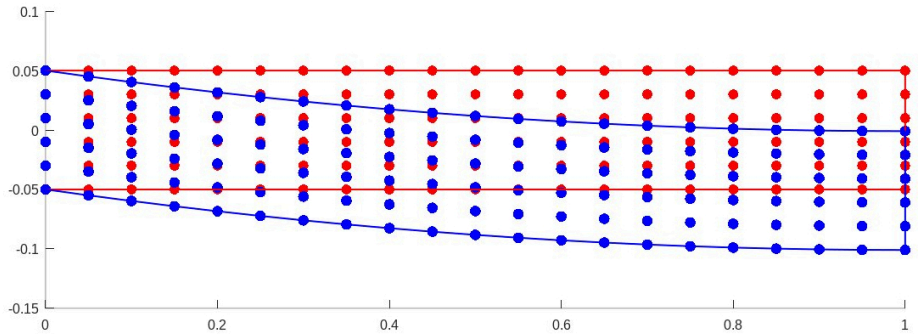

Figure 4.4 Deformed geometry (with the displacement scaled by a factor of 10^5 to render it visible) obtained with the variational method when using $k_1 = 1$ and $k_2 = 2$ (9 basis functions). Shown are the reference and deformed positions of a set of sampling points (in red and blue, respectively). Also shown are the boundaries of the reference and deformed configurations. This solution is physically unrealistic.

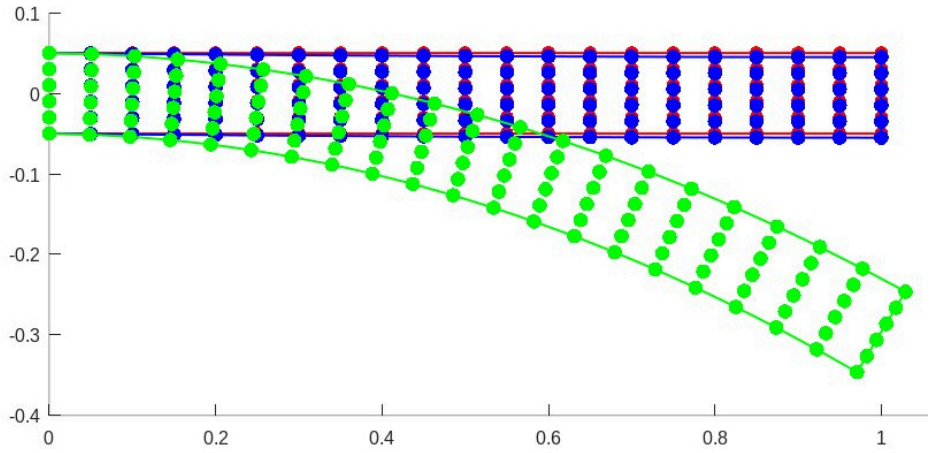

Figure 4.5 In green we show the deformed geometry (with the displacement scaled by a factor of 10^4 to render it visible) obtained with the variational method when using $k_1 = k_2 = 2$ (12 basis functions). In red and blue we show the same reference and approximate deformation as in Fig. 4.4, but now they are almost superposed because α has been decreased to 10^4.

4.4 Finite Element Spaces for Multifield problems in 2D

In Chapter 2 we introduced the P_1 finite element space in two space dimensions. The P_1 finite element provides basis functions that are **continuous scalar functions** which, by refining the mesh, can approximate any function $u \in C^1(\Omega)$.

Though they have not yet been discussed in this notes, there exist P_k finite elements that provide continuous scalar functions which are piecewise polynomials of any order $k \geq 1$ in the variables $x_1 - x_2$.

None of these finite element spaces can *per se* approximate problems with several unknowns such as the 2D elasticity problem, in which the unknown is a vector field **u** which, upon selecting a suitable coordinate frame, becomes **a pair** of scalar functions (u_1, u_2). How to build a finite element space that can approximate **u**? This is an example of one among many **multifield** problems in Engineering.

Multifield problems. These are problems in which there exist $p > 1$ coupled unknowns, $u_1, \ldots, u_p$. Typical examples are thermoelasticity (temperature and displacement), thermal convection (temperature and fluid velocity and pressure), electrodynamics (electric and magnetic fields), etc. The solution is thus a p-tuple $(u_1, u_2, \ldots, u_p)$, which will be organized as a column array for convenience.

We thus pose the general question of **how to build multifields using scalar finite elements as raw material**. The following lemma describes a procedure to do that for **a two-field in two-dimensional domain**. The extension to p-fields or to other space dimensions is straightforward.

Lemma 4.2. *Let $\mathcal{W}_{h1}$ and $\mathcal{W}_{h2}$ be two finite element spaces with bases $\mathcal{N}^\Omega = \{N_1, N_2, \ldots, N_{m_1}\}$ and $\mathcal{M}^\Omega = \{M_1, M_2, \ldots, M_{m_2}\}$. Then,*

a) *the space of two-fields*

$$\mathcal{W}_h = \left\{ \underline{w_h}(x_1, x_2) = \begin{pmatrix} w_{h1}(x_1, x_2) \\ w_{h2}(x_1, x_2) \end{pmatrix} \mid w_{h1} \in \mathcal{W}_{h1},\ w_{h2} \in \mathcal{W}_{h2} \right\} = \mathcal{W}_{h1} \times \mathcal{W}_{h2} \tag{4.42}$$

is a vector space of dimension $m_1 + m_2$.

b) *The set of two-fields*

$$\mathcal{N}^\Omega = \left\{ \underbrace{\begin{pmatrix} N_1 \\ \\ 0 \end{pmatrix}}_{\underline{N}_1}, \underbrace{\begin{pmatrix} N_2 \\ \\ 0 \end{pmatrix}}_{\underline{N}_2}, \ldots \underbrace{\begin{pmatrix} N_{m_1} \\ \\ 0 \end{pmatrix}}_{\underline{N}_{m_1}}, \underbrace{\begin{pmatrix} 0 \\ \\ M_1 \end{pmatrix}}_{\underline{N}_{m_1+1}}, \underbrace{\begin{pmatrix} 0 \\ \\ M_2 \end{pmatrix}}_{\underline{N}_{m_1+2}}, \ldots \underbrace{\begin{pmatrix} 0 \\ \\ M_{m_2} \end{pmatrix}}_{\underline{N}_{m_1+m_2}} \right\}$$

is a basis of $\mathcal{W}_h$.

c) *Assume that $\mathcal{W}_{h1}$ and $\mathcal{W}_{h2}$ are generated, respectively, by* **scalar finite elements** *$(\Omega_e, \mathcal{N}^e)$ and $(\Omega_e, \mathcal{M}^e)$* **on the same partition** *$\Omega = \cup_e \Omega_e$, where*

$$\mathcal{N}^e = \{N_1^e, N_2^e, \ldots, N_{\ell_1}^e\}, \qquad \mathcal{M}^e = \{M_1^e, M_2^e, \ldots, M_{\ell_2}^e\}$$

are the corresponding sets of **scalar shape functions**. *Then* $(\Omega_e, \mathcal{N}^e)$ *is a* **two-field finite element**, *where*

$$\mathcal{N}^e = \Big\{ \underbrace{\begin{pmatrix} N_1^e \\ \\ 0 \end{pmatrix}}_{\underline{N}_1^e}, \underbrace{\begin{pmatrix} N_2^e \\ \\ 0 \end{pmatrix}}_{\underline{N}_2^e}, \ldots \underbrace{\begin{pmatrix} N_{\ell_1}^e \\ \\ 0 \end{pmatrix}}_{\underline{N}_{\ell_1}^e}, \underbrace{\begin{pmatrix} 0 \\ \\ M_1^e \end{pmatrix}}_{\underline{N}_{\ell_1+1}^e}, \underbrace{\begin{pmatrix} 0 \\ \\ M_2^e \end{pmatrix}}_{\underline{N}_{\ell_1+2}^e}, \ldots \underbrace{\begin{pmatrix} 0 \\ \\ M_{\ell_2}^e \end{pmatrix}}_{\underline{N}_{\ell_1+\ell_2}^e} \Big\}$$

d) *Let* LG1 *and* LG2 *be the* **local-to-global maps** *that establish the correspondences between* $\mathcal{N}^e$ *and* $\mathcal{N}^\Omega$ *and between* $\mathcal{M}^e$ *and* $\mathcal{M}^\Omega$, *respectively; i.e.,*

$$N_A = \mathop{\mathring{\sum}}_{\{(a,e)|\mathsf{LG1}(a,e)=A\}} N_a^e, \qquad M_B = \mathop{\mathring{\sum}}_{\{(b,e)|\mathsf{LG2}(b,e)=B\}} M_b^e \,. \tag{4.43}$$

Then the **local-to-global map** LG **of the two-field finite element** *that satisfies*

$$\underline{N}_A = \mathop{\mathring{\sum}}_{\{(a,e)|\mathsf{LG}(a,e)=A\}} \underline{N}_a^e \tag{4.44}$$

is given by

$$\mathsf{LG}(a,e) = \begin{cases} \mathsf{LG1}(a,e) & \text{if } 1 \le a \le \ell_1 \\ \mathsf{LG2}(a-\ell_1,e) + m_1 & \text{if } \ell_1 + 1 \le a \le \ell_1 + \ell_2 \end{cases}$$

In Octave/MATLAB the array LG *can be formed with the instruction*

```
LG = [ LG1 ; LG2 + m1 ];
```

e) *Assume that both spaces* $\mathcal{W}_{h1}$ *and* $\mathcal{W}_{h2}$ *satisfy the approximability property, i.e., assume that for any smooth two-field* $\underline{w}$ *there exist constants* C_1, C_2, k_1 *and* k_2 *and norms* $\|\cdot\|_{W_1}$ *and* $\|\cdot\|_{W_2}$, *all independent of* h, *such that*

$$\min_{v_h \in \mathcal{W}_{h1}} \|w_1 - v_h\|_{W_1} \le C_1 h^{k_1} \quad \text{and} \quad \min_{v_h \in \mathcal{W}_{h2}} \|w_2 - v_h\|_{W_2} \le C_2 h^{k_2} \,. \tag{4.45}$$

Then,

$$\min_{\underline{w_h} \in \mathcal{W}_h} \left(\|w_1 - w_{h1}\|_{W_1}^2 + \|w_2 - w_{h2}\|_{W_2}^2 \right)^{\frac{1}{2}} \le C_1 h^{k_1} + C_2 h^{k_2} \,. \tag{4.46}$$

This lemma allows us to exploit the scalar finite element spaces we have introduced in previous chapters to build multifield finite element spaces. We do this with the P_1-space in two dimensions in the following section.

4.5 Solving Linear Elasticity Problems in 2D with P_1 Finite Elements

In 2D Cartesian coordinates the displacement field **u** is represented by a two-field $\underline{u} = (u_1, u_2)^T$. Also, the space $\mathcal{W} = \mathbf{H}^1(\Omega)$ of linear elasticity coincides, by its definition (4.25), with $\mathcal{W}_1 \times \mathcal{W}_2$ where $\mathcal{W}_1 = \mathcal{W}_2 = H^1(\Omega)$.

We can thus identify the two-fields with the vector fields and keep using the boldface vector notation that was adopted from the beginning of this chapter.

The **two-field finite element space** that we adopt here is a very popular one. Each component of the field is approximated with P_1 finite elements based on a triangulation of the domain.

Let $\mathcal{T}_h$ be a **conforming triangulation of the domain**, with its corresponding arrays X, LV. Let W_h be the **scalar P_1-space associated to $\mathcal{T}_h$**.

Then, the P_1 **space for 2D elasticity** is simply

$$\mathcal{W}_h = W_h \times W_h = \left\{ \mathbf{w}_h = (w_{h1}, w_{h2})^T : \Omega \to \mathbb{R}^2 \mid w_{h1} \in W_h,\ w_{h2} \in W_h \right\} . \qquad (4.47)$$

In this situation, we can apply Lemma 4.2 to build the basis and the local-to-global map. In particular,

```
nod = size(X,2);
LG = [ LV ; LV+nod ];
```

Let us now turn to the element stiffness matrix and element load vector.

4.5.1 Element stiffness matrix and element load vector

The element basis functions are, in terms of the scalar basis functions N_1^e, N_2^e and N_3^e that we already know well (the barycentric coordinates λ_1, λ_2 and λ_3),

$$\mathbf{N}_1^e = \begin{pmatrix} N_1^e \\ 0 \end{pmatrix}, \quad \mathbf{N}_2^e = \begin{pmatrix} N_2^e \\ 0 \end{pmatrix}, \quad \mathbf{N}_3^e = \begin{pmatrix} N_3^e \\ 0 \end{pmatrix},$$

$$\mathbf{N}_4^e = \begin{pmatrix} 0 \\ N_1^e \end{pmatrix}, \quad \mathbf{N}_5^e = \begin{pmatrix} 0 \\ N_2^e \end{pmatrix}, \quad \mathbf{N}_6^e = \begin{pmatrix} 0 \\ N_3^e \end{pmatrix}.$$

We will use the same array

$$\mathsf{dN}_{ib} = \frac{\partial N_b^e}{\partial x_i}$$

as in Chapter 2. From dN we can compute the strain fields $B^a = \varepsilon(\nabla \mathbf{N}_a)$.

$$B^1 = \begin{pmatrix} \mathsf{dN}_{11} & \frac{1}{2}\mathsf{dN}_{21} \\ \frac{1}{2}\mathsf{dN}_{21} & 0 \end{pmatrix}, B^2 = \begin{pmatrix} \mathsf{dN}_{12} & \frac{1}{2}\mathsf{dN}_{22} \\ \frac{1}{2}\mathsf{dN}_{22} & 0 \end{pmatrix}, B^3 = \begin{pmatrix} \mathsf{dN}_{13} & \frac{1}{2}\mathsf{dN}_{23} \\ \frac{1}{2}\mathsf{dN}_{23} & 0 \end{pmatrix},$$

$$B^4 = \begin{pmatrix} 0 & \frac{1}{2}\mathsf{dN}_{11} \\ \frac{1}{2}\mathsf{dN}_{11} & \mathsf{dN}_{21} \end{pmatrix}, B^5 = \begin{pmatrix} 0 & \frac{1}{2}\mathsf{dN}_{12} \\ \frac{1}{2}\mathsf{dN}_{12} & \mathsf{dN}_{22} \end{pmatrix}, B^6 = \begin{pmatrix} 0 & \frac{1}{2}\mathsf{dN}_{13} \\ \frac{1}{2}\mathsf{dN}_{13} & \mathsf{dN}_{23} \end{pmatrix}.$$

The way to store three-dimensional arrays of matrices in Octave/MATLAB is by adding an additional index **as last index**. The array of strain matrices can thus be coded as

```
for i=1:3
   BB(1:2,1:2,i)  = [dN(1,i),0.5*dN(2,i); 0.5*dN(2,i),0];
   BB(1:2,1:2,i+3)= [0,0.5*dN(1,i); 0.5*dN(1,i),dN(2,i)];
end
```

and the stresses $\Sigma^a = \sigma(\nabla \mathbf{N}^a)$ (again using the Lamé coefficients μ and λ, which are assumed constant in the element with value μ_e and λ_e, respectively) as,

```
for a=1:6
   DD(a)=sum(diag(BB(:,:,a));
   SS(:,:,a)=2*mue*BB(:,:,a)+lambdae*DD(a)*eye(2);
end
```

where `eye(2)` is the two-dimensional identity matrix.

Remark: It is also possible (and convenient) to store these symmetric matrices as one-dimensional arrays by using the **Voigt notation**.

The **element stiffness matrix** K^e is given by

$$\mathsf{K}^e_{ab} = a(\mathbf{N}^b, \mathbf{N}^a) = \int_{\Omega_e} \Sigma^b : B^a \, dx_1 dx_2 \; = A_e \Sigma^b : B^a \,, \tag{4.48}$$

where we have used that **both B^a and Σ^b are constant in the element**.

Remark: The P_1 element in elasticity is also called CST element, for Constant Strain Triangle.

The **element load vector** is also easy to compute. Leaving the treatment of the Neumann boundary conditions for later, let us assume that the body force $\mathbf{b}$ is constant and equal to $\mathbf{b}^e$ in the element. Then,

$$\mathsf{F}^e_a = \ell(\mathbf{N}^e_a) = \int_{\Omega_e} \mathbf{b}^e \cdot \mathbf{N}^e_a \, dx_1 dx_2 = \begin{cases} \frac{A_e}{3} b^e_1 & \text{if } 1 \le a \le 3 \\ \frac{A_e}{3} b^e_2 & \text{if } 4 \le a \le 6 \end{cases}$$

The double contraction ": " of two matrices A and B can be coded in Octave/-MATLAB as

```
contraction = sum(sum(A.*B));
```

Putting it all together, the following lines compute the element stiffness matrix and element load vector.

```
function [Ke, Fe]=elementKandF(xe,Ee,nue,be)
  mue=Ee/(1+nue)/2;
  lambdae=mue*nue/(1-2*nue);
  dN=[xe(2,2)-xe(2,3),xe(2,3)-xe(2,1),xe(2,1)-xe(2,2);...
      xe(1,3)-xe(1,2),xe(1,1)-xe(1,3),xe(1,2)-xe(1,1)];
  Ae2=dN(2,3)*dN(1,2)-dN(1,3)*dN(2,2);
  dN=dN/Ae2; Ae=Ae2/2;
  %% Compute strains and stresses
  for i=1:3
```

```
    BB(1:2,1:2,i)  = [dN(1,i),0.5*dN(2,i); 0.5*dN(2,i),0];
    BB(1:2,1:2,i+3)= [0,0.5*dN(1,i); 0.5*dN(1,i),dN(2,i)];
  end
  for a=1:6
    DD(a)=sum(diag(BB(:,:,a)));
    SS(:,:,a)=2*mue*BB(:,:,a)+lambdae*DD(a)*eye(2);
  end
  %% Build Ke and Fe
  for a=1:6
   for b=1:6
    Ke(a,b)=Ae*sum(sum(SS(:,:,b).*BB(:,:,a)));
   end
  end
  Fe=Ae/3*[be(1);be(1);be(1);be(2);be(2);be(2)];
end
```

4.5.2 Boundary conditions

The Dirichlet and Neumann boundary conditions can be treated with a slight adaptation of the techniques used in Chapter 2 to make it suitable for vector fields.

The array of boundary edges BE, remember, has as many columns as boundary edges in the mesh. The first line and second lines contain the two nodes of each boundary edge. The third line contains the geometrical line to which each edge belongs.

The array η_g (EtaG) must now include both unknowns (u_{h1} and u_{h2}) of all nodes that have Dirichlet conditions. It is possible to have boundaries along which only one of the unknowns is imposed, as would be the case of a symmetry line. Since η_g imposes constraints on each degree of freedom, η_g contains all the indices of degrees of freedom whose values are prescribed by the essential boundary conditions.

Similarly, the array H has now two lines. Each column gives us the surface force **H** that is applied to each boundary edge (assumed constant each edge).

We show below the procedure to build η_g, GG and H in a case in which the Dirichlet boundary conditions (equal to zero) are applied along geometrical line #4 and a surface force with value $(0,-100)^T$ is applied along geometrical line #2. Notice that we create the scalar version on η_g first and then transform it to the vector version. This would not be correct if some nodes have the u_1 component imposed but not u_2 or viceversa.

```
bcaux=zeros(1,size(X,2));
nbe=size(BE,2);
HH=zeros(2,nbe);
for k=1:nbe
   if (BE(3,k)==4)
      bcaux(BE(1,k))=1;
      bcaux(BE(2,k))=1;
```

```
   end
   if (BE(3,k)==2)
      HH(:,k)=[0;-1000];
   end
end
EtaG=find(bcaux==1);
%% This is the EtaG for scalars, for vectors it is
EtaG=[EtaG EtaG+nod];
%% Set boundary values to zero, both components.
GG=0*ones(1,length(EtaG));
```

Let us use the same trick as in Chapter 2. We build the stiffness matrix and load vector as if there were no Dirichlet conditions or Neumann conditions, then we add the Neumann contributions, and just before solving the system we correct K and F according to η_g and GG (the list of boundary values).

The procedure can be easily coded as

```
%% Neumann contributions
for ied=1:nbe
  lged=BE(1:2,ied);
  xed(1:dd,1:2)=X(1:dd,lged);
  Led=norm(xed(:,1)-xed(:,2));
  Hed=HH(:,ied);
  F(lged)+=Hed(1)*Led/2;
  F(lged+nod)+=Hed(2)*Led/2;
end
%% unknowns with specified value
ng=length(EtaG);
II=eye(nunk);
for ig=1:ng
 K(EtaG(ig),:)=II(EtaG(ig),:);
 F(EtaG(ig))=GG(ig);
end
```

4.5.3 Example: A cantilever bar

We go back to the geometry and conditions of Fig. 4.3, in which a cantilever bar is fixed at its left side to a rigid vertical wall. We use the same coefficients as in Example 4.5, namely $H = 0.1$ m, $L = 1$ m, $E = 2 \times 10^{11}$ Pa, $\nu = 0.3$, $\rho = 8000$ kg/m^3, $g = 9.8$ m/s^2. We want to illustrate that the finite element method indeed provides a convergent approximation for this problem.

As done in Example 4.5, we will mainly visualize the deformed geometry (affected by $\alpha = 10^4$ as in Fig. 4.5) and report the vertical displacement of the upper right corner of the bar (node #3).

The polygonal boundary was defined from the points $(0, -\frac{H}{2})^T$, $(L, -\frac{H}{2})^T$, $(L, \frac{H}{2})^T$ and $(0, \frac{H}{2})^T$. The Dirichlet boundary is geometrical line #4 (between points 4 and 1). The Neumann boundary is defined as line #2 (between points 2 and 3), but **H** is taken as zero and thus has no effect.

The vertical displacement of node 3, as obtained with meshes of different refinement, is as follows

n_{nod}	n_{el}	u_{h2}(node 3)
32	40	-32.572 μm
66	86	-42.112 μm
246	402	-55.186 μm
938	1698	-59.290 μm
3530	6706	-60.221 μm
13756	26806	-60.471 μm

The corresponding deformed meshes (with displacement scaled by $\alpha = 10^4$) are shown in Fig. 4.6. As observed with the global polynomial case, the discretized bar shows to be stiffer when the mesh is coarser. The vertical displacement of node 3 converges to $\simeq -60.4\,\mu$m as the mesh is refined. In the bottom part of the figure the difference in u_{h2}(node 3) between successive meshes is plotted vs. the number of elements. The logarithmic plot shows that the pointwise value converges with order $O(h^2)$.

4.5.4 Computing the stresses

As mentioned before, one of the main objectives of solving an elastic problem is to compute the stress field.

For the P_1 finite element space, the stress is constant within each element. How to compute it? Once one has computed $\mathbf{u}_h$, it is quite simple. The function `elementStress` below does just that. It simply computes $\sigma = \sum_{a=1}^{6} U_a \Sigma^a$. Notice the similarity with `elementKandF`.

```
function sigmae=elementStress(xe,Ee,nue,Ue)
  mue=Ee/(1+nue)/2;
  lambdae=mue*nue/(1-2*nue);
  dN=[xe(2,2)-xe(2,3),xe(2,3)-xe(2,1),xe(2,1)-xe(2,2);...
      xe(1,3)-xe(1,2),xe(1,1)-xe(1,3),xe(1,2)-xe(1,1)];
  Ae2=dN(2,3)*dN(1,2)-dN(1,3)*dN(2,2);
  dN=dN/Ae2; Ae=Ae2/2;
  %% Compute strains and stresses
  for i=1:3
    BB(1:2,1:2,i)  = [dN(1,i),0.5*dN(2,i); 0.5*dN(2,i),0];
    BB(1:2,1:2,i+3)= [0,0.5*dN(1,i); 0.5*dN(1,i),dN(2,i)];
  end
```

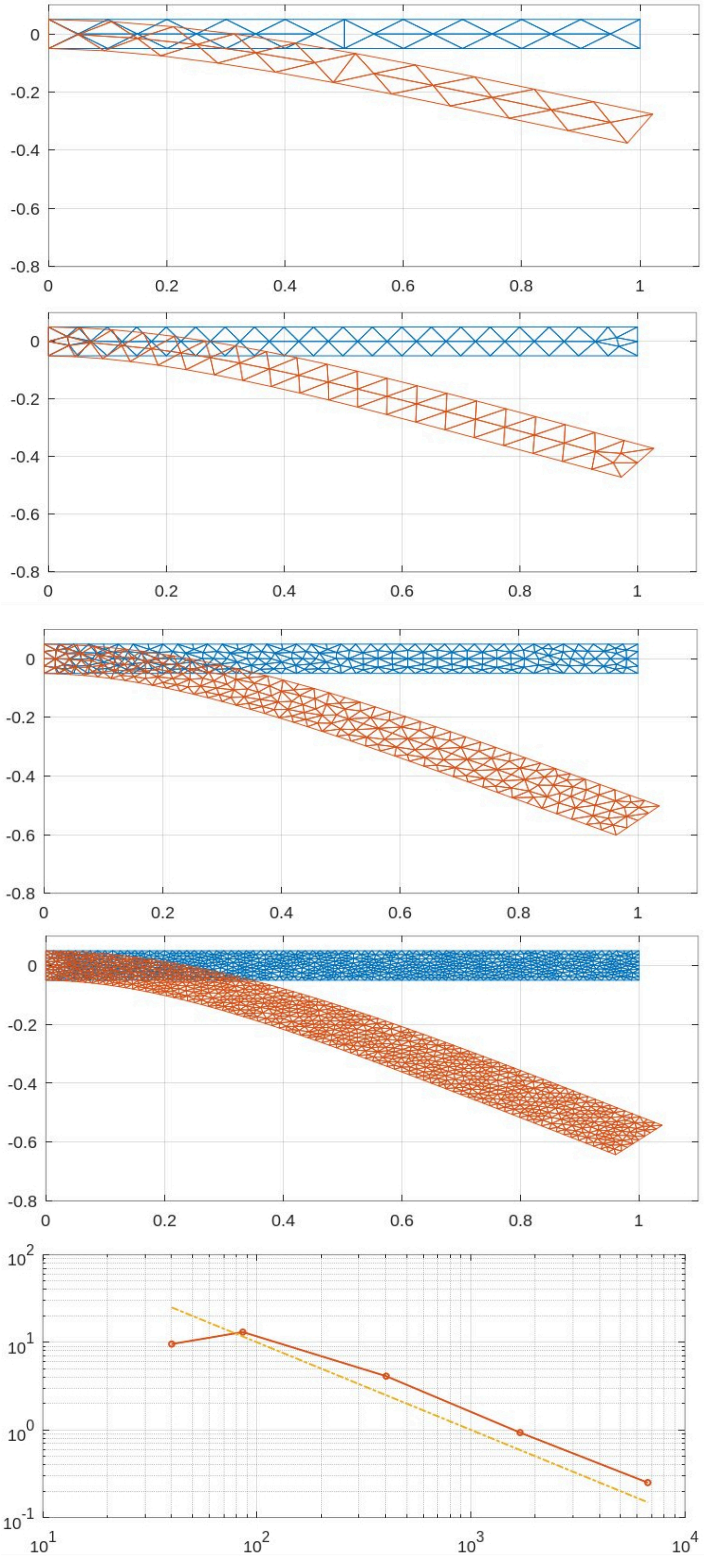

Figure 4.6 Results of a cantilever bar problem under its own weight. The first four graphs show the undeformed and deformed meshes (with the displacements scaled by $\alpha = 10^4$) for several mesh sizes. The bottom graph shows the difference (in microns) in u_{h2} at the top right corner between two successive meshes. The dotted line is a reference line with $O({n_{\text{el}}}^{-1})$ which corresponds to $O(h^2)$.

```
13   for i=1:6
14     DD(i)=sum(diag(BB(:,:,i)));
15     SS(:,:,i)=2*mue*BB(:,:,i)+lambdae*DD(i)*eye(2);
16   end
17   %% Build sigmae
18   sigmae=zeros(2,2);
19   for i=1:6
20       sigmae=sigmae+Ue(i)*SS(:,:,i);
21   end
22 end
```

In Figure 4.7 we show the von Mises stress $\|\sigma\| = \sqrt{\sigma : \sigma}$ by plotting a circle at the centroid of each element, colored according to $\|\sigma\|$. The maximum stresses are seen to occur, as expected, in the top left and bottom left corners of the bar. The predicted value of the von Mises stress is 2.414 MPa. In the top left element, in particular, we obtain (for a mesh with 1698 elements)

$$\sigma_h = \begin{pmatrix} 2.40 & -0.176 \\ -0.176 & 0.150 \end{pmatrix} \text{ MPa},$$

showing that the main component of the stress is horizontal traction.

Since the yield stress of steel is about 250 MPa, we can be quite sure that a cantilever steel bar of such 1 m × 0.1 m will not collapse under its own weight.

Because the stress and the strain are linear in $\nabla \mathbf{u}$, we expect that they converge to the exact value more slowly (as we refine the mesh) than the displacement $\mathbf{u}$ itself. For the P_1-element we expect

$$\|\varepsilon - \varepsilon_h\|_{0,\Omega} \leq c_1 \, h$$

and

$$\|\sigma - \sigma_h\|_{0,\Omega} \leq c_2 \, h \, .$$

for some c_1 and c_2 that depend on the (smooth) exact solution. The convergence of σ_h in the **maximum norm** is a more delicate problem, since it is very sensitive to the boundary conditions and the mesh geometry. For the meshes considered for the convergence of u_2 at node 3 we obtain the results tabulated below.

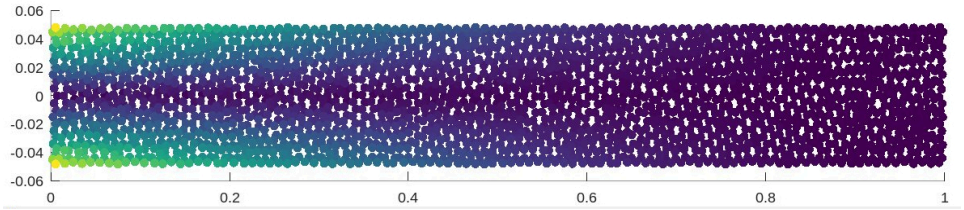

Figure 4.7 Scatter plot colored with the von Mises stress ($\|\sigma\|$). Each point is located at the centroid of an element. The maximum von Mises stress (in yellow) is 2.414 MPa and takes place at the top left and bottom left corners.

n_{nod}	n_{el}	max von Mises stress
32	40	1.098 MPa
66	86	1.802 MPa
246	402	2.103 MPa
938	1698	2.414 MPa
3530	6706	2.744 MPa
13756	26806	3.135 MPa

4.6 A Variational Method as a Minimum Principle

Let us recapitulate the steps that were followed to arrive at the discrete problem (4.37):

1. The point of departure was the Variational Problem 4.1, which introduces the potential energy function V and defines the solution $\mathbf{u}$ as the minimum of V over $\mathscr{S}$.

2. Then, using Theorem 4.1, we saw that the minimum of V over $\mathscr{S}$ satisfies the weak formulation $a(\mathbf{u},\mathbf{v}) = \ell(\mathbf{v})$ for all $\mathbf{v} \in \mathscr{V}$.

3. Once we had the weak formulation, we proceeded to the variational method of approximation just as in the previous chapters.

There is however another intuitive, even more natural, discretization method. We can go straight from item 1 above to a discrete problem by simply restricting the **minimization problem** to the discretized trial space $\mathscr{S}_h$.

In other words, one can consider the problem

Problem 4.4 (Discrete Variational Problem)**.** *Find* $\mathbf{u}_h \in \mathcal{W}_h$ *that minimizes* V *over* $\mathcal{S}_h$*; i.e., find* $\mathbf{u}_h \in \mathcal{S}_h$ *such that*

$$V(\mathbf{u}_h) \leq V(\mathbf{w}_h), \qquad \forall \mathbf{w}_h \in \mathcal{S}_h. \tag{4.49}$$

It is not difficult to see that Problem 4.4 has a unique solution, and that this solution coincides with that of the variational method. In fact, we can apply Theorem 4.1 taking as $\mathcal{W}$ the discrete space $\mathcal{W}_h$. Then $\mathcal{S}_h$ is an affine subspace of $\mathcal{W}_h$ and $\mathcal{V}_h$ is its direction, as required by the theorem. Further, notice that hypothesis (a) of the theorem is automatically satisfied, because $\mathcal{V}_h \subset \mathcal{V}$. Finally, to check that hypothesis (b) holds, notice that if $\mathbf{u}$ is the **exact solution** (assumed to exist) then

$$V(\mathbf{w}_h) \geq V(\mathbf{u}) , \qquad \forall \mathbf{w}_h \in \mathcal{S}_h .$$

This means that V is bounded from below in $\mathcal{S}_h$. In a finite-dimensional space, a function V that is bounded from below and such that $V(\mathbf{w}_h)$ tends to $+\infty$ when $\mathbf{w}_h \to \infty$ always has at least one minimum. And this is certainly the case, since the strain energy $\frac{1}{2}a(\mathbf{w}_h, \mathbf{w}_h)$ is quadratic in $\mathbf{w}_h$ and will eventually dominate and tend to $+\infty$ as $\mathbf{w}_h \to \infty$. Then V has a minimum in $\mathcal{S}_h$ as required.

Theorem 4.1 then guarantees that the minimum $\mathbf{u}_h$ of V over $\mathcal{S}_h$ is unique, and is further characterized as the unique element in $\mathcal{S}_h$ that satisfies

$$a(\mathbf{u}_h, \mathbf{v}_h) = \ell(\mathbf{v}_h) \qquad \forall \mathbf{v}_h \in \mathcal{V}_h.$$

That is, $\mathbf{u}_h$ is the solution of the variational method.

Therefore, we have two ways of getting to $\mathbf{u}_h$: either through a variational method, or through minimization of V in $\mathcal{S}_h$. In terms of code, it tells us that we could implement our variational method solver in a totally different way. Instead of a stiffness matrix K and a load vector F, we would just need to code a *potential energy function* V, say for example

```
function V = PotentialEnergy(Y)
```

which would have as variables the unknown coefficients U_i for $i \notin \eta_g$, collected in the array Y. This function can be computed element by element in a procedure similar to the assembly procedure.

Then, instead of calling the linear system solver "\" we would call a minimization routine (such as `sqp` in Octave). The solution would be exactly the same, but the conceptual approach is quite different.

Remark: The minimization approach makes some generalizations straightforward. One could for example incorporate some restrictions to the minimization problem which would be difficult, or at least not intuitive, to formulate in both the strong or weak forms. Consider an elastic solid that is placed on top of a rigid horizontal surface. The points that are in contact with the surface are then constrained to have non-negative vertical displacement, i.e., $u_2(\mathbf{x}) \geq 0$ if $x_2 = 0$. This is a constraint that is easily incorporated into the discrete minimization

problem (not more than ten lines of code!) and leads to the correct mechanical solution. To arrive at the same numerical approximation starting from the differential equation or the weak formulation is quite involved.

We should point out here that minimization software is much more efficient if you provide the Gradient and the Hessian of the function to be minimized. To compute the gradient one needs F, and the Hessian is nothing but K, so there is no real coding difference when efficiency is important.

The variational (or minimization) approach that we have illustrated for linear elastic problems is possible whenever the bilinear form in the weak formulation is symmetric and coercive. This observation is interesting enough to devote a small section to it.

4.7 Minimization problems and variational method

Let us organize our ideas in the form of a theorem.

Theorem 4.2. *Let $\mathcal{W}$ be a normed vector space, $\mathcal{S} \subset \mathcal{W}$ an affine subspace and $\mathcal{V}$ the direction of $\mathcal{S}$. Further, let $\mathcal{W}_h \subset W$ be a finite-dimensional vector subspace, $\mathcal{S}_h = \mathcal{S} \cap \mathcal{W}_h$ and $\mathcal{V}_h = \mathcal{V} \cap \mathcal{W}_h$.*

Consider a **symmetric** *continuous bilinear form $a(\cdot,\cdot)$ defined in $\mathcal{W}$,* **coercive** *on $\mathcal{V}$, and a continuous linear form $\ell(\cdot)$, also defined in $\mathcal{W}$.*

First part: *Define the following problems:*

PA: *Find $u \in \mathcal{S}$, $V(u) < V(w)$ for all $w \in \mathcal{S}$, $w \neq u$, where*

$$V(w) = \frac{1}{2} a(w,w) - \ell(w) \,. \tag{4.50}$$

PB: *Find $u \in \mathcal{S}$ such that*

$$a(u,v) = \ell(v) \,, \qquad \forall\, v \in \mathcal{V} \,. \tag{4.51}$$

Then problems PA and PB are equivalent. *They both have unique, coincident solutions.*

Second part: *Define the following* **discrete** *problems:*

PA$_h$: *Find $u_h \in \mathcal{S}_h$, $V(u_h) < V(w_h)$ for all $w_h \in \mathcal{S}_h$, $w_h \neq u_h$.*

PB$_h$: *Find $u_h \in \mathcal{S}_h$ such that*

$$a(u_h, v_h) = \ell(v_h) \,, \qquad \forall\, v_h \in \mathcal{V}_h \,. \tag{4.52}$$

Then problems PA$_h$ and PB$_h$ are equivalent. *They both have unique, coincident solutions.*

Now let us apply this theorem to different from that of linear elasticity, in a suite of examples.

Example 4.6 (Minimization form of 1D second-order symmetric problems)

Consider, as in Sections 1 and 2 of Chapter 1, that u is the solution of the differential equation

$$-(k(x)u'(x))' + c(x)u(x) = f(x)$$

with boundary conditions $u(0) = g_0$ and $u'(L) = d_L$, where the coefficient functions k, c and f are continuous and bounded, with $k(x) \geq k_0 > 0$ and $c(x) \geq 0$ for all x.

Then u **is the minimum of**

$$V(w) = \frac{1}{2}\int_0^L \left(k(x)w'(x)^2 + c(x)w(x)^2\right) dx - \int_0^L f(x)w(x)\,dx - k(L)d_L w(L)$$

over all smooth functions w **satisfying** $w(0) = g_0$**.**

The solution u_h of the variational method minimizes V over $\mathscr{S}_h$. The function V in this case is not interpreted as a potential energy, but rather as a *potential*.

Notice that we have removed the first-order term $b(x)u'(x)$, because if $b(x) \neq 0$ then the bilinear form is no longer symmetric and thus there is no equivalence to a minimization problem.

Example 4.7 Minimization form of 1D fourth-order symmetric problems

Remember the fourth-order problem presented in strong form in Problem 1.5. From Theorem 4.2 we know that **the solution** u **minimizes the function**

$$V(w) = \frac{1}{2}\int_0^L \left(q(x)w''(x)^2 + c(x)w(x)^2\right) dx - \int_0^L f(x)w(x)\,dx - W\,w(L) - T\,w'(L)$$

over all smooth functions w **satisfying** $w(0) = g_0$ **and** $w'(0) = d_0$**.** For simplicity, we have written the boundary conditions at $x = L$ in terms of the force W and torque T (see (1.160).

The solution of the variational method minimizes V over $\mathscr{S}_h$.

When the fourth-order problem models a beam in bending, V is indeed the potential energy of the beam.

Example 4.8 Minimization form of 2D diffusion problems

Let us now consider that u is the solution of Problem 2.1, i.e. that

$$-\text{div}\,(K\nabla u) = f$$

in a 2D domain Ω, with boundary conditions $u = g$ in $\partial\Omega_D$ and $(K\nabla u)\cdot\check{n} = H$ in $\partial\Omega_N$. Then from Theorem 4.2 we conclude that u **minimizes the function**

$$V(w) = \frac{1}{2}\int_\Omega (K\nabla w)\cdot\nabla w\, d\Omega - \int_\Omega f\, w\, d\Omega - \int_{\partial\Omega_N} H\, w\, d\partial\Omega$$

over all functions w **satisfying** $w = g$ **on** $\partial\Omega_D$. Also, that **the solution** u_h **of the variational method minimizes** V **over** $\mathscr{S}_h$.