Homework 1

BMEGEMMNWCM, Continuum Mechanics

Problem description: The reference and spatial configurations of a 4-node linear plane strain quad element are known according to a given figure (see your particular figure at the link provided below). The grid size in the figure is 1 mm in both directions. 1-point Gaussian quadrature is used in the element formulation. The material particle located at the Gauss integration point is labelled with **G**. The nodes are located at intersections of the gridlines, thus the coordinates (reference and spatial) of the nodes are integer values.

Tasks: Determine the following quantities for material particle **G**:

1. Position of the reference and the spatial configurations.
2. The displacement vector.
3. Deformation gradient.
4. Green–Lagrange strain tensor; Euler–Almansi strain tensor.
5. Volume ratio, volume strain.
6. Isochoric and volumetric parts of the deformation gradient.
7. Principal stretches.
8. Engineering strain in the direction designated by the straight line connecting node 2 and node 4 in the reference configuration.
9. Consider the infinitesimal surface element vector $\mathrm{d}\boldsymbol{A} = \mathrm{d}A\boldsymbol{E}_3$ at **G**. It deforms to $\mathrm{d}\boldsymbol{a} = \mathrm{d}a\boldsymbol{e}_3$ in the spatial configuration, where $\boldsymbol{E}_3$ and $\boldsymbol{e}_3$ are the basis vectors perpendicular to the plane. Determine the ratio $\mathrm{d}a/\mathrm{d}A$.
10. Angle of shear for material line elements with initial orientations $\boldsymbol{N}_1 = \boldsymbol{E}_1$ and $\boldsymbol{N}_2 = \boldsymbol{E}_2$.
11. Amount of rigid body rotation associated with material particle at **G**.
12. Components of the spatial Hencky's strain tensor.

Your figure can be downloaded from the following link:

`http://www.mm.bme.hu/~kossa/NEPTUN-HW1.pdf`

Where the string `NEPTUN` has to be replaced with the Student's NEPTUN code. The link above is case-sensitive! Use uppercase letters for the file-name! Students must solve the problem correspondig to their NEPTUN's code!

Budapest University of Technology and Economics
Faculty of Mechanical Engineering

Continuum Mechanics
I. Homework

Gergő Kelle

October 17, 2023

Contents

1 Position of the reference and the spatial configurations

Key informations: It is a 4-node linear plane strain quad element. 1-point Gaussian quadrature is used in the element formulation. The material particle located at the Gauss integration point is labelled with G.

As the Homework instruction page suggest I downloaded the figure about the problem with my NEPTUN code, which can be seen at fig. 1.

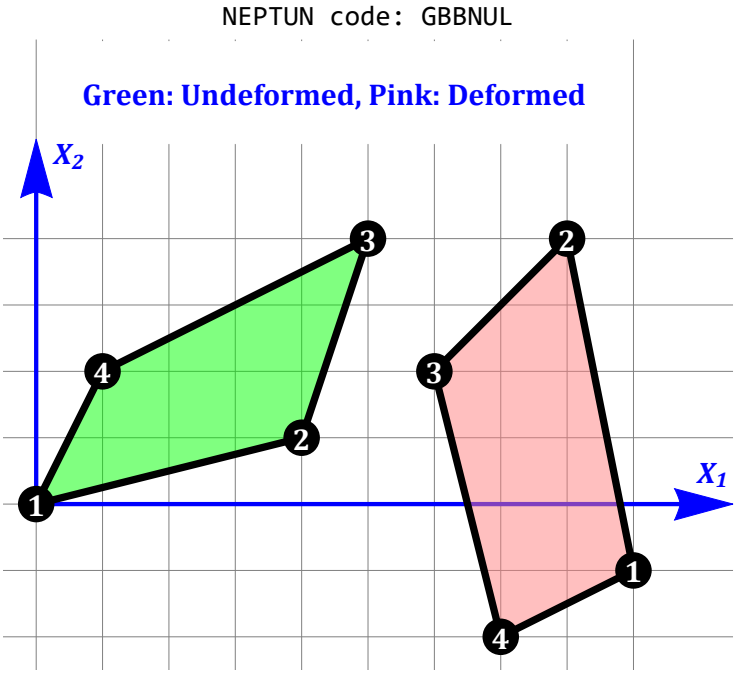


Figure 1: Your Figure Caption

Since the node points and the corresponding vectors will be neccesarry I've listed them in Tab. 1.

Table 1: Table of position vectors

Node	Reference	Current
1	$\mathbf{X}_{0,1}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$	$\mathbf{x}_{0,1}^T = \begin{bmatrix} 9 & -1 & 0 \end{bmatrix}$
2	$\mathbf{X}_{0,2}^T = \begin{bmatrix} 4 & 1 & 0 \end{bmatrix}$	$\mathbf{x}_{0,2}^T = \begin{bmatrix} 8 & 4 & 0 \end{bmatrix}$
3	$\mathbf{X}_{0,3}^T = \begin{bmatrix} 5 & 4 & 0 \end{bmatrix}$	$\mathbf{x}_{0,3}^T = \begin{bmatrix} 6 & 2 & 0 \end{bmatrix}$
4	$\mathbf{X}_{0,4}^T = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}$	$\mathbf{x}_{0,4}^T = \begin{bmatrix} 7 & -2 & 0 \end{bmatrix}$

The undeformed (reference) and the deformed (current) position vectors are compiled according to the Tab. 1 in Eq. 1 and 1.

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 \\ 4 & 1 & 0 \\ 5 & 4 & 0 \\ 1 & 2 & 0 \end{bmatrix} \text{mm} \qquad (1) \qquad \mathbf{x} = \begin{bmatrix} 9 & -1 & 0 \\ 8 & 4 & 0 \\ 6 & 2 & 0 \\ 7 & -2 & 0 \end{bmatrix} \text{mm} \qquad (2)$$

Before any further calculations, the position of the Gauss point ($\mathbf{G}$) needs to be determined. Since $\mathbf{G}$ is positioned at the quad element center the average of every corresponding node coordinate value will give us the solution.

$$G_x = \frac{1}{k}\sum_{i=1}^{k} \mathbf{X}_{i,1} \qquad (3)$$

$$G_y = \frac{1}{k}\sum_{i=1}^{k} \mathbf{X}_{i,2} \qquad (4)$$

$$G_z = \frac{1}{k}\sum_{i=1}^{k} \mathbf{X}_{i,3} \qquad (5)$$

$$\mathbf{X}_G^T = \begin{bmatrix} 2.5 & 1.75 & 0 \end{bmatrix} \qquad (6)$$

$$\mathbf{x}_G^T = \begin{bmatrix} 7.5 & 0.75 & 0 \end{bmatrix} \qquad (7)$$

```python
def calculate_gauss_point(nodes):
  num_nodes = len(nodes)
  gx, gy, gz = 0, 0, 0
  for x, y, z in nodes:
      gx += x
      gy += y
      gz += z
  gx /= num_nodes
  gy /= num_nodes
  gz /= num_nodes
  return gx, gy, gz

undeformed_gauss_point =
    calculate_gauss_point(
    undeformed_nodes)
deformed_gauss_point =
    calculate_gauss_point(deformed_nodes
    )
```

i: For later calculations the whole code is available as appendix.

2 Displacement vector

The displacement vector of $\mathbf{G}$ can be determined as the difference of reference and current position

$$\mathbf{U}_G^T = \mathbf{x}_G^T - \mathbf{X}_G^T = \begin{bmatrix} 5.0 & -1.0 & 0 \end{bmatrix}. \qquad (8)$$

Displacement can be determined for every node as

$$\mathbf{U}_{nodes} = \mathbf{x} - \mathbf{X} = \begin{bmatrix} 9 & -1 & 0 \\ 4 & 3 & 0 \\ 1 & -2 & 0 \\ 6 & -4 & 0 \end{bmatrix} \qquad (9)$$

.

3 Deformation gradient

Deformation gradiend most commonly calculated with one of the following methods:

$$\mathbf{F}(\mathbf{X}) = \text{Grad}\boldsymbol{\chi}(\mathbf{X}, t) = \frac{\partial \boldsymbol{\chi}(\mathbf{X}, t)}{\partial \mathbf{X}} \quad \text{or} \quad \mathbf{F}(\mathbf{X}) = \text{Grad}\mathbf{U}(\mathbf{X}, t) + \mathbf{I} \qquad (10)$$

Where	Deformation gradient	:	$\mathbf{F}(\mathbf{X},t)$
	Motion	:	$\boldsymbol{\chi}(\mathbf{X},t)$
	Material displacement field	:	$\mathbf{U}(\mathbf{X},t)$
	Material displacement gradient	:	$\mathbf{K}(\mathbf{X},t) = \text{Grad}\mathbf{U}(\mathbf{X},t)$

The determination of the motion function from the provided data for the computation of the deformation gradient remains unattainable. Therefore calculating the material displacement field is the better solution. Utilizing the shape functions as Equation 11 shows, the displacement field can be computed using the nodal displacements.

$$N_1(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta) \tag{11}$$

$$N_2(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta) \tag{12}$$

$$N_3(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta) \tag{13}$$

$$N_4(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta) \tag{14}$$

> *i: To elucidate the notational conventions, as illustrated in Figure 1, we denote X_1 as the nonzero coordinate of the vector $\boldsymbol{E}_1$, and X_2 as the nonzero coordinate of $\boldsymbol{E}_2$ (analogously, X_3).*

$$X_1(\xi,\eta) = \sum_{i=1}^{k} \mathbf{X}_{i1} \cdot N_i(\xi,\eta) \tag{15}$$

$$X_2(\xi,\eta) = \sum_{i=1}^{k} \mathbf{X}_{i2} \cdot N_i(\xi,\eta) \tag{16}$$

To obtain the displacement field components, we must transform the ξ and η coordinates with respect to X_1 and X_2. This involves inverting the previously mentioned expressions. The inversion yields two separate solutions, and we must choose the suitable one. The resultant solutions are as follows:

Table 2: Inverse function solutions

	$\xi(X_1,X_2)$	$\eta(X_1,X_2)$
solution 1	$\frac{1}{4}\left(-11+X_1-\sqrt{49+18X_1+X_1^2-16X_2}\right)$	$6+X_1+\sqrt{49+18X_1+X_1^2-16X_2}$
solution 2	$\frac{1}{4}\left(-11+X_1+\sqrt{49+18X_1+X_1^2-16X_2}\right)$	$6+X_1-\sqrt{49+18X_1+X_1^2-16X_2}$

The selected solution should satisfy the condition where $\xi = 0$ and $\eta = 0$ for the Gauss point's coordinates (X_G) in the local coordinate system.

Table 3: Solutions given value at $\xi = 0$ and $\eta = 0$

	$\xi(X_1, X_2)$	$\eta(X_1, X_2)$
solution 1	-4.25	17
solution 2	0	0

Thus the second

$$\xi(X_1, X_2) = \frac{1}{4}\left(-11 + X_1 + \sqrt{49 + 18X_1 + X_1^2 - 16X_2}\right), \tag{17}$$

$$\eta(X_1, X_2) = 6 + X_1 - \sqrt{49 + 18X_1 + X_1^2 - 16X_2} \tag{18}$$

solutions shall be used. Using these functions the material displacement field can be determined using the displacement of the nodes ($\mathbf{U}_{nodes}$)

$$\mathbf{U}(X_1, X_2, X_3) = \begin{bmatrix} \sum_{i=1}^{4} U_{nodes,i1} \cdot N_i(X_1, X_2) \\ \sum_{i=1}^{4} U_{nodes,i2} \cdot N_i(X_1, X_2) \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{8}\left(23 - 17X_1 + 7\sqrt{49 + 18X_1 + X_1^2 - 16X_2}\right) \\ \frac{1}{4}\left(-11 + 5X_1 + \sqrt{49 + 18X_1 + X_1^2 - 16X_2} - 8X_2\right) \\ 0 \end{bmatrix} \tag{19}$$

$$\mathbf{U}(\mathbf{X}_G)^T = \begin{bmatrix} 5 & -1 & 0 \end{bmatrix} = \mathbf{x}_G^T - \mathbf{X}_G^T \quad \checkmark \tag{20}$$

From the material displacement field the deformation gradient can be calculated as

$$\mathbf{F}(\mathbf{X}_G) = \mathrm{Grad}\mathbf{U}(\mathbf{X}_G) + \mathbf{I} = \begin{bmatrix} 0.0588 & -0.8235 & 0 \\ 1.588 & -1.235 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

4 Green-Lagrange and Euler-Almansi strain tensor

In order to calculate Green-Lagrange strain tensor I've calculated Cauchy-Green deformation tensor beforehand. Since our task is to calculate every parameter for the Gauss point thus the numerical solutions are represented the solution at $\mathbf{G}$.

$$\mathbf{C}(\mathbf{X}) = \mathbf{F}^T\mathbf{F} \tag{22}$$

$$\mathbf{E}(\mathbf{X}) = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right) = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) \tag{23}$$

Where
- Cauchy-Green deformation tensor: $\mathbf{C}(\mathbf{X})$
- Green-Lagrange strain tensor : $\mathbf{E}(\mathbf{X})$
- Second-order identity tensor : $\mathbf{I}$

The resulting solutions are

$$\mathbf{C}(\mathbf{X}_G) = \begin{bmatrix} 2.5252 & -2.0096 & 0 \\ -2.0096 & 2.2034 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{24}$$

$$\mathbf{E}(\mathbf{X}_G) = \begin{bmatrix} 0.7626 & -1.0048 & 0 \\ -1.0048 & 0.6017 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{25}$$

$$\mathbf{c} = \mathbf{F}^{-T}\ \mathbf{F}^{-1} \tag{26}$$

$$\mathbf{e}(\mathbf{X}) = \frac{1}{2}\left(\mathbf{I} - \mathbf{c}\right) = \frac{1}{2}\left(\mathbf{I} - \mathbf{F}\ \mathbf{F}^{T}\right) \tag{27}$$

Where

Cauchy deformation tensor :	$\mathbf{c}(\mathbf{X})$
Euler-Almansi strain tensor:	$\mathbf{e}(\mathbf{X})$

The resulting solutions are

$$\mathbf{c}(\mathbf{X}_G) = \begin{bmatrix} 2.6530 & -0.72795 & 0 \\ -0.72795 & 0.4468 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{28}$$

$$\mathbf{e}(\mathbf{X}_G) = \begin{bmatrix} -0.8265 & 0.3640 & 0 \\ 0.3640 & 0.2766 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{29}$$

5 Volume element dv/dV ratio and volume strain

Volume ratio and volume strain are both based on change of a volume element size between the reference and current configuration. Those parameters can be expressed as

$$J = \frac{\mathrm{d}v}{\mathrm{d}V} = \det \mathbf{F}(\mathbf{X}_G) = 1.2351 \tag{30}$$

$$\varepsilon_V = \frac{\mathrm{d}v - \mathrm{d}V}{\mathrm{d}V} = J - 1 = 0.2351. \tag{31}$$

Where

Volume ratio	:	J
Inf. small volume element (reference):		$\mathrm{d}V$
Inf. small volume element (current)	:	$\mathrm{d}v$
Volume strain	:	ε_V

6 Isochoric and volumetric parts of the deformation gradient

The isochoric (volume-preserving) deformation stands for a deformation where the reference and current configuration element does not change in volume ($J = 1$). Thus the deformation gradient ($\mathbf{F}$) can be multiplicatively decomposed into an isochoric and volumetric parts as

$$\mathbf{F} = \mathbf{F}_{iso}\ \mathbf{F}_{vol} = \mathbf{F}_{vol}\ \mathbf{F}_{iso} \tag{32}$$

TheIsochoric and volumetric parts can be calculated as

$$\mathbf{F}_{iso}(\mathbf{X}_G) = J^{-\frac{1}{3}}\mathbf{F}(\mathbf{X}_G) = \begin{bmatrix} 0.0548 & -0.7675 & 0 \\ 1.4800 & -1.1511 & 0 \\ 0 & 0 & 0.9320 \end{bmatrix} \tag{33}$$

$$\mathbf{F}_{vol}(\mathbf{X}_G) = J^{\frac{1}{3}}\mathbf{I} = \begin{bmatrix} 1.0729 & 0 & 0 \\ 0 & 1.0729 & 0 \\ 0 & 0 & 1.0729 \end{bmatrix} \tag{34}$$

$$\tag{35}$$

To make sure the calculations were right we can check the following equilibrium

$$\mathbf{F}_{iso}\ \mathbf{F}_{vol} - \mathbf{F} = \mathbf{0} \tag{36}$$

Since our result satisfies the equilibrium above, the result is acceptable.

7 Principal stretches

Principal stretches can be calculated from the Cauchy-Green deformation tensor determined ar Eq. 24. The resulting eigenvalues can be found in Table 4.

Table 4: Table of position vectors

Notation	Eigenvalue
$\lambda_1 = \sqrt{\mu_1}$	2.0929
$\lambda_2 = \sqrt{\mu_2}$	1
$\lambda_3 = \sqrt{\mu_3}$	0.5901

8 Engineering strain of a line element

To determine the engineering strain on the selected line element we have to calculate the length of the elemnt at the reference and the current state. The comparison of theese two length value will give us the engineering strain due to its deformation.

$$\varepsilon_e = \frac{||\mathbf{x}_{0,2} - \mathbf{x}_{0,4}||}{||\mathbf{X}_{0,2} - \mathbf{X}_{0,4}||} - 1 = 0.923538 \tag{37}$$

9 Surface element da/dA ratio

The surface element $\frac{\mathrm{d}a}{\mathrm{d}A}$ ratio can be determined similarly as the volume ratio. In order to determine the surface element ration we need a vector thats perpendicular to it. Since the body is spread over two dimensions, the unit vector perpendicular to it is the unit vector of $\mathbf{E}_3$ or $\mathbf{e}_3$ direction.

$$\mathbf{N}_1^T = \mathbf{E}_1^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{38}$$
$$\mathbf{N}_2^T = \mathbf{E}_2^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{39}$$
$$\mathbf{N}_3^T = \mathbf{E}_3^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$\frac{\mathrm{d}a}{\mathrm{d}A} = \det \mathbf{F} \sqrt{\mathbf{N}_3 \ \mathbf{F}^{-1} \ \mathbf{F}^{-T} \ \mathbf{N}_3} = 1.2351 \tag{41}$$

The surface element ratio between the reference and the current configuration at $\mathbf{G}$ point is 1.2351.

10 Angle of shear material line elements

This task is about to determine the angle of shear in case of the material line elements parallel to the unit vectors of $\mathbf{E}_1$ (or $\mathbf{N}_1$) and $\mathbf{E}_2$ (or $\mathbf{N}_2$). To be able to calculate the angle, the stretch ratios associated with these orientations needs to be determined with the following equations

$$\lambda_{N_1} = \sqrt{\mathbf{N}_1 \ \mathbf{C} \ \mathbf{N}_1} = 1.5891 \tag{42}$$
$$\lambda_{N_2} = \sqrt{\mathbf{N}_2 \ \mathbf{C} \ \mathbf{N}_2} = 1.4844 \tag{43}$$

With the help of the stretch ratios the angle of shear can be determined as

$$\gamma = \frac{\pi}{2} - \arccos\left(\frac{\mathbf{N}_1 \ \mathbf{C} \ \mathbf{N}_2}{\lambda_{N_1}\lambda_{N_2}}\right) = -1.0197 \text{ rad} \quad \rightarrow \quad \gamma = -58.4259^\circ \tag{44}$$

11 Rigid body rotation

The rigid body rotation tensor can be calculated with the following equations

$$\mathbf{U} = \sqrt{\mathbf{C}}, \tag{45}$$
$$\mathbf{R} = \mathbf{F} \ \mathbf{U}^{-1} = \begin{bmatrix} -0.4384 & -0.8988 & 0 \\ 0.8988 & -0.4384 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{46}$$

To get the actual rotation angle and the deriction we can use the rotation tensor first scalar invariant value the following way:

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) = 2.0246 \text{ rad} \quad \rightarrow \quad \theta = 116^\circ \tag{47}$$
$$\mathbf{N} = \frac{\mathbf{R} - \mathbf{R}^\top}{2\sin\theta} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{48}$$

12 Hencky's strain tensor

The Hencky's strain tensor can be calculated with the help of the left stretch tensor $\mathbf{V}$ which is easy to determine with the help of deformation gradient and rotation tensor

$$\mathbf{V} = \mathbf{F}\ \mathbf{R}^T = \begin{bmatrix} 0.7144 & 0.4139 & 0 \\ 0.4139 & 1.9687 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{49}$$

Hencky's strain tensor is the natural logarithm of the left stretch tensor $\mathbf{V}$

$$\mathbf{h} = \ln\mathbf{V} = \begin{bmatrix} -0.4227 & 0.3487 & 0 \\ 0.3487 & 0.6339 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{50}$$

CM_I_HW

October 17, 2023

1 Cont mech I. HW

Name: *Gergő KELLE*

NEPTUN Code: *GBBNUL*

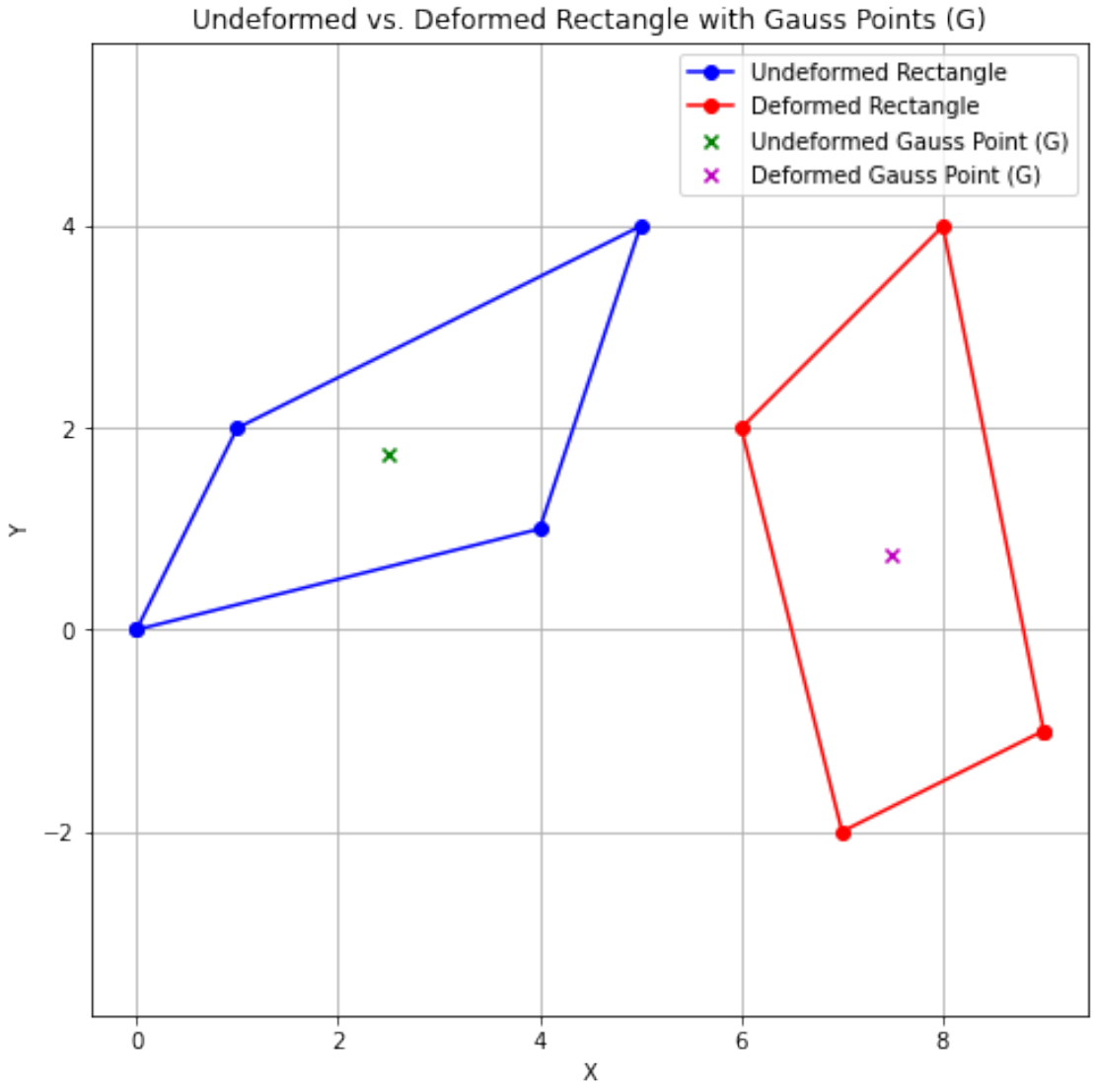
1.1 TOC:

1.2 1. Position of the reference and the spatial configurations

[38]:
```
import matplotlib.pyplot as plt
import numpy as np
import sympy as sp

# Coordinates of the undeformed rectangle
undeformed_rectangle = {
    'Point1': (0, 0, 0),
    'Point2': (4, 1, 0),
    'Point3': (5, 4, 0),
    'Point4': (1, 2, 0)
}

# Coordinates of the deformed rectangle
deformed_rectangle = {
    'Point1': (9, -1, 0),
    'Point2': (8, 4, 0),
```

```
    'Point3': (6, 2, 0),
    'Point4': (7, -2, 0)
}

# Extract coordinates to create vectors
undeformed_nodes = [undeformed_rectangle[point] for point in␣
 ↪sorted(undeformed_rectangle.keys())]
deformed_nodes = [deformed_rectangle[point] for point in␣
 ↪sorted(deformed_rectangle.keys())]

# Calculate the Gauss point for undeformed and deformed elements
def calculate_gauss_point(nodes):
    num_nodes = len(nodes)
    gx, gy, gz = 0, 0, 0
    for x, y, z in nodes:
        gx += x
        gy += y
        gz += z
    gx /= num_nodes
    gy /= num_nodes
    gz /= num_nodes
    return gx, gy, gz

undeformed_gauss_point = calculate_gauss_point(undeformed_nodes)
deformed_gauss_point = calculate_gauss_point(deformed_nodes)

# Extract x and y coordinates for plotting
undeformed_x, undeformed_y, _ = zip(*undeformed_nodes)
deformed_x, deformed_y, _ = zip(*deformed_nodes)

# Plot the undeformed and deformed rectangles
plt.figure(figsize=(8, 8))
plt.plot(undeformed_x + (undeformed_x[0],), undeformed_y + (undeformed_y[0],),␣
 ↪marker='o', label='Undeformed Rectangle', linestyle='-', color='b')
plt.plot(deformed_x + (deformed_x[0],), deformed_y + (deformed_y[0],),␣
 ↪marker='o', label='Deformed Rectangle', linestyle='-', color='r')
plt.scatter(undeformed_gauss_point[0], undeformed_gauss_point[1], marker='x',␣
 ↪color='g', label='Undeformed Gauss Point (G)')
plt.scatter(deformed_gauss_point[0], deformed_gauss_point[1], marker='x',␣
 ↪color='m', label='Deformed Gauss Point (G)')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Undeformed vs. Deformed Rectangle with Gauss Points (G)')
plt.legend()
plt.grid(True)
plt.axhline
```

```
plt.axvline
plt.axis('equal')
plt.show()

# Print the Gauss points
print("Undeformed Gauss Point (G_undef):", undeformed_gauss_point)
print("Deformed Gauss Point (G_def):", deformed_gauss_point)
```




```
Undeformed Gauss Point (G_undef): (2.5, 1.75, 0.0)
Deformed Gauss Point (G_def): (7.5, 0.75, 0.0)
```

1.3 2. Displacement vector

[39]:
```
# Calculate the displacement vector
displacement_vector = (
    deformed_gauss_point[0] - undeformed_gauss_point[0],
    deformed_gauss_point[1] - undeformed_gauss_point[1],
    deformed_gauss_point[2] - undeformed_gauss_point[2]
)
print("displacement vector of G point (U_G)",'\n', displacement_vector)
```

```
displacement vector of G point (U_G)
 (5.0, -1.0, 0.0)
```

[40]:
```
# Extract coordinates to create vectors
undeformed_nodes = np.array([undeformed_rectangle[point] for point in␣
 ↪sorted(undeformed_rectangle.keys())])
deformed_nodes = np.array([deformed_rectangle[point] for point in␣
 ↪sorted(deformed_rectangle.keys())])

# Calculate the displacement matrix
displacement_matrix = deformed_nodes - undeformed_nodes

# Print the displacement matrix
print("Displacement Matrix (4x3):")
print(displacement_matrix)
```

```
Displacement Matrix (4x3):
[[ 9 -1  0]
 [ 4  3  0]
 [ 1 -2  0]
 [ 6 -4  0]]
```

1.4 3. Deformation gradient

Due to python's poor symbolic tools or perhabs my level of knowledge about this programming language I've calculated the Deformation gradient in Wolfram Mathematica which is a better, symbolic solvers.

The code was the following:

ClearAll; X0 = {{0, 0}, {4, 1}, {5, 4}, {1, 2}}; U = {{9, -1}, {4, 3}, {1, -2}, {6, -4}}; Xt = X0 + U

parameters = {[Xi] -> 0, [Eta] -> 0};

NN = 1/4 {(1 - [Xi]) (1 - [Eta]), (1 + [Xi]) (1 - [Eta]), (1 + [Xi]) (1 + [Eta]), (1 - [Xi]) (1 + [Eta])};

XS0 = {Total[(X0[[All, 1]]*NN)], Total[(X0[[All, 2]]*NN)]}

XS1 = {Total[(Xt[[All, 1]]*NN)], Total[(Xt[[All, 2]]*NN)]}

N[XS0 /. {[Xi] -> 0, [Eta] -> 0}]

N[XS1 /. {[Xi] -> 0, [Eta] -> 0}]

ux = Expand[U[[All, 1]] . NN]

uy = Expand[U[[All, 2]] . NN]

Expand[4 ux]

Expand[4 uy]

x = Simplify[X0[[All, 1]] . NN]

y = Simplify[X0[[All, 2]] . NN]

Expand[4 x]

Expand[4 y]

sol = FullSimplify[Solve[{x == XX, y == YY}, {[Xi], [Eta]}]]

N[sol /. {XX -> XS0[[1]], YY -> XS0[[2]]}];

UXY = FullSimplify[{ux, uy} /. (sol[[2]])]

KK = FullSimplify[Grad[UXY, {XX, YY}]];

KK = {{KK[[1, 1]], KK[[1, 2]], 0}, {KK[[2, 1]], KK[[2, 2]], 0}, {0, 0, 0}};

MatrixForm[KK]

MatrixForm[Expand[KK]]

F = Simplify[KK + IdentityMatrix[3]]

F // MatrixForm

{XX, YY} = N[XS0 /. {[Xi] -> 0, [Eta] -> 0}];

F // MatrixForm

The result of the Wolfram Mathematica code presented above:

$$\mathbf{F} = \begin{bmatrix} 0.0588235 & -0.823529 & 0 \\ 1.58824 & -1.23529 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[41]:
```
F_X_G = sp.Matrix([[0.05882, -0.8235, 0], [1.588, -1.235, 0], [0, 0, 1]])
```

1.5 4. Green-Lagrange and Euler-Almansi strain tensor

[42]:
```
# Cauchy-Green deformation tensor
C_G = F_X_G.transpose()*F_X_G
print(C_G)
```

```
Matrix([[2.52520379240000, -2.00961827000000, 0], [-2.00961827000000,
2.20337725000000, 0], [0, 0, 1]])
```

[43]:
```
# Green-Lagrange strain tensor
GL_strain = 1/2*(C_G - sp.eye(3))
GL_strain
```

[43]:
$$\begin{bmatrix} 0.7626018962 & -1.004809135 & 0 \\ -1.004809135 & 0.601688625 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

[44]:
```
# Cauchy deformation tensor
cauchy_d = F_X_G.transpose().inv()* F_X_G.inv()
cauchy_d
```

[44]:
$$\begin{bmatrix} 2.65303515500699 & -0.727953753069842 & 0 \\ -0.727953753069842 & 0.446838290721603 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[45]:
```
#Euler-Almansi strain tensor
EA_strain = 1/2*(sp.eye(3)-cauchy_d)
EA_strain
```

[45]:
$$\begin{bmatrix} -0.826517577503495 & 0.363976876534921 & 0 \\ 0.363976876534921 & 0.276580854639198 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

1.6 5. Volume ratio, volumetric strain

[46]:
```
#Volume ratio
J = sp.det(F_X_G)
J
```

[46]:
1.2350753

[47]:
```
#volumetric strain
eps_v= J -1
eps_v
```

[47]:
0.2350753

1.7 6. Isochoric and volumetric parts of the deformation gradient

[48]:
```
F_iso = 1/ J**(1/3) * F_X_G
F_vol = J**(1/3) * sp.eye(3)
print("F_ISO: ", F_iso, '\n', "F_VOL: ", F_vol)
F_iso*F_vol-F_X_G
#must be 0 matrix
```

```
F_ISO:  Matrix([[0.0548227152446906, -0.767536654267302, 0], [1.48008282571521,
-1.15107197088053, 0], [0, 0, 0.932042081684641]])
 F_VOL:  Matrix([[1.07291292920222, 0, 0], [0, 1.07291292920222, 0], [0, 0,
1.07291292920222]])
```

[48]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

1.8 7. Principal stretches

[49]:
```
eigenvalues2= C_G.eigenvals()
eigenvalues2
```

[49]:
```
{4.38034078643797: 1, 0.348240255962033: 1, 1.00000000000000: 1}
```

[50]:
```
# Extract eigenvalues and store them in value1, value2, and value3
eigenvalue_list = list(eigenvalues2)
value1, value2, value3 = eigenvalue_list[:3]  # Assuming there are exactly␣
 ↪three eigenvalues
lambda1 = value1**(1/2)
lambda2 = value3**(1/2)
lambda3 = value2**(1/2)

# Print the values
print("lambda1:", lambda1)
print("lambda2:", lambda2)
print("lambda3:", lambda3)
```

```
lambda1: 2.09292636909137
lambda2: 1.00000000000000
lambda3: 0.590118849014360
```

1.9 8. Engineering strain of a line element

[51]:
```
def length(x):
    return np.sqrt(sum(i**2 for i in x))

L = length(undeformed_nodes[1]- undeformed_nodes[3])
l = length(deformed_nodes[1]- deformed_nodes[3])
dL = l - L
eps_eng = l/L-1
print("Undeformed line length: ", L, " mm", '\n',
        "Deformed line length: ", l, " mm", '\n',
        "The line element length changed with: ", dL, " mm", '\n',
         "Engineering strain of the line element: ", eps_eng)
```

```
Undeformed line length:  3.1622776601683795  mm
 Deformed line length:  6.082762530298219  mm
 The line element length changed with:  2.92048487012984  mm
 Engineering strain of the line element:  0.9235384061671343
```

1.10 9. Surface element da/dA ratio

[52]:
```
No_1 = sp.Matrix([[1],
                  [0],
                  [0]])
No_2 = sp.Matrix([[0],
                  [1],
                  [0]])
No_3 = sp.Matrix([[0],
                  [0],
                  [1]])

AA = F_X_G.inv() * F_X_G.transpose().inv()
sp.det(F_X_G) * (No_3.T*AA*No_3)**(1/2)
```

[52]:
$$\left[1.2350753\right]$$

1.11 10. Angle of shear material line elements

[53]:
```
lamndba_N1 = (No_1.T*C_G*No_1)**(1/2)
lamndba_N1_value = lamndba_N1[0]
lamndba_N2 = (No_2.T*C_G*No_2)**(1/2)
lamndba_N2_value = lamndba_N2[0]
print("Lambda_N1: ", lamndba_N1_value)
print("Lambda_N2: ", lamndba_N2_value)
```

```
Lambda_N1:  1.58908898190127
Lambda_N2:  1.48437773157643
```

[54]:
```
tmp = No_1.T*C_G*No_2
tmp_value = tmp[0]

import math as m
rad = m.pi/2 - m.acos((tmp_value)/(lamndba_N1_value* lamndba_N2_value))
deg = rad*180 / m.pi
print(rad,"rad", '\n',deg, "°")
```

```
-1.019723676858753 rad
 -58.42586295356872 °
```

1.12 11. Rigid body rotation

[55]:
```
UU = (C_G)**(1/2)
UU
```

[55]:
$$\begin{bmatrix} 1.40149672731003 & -0.749006485779179 & 0 \\ -0.749006485779179 & 1.2815484907957 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

[56]:
```
R = F_X_G * UU.inv()
R
```

[56]:
$$\begin{bmatrix} -0.438375019572127 & -0.898792157406337 & 0 \\ 0.898792157406337 & -0.438375019572127 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

[57]:
```
theta_rad = m.acos((np.trace(R) - 1) /2)
theta_deg = theta_rad *180 / m.pi
print(theta_rad,"rad", '\n',theta_deg, "°")
```

```
2.0245862411838864 rad
 116.00024688009206 °
```

[58]:
```
NN = (R - R.transpose())/(2*m.sin(theta_rad))
NN
```

[58]:
$$\begin{bmatrix} 0 & -1.0 & 0 \\ 1.0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

1.13 12. Hencky's strain tensor

[59]:
```
V = F_X_G * R.transpose()
V
```

[59]:
$$\begin{bmatrix} 0.714370122972886 & 0.413868783316287 & 0 \\ 0.413868783316288 & 1.96867509513284 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

[60]:
```
import scipy.linalg
V = np.dot(F_X_G, R.transpose())
VV = np.array(V).astype(float)
scipy.linalg.logm(VV)
```

```
array([[-0.42275961,  0.3486512 ,  0.        ],
       [ 0.3486512 ,  0.63389155,  0.        ],
       [ 0.        ,  0.        ,  0.        ]])
```