Norges teknisk–naturvitenskapelige universitet
Department of Mathematical Sciences

TMA4220 FEM
Fall 2022

**Project 1**

Group 6: Mats Seglem and Gard Strøm

## Introduction

This is the first part of the programming exercise in the course TMA4220 Numerical solution of differential equations using element methods. This part will build a solid code base for the second part.

## Gaussian quadrature

For the finite element code to work, we need a way to evaluate definite integrals numerically. In this project, we use Gaussian Quadrature.

1.1 In one dimension, the Gauss quadrature takes the form

$$\int_{-1}^{1} g(z)\mathrm{d}z \approx \sum_{q=1}^{N_q} \rho_q g(z_q),$$

where $N_q$ is the number of number of integration points, $z_q$ are the Gaussian quadrature points and $\rho_q$ are the associated Gaussian weights. The Gauss quadrature nodes and weights can be seen in table 1 in the project description. We implemented the function quadrature1D($a$, $b$, $N_q$, $g$) in python to evaluate the integral of 1D functions numerically. To check our implementation, we tested our implementation on the integral

$$\int_{1}^{2} e^x \mathrm{d}x = e^2 - e^1 \approx 4.67077.$$

This comparison can be seen in figure 1.

| Nq | Value | Difference from exact |
|---|---|---|
| 1 | 2.71828 | 1.95249 |
| 2 | 4.66973 | 0.00104776 |
| 3 | 4.67077 | 2.2401e-06 |
| 4 | 4.67077 | 2.53607e-09 |

Figur 1: Comparison between our implementation of 1D Gaussian quadrature and the exact solution of $\int_1^2 e^x dx$. $N_q$ denotes number of integration points, value denotes the evaluation of the integral with $N_q$ integration points and the last column denotes the difference between approximated and exact solution.

We can see that this function works as it should.

1.2 We also implemented a function quadrature2D($p_1$, $p_2$, $p_3$, $N_q$, g) to evaluate the integral of functions in $\mathbb{R}^2$ numerically. To check our implementation, we compared with the analytically solution of the integral

$$\iint_\Omega \log(x+y)dxdy \approx 1.16541$$

where $\Omega$ is the triangle with corners in $(1,0)$, $(3,1)$, $(3,2)$. The comparison can be seen in figure 2

| Nq | Value | Difference from exact |
|---|---|---|
| 1 | 1.20397 | -0.0385628 |
| 3 | 1.17299 | -0.00758347 |
| 4 | 1.16792 | -0.00250996 |

Figur 2: Comparison between our implementation of 2D Gaussian quadrature and the exact solution of $\iint_\Omega \log(x+y)dxdy$. $N_q$ denotes number of integration points, value denotes the evaluation of the integral with $N_q$ integration points and the last column denotes the difference between approximated and exact solution.

We can see that this function also works as it should.

## 2D Poisson

We are going to solve the two-dimensional Poisson problem, given by

$$\begin{cases} \nabla^2 u(x,y) = -f(x,y), \ (x,y) \in \Omega \\ u(x,y) = 0, \ (x,y) \in \partial\Omega \end{cases} \tag{1}$$

with $f(x,y)$ given as

$$f(x,y) = -8\pi\cos(2\pi(x^2+y^2)) + 16\pi^2(x^2+y^2)\sin(2\pi(x^2+y^2))$$

on the the domain $\Omega$ given by the unit disk, $\Omega = \{(x,y) : x^2+y^2 \leq 1\}$. Then $\partial\Omega = \{(x,y) : x^2+y^2 = 1\}$.

2.1 First, we will verify that

$$u(x,y) = \sin\left(2\pi\left(x^2+y^2\right)\right)$$

is a solution to problem 1. We calculate the partial derivatives,

$$\begin{aligned} \frac{\partial u}{\partial x} &= 4\pi x\cos\left(2\pi\left(x^2+y^2\right)\right) \\ \frac{\partial^2 u}{\partial x^2} &= 4\pi\cos\left(2\pi\left(x^2+y^2\right)\right) - 16\pi^2 x^2\sin\left(2\pi\left(x^2+y^2\right)\right) \\ \frac{\partial u}{\partial y} &= 4\pi y\cos\left(2\pi\left(x^2+y^2\right)\right) \\ \frac{\partial^2 u}{\partial y^2} &= 4\pi\cos\left(2\pi\left(x^2+y^2\right)\right) - 16\pi^2 y^2\sin\left(2\pi\left(x^2+y^2\right)\right). \end{aligned}$$

Ultimately, we then have

$$\nabla^2 u(x,y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 8\pi\cos\left(2\pi\left(x^2+y^2\right)\right) - 16\pi^2\left(x^2+y^2\right)\sin\left(2\pi\left(x^2+y^2\right)\right),$$

and by comparison, we have

$$\nabla^2 u(x,y) = -f(x,y),$$

and we have verified that $u(x,y) = \sin\left(2\pi\left(x^2+y^2\right)\right)$ is a solution to problem 1.

2.2 We will now write up the weak formulation of the problem. Assume $u$ satisfies problem 1. Then, multiplying with an arbitrary test function $v$ and integrating over $\Omega$, we get

$$\iint_\Omega \nabla^2 u v \mathrm{d}\Omega = -\iint_\Omega f v \mathrm{d}\Omega. \tag{2}$$

Greens formula for the divergence operator is

$$\int_\Omega \varphi \mathrm{div}(\mathbf{b})\mathrm{d}\Omega = -\int_\Omega \mathbf{b}\nabla\varphi\mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{b}\cdot\mathbf{n}\varphi\mathrm{d}s.$$

Applying this formula on 2 with $\varphi = v$ and $\mathbf{b} = \nabla u$ yields

$$\iint_\Omega \nabla^2 u v \mathrm{d}\Omega = -\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega + \int_{\partial\Omega} \frac{\partial u}{\partial n} v \mathrm{d}s. \tag{3}$$

From 1, we can see that $u = 0$ on $\partial\Omega$, thus $\frac{\partial u}{\partial n} = 0$ on $\partial\Omega$. Hence,

$$\iint_\Omega \nabla^2 u v \mathrm{d}\Omega = -\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega$$

and thus the problem can be rewritten as

$$\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega = \iint_\Omega f v \mathrm{d}\Omega \iff a(u,v) = l(v)\ \forall v \in X. \tag{4}$$

For these integrals to make sense, we need $\nabla u \nabla v \in L^1(\Omega)$, and for this we need $\nabla u$, $\nabla v \in [L^2(\Omega)]^2$. Including that $u, v$ vanishes at the boundary, this gives us the space

$$X = H_0^1(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}$$
$$H^1(\Omega) = \{v : \Omega \to \mathbb{R} \text{ s.t } v \in L^2(\Omega), \frac{\partial v}{\partial x_i} \in L^2(\Omega), i = 1, 2\}.$$

Bilinearity of $a$ and linearity of $l$ follows trivially from linearity of weak derivatives and integrals.

In conclusion, the weak formulation is

$$\begin{aligned} &\text{find } u \in H_0^1(\Omega): \\ &a(u,v) = l(v)\ \forall v \in H_0^1(\Omega) \\ &a(u,v) = \iint_\Omega \nabla u \nabla v \mathrm{d}\Omega \\ &l(v) = \iint_\Omega f v \mathrm{d}\Omega \end{aligned}$$

2.3 We are now looking for a solution in a smaller space $X_h \subset X$. Let our domain $\Omega$ be discretized into $M$ triangles, such that $\Omega = \cup_{k=1}^M K_k$. Each triangle $K_k$ is then defined by its three corner nodes $(x_i, y_i)$, and there is a basis function corresponding to each node. The space $X_h$ is defined by

$$X_h = \left\{v \in X = H_0^1 : v|_{K_k} m \in \mathbb{P}_1(K_k), 1 \le k \le M\right\}.$$

and the basis functions $\{\varphi_i\}_{i=1}^n$ satisfy

$$X_h = \mathrm{span}\{\phi_i\}_{i=1}^n, \quad \varphi_j(\mathbf{x_i}) = \delta_{ij}.$$

We search to find $u_h \in X_h$, $\forall v \in X_h$. We write $u_h$ and $v_h$ as weighted sums of the basis functions

$$u_h = \sum_{i=1}^n u_h^i \varphi_i(x,y)$$
$$v_h = \sum_{j=1}^n v_h^j \varphi_j(x,y)$$

From the weak formulation, we then get

$$
\begin{aligned}
a(u_h, v_h) &= l(v_h) \\
\iint_\Omega \nabla u_h \nabla v_h \mathrm{d}\Omega &= \iint_\Omega f v_h \mathrm{d}\Omega \\
\iint_\Omega \sum_{i=1}^{n} u_h^i \nabla \varphi_i \sum_{j=1}^{n} v_h^j \nabla \varphi_j \mathrm{d}\Omega &= \iint_\Omega f \sum_{j=1}^{n} v_h^j \varphi_j \mathrm{d}\Omega \\
\sum_{i=1}^{n}\sum_{j=1}^{n} u_h^i v_h^j \iint_\Omega \nabla \varphi_i \nabla \varphi_j \mathrm{d}\Omega &= \sum_{j=1}^{n} v_h^j \iint_\Omega f \varphi_j \\
\sum_{i=1}^{n}\sum_{j=1}^{n} u_h^i v_h^j a(\varphi_i, \varphi_j) &= \sum_{j=1}^{n} v_h^j l(\varphi_j) \qquad (5) \\
\mathbf{v}^T A \mathbf{u} &= \mathbf{v}^T \mathbf{f} \ \forall v \in X_h \\
A\mathbf{u} &= \mathbf{f}
\end{aligned}
$$

with use of the linearity of a() and l(). The Galerkin formulation; Find $u_h \in X_h$ such that $a(u_h, v) = l(v) \forall v \in X_h$, is then equivalent with solving the linear system

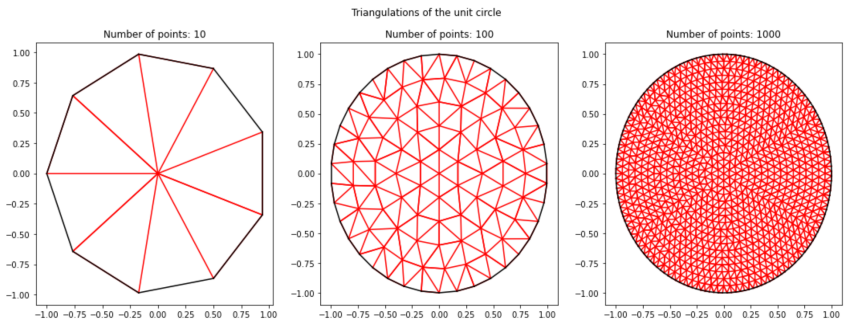
$$A\mathbf{u} = \mathbf{f},$$

with

$$
\begin{aligned}
A &= [A_{ij}] = [a(\varphi_i, \varphi_j)] \\
\mathbf{u} &= [u_h^i] \\
\mathbf{f} &= [f_j] = [l(\varphi_j)]
\end{aligned}
$$

as we can see from equation 5.

**2.4** By using the given code in the script getdisc.py, we generated a mesh on the unit disc. This function takes a number $N$, representing number of nodes in the triangulation. We have plotted the triangulation with $N = 10$, $N = 100$ and $N = 1000$. These meshes can be seen in figure 3.




Figur 3: Meshes of the unit disc, with 10, 100 and 1000 nodes.

2.5 Now we will start to solve the linear system derived in problem 2.4. First we need to build the matrix $A$. This is done in the function generateAandF() in the code. To do this, we follow the pseudocode in figure 4.

**4.4.2 Procedure for $\underline{\bar{A}}_h$**

*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 $\beta = 1,2,3$

$j = \theta(k, \beta)$ ;

$\widetilde{A}_{h\,i\,j} = \widetilde{A}_{h\,i\,j} + A^k_{\alpha\beta}$ ;} } }

Figur 4: Pseudocode for assembly of the matrix A.

This pseudocode is retrieved from lecture note 4. The local to global map $\theta(k, \alpha)$ are given by the *trig* parameter in the function GetDisc(). To calculate the elemental matrices we need the local basis functions

$$\mathcal{H}^k_\alpha = c^k_\alpha + c^k_{x\alpha} x + c^k_{y\alpha} y, \tag{6}$$

where we find the coefficients $c^k_\alpha$, $c^k_{x\alpha}$ and $c^k_{y\alpha}$ from

$$\begin{bmatrix} 1 & x^k_1 & y^k_1 \\ 1 & x^k_2 & y^k_2 \\ 1 & x^k_3 & y^k_3 \end{bmatrix} \begin{bmatrix} c^k_1 \\ c^k_{x_1} \\ c^k_{y_1} \end{bmatrix} or \begin{bmatrix} c^k_2 \\ c^k_{x_2} \\ c^k_{y_2} \end{bmatrix} or \begin{bmatrix} c^k_3 \\ c^k_{x_3} \\ c^k_{y_3} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} or \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} or \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The elemental matrices is then

$$\begin{aligned} (A^k)_{\alpha,\beta} &= \int_{T^k_h} \frac{\partial \mathcal{H}^k_{1,2or3}}{\partial x} \frac{\partial \mathcal{H}^k_{1,2or3}}{\partial x} + \frac{\partial \mathcal{H}^k_{1,2or3}}{\partial y} \frac{\partial \mathcal{H}^k_{1,2or3}}{\partial y} d\Omega \\ &= Area(T^k_h) \cdot (c^k_{x\alpha} c^k_{x\beta} + c^k_{y\alpha} c^k_{y\beta}). \end{aligned}$$

We calculate the coefficients in our code, and solve the integral using the quadrature2D() function. When running our implementation of this code, we get that the matrix $A$ is singular. We verify this by checking the condition number of the matrix, and check that it is "infinite"up to machine precision.

The reason that the matrix is singular, is that we build the matrix $A$ to solve the linear system $A\mathbf{u} = \mathbf{f}$. So far, we have not considered any boundary conditions in $A$, so this linear system does not have a unique solution, and thus the matrix is singular.

2.6 The next step in solving the problem, is to build the right hand side vector $\mathbf{f}$. This is also done in the function generateAandF() in our code. To do this, we follow the pseudocode in figure 5.

**4.4.3 Procedure for $\underline{\bar{F}}_h$**

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

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

Figur 5: Pseudocode for assembly of F

This pseuducode is also retrieved from lecture note 4. As in problem 2.5, the local to global mapping is given by the *trig* parameter from the GenerateDisc() function. The elemental loads are

$$F^k_\alpha = \int_\Omega f \cdot \mathcal{H}^k_{1,2or3},$$

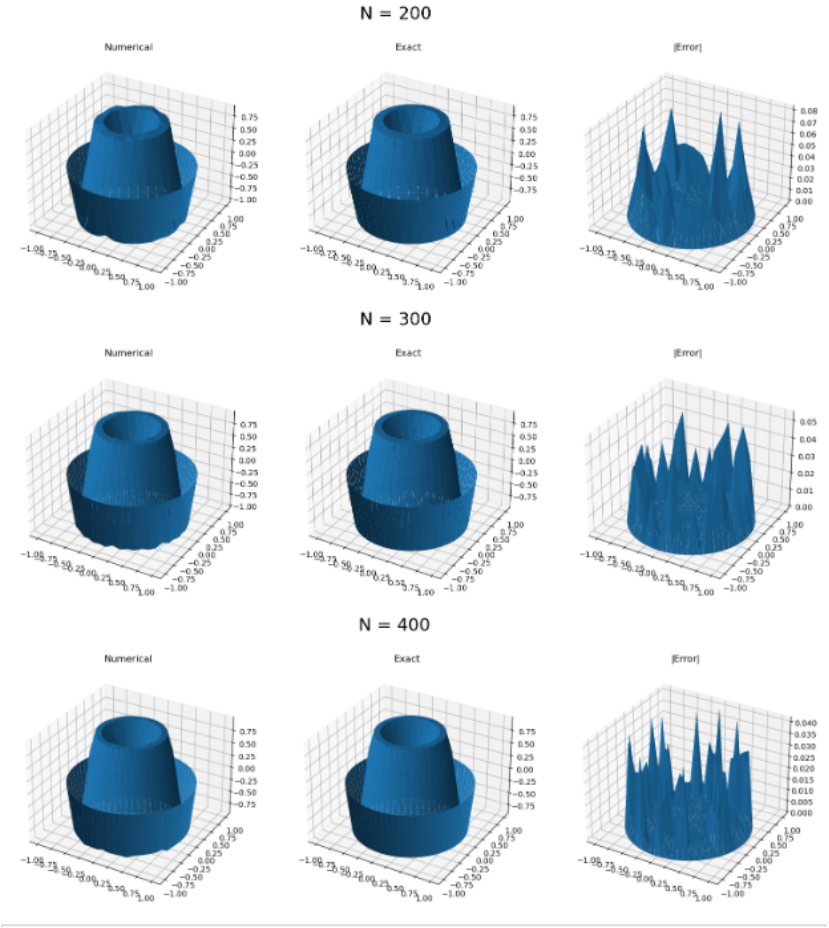
where $\mathcal{H}^k_\alpha$ is calculated as in equation 6 and $f$ is the right hand side of problem 1. This integral is also solved using our implementation of the Gaussian quadrature in the function quadrature2D() in our code.

2.7 Next, we implement the homogeneous Dirichlet boundary conditions on the whole boundary. To do this, we use the big number approximation method. We have $\tilde{n}$ nodes, and $n < \tilde{n}$ interior nodes. For nodes on the boundary $(x_i, y_i)$, $i = n+1, n+2, \ldots, \tilde{n}$, we set:

$$A_{ii} = \frac{1}{\varepsilon},\ (\varepsilon \ll 1)$$
$$\mathbf{f}_i = 0.$$

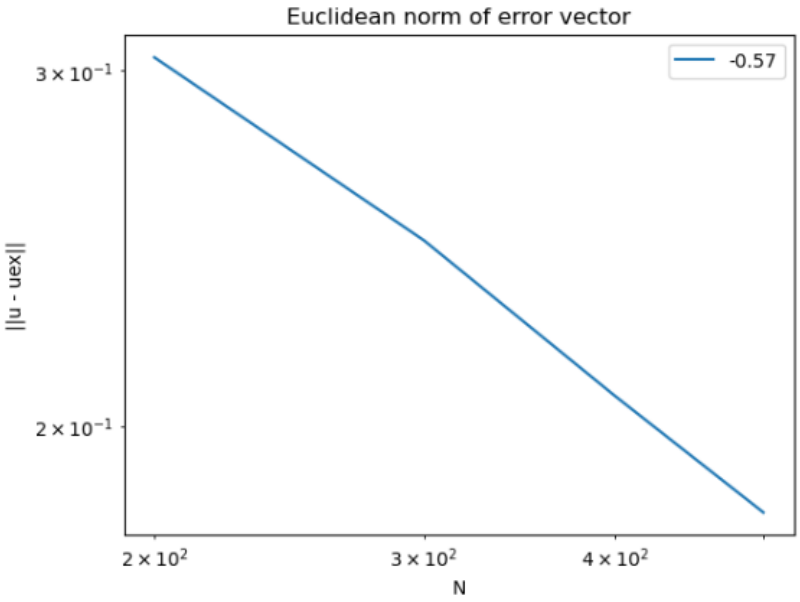
In this way, we get $u^{n+1}_h \cong u^{n+2}_h \cong \ldots \cong u^{\tilde{n}}_h \cong 0$ in a nice and symmetric way.

2.8 After we have implemented all this, we can finally solve the linear system. In figure 6 we have plotted the finite element solution for different meshes and the exact solution together with the difference between them.




Figur 6: Numerical solution for 200, 300 and 400 nodes in the mesh, plotted alongside the exact solution and the error.

Just from inspection, we can see that the numerical solution approaches the exact solution for finer mesh. In figure 7 we have plotted the norm of the error vector, against number of nodes in the mesh. From this we can see that the error decreases for increasing number of nodes.




Figur 7: Plot of euclidean norm of error against number of nodes in mesh.

## Neumann Boundary Conditions

Now we introduce Neumann boundary conditions on some parts of the boundary. The problem then changes to

$$\begin{cases} \nabla^2 u(x,y) = -f(x,y) \\ u(x,y)|_{\partial\Omega_D} = 0 \\ \frac{\partial u(x,y)}{\partial n}|_{\Omega_N} = g(x,y) \end{cases} \tag{7}$$

with the exact solution $u$ as given in problem 2.1, the source term $f$ stated in equation 1 and $g$ as

$$g(x,y) = 4\pi\sqrt{x^2+y^2}\cos\left(2\pi\left(x^2+y^2\right)\right). \tag{8}$$

The Dirichlet boundary conditions is defined on $\partial\Omega_D = \left\{x^2+y^2 = 1, y < 0\right\}$ and the Neumann boundary conditions on $\partial\Omega_N = \left\{x^2+y^2 = 1, y > 0\right\}$.

3.1 We will now verify that the analytical solution $u$ in problem 2.1 satisfies equation 8 on the boundary $\partial\Omega_N$. The derivative with respect to the normal is defined as

$$\begin{aligned} \frac{\partial u(x,y)}{\partial n}\Big|_{\partial\Omega_N} &= n \cdot \nabla u(x,y) \\ &= \frac{[x,y]^T}{\sqrt{x^2+y^2}} \cdot \left[\frac{\partial u(x,y)}{\partial x}, \frac{\partial u(x,y)}{\partial y}\right]^T \end{aligned}$$

where $\frac{[x,y]^T}{\sqrt{x^2+y^2}}$ is the surface normal to the unit disc. We calculate the partial derivatives of $u$ with regard to $x$ and $y$

$$\frac{\partial u(x,y)}{\partial x} = 4\pi x \cos(2\pi(x^2+y^2))$$

$$\frac{\partial u(x,y)}{\partial y} = 4\pi y \cos(2\pi(x^2+y^2)).$$

Combining these with the above result yields

$$\begin{aligned}
\frac{[x,y]^T}{\sqrt{x^2+y^2}} \cdot \left[\frac{\partial u(x,y)}{\partial x}, \frac{\partial u(x,y)}{\partial y}\right]^T &= \frac{[x,y]^T}{\sqrt{x^2+y^2}} \cdot [4\pi x \cos(2\pi(x^2+y^2)), 4\pi y \cos(2\pi(x^2+y^2))]^T \\
&= \frac{x}{\sqrt{x^2+y^2}} 4\pi x \cos(2\pi(x^2+y^2)) + \frac{y}{\sqrt{x^2+y^2}} 4\pi y \cos(2\pi(x^2+y^2)) \\
&= 4\pi \left(\frac{x^2}{\sqrt{x^2+y^2}} + \frac{y^2}{\sqrt{x^2+y^2}}\right) \cos(2\pi(x^2+y^2)) \\
&= 4\pi\sqrt{x^2+y^2} \cos(2\pi(x^2+y^2))
\end{aligned}$$

Thus,

$$\frac{\partial u(x,y)}{\partial n}\Big|_{\partial\Omega_N} = 4\pi\sqrt{x^2+y^2} \cos(2\pi(x^2+y^2)) = g(x,y)$$

and the analytic solution presented in task 2.1 satisfies equation 8 at the boundary.

3.2 With the introduction of Neumann boundary condition, we need a change in the weak formulation of the problem. From our derivation of the weak formulation, we have equation 3:

$$\iint_\Omega \nabla^2 u v \mathrm{d}\Omega = -\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega + \int_{\partial\Omega} \frac{\partial u}{\partial n} v \mathrm{d}s.$$

When we only considered homogeneous Dirichlet boundary conditions, $\frac{\partial u}{\partial n} = 0$ on the whole boundary. This is no longer the case, as $\frac{\partial u}{\partial n} = g$ on $\partial\Omega_n$. Hence, we get the integral equation

$$\begin{aligned}
\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega - \int_{\partial\Omega_N} \frac{\partial u}{\partial n} v \mathrm{d}s &= \iint_\Omega f v \mathrm{d}\Omega \\
\iint_\Omega \nabla u \nabla v \mathrm{d}\Omega &= \iint_\Omega f v \mathrm{d}\Omega + \int_{\partial\Omega_N} g v \mathrm{d}s.
\end{aligned}$$

In addition, with the introduction of Neumann boundary conditions, we have a new space of test functions, namely

$$H^1_{\Omega_D} = \{v \in H^1(\Omega) : v|_{\partial\Omega_D} = 0\}.$$

So, for Neumann boundary conditions, the weak formulation changes to

$$\begin{aligned}
&\text{find } u \in H^1_{\Omega_D}(\Omega) : \\
&a(u,v) = l(v)\ \forall v \in H^1_{\Omega_D}(\Omega) \\
&a(u,v) = \iint_\Omega \nabla u \nabla v \mathrm{d}\Omega \\
&l(v) = \iint_\Omega f v \mathrm{d}\Omega + \int_{\partial\Omega_N} g v \mathrm{d}s.
\end{aligned}$$

In conclusion, both $a$ and $l$ changes as they now operate on a new function space. In addition, the form of $l$ changes, as seen above.

3.3 To be able to handle this new Neumann boundary term in our weak formulation, we changed the function quadrature1D($a$, $b$, $N_q$, g) from problem 1.1 to be able to handle inputs where $a, b \in \mathbb{R}^2$. The reason for this, is that we need this function to be able to calculate the line integral in the new $l$.

We tested the modified function on the integral

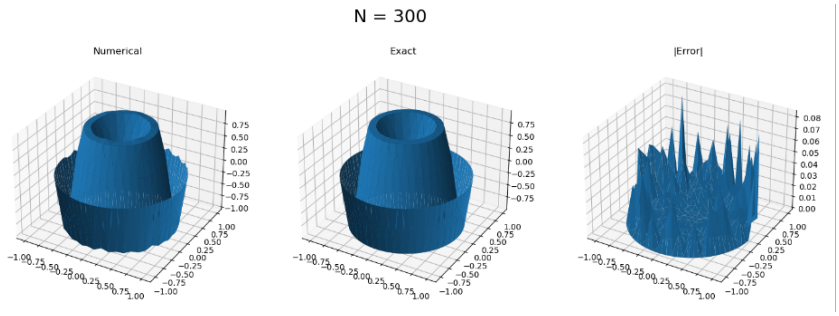
$$\int_a^b e^x \mathrm{d}s = e^2 - e \approx 4.67077,$$

where $a = (1, 0)$ and $b = (2, 0)$. The comparison between this value, and the value computed by our modified function can be seen in figure 8.

| Nq | Value | Difference from exact |
|---|---|---|
| 1 | 2.71828 | 1.95249 |
| 2 | 4.66973 | 0.00104776 |
| 3 | 4.67077 | 2.2401e-06 |
| 4 | 4.67077 | 2.53607e-09 |

Figur 8: Comparison between our implementation of 1D Gaussian quadrature for line integrals and the exact solution of $\int_a^b e^x ds$. $N_q$ denotes number of integration points, value denotes the evaluation of the integral with $N_q$ integration points and the last column denotes the difference between approximated and exact solution.

3.4 To solve the problem with mixed boundary conditions, we used the code from problem 2 and added code to implement the Neumann boundary. The solution with mixed boundary condition can be seen in figure 9.




Figur 9: Numerical solution with 300 nodes of the problem 7, the exact solution and the error.

From the error plot, we can see that the error is bigger on the Neumann boundary, than on the Dirichlet boundary. By comparing this figure with figure 6, we can again see that the error is bigger on the Neumann boundary than on the Dirichlet boundary. The error in the interior seems to be quite equal for both problems.