of the imposition of boundary conditions. In actual practice, in particular for more general boundary conditions, our numbering will not be the best as regards efficiency (for direct solvers); but we know we can always renumber at the end through our $\theta(k, \alpha)$ array.

In general, finite elements are based on largely unstructured *meshes; this has the advantage of flexibility, but also precludes the application of some structured-mesh notions (e.g., FFT or tensorization concepts).*

Note 3 ***Triangulation in $\mathbb{R}^2$***

In two (and particularly three) space dimensions, triangulation can be a difficult task. First, the union of (the closure of) our triangles must cover the domain; second, the (open) elements must not intersect (overlap) with each other; third, the intersection of the closure of one element with the closure of another element must be either an entire edge of both elements or a vertex of both (or null). These conditions figure prominently in the definition of the space that we hang upon this triangulation, though they are so "obvious" that we sometimes forget their presence. (Note in nonconforming approaches, some of these constraints can be relaxed.)

In addition of the above zeroth-order constraints, there are first-order constraints that must, or at least should, be satisfied in order to ensure that the approximation properties of the associated finite element space are good. The first is *regularity*; there are many ways to state this condition (and it goes by many names); in short, it requires that the minimum angle of any triangle be bounded away from zero as h tends to zero. This has many implications as regards the geometry and topology of the mesh (e.g., it bounds the number of triangles that can share a common vertex, ensures that the length of a short side of a triangle does not tend to zero relative to the length of a long side, ...). Note that h, the diameter of the triangulation $\mathcal{T}_h$, is the maximum of the h^k; h^k, in turn, is the diameter — length of the longest edge — of triangle T_h^k.

The second condition often imposed on $\mathcal{T}_h$ has already been discussed in the context of one space dimension: *quasi-uniformity* requires that the minimum of h^k, $k = 1, \ldots, K$, divided by the maximum of h^k, $k = 1, \ldots, K$, remain bounded away from zero as h tends to zero — the elements must not be increasingly disparate in size as $h \to 0$.

Finally, there are second-order requirements. We would like our triangulation to place more elements in regions where the solution will vary more rapidly. We would like the flexibility to locally adapt and refine the mesh based on *a posteriori* error indicators. We would like to specify element aspect ratios and perhaps the alignment of the elements in order to capture certain features of the solution, or to control the conditioning of the stiffness matrix.

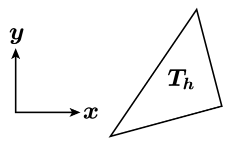
There are, fortunately, triangulation methods that can often honor most of the requests above; several very good third–party software packages are available that provide these capabilities. (The most general packages are quite literally triangulators, or, in $\mathbb{R}^3$, tetrahedron-based; quadrilateral elements do not admit the same level of generality, at least not in as convenient a fashion, as "simplex" elements.) Nevertheless, mesh generation remains a major task, sometimes more time-consuming than a calculation itself. For this reason it is often of interest to avoid the construction of a volume-filling mesh if possible (e.g., as in boundary element methods, discussed next in the course).

2.2 Approximation

2.2.1 Space (Linear Elements)

Slide 6

$$X_h = \{ \underbrace{v \in X}_{\substack{v|_\Gamma = 0, \\ v \in C^0(\Omega)}} \;\; | \, v|_{T_h} \in \mathbb{P}_1(T_h), \; \forall \, T_h \in \mathcal{T}_h \}$$



$$\mathbb{P}_1(T_h):\ v|_{T_h} = c_0 + \underbrace{c_x}_{v_x} x + \underbrace{c_y}_{v_y} y, \quad c, c_x, c_y \in \mathbb{R}$$

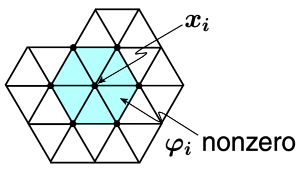
We restrict our attention, as in one space dimension, to linear polynomial approximations. The extension to higher order elements is not difficult. In practice, quadratic elements are often the best in terms of accuracy per unit of work. We also consider only triangular elements, though in many cases quadrilateral (bilinear or biquadratic) elements have advantages.

2.2.2 Basis (Nodal)

SLIDE 7

$X_h = \text{span}\,\{\varphi_1, \ldots, \varphi_n\}$:
$\varphi_i \in X_h, \quad \varphi_i\,(\boldsymbol{x}_j) = \delta_{ij}, \quad 1 \le i, j \le n.$

Support of φ_i:



We see how our requirements on the topology of the triangulation are reflected in the space and basis. For example, because we require that elements that intersect on an edge must intersect over an entire edge, we are ensured that, given two nodal values, a function is perforce continuous over the associated edge.

Note by our node numbering the $\boldsymbol{x}_j$, $j = 1, \ldots, n$, are all in the interior; it follows from $\varphi_i \in X_h \subset X$ that $\varphi_i(\boldsymbol{x}_j) = 0$ for $j = n+1, \ldots, \tilde{n}$.

SLIDE 8

Nodal interpretation: $v \in X_h$,

$$v = \sum_{i=1}^{n} v_i\,\varphi_i(\boldsymbol{x})\ ;$$

$$v(\boldsymbol{x}_j) = \sum_{i=1}^{n} v_i\,\varphi_i\,(\boldsymbol{x}_j) = \sum_{i=1}^{n} v_i\,\delta_{ij} \Rightarrow \boxed{v_j = v(\boldsymbol{x}_j)}\ .$$

2.3 "Projection"

2.3.1 Rayleigh-Ritz or Galerkin

Slide 9

Rayleigh-Ritz:

$$u_h = \arg\min_{w \in X_h} \underbrace{\tfrac{1}{2}a(w,w) - \ell(w)}_{J(w)}$$

Galerkin: $u_h \in X_h$ satisfies

$$a(u_h, v) = \ell(v), \quad \forall\, v \in X_h\ .$$

2.4 Discrete Equations

2.4.1 General Case

Slide 10

Let $u_h(\boldsymbol{x}) = \displaystyle\sum_{j=1}^{n} u_{h\,j}\, \varphi_j(\boldsymbol{x})$; $v = \varphi_i(\boldsymbol{x}),\ i = 1, \ldots, n$:

$$\boxed{\underline{A}_h\, \underline{u}_h = \underline{F}_h} \qquad\qquad \underline{u}_h \in \mathbb{R}^n$$

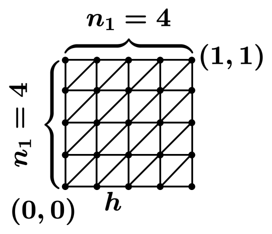
$A_{h\,ij} = a(\varphi_i, \varphi_j)$, $1 \le i, j \le n$,
$F_{h\,i} = \ell(\varphi_i)$, $1 \le i \le n$.

The procedures, either Rayleigh-Ritz or Galerkin, are identical *to those we developed in detail in the one-dimensional case.*

2.4.2 Particular Illustrative Case

Slide 11

Uniform Mesh:



$$\begin{aligned} K &= 2n_1^2 \\ \tilde{n} &= (n_1+1)^2 \\ n &= (n_1-1)^2 \\ h &= 1/n_1 \end{aligned}$$

When we refer to a uniform mesh, we shall mean the particular triangulation above. (Note the diameter of the elements, and hence mesh, is in fact $\sqrt{2}h$, not h.)

SLIDE 12

Expression for $\underline{A}_h$:

$$a(w,v) = \int_\Omega \nabla w \cdot \nabla v \, dA = \int_\Omega w_x v_x + w_y v_y \, dA$$

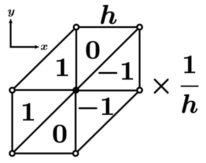
$$\Downarrow$$

$$A_{h\,ij} = a(\varphi_i, \varphi_j) = \int_\Omega \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_j}{\partial x} + \frac{\partial \varphi_i}{\partial y}\frac{\partial \varphi_j}{\partial y} \, dA$$

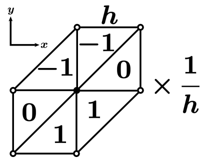
$$1 \le i, j \le n.$$

SLIDE 13

Derivatives of φ_i: $\bullet x_i$



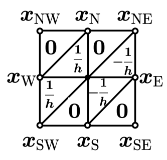
$\partial\varphi_i/\partial x$
(piecewise constant)



$\partial\varphi_i/\partial y$
(piecewise constant)

SLIDE 14

Evaluation of $\int_\Omega (\partial\varphi_i/\partial x)\,(\partial\varphi_j/\partial x)\,dA$ $\bullet\ x_i$

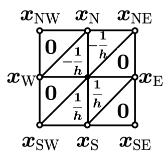


$\boldsymbol{\partial\varphi_i/\partial x}$

$$\int_\Omega \frac{\partial \varphi_i}{\partial x} \left\{ \begin{array}{c} \partial\varphi_N/\partial x \\ \partial\varphi_{NE}/\partial x \\ \partial\varphi_E/\partial x \\ \partial\varphi_{SE}/\partial x \\ \partial\varphi_S/\partial x \\ \partial\varphi_{SW}/\partial x \\ \partial\varphi_W/\partial x \\ \partial\varphi_{NW}/\partial x \\ \partial\varphi_i/\partial x \end{array} \right\} dA = \left\{ \begin{array}{c} 0 \\ 0 \\ -2/h^2 \\ 0 \\ 0 \\ 0 \\ -2/h^2 \\ 0 \\ 4/h^2 \end{array} \right\} \frac{h^2}{2}$$

SLIDE 15

Evaluation of $\int_\Omega (\partial\varphi_i/\partial y)\,(\partial\varphi_j/\partial y)\,dA$ $\bullet\ x_i$



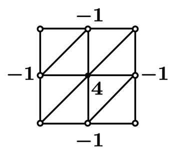
$\boldsymbol{\partial\varphi_i/\partial y}$

$$\int_\Omega \frac{\partial \varphi_i}{\partial y} \left\{ \begin{array}{c} \partial\varphi_N/\partial y \\ \partial\varphi_{NE}/\partial y \\ \partial\varphi_E/\partial y \\ \partial\varphi_{SE}/\partial y \\ \partial\varphi_S/\partial y \\ \partial\varphi_{SW}/\partial y \\ \partial\varphi_W/\partial y \\ \partial\varphi_{NW}/\partial y \\ \partial\varphi_i/\partial y \end{array} \right\} dA = \left\{ \begin{array}{c} -2/h^2 \\ 0 \\ 0 \\ 0 \\ -2/h^2 \\ 0 \\ 0 \\ 0 \\ 4/h^2 \end{array} \right\} \frac{h^2}{2}$$

SLIDE 16

Summary $\bullet x_i$

Nonzero entries of row i of $\underline{A}_h$:



identical to finite differences.

Actually, not quite identical; the FEM matrix is h^2 times the finite difference matrix. Obviously, one of the the advantages of the FEM — to develop approximations based on general triangulations $\mathcal{T}_h$ of arbitrary domains — is not exercised in this example. Our purpose rather is to get some sense of how the φ_i "interact" to form $\underline{A}_h$.

3 Theoretical Analysis

3.1 General Results

3.1.1 Energy Norm

SLIDE 17

Recall $|||v|||^2 \equiv a(v,v) = |v|^2_{H^1(\Omega)} = \int_\Omega |\nabla v|^2 \, dA$.

Then

$$\boxed{|||e||| = \inf_{w_h \in X_h} |||u - w_h|||} \qquad (e = u - u_h) \ ;$$

u_h is the *projection* of u on X_h in the energy norm.

The proof is identical *to that for the one-dimensional case.*

3.1.2 H^1 Norm

SLIDE 18

Recall $\|v\|^2_{H^1(\Omega)} = \int_\Omega |\nabla v|^2 + v^2 \, dA$.

Then

$$\boxed{\|e\|_{H^1(\Omega)} \le \left(1 + \frac{\beta}{\alpha}\right) \inf_{w_h \in X_h} \|u - w_h\|_{H^1(\Omega)} \ ;}$$

α:	coercivity constant (> 0);
β:	continuity constant ($= 1$).

The proof is identical *to that for the one-dimensional case.*

3.2 Particular Results

3.2.1 H^1 and L^2 Norms

SLIDE 19

For $f \in L^2(\Omega)$ and Ω convex,

$$|||e||| \leq C\, h\, \|u\|_{H^2(\Omega)}\ ;$$

$$\|e\|_{H^1(\Omega)} \leq C\, h\, \|u\|_{H^2(\Omega)}\ ;$$

and

$$\|e\|_{L^2(\Omega)} \leq C\, h^2\, \|u\|_{H^2(\Omega)}\ .$$

$\boxed{\text{N4}}$

Note 4 ***Convergence rate and regularity (Optional)***

The form of the proofs are very similar, in fact identical, to the forms of the corresponding proofs in one space dimension. However, the bound for $\|u - \mathcal{I}_h u\|_{H^1(\Omega)}$ is now more difficult to derive (note the mesh regularity plays an important role), requiring some general tools that we will not take the time to develop here. Nevertheless, in the end, the same h dependence as in one space dimension is recovered.

We can of course replace the H^2 norm with the broken H^2 norm, thus permitting less regular f. However, as indicated earlier, the real issue in higher space dimensions is geometry–induced singularities. In the above we exploit the result that, if $f \in L^2(\Omega)$ and Ω is convex, then necessarily $u \in H^2(\Omega)$ (a similar assumption is required for the dual problem associated with the L^2 estimate). Of course, even for Ω nonconvex u might be in $H^2(\Omega)$ (depending on f), but it will not be the case in general for any $f \in L^2(\Omega)$.

Just as in our simple example in one space dimension (in which we place a delta distribution load at a point which is not a node, thus obtaining convergence in the energy norm as $\sqrt{h}$ rather than h), if u is not in $H^2(\Omega)$, convergence (say) in the H^1 norm will degrade to h^σ for some $\sigma < 1$. This effect can be mitigated by (adaptive) refinement, in which we place progressively smaller elements near the singularity, compensating for the smaller exponent with a smaller value of h; there are also more sophisticated procedures that can be applied. In some problems the singularities dominate; however, in certain other problems, the singularities are less important — the convergence rate may be slow, but the constants are so small that the degraded performance is not observed except for very small h (and hence very small errors).

We note that if the boundary Γ is some smooth function (that is, Ω is not polygonal), then higher-order isoparametric formulations will recover the optimal order of convergence.

3.2.2 Output Functionals

SLIDE 20

Recall $s = \ell^O(u) + c^O$, $s_h = \ell^O(u_h) + c^O$.

For $f \in L^2(\Omega)$ and Ω convex, if $\quad \ell^O \in H^{-1}(\Omega), \quad |s - s_h| = |\ell^O(e)| \leq C\, h\, \|u\|_{H^2(\Omega)}$;

if $\quad \ell^O \in L^2(\Omega), \quad |s - s_h| = |\ell^O(e)| \leq C\, h^2\, \|u\|_{H^2(\Omega)}$.

In fact, we can apply the same adjoint arguments as in $\mathbb{R}^1$ to obtain h^2 convergence rates for an even wider class of output functionals.

4 Implementation

4.1 Overview

SLIDE 21

Four steps:

A Proto-Problem;

Elemental Quantities;

Assembly;

Boundary Conditions;

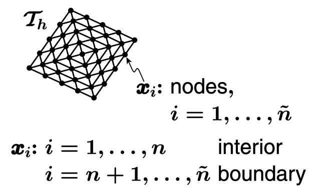
and Numerical Quadrature.

4.2 A Proto-Problem

4.2.1 Space and Basis

SLIDE 22

Let $\tilde{X}_h = \{v \in H^1(\Omega) \mid v|_{T_h} \in \mathbb{P}_1(T_h),\ \forall\, T_h \in \mathcal{T}_h\}$
$= \text{span}\,\{\varphi_1, \ldots, \varphi_n,\ \varphi_{n+1}, \ldots, \varphi_{\tilde{n}}\}$


$\mathcal{T}_h$
$\boldsymbol{x}_i$: nodes,
$i = 1, \ldots, \tilde{n}$
$\boldsymbol{x}_i$: $i = 1, \ldots, n$ interior
$i = n+1, \ldots, \tilde{n}$ boundary

We now include the hat functions associated with the boundary nodes. Any member v of $\tilde{X}_h$ can be expressed as

$$v(\boldsymbol{x}) = \sum_{i=1}^{\tilde{n}} v_i \varphi_i(\boldsymbol{x}),$$

with $v_i = v(\boldsymbol{x}_i)$.

4.2.2 Statement

SLIDE 23

"Find" $\tilde{u}_h \in \tilde{X}_h$ such that

$$a(\tilde{u}_h, v) = \ell(v), \qquad \forall v \in \tilde{X}_h \ .$$

We never actually solve this problem;
it serves only as a convenient pre-processing step.

See comments in $\mathbb{R}^1$ *related to this all–Neumann problem.*

4.2.3 Discrete Equations

SLIDE 24

$$\underline{\tilde{A}}_h \, \underline{\tilde{u}}_h = \underline{\tilde{F}}_h \qquad \tilde{u}_h(\boldsymbol{x}) = \sum_{i=1}^{\tilde{n}} \tilde{u}_{h\,i} \, \varphi_i(\boldsymbol{x})$$

$$\tilde{A}_{h\,ij} = a(\varphi_i, \varphi_j) = \int_\Omega \frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} + \frac{\partial \varphi_i}{\partial y} \frac{\partial \varphi_j}{\partial y} \, dA \qquad 1 \le i, \ j \le \tilde{n} \ ;$$

$$\tilde{F}_{h\,i} = \ell(\varphi_i) \left(= \int_\Omega f \varphi_i \right), \ 1 \le i \le \tilde{n} \ .$$

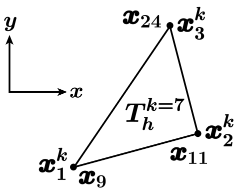
See comments in $\mathbb{R}^1$*. Note in this case, thanks to our convenient numbering,* $\underline{A}_h$ *is very simply related to* $\underline{\tilde{A}}_h$*:* $\underline{A}_h$ *is the left upper* $n \times n$ *submatrix of* $\underline{\tilde{A}}_h$*. For more efficient and natural numbering schemes, this will no longer be the case —* $\underline{A}_h$ *will not be a "contiguous" submatrix of* $\underline{\tilde{A}}_h$*.*

4.3 Elemental Quantities

4.3.1 Local Definitions

SLIDE 25

Local Nodes



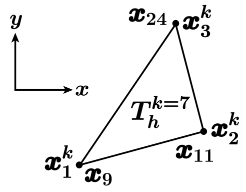
Area^k: area of T_h^k.

$\boldsymbol{x}_1^k, \boldsymbol{x}_2^k, \boldsymbol{x}_3^k$: local nodes in element T_h^k,
corresponding to global nodes $\boldsymbol{x}_9, \boldsymbol{x}_{11}, \boldsymbol{x}_{24}$ (say).

SLIDE 26

Local Basis Functions $\mathcal{H}_\alpha^k, \alpha = 1, 2, 3$:

$\mathcal{H}^k_\alpha \in \mathbb{P}_1(T^k_h)$
$\mathcal{H}^k_\alpha(\boldsymbol{x}^k_\beta) = \delta_{\alpha\beta}$



$$\mathcal{H}^7_1 = \varphi_9|_{T^7_h}; \quad \mathcal{H}^7_2 = \varphi_{11}|_{T^7_h}; \quad \mathcal{H}^7_3 = \varphi_{24}|_{T^7_h} \ . \qquad \boxed{\text{N5}}$$

Since we can represent any $v \in \tilde{X}_h$ restricted to T^k_h in terms of our $\mathcal{H}^k_\alpha$, we will be able to express the contribution of element T^k_h to $\underline{\tilde{A}}_h$ and $\underline{\tilde{F}}_h$ in terms of these local Lagrangian interpolants — just as in $\mathbb{R}^1$.

Note 5 — *Reference element and barycentric coordinates*

In actual practice, we would proceed as we did in $\mathbb{R}^1$. We would introduce a reference triangle $\hat{T}$ — in fact the triangle with vertices $(0,0),(1,0)$, and $(0,1)$ — which we would imbue with a local space and local basis functions. We would then introduce affine mappings that would permit us to recreate each element as a simple image of the reference element. (This procedure is then readily extended to the isoparametric case, in which we have curved edges.) However the mapping and associated change of variables is a little bit more tedious than in $\mathbb{R}^1$, and so we proceed with a slightly simpler formulation in which we retain the concepts of local nodes and local basis functions, but without appeal to the reference element. (It is for this reason that our $\mathcal{H}^k_\alpha$ now bear a k superscript — they are, essentially, the *already-mapped* version of the $\mathcal{H}_\alpha$ associated with the reference element, and thus will be different for each element T^k_h.)

The local coordinates associated with the reference element are denoted *area*, or *barycentric*, coordinates. They have a very nice geometric interpretation — in effect, the reference element coordinates of a point $\boldsymbol{x}$ in element T^k_h correspond to the relative areas marked off by the point $\boldsymbol{x}$ and each of the three sides of the triangle. (In fact, only two of these three coordinates are used — the third is directly related to the other two, since the sum of the relative areas must sum to unity.) A complete discussion of barycentric coordinates and reference element mappings may be found in most texts (e.g., Quarteroni and Valli).

Slide 27

Expression for $\mathcal{H}^k_\alpha$, $\alpha = 1,2,3$:
$\mathcal{H}^k_\alpha = c^k_\alpha + c^k_{x\,\alpha}\, x + c^k_{y\,\alpha}\, y,$

$$\begin{pmatrix} 1 & x^k_1 & y^k_1 \\ 1 & x^k_2 & y^k_2 \\ 1 & x^k_3 & y^k_3 \end{pmatrix} \begin{pmatrix} c^k_\alpha \\ c^k_{x\,\alpha} \\ c^k_{y\,\alpha} \end{pmatrix} = \overbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}}^{\alpha=1} \text{ or } \overbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}^{\alpha=2} \text{ or } \overbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}^{\alpha=3}$$

$$\Rightarrow \quad c^k_\alpha,\ c^k_{x\,\alpha},\ c^k_{y\,\alpha}\ , \qquad \alpha = 1,2,3\ .$$

4.3.2 Elemental Matrices

SLIDE 28

$$\tilde{A}_{h\,ij} = a(\varphi_i, \varphi_j) = \int_\Omega \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_j}{\partial x} + \frac{\partial \varphi_i}{\partial y}\frac{\partial \varphi_j}{\partial y}\, dA$$

Element T_h^7 (say) contributes

$$\int_{T_h^7} \frac{\partial \varphi_{9,11,\text{ or }24}}{\partial x}\frac{\partial \varphi_{9,11,\text{ or }24}}{\partial x} + \frac{\partial \varphi_{9,11,\text{ or }24}}{\partial y}\frac{\partial \varphi_{9,11,\text{or }24}}{\partial y}\, dA\ .$$

SLIDE 29

But

$$\int_{T_h^7} \frac{\partial \varphi_{9,11,\text{ or }24}}{\partial x}\frac{\partial \varphi_{9,11,\text{ or }24}}{\partial x} + \frac{\partial \varphi_{9,11,\text{ or }24}}{\partial y}\frac{\partial \varphi_{9,11,\text{ or }24}}{\partial y}\, dA$$

$$= \int_{T_h^7} \underbrace{\frac{\partial \mathcal{H}^7_{1,2,\text{ or }3}}{\partial x}}_{\text{constant}}\underbrace{\frac{\partial \mathcal{H}^7_{1,2,\text{ or }3}}{\partial x}}_{\text{constant}} + \underbrace{\frac{\partial \mathcal{H}^7_{1,2,\text{ or }3}}{\partial y}}_{\text{constant}}\underbrace{\frac{\partial \mathcal{H}^7_{1,2,\text{ or }3}}{\partial y}}_{\text{constant}}\, dA.$$

SLIDE 30

Define elemental matrices $\underline{A}^k \in \mathbb{R}^{3\times 3}$:

$$\begin{aligned} A^k_{\alpha\beta} &= \int_{T_h^k} \frac{\partial \mathcal{H}^k_\alpha}{\partial x}\frac{\partial \mathcal{H}^k_\beta}{\partial x} + \frac{\partial \mathcal{H}^k_\alpha}{\partial y}\frac{\partial \mathcal{H}^k_\beta}{\partial y}\, dA \\ &= \text{Area}^k (c^k_{x\,\alpha}\, c^k_{x\,\beta} + c^k_{y\,\alpha}\, c^k_{y\,\beta}), \qquad 1 \le \alpha, \beta \le 3 \end{aligned}$$

since derivatives are all *constant* over T_h^k. $\boxed{\text{E1}}$

▷ ***Exercise 1***

(a) Find the elemental matrix $\underline{A}^k \in \mathbb{R}^{3\times 3}$ for our an element with nodes (vertices) at $\boldsymbol{x}^k_1 = (0,0)$, $\boldsymbol{x}^k_2 = (h,0)$, $\boldsymbol{x}^k_3 = (0,h)$.

(b) Prove, by invariance arguments or wild hand-waving, that your result applies to *any* right–triangular element (with hypotenuse $\sqrt{2}h$) at *any* orientation or position.

(c) Find the element *mass matrix* $\underline{M}^k \in \mathbb{R}^{3\times 3}$ for any right–triangle element with hypotenuse $\sqrt{2}h$.

■

4.3.3 Elemental Loads

Slide 31

$$\tilde{F}_{h\,i} = \ell(\varphi_i) = \int_\Omega f\,\varphi_i\,dA$$

Element T_h^7 (say) contributes

$$\int_{T_h^7} f\,\varphi_{9,11,\,\text{or}\,24}\,dA$$
$$= \int_{T_h^7} f\,\mathcal{H}^7_{1,2,\,\text{or}\,3}\,dA\ .$$

Slide 32

Define elemental load vectors $\underline{F}^k \in \mathbb{R}^3$:

$$F^k_\alpha = \int_{T_h^k} f\,\mathcal{H}^k_\alpha\,dA, \qquad \alpha = 1,2,3\ ;$$

evaluation — approximation — of integral typically by
numerical quadrature techniques.

4.4 Assembly

4.4.1 The $\theta(k,\alpha)$ Array

Slide 33

The "idea" is identical to that in $\mathbb{R}^1$*, and already clear from the previous few slides; we move directly to the algorithm.*

Introduce local-to-global mapping

$$\theta(k,\alpha)\colon\ \underbrace{\{1,\ldots,K\}}_{\text{element}} \times \underbrace{\{1,2,3\}}_{\text{local node}} \to \underbrace{\{1,\ldots,\tilde{n}\}}_{\text{global node}}$$

such that
$\boldsymbol{x}^k_\alpha$ (local node α in element k) =
$\boldsymbol{x}_{\theta(k,\alpha)}$ (global node $\theta(k,\alpha)$).

For example, in our illustrative case above for element $k = 7$*, we see that* $\theta(7,1) = 9$*,* $\theta(7,2) = 11$*, and* $\theta(7,3) = 24$*.*

4.4.2 Procedure for $\underline{\tilde{A}}_h$

SLIDE 34

To form $\underline{\widetilde{A}}_h$:

```
zero $\underline{\widetilde{A}}_h$;
{for  $k = 1, \ldots, K$
      {for  $\alpha = 1, 2, 3$
                $i = \theta(k, \alpha)$ ;
      {for  $\beta = 1, 2, 3$
                $j = \theta(k, \beta)$ ;
        $\widetilde{A}_{h\,i\,j} = \widetilde{A}_{h\,i\,j} + A^k_{\alpha\beta}$ ;} } }
```

$\boxed{\text{E2}}$

Although the algorithm in $\mathbb{R}^2$ *is essentially the same as the algorithm in* $\mathbb{R}^1$ *(only the number of local nodes changes), the algorithm is in fact much more useful in* $\mathbb{R}^2$*. In particular, it is in* $\mathbb{R}^2$ *(and* $\mathbb{R}^3$*) that direct calculation via* φ_i, φ_j *of the entries of* $\underline{\tilde{A}}_h$ *would be immensely tedious; by contrast, the assembly procedure is very simple. In short, it is much better to run over the elements and put their entries in the appropriate place than to run over each node and find all elements and basis functions that contribute.*

▷ ***Exercise 2*** Consider four right-triangle elements with hypotenuse $\sqrt{2}h$ that are joined in a diamond such that the nodes opposing the hypotenuse in each element all correspond to the same global node, say $\boldsymbol{x}_{i^*}$. Find the row $\underline{\widetilde{A}}_{h\,i^*j}$. ■

4.4.3 Procedure for $\underline{\tilde{F}}_h$

SLIDE 35

To form $\underline{\widetilde{F}}_h$:

```
zero $\underline{\widetilde{F}}_h$;
{for  $k = 1, \ldots, K$
      {for  $\alpha = 1, 2, 3$
                $i = \theta(k, \alpha)$ ;
          $\widetilde{F}_{h\,i} = \widetilde{F}_{h\,i} + F^k_\alpha$ ;} }
```

4.5 Boundary Conditions

4.5.1 Homogeneous Dirichlet

SLIDE 36

We indicate here only the case of homogeneous Dirichlet conditions; inhomogeneous Dirichlet conditions and Neumann conditions then follow directly from the analogous developments in $\mathbb{R}^1$.

Recall:

$$u_h \in X_h \qquad \boxed{u_h|_\Gamma = 0}$$

$$a(u_h, v) = \ell(v), \quad \forall\, v \in X_h \qquad \boxed{v|_\Gamma = 0\ ;}$$

$$\begin{aligned} X_h &= \text{span}\,\{\varphi_1, \ldots, \varphi_n\} \ \textit{versus} \\ \tilde{X}_h &= \text{span}\,\{\varphi_1, \ldots, \varphi_n,\ \varphi_{n+1}, \ldots, \varphi_{\tilde{n}}\}\ . \end{aligned}$$

SLIDE 37

Explicit Elimination

$X_h \ \Rightarrow \varphi_{n+1}, \ldots, \varphi_{\tilde{n}}$ not admissible variations, so

REMOVE $Rn+1, \ldots, R\tilde{n}$ from $\underline{\tilde{A}}_h$;

$\tilde{u}_{h\,n+1}, \ldots, \tilde{u}_{h\,\tilde{n}} = 0$, so

REMOVE $Cn+1, \ldots, C\tilde{n}$ from $\underline{\tilde{A}}_h$.

As indicated earlier, in the case of our particular numbering, the above is equivalent to just taking the $n \times n$ left upper submatrix of $\underline{\tilde{A}}_h$.

SLIDE 38

Big Number Approach

Place $\frac{1}{\varepsilon}$ $(\varepsilon \ll 1)$ on entries $\tilde{A}_{h\,ii},\ i = n+1, \ldots, \tilde{n}$.

Place 0 on entries $\tilde{F}_{h\,i},\ i = n+1, \ldots, \tilde{n}$.

This replaces $Rn+1, \ldots, R\tilde{n}$ with $u_{h\,n+1} \cong \cdots \cong u_{h\,\tilde{n}} \cong 0$ in an easy, symmetric way.

4.6 Quadrature

SLIDE 39

How do we evaluate

$$F_\alpha^k = \int_{T_h^k} f(\boldsymbol{x})\, \mathcal{H}_\alpha^k(\boldsymbol{x})\, dA$$

for general f?

In actual practice the integral would typically be mapped to the reference element, however even in that case the basic procedure is similar to that discussed here. As in $\mathbb{R}^1$, numerical quadrature is not only important for the load vector, but also for the elemental matrices.

4.6.1 Gauss Quadrature

SLIDE 40

As in $\mathbb{R}^1$, we can also pursue other approaches (e.g., integration by interpolation via the mass matrix). We restrict attention here to Gauss quadrature.

Approximate

$$\begin{aligned} F_\alpha^k &= \int_{T_h^k} f(\boldsymbol{x})\, \mathcal{H}_\alpha^k \, dA \\ &\approx \sum_{q=1}^{N_q} \rho_q^k \, f(\boldsymbol{z}_q^k)\, \mathcal{H}_\alpha^k(\boldsymbol{z}_q^k) \ ; \end{aligned}$$

ρ_q^k:	quadrature weights,
$\boldsymbol{z}_q^k$:	quadrature points.

On the reference element the weights and points are universal; our superscript k is needed because we have not mapped to the reference element.

SLIDE 41

For example:

$N_q = 1,\ \rho_1^k = \text{Area}^k,\ \boldsymbol{z}_1^k = \frac{1}{3}(\boldsymbol{x}_1^k + \boldsymbol{x}_2^k + \boldsymbol{x}_3^k)$ $\boxed{\text{N6}}$

integrates exactly $\int_{T_h^k} g(\boldsymbol{x})\, dA$

for all $g \in \mathbb{P}_1(T_h^k)$;

higher order formulas tabulated.

Generation of Gauss quadrature (or "cubature") formulas in $\mathbb{R}^2$ and $\mathbb{R}^3$ is a more challenging task than in $\mathbb{R}^1$, however effective formulas up to rather high order have been developed for triangles.

For our linear elements, the one-point formula suffices to preserve our ideal convergence rates, though often a slightly higher order quadrature is desired in order to integrate exactly certain quantities (e.g., the mass matrix, which involves quadratics). There are other cases where inexact quadrature is deliberately pursued (e.g., to force a diagonal mass matrix for explicit time integration).

Note 6 **Derivation of one–point Gauss formula**

We prove here that this one–point formula does indeed integrate all $g \in \mathbb{P}_1(T_h^k)$ exactly. First, we see that $\boldsymbol{z}_1^k$ is the centroid of T_h^k (most easily proven by mapping to the reference element...), and thus

$$\boldsymbol{z}_1^k = \int_{T_h^k} \boldsymbol{x} \, dA/\text{Area}^k .$$

Now consider any $g \in \mathbb{P}_1(T_h^k)$, which we can write as

$$g = g(\boldsymbol{z}_1^k) + \nabla g \cdot (\boldsymbol{x} - \boldsymbol{z}_1^k) \ ,$$

where ∇g is constant since $g \in \mathbb{P}_1(T_h^k)$. Then

$$\begin{aligned} \int_{T_h^k} g(\boldsymbol{x}) \, dA &= \int_{T_h^k} g(\boldsymbol{z}_1^k) + \nabla g \cdot (\boldsymbol{x} - \boldsymbol{z}_1^k) \, dA \\ &= \text{Area}^k \ g(\boldsymbol{z}_1^k) + \nabla g \cdot \int_{T_h^k} (\boldsymbol{x} - \boldsymbol{z}_1^k) \, dA \\ &= \text{Area}^k \ g(\boldsymbol{z}_1^k) + \nabla g \cdot [\text{Area}^k \ \boldsymbol{z}_1^k - \int_{T_h^k} \boldsymbol{z}_1^k \, dA] \\ &= \text{Area}^k \ g(\boldsymbol{z}_1^k) \end{aligned}$$

which is our one-point formula. Note only for $\boldsymbol{z}_1^k$ chosen to be the centroid do we integrate exactly all *linear* polynomials — for any other choice, only constant functions are integrated exactly.

---

5 Solution Methods for $\underline{A}_h \, \underline{u}_h = \underline{F}_h$

5.1 Overview

SLIDE 42

We make our remarks here very brief, with specific attention to the FEM context.

Topics

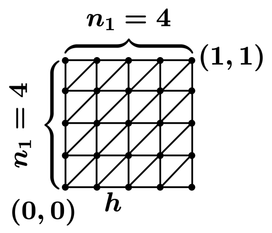
Direct Methods — Banded LU.

Iterative Methods — Conjugate Gradients:
algorithm and interpretation;
convergence rate and conditioning;
action of $\underline{A}_h$.

5.2 Direct Methods — Banded LU

5.2.1 Uniform Mesh

SLIDE 43

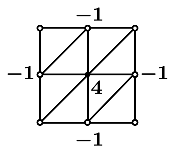


$$
\begin{aligned}
K &= 2n_1^2 \\
\tilde{n} &= (n_1 + 1)^2 \\
n &= (n_1 - 1)^2 \\
h &= 1/n_1
\end{aligned}
$$

SLIDE 44

Stencil

Nonzero entries of row i of $\underline{A}_h$:



5.2.2 Operation Count and Storage

SLIDE 45

For "x–then–y" node numbering,

bandwidth $b = O(n_1)$.

LU: $O(n_1^2\, n_1^2)$ operations; $O(n_1^2\, n_1)$ storage.

Forward/Back Solves: $O(n_1^2\, n_1)$ operations.

Banded LU is viable in $\mathbb{R}^2$, though not in $\mathbb{R}^3$. Note for general domains and unstructured meshes Ω (without any particular obvious ordering) there are a number of heuristic methods to minimize bandwidth. More generally, graph-based sparse matrix techniques can be applied — the edges and vertices of the matrix graph are simply the edges and vertices of the triangulation.

5.3 Conjugate Gradient Iteration

5.3.1 Algorithm

SLIDE 46

$\underline{u}_h = 0$ (say); $\underline{r}^0 = \underline{F}_h$; $\boxed{\text{N7}}$

For $k = 1, \ldots$:

$$
\left.\begin{aligned}
\beta^k &= (\underline{r}^{k-1})^T\, \underline{r}^{k-1} / (\underline{r}^{k-2})^T\, \underline{r}^{k-2} \\
\underline{p}^k &= \underline{r}^{k-1} + \beta^k\, \underline{p}^{k-1}
\end{aligned}\right\} \underline{p}^1 = \underline{r}^0
$$

$$\begin{aligned}
\alpha^k &= (\underline{r}^{k-1})^T\,\underline{r}^{k-1}/(\underline{p}^k)^T\,(\underline{A}_h\,\underline{p}^k)\\
\underline{u}_h^k &= \underline{u}_h^{k-1} + \alpha^k\,\underline{p}^k\\
\underline{r}^k &= \underline{r}^{k-1} - \alpha^k(\underline{A}_h\underline{p}^k)\ .
\end{aligned}$$

Here α^k and β^k are obviously not related to our coercivity and continuity constants, α and β, and the k obviously refers to iteration index, not element. The $\underline{r}^k$ are the residuals, $\underline{F}_h - \underline{A}_h\,\underline{u}_h^k$, and the $\underline{p}^k$ are the search directions.

Note 7 *Conjugate gradients and the finite element method*

The conjugate gradient (CG) method is a Krylov method particularly well suited (in fact, restricted) to SPD systems. At each iteration, the CG method (implicitly) minimizes the iteration error $\underline{u}_h - \underline{u}_h^k$ in the "$\underline{A}_h$" norm,

$$(\underline{u}_h - \underline{u}_h^k)^T\,\underline{A}_h\,(\underline{u}_h - \underline{u}_h^k)$$

(hence the requirement that $\underline{A}_h$ be SPD) over the Krylov space $\mathcal{K}^k \equiv$ span $\{\underline{r}^0,\ldots,\underline{A}_h^{k-1}\,\underline{r}^0\}$ = span$\{\underline{\mathrm{p}}^0,\underline{\mathrm{p}}^1,\ldots,\underline{\mathrm{p}}^{k-1}\}$. The method exploits the symmetry of $\underline{A}_h$ in order to perform an *orthogonalization* which, in turn, permits very efficient calculation of $\underline{u}_h^k$. The CG technique is truly remarkable: simple to implement, yet very effective.

The CG method has a particularly nice interpretation in the FEM context. To show this, we first note that, for $\underline{e}_I^k = \underline{u}_h - \underline{u}_h^k$, and hence

$$e_I^k(\boldsymbol{x}) = \sum_{i=1}^n e_{Ii}^k \varphi_i(\boldsymbol{x})\ ,$$

we have that

$$\begin{aligned}
(\underline{e}_I^k)^T\underline{A}_h\underline{e}_I^k &= \sum_{i=1}^n\sum_{j=1}^n e_{Ii}^k a(\varphi_i,\varphi_j)e_{Ij}^k\\
&= a\left(\sum_{i=1}^n e_{Ii}^k\varphi_i, \sum_{j=1}^n e_{Ij}^k\varphi_j\right)\\
&= |||e_I^k|||^2,
\end{aligned}$$

which is the *energy norm* of e_I^k. Note that e_I^k, the iteration error, is not our discretization error of earlier. In particular, even if we run the CG algorithm to termination and $e_I = 0$, $e = u - u_h$, the discretization error, will still of course be nonzero. Indeed, this tells us that we should balance errors: there is no reason for e_I to be zero if e is nonzero.

We can now readily visualize the CG method in terms of our $J(w)$ paraboloid. We recall that the finite element approximation u_h minimizes J over all functions in X_h; this is equivalent (our MIRACLE) to minimizing $|||u - u_h|||$. The

CG method iterate u_h^k minimizes J over all functions engendered by $\mathcal{K}^k = \text{span}\,\{\underline{p}^0, \underline{p}^1, \ldots, \underline{p}^{k-1}\}$, that is, all functions which can be expressed as

$$v_h^k(x) = \sum_{j=0}^{k-1} \sum_{i=1}^{n} p_i^j \, \varphi_i(x) \ ;$$

this is equivalent (by our same MIRACLE) to minimizing $|||u_h - u_h^k|||$. In short, the CG method applied to our finite element discretization is a Rayleigh-Ritz method *within* a Rayleigh-Ritz method: minimization over a k-dimensional subslice (the space engendered by $\mathcal{K}^k$) of an n-dimensional slice (the space X_h) of an infinite-dimensional paraboloid (the space X). As always, choice of basis is important: orthogonality for the CG subslice, and sparsity for the finite element slice.

Note the Krylov nature of CG is crucial to these arguments. If we were to simply perform steepest descent minimization ($\underline{p}^k = \underline{r}^k$) we would not obtain the same error minimization statement over $\mathcal{K}^k$, with corresponding detriment to the convergence rate.

5.3.2 Convergence Rate

SLIDE 47

In general,

$$\frac{(\underline{u}_h - \underline{u}_h^k)^T \underline{A}_h (\underline{u}_h - \underline{u}_h^k)}{(\underline{u}_h)^T \underline{A}_h \underline{u}_h} \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k ,$$

$$\kappa(\underline{A}_h) = \frac{\lambda_{\max}(\underline{A}_h)}{\lambda_{\min}(\underline{A}_h)} .$$

Here $\lambda_{\max}(\underline{A}_h)$ and $\lambda_{min}(\underline{A}_h)$ refer to the maximum eigenvalue and minimum eigenvalue of A_h.

In addition to the above result, we also obtain, at least in infinite precision, the finite termination property $\underline{u}_h^n = \underline{u}_h$, though this is generally not of much interest. For steepest descent we lose this property (except for very particular starting vectors); also, for steepest descent, $\sqrt{\kappa}$ in the above relation must be replaced by κ — a significant degradation with great practical import.

SLIDE 48

For FEM $\underline{A}_h$:

$$\kappa(\underline{A}_h) \leq C h^{-2}$$

for quasi-uniform, regular meshes $\mathcal{T}_h$;

thus $n_{\text{iter}} \sim O(\frac{1}{h})$.

The condition number result is proven in Quarteroni and Valli, pp. 192–196. It is valid for any SPD second–order elliptic PDE and any order of polynomial approximation, though C will depend on the polynomial order and the problem particulars (coercivity and continuity constants) — but not *on h.*

In the above slide, n_{iter} is the number of iterations required to reduce the iteration error in the energy norm by some fixed fraction ε. The result can be obtained starting with the equation

$$\ln \frac{\varepsilon}{2} = n_{\text{iter}} \ln \left(\frac{1 - 1/\sqrt{\kappa}}{1 + 1/\sqrt{\kappa}} \right) ,$$

expanding out the expression involving κ, and using the Taylor series for $\ln(1+z)$. *The dependence on ε is only logarithmic.*

We see that n_{iter} depends on h: as h decreases, κ increases, which in turn decreases the convergence rate. However, the dependence on h is not so strong, and is also independent of spatial dimension.

5.3.3 Computational Effort

SLIDE 49

For uniform FEM mesh: $\frac{1}{h} = n_1$

$$n_{\text{iter}} \sim O(n_1) \; ;$$

$$\text{work/iteration} \sim O(n_1^2) \; ; \qquad \text{(Slide 44)}$$

$\Rightarrow O(n_1^3)$ total operations, $O(n_1^2)$ storage.

Note in the above, operations refers to total operations to reduce the iteration error in the energy norm to some fixed small fraction ε.

We see that already in $\mathbb{R}^2$ CG can be better (depending on the constants...) than banded LU. In $\mathbb{R}^3$ the improvement is even more dramatic. Despite the relatively good convergence rate of CG, it is often of interest to improve things further by preconditioning; good preconditioners can be developed even for very unstructured or only semi-structured triangulations (e.g., domain decomposition methods).

5.3.4 General Evaluation of $\underline{y} = \underline{A}_h\, \underline{p}$

SLIDE 50

In the operation and storage estimates of the previous slide, we implicitly used the finite different stencil to conclude that work/iteration is $O(n_1^2)$ and storage only $O(n_1^2)$. But more generally — on any mesh — we need to have an algorithm that allows us to evalute the action of $\underline{A}_h$ efficiently without every forming $\underline{A}_h$.

Given $\underline{\tilde{p}} \in \mathbb{R}^{\tilde{n}},\ \tilde{p}_i = p_i,\ i = 1, \ldots, n$

$$\tilde{p}_i = 0,\ i = n+1, \ldots, \tilde{n}:$$

Evaluate $\underline{\tilde{y}} = \underline{\tilde{A}}_h\, \underline{\tilde{p}}$;
Set $y_i = \tilde{y}_i,\ i = 1, \ldots, n, ;\ \tilde{y}_i = 0,\ i = n+1, \ldots, \tilde{n}$.

$\boxed{\text{N8}}$

Note 8 **Boundary conditions in the iterative context**

In practice, the CG method of Slide 46 is implemented with a $\underline{\tilde{u}}_h$ rather than a $\underline{u}_h$. The homogeneous Dirichlet boundary conditions are first imposed on $\tilde{u}^0_{h\,n+1}, \ldots, \tilde{u}^0_{h\,\tilde{n}}$ (more generally, at the nodes on the boundary, however they might be numbered). Then, by virtue of the masking step implemented in the last line of Slide 50, we maintain this boundary condition as we proceed — we *effectively* operate with $\underline{A}_h$. The advantage of working with all $\tilde{n}$ nodes, and first finding the action of $\underline{\tilde{A}}_h$ and then masking, is ease of implementation. The notion is readily extended to inhomogeneous Dirichlet conditions and Neumann conditions as well.

SLIDE 51

Evaluation of $\underline{\tilde{A}}_h \underline{\tilde{p}}$: *$O(K)$ operations*

```
zero $\underline{\tilde{y}}$; {for $k = 1, \ldots, K$ (elements)
        {for  $\alpha = 1, 2, 3$
              $i = \theta(k, \alpha)$ ;
        {for  $\beta = 1, 2, 3$
              $j = \theta(k, \beta)$ ;
        $\tilde{y}_i = \tilde{y}_i + \boxed{A^k_{\alpha\beta}}\, \tilde{p}_j$ ; } } }
```

In essense, the above procedure performs assembly, or direct stiffness summation, on the fly, never forming $\underline{\tilde{A}}_h$ — only the element matrices are needed, with the necessary matrix multiplication being done triangle by triangle. There are several variants of the above, some more efficient than others.

Note that for our uniform mesh, $K = O(n_1^2)$, thus verifying our earlier estimate for work/iteration (the other work required by the CG iteration, inner products, is also $O(K) = O(n_1^2)$). We also confirm that the storage requirement for CG is only the elemental matrices and the field vectors, both of which are $O(K) = O(n_1^2)$.