Renato Poli (rep2656)

CSE397/EM397 – HW#2 03/29/2024

2.1 $P_e \gg 1$

$\|u\|_2 \sim L^{1/2} P_e^{3/2}$ — H^2 - norm

$a\, u_{,x} - K u_{,xx} = 0$ $f = 0$ $u(0) = 0$

$u(L) = 1$

$a\, u_{,x} = K u_{xx}$

Set $v = u_{,x}$

$\frac{a}{K} = \frac{v_{,x}}{v} \Rightarrow \frac{a}{K} x + C_0 = \ln(v)$

$v = C_1 e^{a/K\, x} \Rightarrow u_{,x} = C_1 e^{a/K\, x}$

$u = C_1 \frac{K}{a} \exp\left(\frac{a}{K} x\right) + C_2 \Rightarrow u = C_3 \exp\left(\frac{a}{K} x\right) + C_2$

$u(0) = 0 \Rightarrow C_3 = -C_2$

$u(L) = 1 \Rightarrow 1 = -C_2 \exp\left(\frac{a}{K} L\right) + C_2$

$C_2 = \left(1 - \exp\frac{a}{K} L\right)^{-1}$

$u = \frac{-\exp\frac{a}{K} x + 1}{1 - \exp\frac{a}{K} L}$ Def $P_e = \frac{a L}{K}$

$$\Rightarrow \boxed{u = \frac{1 - \exp\left(P_e \frac{x}{L}\right)}{1 - \exp(P_e)}}$$

$$\|u\|_2 = (u,u)_2^{1/2}$$

$$(u,u)_2 = \int_0^L (u.u + L^2 u_{,x} u_{,x} + L^4 u_{,xx} u_{,xx})\, dx$$

for $Pe \gg 1 \Rightarrow u \simeq \dfrac{\exp Pe \frac{x}{L}}{\exp Pe}$

let $C = \dfrac{1}{\exp Pe}$ $\quad D = \dfrac{Pe}{L}$ $\Bigg\}$ $u = C \exp(Dx)$

$u_{,x} = CD \exp Dx \qquad u_{,xx} = CD^2 \exp(Dx)$

$(u,u)_2 \Rightarrow \quad (u)^2 = C^2 \exp(2Dx)$

$\qquad (u_{,x})^2 = C^2 D^2 \exp(2Dx)$

$\qquad (u_{,xx})^2 = C^2 D^4 \exp(2Dx)$

as $Pe \gg 1$, $(u_{,xx})^2 \gg (u_{,x})^2 \gg (u)^2$

$$\int_0^L C^2 D^4 L^4 \exp(2Dx)\, dx = \int_0^L C^2 Pe^4 \exp(2Dx)\, dx$$

$$\Rightarrow \int_0^L \exp(2Dx)\, dx = \frac{1}{2D}\Big[\exp(2Dx)\Big]_0^L = \frac{1}{2D}\Big[\exp 2DL - 1\Big]$$

$$\exp\left(\frac{2Pe}{2}\right) \gg 1 \Rightarrow \exp 2DL - 1 \approx \exp(2DL)$$

$$(u,u)_2 = \frac{1}{\cancel{\exp(2Pe)}} \, Pe^4 \, \frac{L}{2Pe} \, \cancel{\exp(2Pe)}$$

$$(u,u)_2 = \frac{Pe^3 L}{2}$$

Hence:

$$\|u\| = \sqrt{(u,u)_2} \cong \sqrt{\frac{Pe^3 L}{2}} \approx Pe^{3/2} L^{1/2}.$$

when $Pe \gg 1$.

$e = u - u^h$

(22) $\|e\|^2 = \int_0^1 e^2 dx$

$|e|_1^2 = \int_0^1 e_{,x}^2 dx$

$\|e\|_1^2 = \int_0^1 (e^2 + l^2 e_{,x}^2) dx$

$\|e\| = \beta N_{el}^{-\gamma} \rightarrow \log\|e\| = \log\beta - \gamma \log N_{el}$

—x—

H1 - seminorm

$u = \frac{e^{ax} - 1}{e^a - 1} \longrightarrow u_{,x} = \frac{a}{e^a - 1}(e^{ax} - 1)$

for $a = 500$. $\boxed{u_{,x} \cong a e^{ax-a}}$

$u^h_{,x} = \frac{1}{h}(u_{A+1} - u_A) = C \qquad e_{,x} = u_{,x} - u^h_{,x}$

$e_{,x}^2 = C^2 + a^2 e^{2ax} e^{-2a} - 2a C e^{ax} e^{-a}$

$$\int_{x_0}^{x_1} e_{,x}^2 dx = \left| \begin{array}{l} C^2(x_1 - x_0) \\ \frac{a}{2} e^{-2a}(e^{2ax_1} - e^{2ax_0}) \\ -2C e^{-a}(e^{ax_1} - e^{ax_0}) \end{array} \right.$$

//

2.2 /

As expected, L2-norm of the error convergence rate is $\gamma \approx 2$, and H^1 norms and seminorms convergence rate is $\gamma \approx 1$.

The theoretical convergence rate is observed for Galerkin and SUPG methods only when using a large number of elements (> 200).

Galerkin oscillates with a small number of elements. SUPG is nodally exact in all cases.

In [1]:

```python
import numpy as np
from numpy import tanh
import sys

import matplotlib.pyplot as plt

plt.style.use('paper.mplstyle')
np.set_printoptions(threshold=200, linewidth=200)
plt.figure(figsize=(12,6))

def XI_KB() :
    global METHOD, A, H, KAPPA
    if METHOD == "GALERKIN" :
        return 0
    elif METHOD == "SUPG" :
        if KAPPA != 0 :
            alpha_h = A * H / 2 / KAPPA
            return 1/tanh(alpha_h) - 1/alpha_h
        else:
            return 1
    elif METHOD == "UPWIND" :
        return 1
    else :
        fail(f"XI(): Unknown method {METHOD}")

def XI_F( ) :
    global METHOD, A, H, KAPPA
    if METHOD == "GALERKIN" :
        return 0
    elif METHOD == "SUPG" :
        if KAPPA != 0 :
            alpha_h = A * H / 2 / KAPPA
            return 1/tanh(alpha_h) - 1/alpha_h
        else:
            return 1
    elif METHOD == "UPWIND" :
        return 0
    else :
        fail(f"XI(): Unknown method {METHOD}")


def Usolve() :
    global KAPPA, H, A, N, F, G0, G1
    if N <= 1 :
        return np.array( [ G0, G1 ] )

    Sdif = np.array( [ -1, 2, -1 ] )
    Sadv = np.array( [ -1, 0, 1] )

    K  = np.zeros( [ N-1, N-1 ] )
    B0 = np.zeros( N-1 )
    BN = np.zeros( N-1 )

    xi = XI_KB()
```

```
    B0[0]   = - ( KAPPA + A*H/2*xi )/H - A/2
    BN[N-2] = - ( KAPPA + A*H/2*xi )/H + A/2

    K[0,0]     = 2*( KAPPA + A*H/2*xi )/H
    K[N-2,N-2] = 2*( KAPPA + A*H/2*xi )/H
    if N > 2 :
        K[0,1]     = - ( KAPPA + A*H/2*xi )/H + A/2
        K[N-2,N-3] = - ( KAPPA + A*H/2*xi )/H - A/2

    for i in range( 1, N-2 ) :
        [ K[i,i-1],K[i,i],K[i,i+1] ] = ( KAPPA/H + A/2*xi ) * Sdif + A/2 * Sadv

    U = np.linalg.solve( K,  F - B0 * G0 - BN * G1 )
    U = np.append(U,G1)
    U = np.insert(U,G0,0)

    return U
```

```
<Figure size 864x432 with 0 Axes>
```

In [2]:

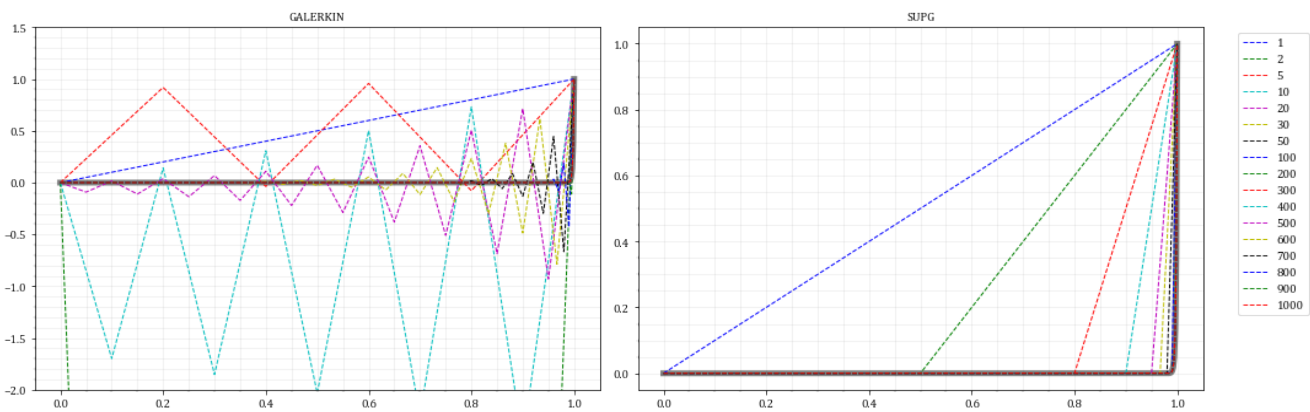
```
from scipy.stats import linregress

# GALERKIN or SUPG

KAPPA = 1
G0 = 0
G1 = 1
A=500

NN = [ 1, 2, 5, 10, 20, 30, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 ]
#NN = [2,10]
# X and solution vectors
XX = []
UUgal = []
UUsupg = []
# Errors
L2Ngal = []
L2Nsupg = []
H1SNgal = []
H1SNsupg = []
H1Ngal = []
H1Nsupg = []

# NUMERICAL SOLUTIONS
X_EXACT = np.linspace( 0, 1, 500 )
U_EXACT = ( np.exp( A*X_EXACT) - 1 ) / ( np.exp(A) - 1)
#
#
#
def _exact(X) :
    global A
    ret = []
    for x in X :
        ret.append( ( np.exp( A*x) - 1 ) / ( np.exp(A) - 1) )
    return ret
```

```python
def _exact_dx(X) :
    global A
    ret = []
    for x in X :
        ret.append( A * ( np.exp( A*x) - 1 ) / ( np.exp(A) - 1) )
    return ret

#
# Calculate the errors
#
def _err( X, U ) :
    global H, A

    L2_NORM = 0
    H1_SEMINORM = 0
    H1_NORM = 0
    for i in range( len(X) - 1 ) :
        X0 = X[i]; X1 = X[i+1]; U0 = U[i]; U1 = U[i+1] ;
        n = 100                       # split the space in small pieces (brute force

        # H1 SEMINORM
        uxh = (U1-U0)/H
        c1 =  A/2 * ( np.exp(2*A*(X1-1)) - np.exp(2*A*(X0-1)) )
        c2 = uxh**2 * (X1 - X0 )
        c3 = -2*uxh*np.exp(-A)*(np.exp(A*X1) - np.exp(A*X0) )
        H1_SEMINORM += c1 + c2 + c3

        # L2 NORM
        dx = ( X1 - X0 ) / n
        _ex_an = U1/H - U0/H
        for j in range(n) :
            x0 = ( X0 * (n-j) + X1 * j ) /n
            x1 = ( X0 * (n-j-1) + X1 * (j+1) ) /n
            u0 = ( (n-j)*U0 + j*U1 ) / n
            u1 = ( (n-j-1)*U0 + (j+1)*U1 ) / n
            u = u0/2 + u1/2
            x = x0/2 + x1/2
            u_exact = _exact([ x0, x1, x ])

            # error
            _e = u_exact[2] - u
            L2_NORM += ( _e **2 ) * dx  # L2 nor: \int e^2 dx

    H1_NORM = ( L2_NORM + H1_SEMINORM ) ** 0.5
    L2_NORM = ( L2_NORM ) ** 0.5
    H1_SEMINORM = ( H1_SEMINORM ) ** 0.5

    return L2_NORM, H1_SEMINORM, H1_NORM

for i in range( len(NN) ) :
    N = NN[i]
    F = np.zeros( N-1 )
    H = 1/N
    X = np.linspace( 0, 1, N+1 )
    XX.append(X)
```

```
    METHOD = "GALERKIN"
    UUgal.append( Usolve() )
    ( l2n, h1sn, h1n ) = _err(X, UUgal[-1] )
    L2Ngal.append( l2n )
    H1SNgal.append( h1sn )
    H1Ngal.append( h1n )

    METHOD = "SUPG"
    UUsupg.append( Usolve() )
    ( l2n, h1sn, h1n ) = _err(X, UUsupg[-1] )
    L2Nsupg.append( l2n )
    H1SNsupg.append( h1sn )
    H1Nsupg.append( h1n )
```

In [10]:

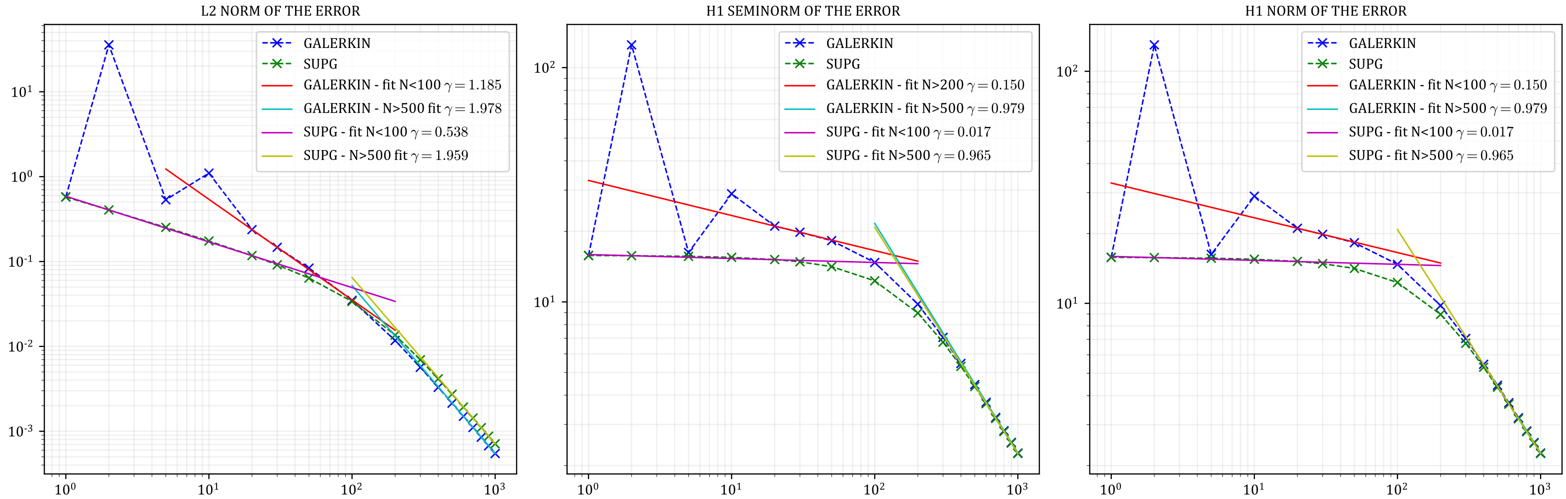
```
## PLOT
fig, [ax1, ax2] = plt.subplots( 1,2, figsize=(15,5))

ax1.plot( X_EXACT, U_EXACT, c='k', lw=5, alpha=.5)
for i in range(len(UUgal)):
    U = UUgal[i]
    X=XX[i]
    ax1.plot( X, U, ls='--', lw=1, label=NN[i])

ax1.set_title(f"GALERKIN")
ax1.set_ylim( -2, 1.5)

###############
ax2.plot( X_EXACT, U_EXACT, c='k', lw=5, alpha=.5)
for i in range(len(UUsupg)):
    U = UUsupg[i]
    X=XX[i]
    ax2.plot( X, U,  ls='--', lw=1, label=NN[i])
ax2.set_title(f"SUPG")
ax2.legend(bbox_to_anchor=(1.05, 1))
fig.tight_layout()
fig.savefig("fig1.pdf")
```

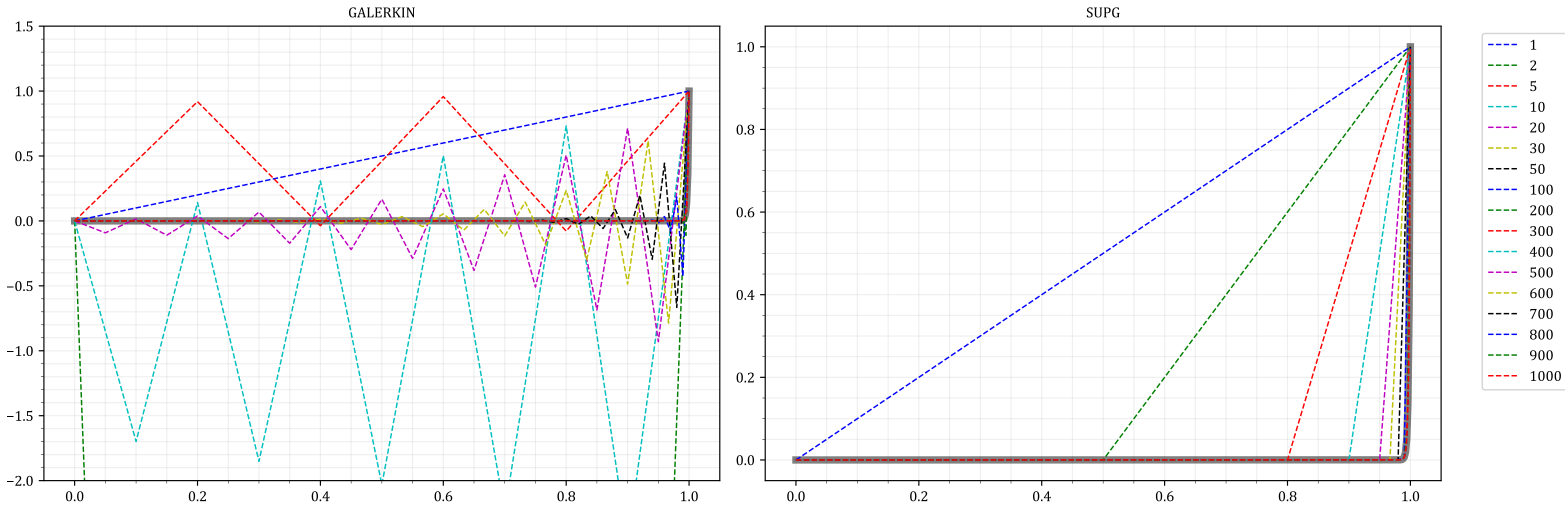


In [11]:

```
fig, [ax3, ax4,ax5] = plt.subplots( 1,3, figsize=(15,5))

## L2 NORMS
ax3.plot( NN, L2Ngal, marker='x', ls='--', lw=1,  label="GALERKIN")
ax3.plot( NN, L2Nsupg, marker='x', ls='--', lw=1,  label="SUPG")
```

```
# Regressions
m, n, r, p, se = linregress(np.log(NN[4:6]), np.log(L2Ngal[4:6]))
ax3.plot( NN[2:9], np.exp(n) * NN[2:9]**m,label=f"GALERKIN - fit N<100 $\gamma={-m:
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(L2Ngal[13:]))
ax3.plot( NN[7:], np.exp(n) * NN[7:]**m,label=f"GALERKIN - N>500 fit $\gamma={-m:.3

m, n, r, p, se = linregress(np.log(NN[0:6]), np.log(L2Nsupg[0:6]))
ax3.plot( NN[:9], np.exp(n) * NN[:9]**m,label=f"SUPG - fit N<100 $\gamma={-m:.3f}$"
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(L2Nsupg[13:]))
ax3.plot( NN[7:], np.exp(n) * NN[7:]**m,label=f"SUPG - N>500 fit $\gamma={-m:.3f}$"

ax3.set_yscale('log')
ax3.set_xscale('log')
ax3.set_title(f"L2 NORM OF THE ERROR")
ax3.legend()

## H1 SEMINORMS
ax4.plot( NN, H1SNgal, marker='x', ls='--', lw=1, label="GALERKIN")
ax4.plot( NN, H1SNsupg, marker='x', ls='--', lw=1, label="SUPG")

m, n, r, p, se = linregress(np.log(NN[4:6]), np.log(H1SNgal[4:6]))
ax4.plot( NN[:9], np.exp(n) * NN[:9]**m,label=f"GALERKIN - fit N>200 $\gamma={-m:.3
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(H1SNgal[13:]))
ax4.plot( NN[7:], np.exp(n) * NN[7:]**m,label=f"GALERKIN - fit N>500 $\gamma={-m:.3

m, n, r, p, se = linregress(np.log(NN[0:6]), np.log(H1SNsupg[0:6]))
ax4.plot( NN[:9], np.exp(n) * NN[:9]**m,label=f"SUPG - fit N<100 $\gamma={-m:.3f}$"
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(H1SNsupg[13:]))
ax4.plot( NN[7:], np.exp(n) * NN[7:]**m,label=f"SUPG - fit N>500 $\gamma={-m:.3f}$"

ax4.set_yscale('log')
ax4.set_xscale('log')
ax4.set_title(f"H1 SEMINORM OF THE ERROR")
ax4.legend()

## H1 NORMS
ax5.plot( NN, H1Ngal, marker='x', ls='--', lw=1, label="GALERKIN")
ax5.plot( NN, H1Nsupg, marker='x', ls='--', lw=1, label="SUPG")

m, n, r, p, se = linregress(np.log(NN[4:6]), np.log(H1Ngal[4:6]))
ax5.plot( NN[:9], np.exp(n) * NN[:9]**m,label=f"GALERKIN - fit N<100 $\gamma={-m:.3
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(H1Ngal[13:]))
ax5.plot( NN[13:], np.exp(n) * NN[13:]**m,label=f"GALERKIN - fit N>500 $\gamma={-m:

m, n, r, p, se = linregress(np.log(NN[0:6]), np.log(H1Nsupg[0:6]))
ax5.plot( NN[:9], np.exp(n) * NN[:9]**m,label=f"SUPG - fit N<100 $\gamma={-m:.3f}$"
m, n, r, p, se = linregress(np.log(NN[13:]), np.log(H1Nsupg[13:]))
ax5.plot( NN[7:], np.exp(n) * NN[7:]**m,label=f"SUPG - fit N>500 $\gamma={-m:.3f}$"

ax5.set_yscale('log')
ax5.set_xscale('log')
ax5.set_title(f"H1 NORM OF THE ERROR")
ax5.legend()
```


L2 NORM OF THE ERROR
GALERKIN
SUPG
GALERKIN - fit N<100 γ = 1.185
GALERKIN - N>500 fit γ = 1.978
SUPG - fit N<100 γ = 0.538
SUPG - N>500 fit γ = 1.959
H1 SEMINORM OF THE ERROR
GALERKIN
SUPG
GALERKIN - fit N>200 γ = 0.150
GALERKIN - fit N>500 γ = 0.979
SUPG - fit N<100 γ = 0.017
SUPG - fit N>500 γ = 0.965
H1 NORM OF THE ERROR
GALERKIN
SUPG
GALERKIN - fit N<100 γ = 0.150
GALERKIN - fit N>500 γ = 0.979
SUPG - fit N<100 γ = 0.017
SUPG - fit N>500 γ = 0.965


GALERKIN
SUPG
1
2
5
10
20
30
50
100
200
300
400
500
600
700
800
900
1000

2.3 $e = \tilde{u} - u$

$$\tilde{u}^h = \frac{x_{A+1} - x}{h} u_A + \frac{x - x_A}{h} u_{A+1}$$

$$u_{A+1} = u + (x_{A+1} - x) u' + \frac{1}{2} (x_{A+1} - x)^2 u''$$

$$u_A = u + (x_A - x) u' + \frac{1}{2} (x_A - x)^2 u''$$

$$h(\tilde{u} - u) = (x_{A+1} - x)\left[\cancel{u} + \cancel{(x_A - x)u'} + \frac{1}{2}(x_A - x)^2 u''\right]$$
$$+ (x - x_A)\left[\cancel{u} + \cancel{(x_{A+1} - x)u'} + \frac{1}{2}(x_{A+1} - x)^2 u''\right] \cancel{- uh}$$

$$2h(e) = (x_{A+1} - x)(x_A - x)^2 u''$$
$$+ (x - x_A)(x_{A+1} - x)^2 u''$$

$$e = u''\left(\frac{1}{2h}\right)\left[(x_{A+1} - x)(x_A - x)^2 + (x - x_A)(x_{A+1} - x)^2\right]$$

$$\sup(e) \Rightarrow x = h/2 = \frac{x_{A+1} + x_A}{2}$$

$$\sup(e) = u''\left(\frac{1}{2h}\right)\left[\left(\frac{x_{A+1} - x_A}{2}\right)\left(\frac{x_A - x_{A+1}}{2}\right)^2 + \left(\frac{x_{A+1} - x_A}{2}\right)\left(\frac{x_{A+1} - x_A}{2}\right)^2\right]$$

$$= u''\left(\frac{1}{2h}\right)\left[\frac{h^3}{8} + \frac{h^3}{8}\right] = \frac{u'' h^2}{8} \quad //$$

For the derivative:

$$\tilde{u}^h_{,x} = \frac{u_{A+1} - u_A}{h}$$

$$e = \tilde{u}^h_{,x} - u_{,x} =$$

$$\frac{1}{h}\Big(\cancel{u} + \cancel{(x_{A+1} - x)u'} + (x_{A+1} - x)^2 u''$$

$$\cancel{-u} - \cancel{(x_A - x)u'} - (x_A - x)^2 u''\Big) - \cancel{u_{,x}}$$

$$e = \frac{u''}{h}\left[(x_{A+1} - x)^2 - (x_A - x)^2\right]$$

$$\sup(|e|) = u''\left(\frac{1}{h}\right)\left[h^2\right] = u'' h$$ //

2.4 - SIMPLE INVERT ESTIMATES

In [5]:

```
import numpy as np
from sympy import *
from IPython.display import display

h_, l_, x_ = symbols('h \lambda x', real=True)

def calc_cinv( Nlhs, Nrhs ) :
    NN1 = Nlhs * Nlhs.transpose()
    NN2 = Nrhs * Nrhs.transpose()

    K = integrate( NN1, (x_, 0, h_) )
    M = integrate( NN2, (x_, 0, h_) )

    L = solve( (K-l_*M).det() , l_ )
    return L

X = [ 0, h_/2, h_ ]
Na_quad = Matrix(
            [ (x_-X[1])*(x_-X[2])/(X[0]-X[1])/(X[0]-X[2]),
              (x_-X[0])*(x_-X[2])/(X[1]-X[0])/(X[1]-X[2]),
              (x_-X[0])*(x_-X[1])/(X[2]-X[0])/(X[2]-X[1])
            ])

Na_lin = Matrix( [ 1 - x_/h_, x_/h_ ] )
```

In [6]:

```
Na = Na_lin
Na_x = diff( Na, x_ )
Lambda = calc_cinv( Na_x, Na )

print("Linear shape function - C/h=");
display(sqrt(Lambda[1]))
```

```
Linear shape function - C/h=
```

$$\frac{2\sqrt{3}}{|h|}$$

In [7]:
```
Na = Na_quad
Na_x = diff( Na, x_ )

Lambda = calc_cinv( Na_x, Na )

print("Quadratic shape function - C/h=");
display(sqrt(Lambda[2]))
```

Quadratic shape function - C/h=

$$\frac{2\sqrt{15}}{|h|}$$

In [8]:
```
Na = Na_quad
Na_xx = diff( Na, x_, x_ )

Lambda = calc_cinv( Na_xx, Na )
print("Quadratic shape function (||w_xx|| < C ||w||)- C/h^2=");
display(sqrt(Lambda[1]))
```

Quadratic shape function (||w_xx|| < C ||w||)- C/h^2=

$$\frac{12\sqrt{5}}{h^2}$$

In [9]:
```
Na = Na_quad
Na_xx = diff( Na, x_, x_ )
Na_x = diff( Na, x_ )
Lambda = calc_cinv( Na_xx, Na_x )

print("Quadratic shape function (||w_xx|| < C ||w_x||)- C/h= 0");
Lambda
```

Quadratic shape function (||w_xx|| < C ||w_x||)- C/h= 0

Out[9]: []

Note:
The quadratic shape function results in a singular matrix.
That is because H1 semi-norm of constant functions are zero, so that semi-norm has a non-trivial kernel.

We observe that the inequality for quadratics is the same as the standard inverse inequality for linears and, therefore, the result for linears applies.