1 Verification of the Exact Solution

$$u(x,y) = sin(y - x^2)$$

1.1 Dirichlet Boundary Condition

$$u(x, 1-x^4) = sin(1 - x^4 - x^2) = sin(1 - x^2 - x^4) = g(x,y)$$

1.2 Neumann Boundary Condition

1.2.1 $y = 0$, $1 < |x| < 2$

$$\frac{\partial u}{\partial n} = -cos(y - x^2)$$

$$k\frac{\partial u}{\partial n}|_{y=0} = (-1)(0+1)cos(0 - x^2) = -cos(-x^2) = -cos(x^2)$$

1.2.2 $x = -2$, $0 \leq y \leq 2$

$$\frac{\partial u}{\partial n} = (-1)(-2x)cos(y - x^2) = 2xcos(y - x^2)$$

$$k\frac{\partial u}{\partial n}|_{x=-2} = (y+1)(2)(-2)cos(y-4) = -4(y+1)cos(y-4)$$

1.2.3 $x = 2$, $0 \leq y \leq 2$

$$\frac{\partial u}{\partial n} = -2xcos(y - x^2)$$

$$k\frac{\partial u}{\partial n}|_{x=2} = (y+1)(-2)(2)cos(y-4) = -4(y+1)cos(y-4)$$

1.2.4 $y = 2$, $-2 \leq x \leq 2$

$$\frac{\partial u}{\partial n} = cos(y - x^2)$$

$$k\frac{\partial u}{\partial n}|_{y=2} = (2+1)cos(2 - x^2) = 3cos(2 - x^2)$$

1.3 Differential Equation

$$\nabla u = \nabla(sin(y - x^2)) = (-2xcos(y - x^2), cos(y - x^2))$$

$$k(x,y)\nabla u = (y+1)(-2xcos(y - x^2), cos(y - x^2))$$

$$-\nabla\cdot(k(x,y)\nabla u) = -(-2(y+1)cos(y-x^2)-(-2x)(-2x)(y+1)sin(y-x^2), cos(y-x^2)-(y+1)sin(y-x^2))$$

$$-\nabla \cdot (k(x,y)\nabla u) = -((-2y - 2 + 1)cos(y - x^2) - (y+1)(4x^2+1)sin(y - x^2))$$

$$-\nabla \cdot (k(x,y)\nabla u) = (2y+1)cos(y - x^2) + (4x^2+1)(y+1)sin(y - x^2)$$

2 Derivation of the Weak Form of the Boundary Value Problem

Let $u = w + G$, such that $G = g(x,y)$ on Γ_1, $w = 0$ on Γ_1.

$$-\nabla \cdot (k(x,y)\nabla u) = -\nabla \cdot (k(x,y)\nabla w) - \nabla \cdot (k(x,y)\nabla G) = f(x,y)$$

Let $v \in V = \{v \in H^1(\Omega) | v = 0 \text{ on } \Gamma_1\}$

$$-\nabla \cdot (k\nabla w)v - \nabla \cdot (k\nabla G)v = fv$$

$$-\iint_\Omega \nabla \cdot (k\nabla w)v\,dx\,dy - \iint_\Omega \nabla \cdot (k\nabla G)v\,dx\,dy = \iint_\Omega fv\,dx\,dy$$

Product Rule

$$-\iint_\Omega \nabla\cdot(k\nabla w v)\,dx\,dy+\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy-\iint_\Omega \nabla\cdot(k\nabla G v)\,dx\,dy+\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy$$

Divergence Theorem

$$-\int_{\partial\Omega} k\frac{\partial w}{\partial n}v\,dS+\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy-\int_{\partial\Omega} k\frac{\partial G}{\partial n}v\,dS+\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy$$

$$-\int_{\Gamma_1} k\frac{\partial w}{\partial n}v\,dS-\int_{\Gamma_2} k\frac{\partial w}{\partial n}v\,dS+\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy-\int_{\Gamma_1} k\frac{\partial G}{\partial n}v\,dS-\int_{\Gamma_2} k\frac{\partial G}{\partial n}v\,dS$$
$$+\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy$$

$v=0$ on Γ_1

$$-\int_{\Gamma_2} k\frac{\partial u}{\partial n}v\,dS+\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy$$

$k\frac{\partial u}{\partial n}=h$ on Γ_2

$$\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy+\int_{\Gamma_2} hv\,dS$$

Weak Problem:
$\Omega=\{(x,y)|-2<x<2,0<y<2\}\setminus\{(x,y)|y\le 1-x^4\}$
Given $G\in H^1(\Omega)$, $G=sin(1-x^2-x^4)$ when $y=1-x^4$, find $u=w+G$, $w\in H^1(\Omega)$ such that $w=0$ when $y=1-x^4$ and

$$\iint_\Omega (y+1)\nabla w\cdot\nabla v\,dx\,dy=\iint_\Omega (2y+1)cos(y-x^2)+(4x^2+1)(y+1)sin(y-x^2)\,dx\,dy$$
$$-\iint_\Omega (y+1)\nabla G\cdot\nabla v\,dx\,dy-\int_{-2}^{-1} cos(x^2)v(x,0)\,dx-\int_1^2 cos(x^2)v(x,0)\,dx$$
$$-\int_0^2 4(y+1)cos(y-4)v(2,y)\,dy-\int_0^2 4(y+1)cos(y-4)v(-2,y)\,dy$$
$$+\int_{-2}^2 3cos(2-x^2)v(x,2)\,dx$$

for all $v\in H^1(\Omega)$ such that $v=0$ when $y=1-x^4$.

3 Justification that the Weak Problem Has a Unique Solution

Let $v\in V=\{v\in H^1(\Omega)|v=0 \text{ when } y=1-x^4\}$
Weak Problem:
Given $G\in H^1(\Omega)$, $G=sin(1-x^2-x^4)$ when $y=1-x^4$, find $u=w+G$, $w\in V$ such that

$$\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy=\iint_\Omega fv\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy+\int_{\Gamma_2} hv\,dS$$

for all $v\in V$.
Let $a(w,v)=\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy$ and $l(v)=\iint_\Omega fv\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v\,dx\,dy+\int_{\Gamma_2} hv\,dS$.

$$a(v,w)=\iint_\Omega k\nabla v\cdot\nabla w\,dx\,dy=\iint_\Omega k\nabla w\cdot\nabla v\,dx\,dy=a(w,v)$$

$$a(\alpha w_1+\beta w_2,v)=\iint_\Omega k\nabla(\alpha w_1+\beta w_2)\cdot\nabla v\,dx\,dy=\iint_\Omega k(\nabla\alpha w_1+\nabla\beta w_2)\cdot\nabla v\,dx\,dy$$
$$=\alpha\iint_\Omega k\nabla w_1\cdot\nabla v\,dx\,dy+\beta\iint_\Omega k\nabla w_2\cdot\nabla v\,dx\,dy=\alpha a(w_1,v)+\beta a(w_2,v)$$

$$a(w,\alpha v_1+\beta v_2)=\iint_\Omega k\nabla w\cdot\nabla(\alpha v_1+\beta v_2)\,dx\,dy=\iint_\Omega k\nabla w\cdot(\alpha\nabla v_1+\beta\nabla v_2)\,dx\,dy$$
$$=\alpha\iint_\Omega k\nabla w\cdot\nabla v_1\,dx\,dy+\beta\iint_\Omega k\nabla w\cdot\nabla v_2\,dx\,dy=\alpha a(w,v_1)+\beta a(w,v_2)$$

$a(w,v)$ is symmetric and bilinear.

$$a(w,w)=\iint_\Omega k\nabla w\cdot\nabla w\,dx\,dy=\iint_\Omega (y+1)|\nabla w|^2\,dx\,dy\geq\iint_\Omega (0+1)|\nabla w|^2\,dx\,dy=\iint_\Omega|\nabla w|^2\,dx\,dy\geq 0$$

If $w=0$, $a(w,w)=0$.

$$a(w,w)\geq\iint_\Omega|\nabla w|^2\,dx\,dy\geq c||w||^2_{H^1(\Omega)}$$

for some $c>0$ by Poincare's Inequality. Then

$$a(w,w)=0\implies c||w||^2_{H^1(\Omega)}\implies w=0$$

Thus $a(w,v)$ is an inner product on V.
Let $||v||_E=\sqrt{a(v,v)}$.

$$a(w,v)=\iint_\Omega h\nabla w\cdot\nabla v\,dx\,dy=\iint_\Omega(y+1)\nabla w\cdot\nabla v\,dx\,dy\leq\iint_\Omega(2+1)|\nabla w\cdot\nabla v|\,dx\,dy$$
$$\leq 3\iint_\Omega|\nabla w||\nabla v|\,dx\,dy\leq 3||\nabla w||_{L^2(\Omega)}||\nabla v||_{L^2(\Omega)}\leq 3||w||_{H^1(\Omega)}||v||_{H^1(\Omega)}$$

so $a(w,v)$ is bounded.

$$a(w,w)\leq 3||w||^2_{H^1(\Omega)}\implies||w||_E=\sqrt{a(w,w)}\leq\sqrt{3}||w||_{H^1(\Omega)}$$

So

$$\sqrt{c}||w||_{H^1(\Omega)}\leq||w||_E\leq\sqrt{3}||w||_{H^1(\Omega)}$$

Then $||\cdot||_E$ and $||\cdot||_{H^1(\Omega)}$ are equivalent. Thus V is also a Hilbert space under $||\cdot||_E$.

$$l(\alpha v_1+\beta v_2)=\iint_\Omega f(x,y)(\alpha v_1+\beta v_2)\,dx\,dy-\iint_\Omega k(x,y)\nabla G\cdot\nabla(\alpha v_1+\beta v_2)\,dx\,dy+\int_{\Gamma_2}h(x,y)(\alpha v_1+\beta v_2)\,dS$$

$$l(\alpha v_1+\beta v_2)=\alpha[\iint_\Omega fv_1\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v_1\,dx\,dy+\int_{\Gamma_2}hv_1\,dS]$$
$$+\beta[\iint_\Omega fv_2\,dx\,dy-\iint_\Omega k\nabla G\cdot\nabla v_2\,dx\,dy+\int_{\Gamma_2}hv_2\,dS]$$
$$l(\alpha v_1+\beta v_2)=\alpha l(v_1)+\beta l(v_2)$$

So $l(v)$ is linear.

$$|l(v)|\leq\iint_\Omega|fv|\,dx\,dy+\iint_\Omega|k||\nabla G\cdot\nabla v|\,dx\,dy+\int_{\Gamma_2}|hv|\,dS$$
$$\leq\iint_\Omega|f||v|\,dx\,dy+\iint_\Omega|y+1||\nabla G||\nabla v|\,dx\,dy+\max_{v\in V}\int_{\Gamma_2}|hv|\,dS$$

Let

$$Q=\frac{\max_{v\in V}\int_{\Gamma_2}|hv|\,dS}{||v||_{H^1(\Omega)}}$$

Then

$$|l(v)|\leq\iint_\Omega|f||v|\,dx\,dy+3\iint_\Omega|\nabla G||\nabla v|\,dx\,dy+Q||v||_{H^1(\Omega)}$$
$$\leq||f||_{L^2(\Omega)}||v||_{L^2(\Omega)}+3||\nabla G||_{L^2(\Omega)}||\nabla v||_{L^2(\Omega)}+Q||v||_{H^1(\Omega)}$$
$$\leq||f||_{L^2(\Omega)}||v||_{H^1(\Omega)}+3||G||_{H^1(\Omega)}||v||_{H^1(\Omega)}+Q||v||_{H^1(\Omega)}$$
$$=(||f||_{L^2(\Omega)}+3||G||_{H^1(\Omega)}+Q)||v||_{H^1(\Omega)}$$

Let $M=||f||_{L^2(\Omega)}+3||G||_{H^1(\Omega)}+Q$. Then $|l(v)|\leq M||v||_{H^1(\Omega)}$ so $l(v)$ is bounded under $||v||_{H^1(\Omega)}$. $||\cdot||_E$ and $||\cdot||_{H^1(\Omega)}$ are equivalent, so $l(v)$ is also bounded under $||\cdot||_E$. Thus $l(v)$ is a continuous linear functional under $||\cdot||_E$.
By the Riesz Representation Theorem, there exists a unique $w^*\in V$ such that $l(v)=a(w^*,v)$ for all $v\in V$ and $u=w^*+G$ is the unique solution of the weak problem.

4 Explanation of Solving the Weak Form

From the weak form of the BVP, we have the weak form on each element Ω_e:

$$a^e(w,v) = l^e(v) \quad \text{for all } v \in V,$$

where

$$a^e(w,v) = \iint_{\Omega_e} k\nabla w \cdot \nabla v \; dxdy,$$

$$l^e(v) = \iint_{\Omega_e} fv \; dxdy \; - \iint_{\Omega_e} k\nabla G \cdot \nabla v \; dxdy \; + \int_{\partial\Omega_e} vk\frac{\partial u}{\partial n} \; ds.$$

Now, for a given triangulation, let h be the length of the longest edge in the triangulation. Let $\{\psi_j^e\}$ be the set of shape functions on an element Ω_e. Now, let the approximation of $u(x,y)$ on the element Ω_e be:

$$u_h^e = w_h^e + G_h^e$$

where $w_h^e = \sum_{j=1}^m w_j^e \psi_j^e$, $G_h^e = \sum_{j=1}^m g_j^e \psi_j^e$. Here, w_j^e and g_j^e are the function values of w_h^e and G_h^e at the j^{th} node in Ω_e. Now, we can substitute this definition for w into our weak form and choose $v = \psi_i^e$ for any $i = 1, 2, ..., m$. Then we have:

$$a^e(\textstyle\sum_{j=1}^m w_j^e \psi_j^e, \psi_i^e) = l^e(\psi_i^e) \quad \text{for } i = 1, 2, ..., m.$$

By the bilinearity of a, we have:

$$\sum_{j=1}^m w_j^e a^e(\psi_j^e, \psi_i^e) = l^e(\psi_i^e) \quad \text{for } i = 1, 2, ..., m.$$

Therefore, if we let $k_{ij}^e = a^e(\psi_j^e, \psi_i^e)$, $f_i^e = l^e(\psi_i^e)$ for $i = 1, 2, ..., m$, then we have the matrix equation:

$$K^e W^e = F^e,$$

where $K^e = (k_{ij}^e), \quad F^e = (f_i^e), \quad W^e = (w_j^e)$.

Now, we cannot calculate the term $\int_{\partial\Omega_e} k\frac{\partial u}{\partial n}\psi_i^e \; ds$ in f_i^e since it requires the unknown function u. However, by the condition of continuity of our approximated solution across the edges of adjacent triangles, we can calculate only the contribution on Γ_2, that is, $\int_{\partial\Omega_e \cap \Gamma_2} h\psi_i^e \; ds$.

Finally, to construct the global solution W, note that we only need the values at every free node, that is, all nodes in Ω not on Γ_1. We let $\{\psi^{f_i}\}$ be the set of global shape functions on the free nodes f_i. Then

$$K = \sum_{e=1}^N k_{ij}^e$$

where local nodes i, j are both free nodes. Also,

$$F = \sum_{e=1}^N f_i^e$$

where local node i is a free node. Then, we can form the global matrix equation

$$KW = F.$$

After solving this equation for W, we construct the approximated solution u_h by:

$$u_h = \sum_{j=1}^m w(n_{f_j})\psi^{f_j} + \sum_{i=1}^k g(n_{c_i})\psi^{c_i},$$

where n_{f_j}, n_{c_i} are the free and constrained nodes, respectively.

In our code, we created a wrapper function myFE2dbvp to compute the approximated solution for a given maximum size of h for our triangulation. Firstly, we set up a loop to continuously refine our triangulation until the maximum side length is less than or equal to the given value. Then we initialize the values for midnodes and triangle midpoints for both linear and quadratic shape functions, and we sort the edges array so that we can expect smaller node values to appear in the first column. Then, we call previously created functions to calculate the boolean array which store whether nodes and edges are on the boundary, and whether they are a part of Γ_1 or Γ_2. Then we create the indVec array, which keeps track of free nodes. Then, with the largest number of quadrature points that we have implemented, we construct the stiffness matrix K and load vector F. We then solve this matrix equation for W, and use the given functions to construct the approximated solution u_h from W.

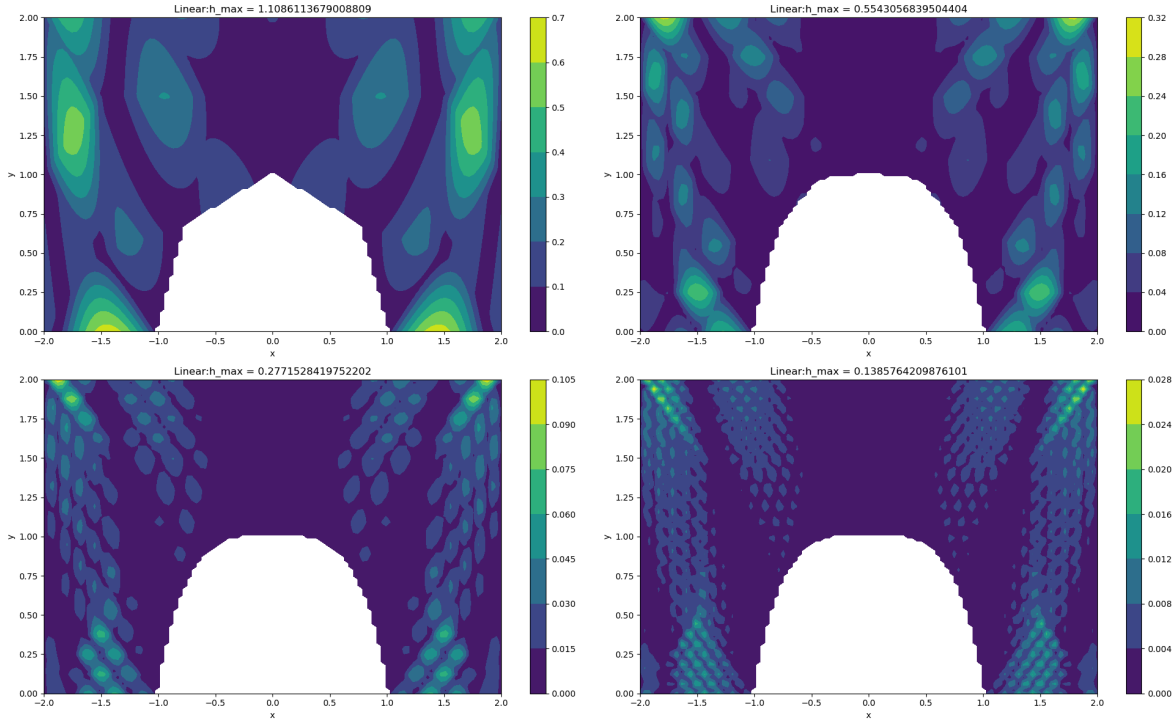
5. Figures that show the absolute differences between the exact solution and the computed solutions obtained by discretizing the domain so the maximal length of the edges does not exceed 1.5, 1, 0.5 and 0.25 by using linear shape functions. Compare the figures and explain your observations.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import tri
import plotly.graph_objects as go
from fn import *
from myFE1dibvp import *
import time
```

In [2]:
```
u_true = lambda x, y: np.sin(y-x**2)
u_true_grad = lambda x, y: np.stack([-2*x*np.cos(y-x**2), np.cos(y-x**2)],
                                    axis=-1)
k = lambda x, y: y+1
f = lambda x, y: (2*y+1)*np.cos(y-x**2)+(4*x**2+1)*(y+1)*np.sin(y-x**2)
g = lambda x, y: np.sin(y-x**2)
h = lambda x, y: (-np.cos(x**2))*((abs(x-1.5)<0.5+1e-12)+
                                  (abs(x+1.5)<0.5+1e-12))*(abs(y)<1e-12)+\
                 (-4*(y+1)*np.cos(y-4))*(abs(x+2)<1e-12)*(abs(y-1)<1+1e-12)+\
                 (-4*(y+1)*np.cos(y-4))*(abs(x-2)<1e-12)*(abs(y-1)<1+1e-12)+\
                 (3*np.cos(2-x**2))*(abs(x)<2+1e-12)*(abs(y-2)<1e-12)+0*x+0*y
mesh = np.load('initial_triangulation.npz')
bdyFn = lambda x: 1-x**4
bdyFnder = lambda x: -4*x**3
maxh_values = [1.5, 1.0, 0.5, 0.25]
x = np.linspace(-2,2,100)
y = np.linspace(0,2,100)
X, Y = np.meshgrid(x,y)
```

In [3]:
```
plt.figure(figsize=(20, 12))
for i, maxh in enumerate(maxh_values):
    start = time.time()
    nodes = mesh['nodes']
    triangles = mesh['triangles']
    edges = mesh['edges']
    bdyNode, bdyEdge = bdyNodeEdge(nodes, edges, triangles)
    curveEdge1 = curveEdge(edges, Gamma1Nodes(nodes))
    uh, _, hMax, nodes, triangles, _ = myFE2dbvp(k, f, g, h, maxh, nodes,
        triangles, edges,bdyNode, bdyEdge, curveEdge1, bdyFn, bdyFnder, shapeFn=
    error = np.abs(u_true(X, Y) - uh(X, Y))
    plt.subplot(2, 2, i + 1)
    contour = plt.contourf(X, Y, error)
    plt.colorbar(contour)
    plt.title(f'Linear:h_max = {hMax}')
    plt.xlabel('x')
    plt.ylabel('y')
    end = time.time()
    print(f'for h_max = {hMax}, time = {end - start}s')
plt.tight_layout()
plt.show()
```

```
for h_max = 1.1086113679008809, time = 0.1771705150604248s
for h_max = 0.5543056839504404, time = 0.7583968639373779s
for h_max = 0.2771528419752202, time = 3.38191556930542s
for h_max = 0.1385764209876101, time = 24.749504804611206s
```



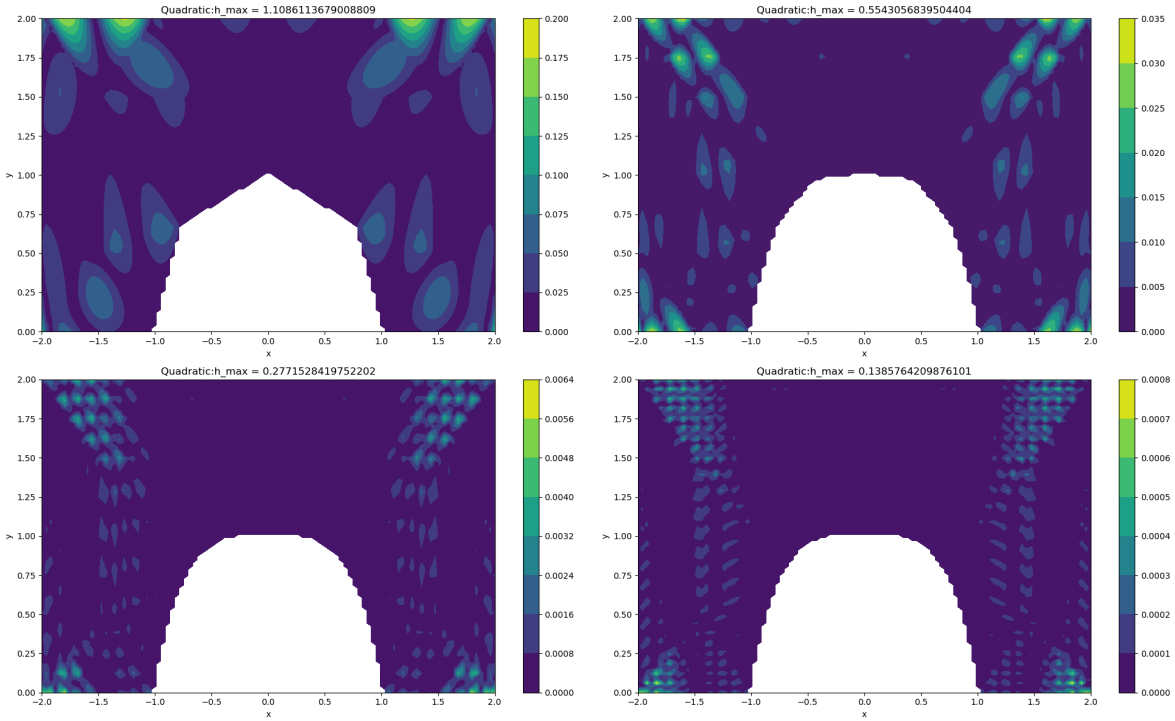
Overall, the error increases along the Neumann boundary, particularly on the four corner points. In contrast, the error near the Dirichlet boundary is smaller, indicating better stability in that region. After refinement of triangle, the overall error decreases; however, it still remains higher error near the Neumann boundary compared with error on other region.

6. Figures that show the absolute differences between the exact solution and the computed solutions obtained by discretizing the domain so the maximal length of the edges does not exceed 1.5, 1, 0.5 and 0.25 by using quadratic shape functions. Compare the figures and explain your observations.

In [4]:
```
x = np.linspace(-2,2,100)
y = np.linspace(0,2,100)
X, Y = np.meshgrid(x,y)
plt.figure(figsize=(20, 12))
for i, maxh in enumerate(maxh_values):
    start = time.time()
    nodes = mesh['nodes']
    triangles = mesh['triangles']
    edges = mesh['edges']
    bdyNode, bdyEdge = bdyNodeEdge(nodes, edges, triangles)
    curveEdge1 = curveEdge(edges, Gamma1Nodes(nodes))
    uh, _, hMax, nodes, triangles, _ = myFE2dbvp(k, f, g, h, maxh, nodes,
            triangles, edges, bdyNode, bdyEdge, curveEdge1, bdyFn, bdyFnder, sha
    error = np.abs(u_true(X, Y) - uh(X, Y))
    error = np.abs(u_true(X, Y) - uh(X, Y))
    plt.subplot(2, 2, i + 1)
    contour = plt.contourf(X, Y, error)
    plt.colorbar(contour)
    plt.title(f'Quadratic:h_max = {hMax}')
```

```
    plt.xlabel('x')
    plt.ylabel('y')
    end = time.time()
    print(f'for h_max = {hMax}, time = {end - start}s')
plt.tight_layout()
plt.show()
```

```
for h_max = 1.1086113679008809, time = 1.035292148590088s
for h_max = 0.5543056839504404, time = 5.072058916091919s
for h_max = 0.2771528419752202, time = 22.638210773468018s
for h_max = 0.1385764209876101, time = 128.17616033554077s
```



Overall, the error increases along the Neumann boundary, particularly at the four corner points. For other regions, the error distribution is quite smooth, especially near the Dirichlet boundary. After refining the triangle, the overall error decreases. However, even with a smaller maximum edge size (h_max), the error near the Neumann boundary persists.

7. Figures that show
   (a) the absolute differences between the exact solution and the best computed solution using linear shape functions.
   (b) the absolute differences between the exact solution and the best computed solution using quadratic shape functions.
   Compare these two figures and explain your observations.

In [6]:
```
maxh = 0.25
plt.figure(figsize=(20, 12))
nodes = mesh['nodes']
triangles = mesh['triangles']
edges = mesh['edges']
bdyNode, bdyEdge = bdyNodeEdge(nodes, edges, triangles)
curveEdge1 = curveEdge(edges, Gamma1Nodes(nodes))

uh_linear, _, hMax1, nodes, triangles1, _ = myFE2dbvp(k, f, g, h, maxh, nodes,
                triangles, edges, bdyNode, bdyEdge, curveEdge1, bdyFn, bdyFnder,
```

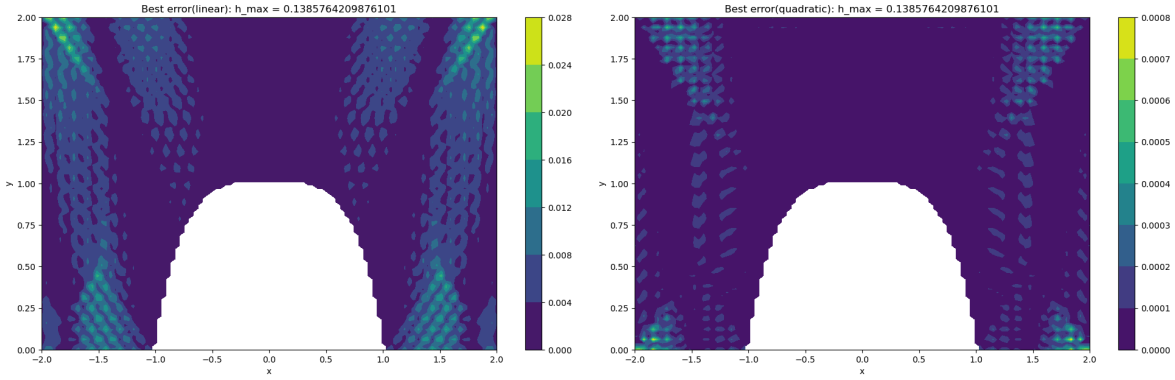
```
error_linear = np.abs(u_true(X, Y) - uh_linear(X, Y))

nodes = mesh['nodes']
triangles = mesh['triangles']
edges = mesh['edges']
bdyNode, bdyEdge = bdyNodeEdge(nodes, edges, triangles)
curveEdge1 = curveEdge(edges, Gamma1Nodes(nodes))
uh_quadratic, _, hMax2, nodes, triangles2, _ = myFE2dbvp(k, f, g, h, maxh, nodes
                triangles, edges, bdyNode, bdyEdge, curveEdge1, bdyFn, bdyFnder,
error_quadratic = np.abs(u_true(X, Y) - uh_quadratic(X, Y))

plt.subplot(2, 2, 1)
contour = plt.contourf(X, Y, error_linear)
plt.colorbar(contour)
plt.title(f'Best error(linear): h_max = {hMax1}')
plt.xlabel('x')
plt.ylabel('y')

plt.subplot(2, 2, 2)
contour = plt.contourf(X, Y, error_quadratic)
plt.colorbar(contour)
plt.title(f'Best error(quadratic): h_max = {hMax2}')
plt.xlabel('x')
plt.ylabel('y')

plt.tight_layout()
plt.show()
```

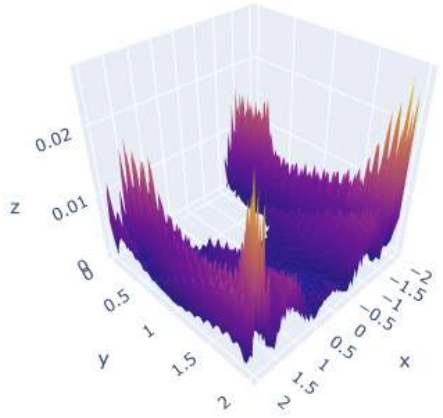


Compared to the error from the linear model, the overall error is significantly reduced when using quadratic method. Although the error remains highly concentrated along the Neumann boundary, its magnitude is greatly diminished. In the quadratic model, the error distribution is smoother and the error near the Dirichlet boundary is smaller. In addition, the ingularity at the boundary corners makes it difficult to completely eliminate the error.

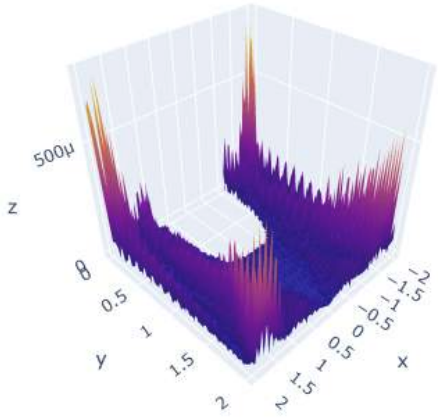
In [ ]:
```
approx1 = go.Surface(x = X, y = Y, z = error_linear)
fig1 = go.Figure(approx1)
fig1.update_layout(title="Error Distribution Using Linear Elements")
fig1.show()

approx2 = go.Surface(x = X, y = Y, z = error_quadratic)
fig2 = go.Figure(approx2)
fig2.update_layout(title="Error Distribution Using Quadratic Elements")
fig2.show()
```

Error Distribution Using Linear Elements



Error Distribution Using Quadratic Elements



8 Convergence of Method

	Linear Shape Functions		Quadratic Shape Functions	
hMax	L^2-norm	H^1-norm	L^2-norm	H^1-norm
1.1086	5.3055×10^{-1}	2.8200×10^{0}	7.2769×10^{-2}	8.4163×10^{-1}
0.5543	1.7360×10^{-1}	1.5695×10^{0}	1.1051×10^{-2}	2.6353×10^{-1}
0.2772	4.6975×10^{-2}	8.1250×10^{-1}	1.4224×10^{-3}	6.8219×10^{-2}
0.1386	1.2016×10^{-2}	4.0997×10^{-1}	1.8162×10^{-4}	1.7361×10^{-2}
Order of Convergence	1.8279	0.92962	2.8897	1.8747

Based on theory, we would expect convergence using linear shape functions to be of order h^2 in the L^2-norm, and order h in the H^1-norm. Using quadratic shape functions, we expect convergence of order h^3 in the L^2-norm and order h^2 in the H^1-norm. We find our code approaches these theoretical values, but is still of a significantly smaller order. We believe this is due to the existence of sharp corners in our domain, where the boundary condition on Γ_2 is discontinuous. If we were to continue to decrease the maximum side length in our triangulation, we would expect to approach the theoretical convergence rate, but due to hardware limitations this would be computationally expensive.