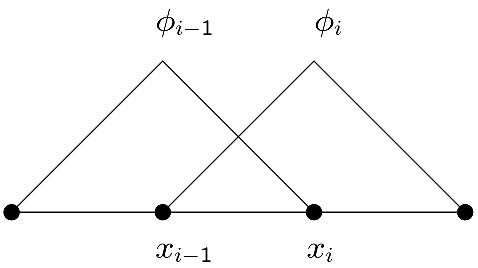
Error approximation

Consider the model problem

$$-u'' = f \quad \text{in } [a,b] \quad \text{and } u(a) = 0 \; u'(b) = 0$$

Let $\mathcal{M} = \{x_0 = a, x_1, \dots x_n = b\}$, be a mesh
$S^{1,0}(\mathcal{M}) = \{v \in C^0[a,b] \,|\, \forall i = 1, \dots n, v_{|[x_{i-1},x_i]} \in \mathbb{P}_1[x_{i-1}, x_i]\}$
the discrete space with hat function basis $\{\phi_i\}_{i=0}^n$



and $S = \{v \in S^{1,0}(\mathcal{M}) | v(0) = 0\}$.

Denote by u be the exact solution and by

$u_h = \sum_{i=1}^n U_i \phi_i \in S$ the discrete solution

of the model problem.

We want to compute the error

$$\text{e}^2 = \int_a^b |u' - u_h'|^2 = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} |u' - u_h'|^2$$

Assume $I_i = [x_{i-1}, x_i]$ and $h_i = x_i - x_{i-1}$

$$[x_{i-1}, x_i] \xrightarrow{A_i} [0,1] \qquad x \mapsto \frac{x - x_{i-1}}{h_i} \qquad\qquad [0,1] \xrightarrow{A_i{}^{-1}} [x_{i-1}, x_i] \qquad t \mapsto h_i\, t + x_{i-1}$$

$\widehat{\phi}_0(t) = 1 - t$ and $\widehat{\phi}_1(t) = t$ for $t \in [0,1]$

note that on $[x_{i-1}, x_i]$

$$\phi_{i-1} = \widehat{\phi}_0 \circ A_i \qquad\qquad \phi'_{i-1} = h_i^{-1} \widehat{\phi}'_0 \circ A_i = -h_i^{-1}$$

$$\phi_i = \widehat{\phi}_1 \circ A_i \qquad\qquad \phi'_i = h_i^{-1} \widehat{\phi}'_1 \circ A_i = h_i^{-1}$$

Therefore

$$\begin{aligned}
\int_{I_i} |u' - u'_h|^2 &= \int_{I_i} |u(x)' - \textstyle\sum_{j=1}^n U_j \phi'_j(x)|^2\, dx \\
&= \int_{I_i} |u'(x) - U_{i-1}\phi'_{i-1}(x) - U_i \phi'_i(x)|^2\, dx \\
&= \int_0^1 |u'(A_i^{-1}(t)) + \underbrace{h_i^{-1}(U_{i-1} - U_i)}_{c}|^2 h_i\, dt \\
&= \int_0^1 |\underbrace{u'(A_i^{-1}(t)) + c}_{G(t)}|^2 h_i\, dt = h_i \int_0^1 G(t)^2 dt \\
&\underset{\text{CS}}{\simeq} \tfrac{h_i}{6}[G^2(0) + 4G^2(1/2) + G^2(1)] \\
&= \tfrac{h_i}{6}\Big[(u'(x_{i-1}) + c)^2 + 4(u'(\tfrac{x_{i-1}+x_i}{2}) + c)^2 \\
&\quad + (u'(x_i) + c)^2\Big]
\end{aligned}$$

Computation of $||u' - u'_h||_{0,2;[01]}$

```
double norm_err(double dim,double nodes[],
                double uh[])
 { int i,k;
   double x[3],g2[3];
   double h, c, err=0;

   for(k=0;k<dim;k++)
     { h=nodes[k+1]-nodes[k];
       x[0]=nodes[k];
       x[1]=(nodes[k]+nodes[k+1])/2;
       x[2]= nodes[k+1];
       c=(uh[k]-uh[k+1])/h;
       for(i=0;i<=2;i++)
          { g2[i]=du(x[i])+c;
            g2[i]*=g2[i];
           }
       err+=(g2[0]+4*g2[1]+g2[2])*h/6;
     }
   return(sqrt(err));
 }
```

Meshes uniform refinements

```
double* refine_mesh(double *nodes,int n)
 {                    /* n=length(nodes)*/
   int i,count;
   double * temp;

   temp=(double *)malloc(n*sizeof(double) );
   for(i=0;i<n;i++)
     temp[i]=nodes[i];
   nodes=(double *)realloc(nodes,
                     (2*n-1)*sizeof(double) );
   count=1;
   nodes[0]=temp[0];
   for(i=0;i<n-1;i++)
     {
       nodes[2*i+1]=(temp[i]+temp[i+1])/2;
       nodes[2*i+2]=temp[i+1];
       count+=2;
     }
   if(!(count==2*n-1))
     printf(" error in nodes vector length\n");
   free(temp);
   return(nodes);
}
```

EOC : Experimental order of convergence

Suppose the error $e = O(h^p)$.
Perform iteratively uniform refinements halfing the meshsize h at every step.

Let e_k be the error at the $k-$th iteration, $h_k = h$ and $h_{k+1} = h/2$
thus we have

$$e_k \simeq C\, h^p \qquad e_{k+1} \simeq C \left(\frac{h}{2}\right)^p$$

$$\frac{e_k}{e_{k+1}} \simeq 2^p \;\rightarrow\; \ln(e_k/e_{k+1}) \simeq p\, \ln(2)$$

and therefore

$$p \simeq \ln(e_k/e_{k+1})/\ln(2)$$

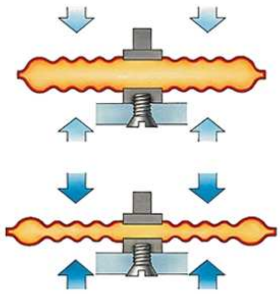
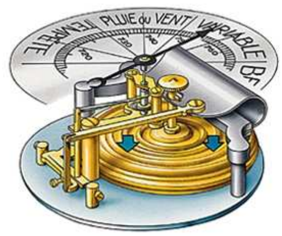
Remark:

For linear finite elements and regular solutions we have $e = O(h)$ and thus $e^2 = O(h^2)$. Therefore in the computation of e^2 we need to use a quadrature formula of order higher than 2 in order to keep quadrature errors "small" in comparison with the discretization error.

```
main (int argc, char * argv[])
{
   int dim, maxiter,iter;
   double xmin,xmax;
   double *uh=0, *mesh=0;
   double err=0, err_o=-1;
   double h,eoc=-1;
   ....
   mesh=MakeUnifMesh(xmin,xmax, dim+1,mesh);
   for(iter=0;iter<maxiter;iter++)
     {
      uh=SolveLins(dim ,mesh, uh);
      err=norm_err(dim,mesh,uh);
      h=mesh[1]-mesh[0];
      if (err_o>=0)
         eoc=log(err_o/err)/log(2);
      err_o=err;
      ....        //print uh err eoc
      if( iter < maxiter-1)
        {
         mesh=refine_mesh(mesh,dim+1);
         dim=2*dim;
        }
     }
}
```

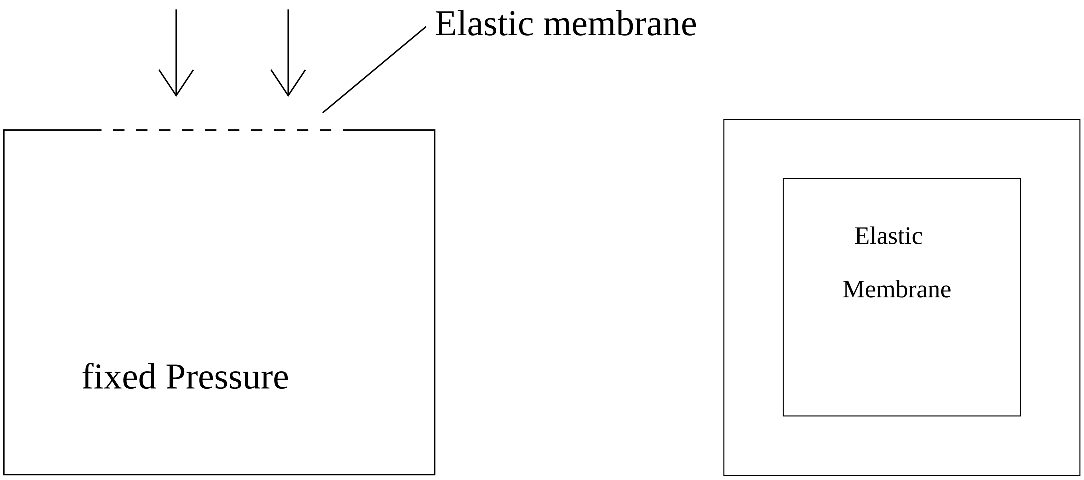
Example 1: a model of pressure sensor

Practical application: aneroid barometer

This is an instrument for measuring pressure without involving any liquid, invented in France by Lucien Vidi in 1844.
It uses a small cilindrical metal box with a flexible basis (aneroid capsule). Inside the capsule there is vacuum so that small changes in the external pressure causes the capsule to expand or contract. With the help of gears and levers these little movements are amplified and dispalyed over a graduate scale on the front of the barometer.

A model problem



$$\begin{cases} -\text{div}(A\,\nabla u) = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases} \qquad (1)$$

where we chose $\Omega = (-1,1)^2 \subset \mathbb{R}^2$

$A = c\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $c \in \mathbb{R}$ material dependent
$f = P_i - P_e$ difference of pressure
$u =$ displacement of the membrane from the reference situation
$g = 0$

We approximate the solution of the model problem with the help of Alberta and look for an answer to the following questions?

(a) What shape does the graph of u assume if the external pressure is bigger than the internal one (i.e f is negative) ?

(b) What happens if the difference of pressure doubles?

In the simulations we set $c = 1$, $g = 0$ and use linear finite elements.

(a) The graph of u is reasonably concave.

(b) with the following values of f we get:

$f = -1$, $u(0,0) \simeq -0.3$ $\quad$ $f = -2$, $u(0,0) \simeq -0.6$

This indicates that u depends linearly on data f.

It is easy to check that the solution u of the model problem (1) depends linearly on the data f and g, infact

$$-\text{div}(c\,\nabla(2u)) = -c\Delta(2u) = -2c\Delta u = 2f \text{ in } \Omega$$

and $2u = 2g$ on $\partial\Omega$.

If $g = 0$ then $2g = g = 0$ and u depends linearly on f i.e.
if u satisfies $-c\Delta u = f$ in Ω and $u = 0$ on $\partial\Omega$ then $2u$ solves $-c\Delta u = 2f$ in Ω and $u = 0$ on $\partial\Omega$.