Portable bridge for off-road wheelchairs

The Granny-Bridge

Introduction

When people lose the ability to get around freely they also lose part of themself and their independence. In later years improvements in battery technology has made electric wheelchairs more viable than ever, and it is helping people regain their independence. Off-road electric wheelchairs are made with the purpose of allowing people to traverse nature and other gnarly environments. Often with tougher frames, beefy suspension, high ground clearance, and knobby tires. But even the toughest off-road wheelchairs struggle with getting over large crevices and holes. Taking inspiration from deployable tank bridges, a solution has been proposed for the problem of traversing crevices.

A portable bridge for off-road electric wheelchairs is going to be the missing key in allowing people of all levels of mobility to climb even the largest mountains, and traverse the most deadly glaciers. Together with every portable bridge an assistant named Tim will be provided. This unit of a human specimen is going to be the one deploying the bridge for the wheelchair user. The bridge is to consist of two profiles that are to be mounted on the top of the wheelchair's roll bars. But to stay compliant with the OSHA guidelines the bridge solution must be exceptionally light weight, as not to cause Tim any work related injuries, the poor assistant has already got some back problems. An added benefit of a lightweight solution is increased range, giving the user more adventure per charge.

The safety of the wheelchair user is the main concern with regards to the design criterions. While the design is to be lightweight, the bridge must tolerate a combined load of up to 200 kg with a reasonable safety factor.

In this report various solutions will be explored, ranging from more economically viable extruded aluminum profiles, to high end laminate structures. Static and buckling analysis will be conducted. As the designs are all based on thin walled structures in one form or another, extra attention will be made regarding the buckling behaviour of the proposed solutions. As buckling is likely to be the dominant failure mode, most likely to introduce instability and pose the highest risk of failure to the user of the bridge. This study will present a feasibility assessment of various design solutions, evaluating both buckling behavior and the Tsai-Wu failure criterion for laminates.

Problem Definition

The portable bridge is constrained to the following rules:

- Height per profile: Maximum 75 mm
- Width per profile: 200 mm
- Length per profile: 3600 mm
- Wheelbase of the vehicle: 1200 mm
- Hold a weight of 200 kg (assumed evenly distributed)

Theory

Buckling

Buckling is often described as a sudden change in the shape of a structural component under compressive load, leading to a dramatic loss of stiffness. A structure subjected to gradually increasing load may experience catastrophic deformations when the load reaches a critical value, known as the buckling load. This critical load is often significantly lower than the material's yield or ultimate tensile strength, making buckling a dominant failure mode in slender structures. (Wikipedia, n.d.)

Lightweight structures frequently employ thin-walled profiles, which are particularly vulnerable to instability and failure due to buckling. In the case of the bridge elements designed in this report, buckling is expected to be the primary mode of failure.
For a simply supported beam under bending, with the top flange in compression and the bottom flange in tension, lateral-torsional buckling (LTB) can occur if the beam is insufficiently restrained. The designs presented in this report primarily utilize box profiles, which provide greater torsional stiffness compared to, for example, I-beams. This increased torsional stiffness reduces the risk of torsional buckling. Consequently, buckling is most likely to occur in the compressed top flange or as lateral instability of the entire profile.

Buckling in Abaqus

Abaqus provides two main approaches for buckling analysis: linear eigenvalue buckling analysis and nonlinear post-buckling analysis (the Riks method). Each method serves a different purpose

in structural assessment, and both can be used to study the stability of lightweight bridge components.

In this study linear eigenvalue buckling analysis is performed to estimate the critical buckling loads of the bridge components. If further refinement is needed, a nonlinear Riks analysis could be conducted to incorporate initial imperfections and capture post-buckling behavior. However, for this study, linear eigenvalue analysis is deemed sufficient.

Eigenvalue buckling analysis is the most commonly used method for estimating the critical load factor at which buckling occurs. In this approach, Abaqus first applies a unit reference load to the structure and then solves the eigenvalue problem based on the system's stiffness properties. The load proportionality factor (LPF), lambda, indicates the multiple of the applied load at which the structure theoretically buckles.

For example:

If LPF = 3.2, the structure will buckle at 3.2 times the applied load.
If LPF < 1, the structure is unstable under the applied load.

Negative eigenvalues will in most cases indicate that the structure would buckle if the load was applied in the opposite direction.
(Dassault Systèmes, n.d.)

Material Properties

Table 1, material properties of aluminium 7050 (Wikipedia, n.d.)

E	**Nu**	**Rho**
70000	0.33	2.7E-9

Table 2, material properties of S-glass/Epoxy

E1	**E2**	**E3**	**Nu12**	**Nu13**	**Nu23**	**G12**	**G13**	**G23**	**Rho**
48000	11000	11000	0.3	0.3	0.4	4200	4200	3600	200E-12

Table 1 provides the material properties of aluminium for the initial assessment. Table 2 provides the material properties of the final design using S-glass/Epoxy.

Method

We aim to evaluate different cross-section designs for the portable bridge and compare their strength to weight performance. To assess the feasibility of the profiles, we will first conduct simulations using aluminum to analyze their structural behavior, and gain insights into the stress distributions. To ensure a fair comparison in line with the lightweight design objective, we will adjust the profile thickness to maintain approximately the same mass across all designs.

Model

The mechanical system describing the portable bridge can be modeled with pinned supports at both ends with the loads applied at the critical position (see Figure 1). We assume that the total weight of 200 kg is evenly distributed between the four wheels.

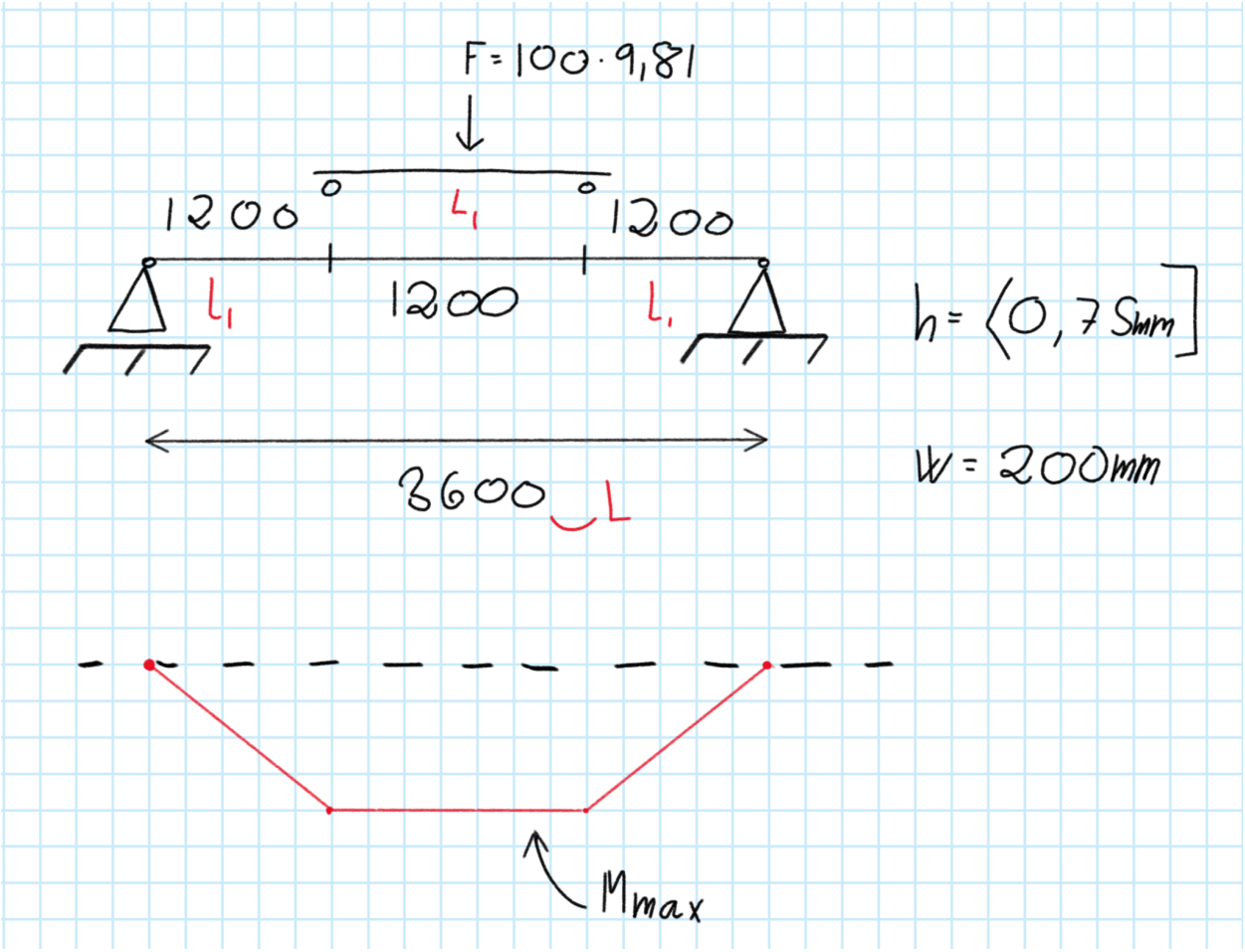


Figure 1, static model of a portable bridge element

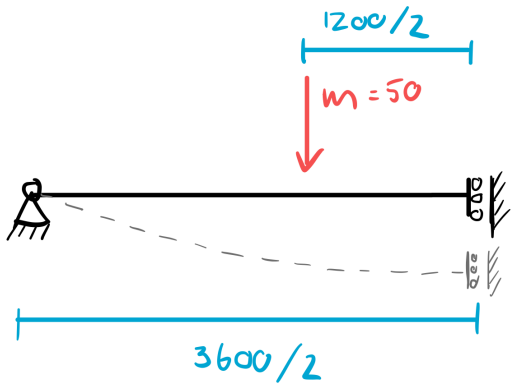


Figure 2, reduced model of portable bridge.

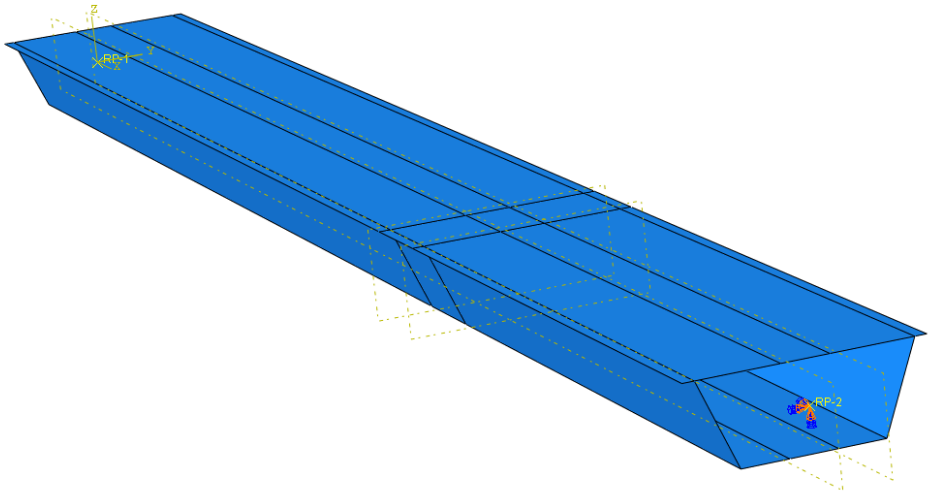


Figure 3, boundary conditions of box profile.

Utilizing symmetry, the portable bridge model is reduced to half its size, as illustrated in Figure 2. The corresponding boundary conditions, shown in Figure 3, are applied accordingly. RP-1, representing the free end, is constrained to rotate only around the y-axis, while RP-2, located at the midplane, remains free to move in the x- and z-directions.

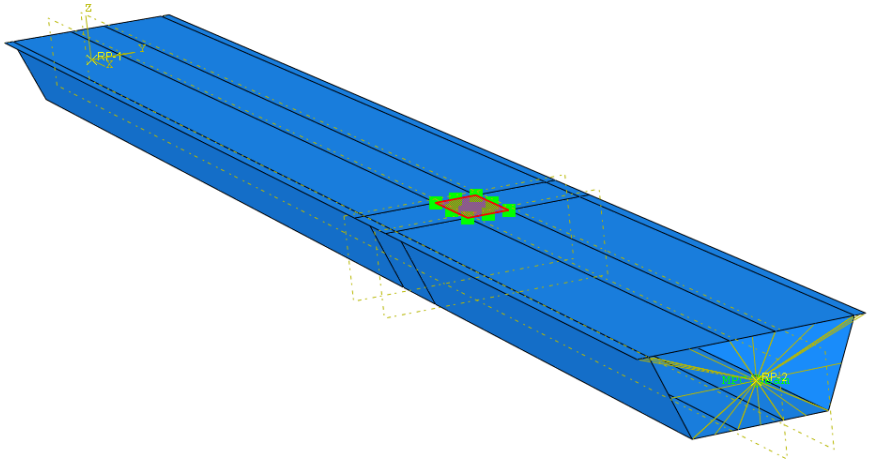


Figure 4, load surface

To model the load of the wheel, a distributed mass of 50 kg was applied over a rectangular surface of 40x80 mm (highlighted in Figure 4).

Shell elements are ideal for modeling the cross-section due to how they efficiently capture bending and in-plane stresses while requiring fewer computational resources than solid elements. Given the bridge's thin-walled structure, shell elements accurately represent its structural behavior without excessive meshing, making them well suited for our case.

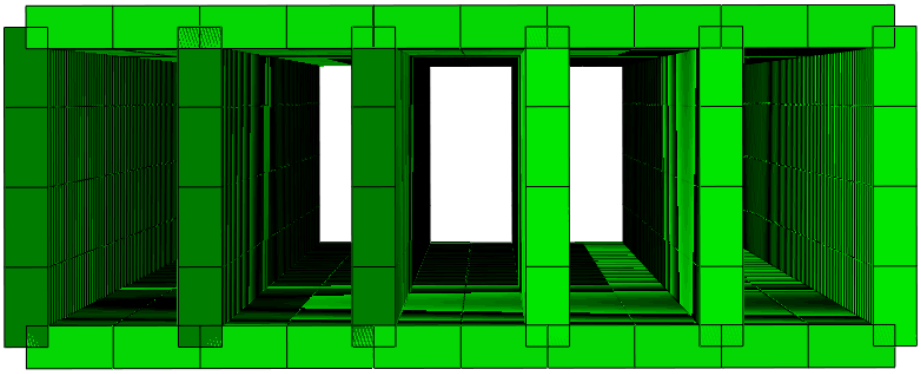


Figure 5, exaggerated view of offset type set as midplane creating overlapping sections.

Second Moment of Area - Simulated / Analytical (%)

	n = 0	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8
t = 1	99.99	100.2	100.39	100.57	100.72	100.87	101.	101.12	101.24
t = 2	99.98	100.39	100.77	101.12	101.44	101.73	102.	102.25	102.48
t = 3	99.94	100.57	101.14	101.66	102.14	102.58	102.99	103.37	103.72

Table 3, the second moment of area for a shell model using midplane offset, divided by the analytical values multiplied by 100. Done for various numbers of inner supports and uniform thicknesses.

Due to the nature of 2D shell element connections, material may either overlap or be missing at the connecting nodes (Figure 5). For simplicity, the offset type is set to midplane for all simulations. Table 3 illustrates the impact of the second moment of area when using shell elements with a midplane offset, compared to the analytical solution (see Appendix for calculations). As expected, we see that thinner walls lead to smaller deviations.

For the initial comparative assessment, we assign each profile a thickness that ensures its weight is equivalent to that of a simple box profile with a thickness of 1 mm. This constant weight basis is then used to determine the thickness for each profile by dividing it by the total length of the cross-sectional perimeter. By doing this, we will get a measure of its specific strength. Only the comparative values for the initial assessment are of interest.

$$\text{perimeter} \cdot t = \text{constant}$$

$$(2 \cdot 200 + 2 \cdot 75) \cdot 1 = 550$$

$$\Rightarrow \quad t = \frac{550}{\text{perimeter}}$$

Although this simplification is not exact (as it does not account for overlaps in the corners), using a small thickness of 1 mm minimizes this effect and provides a good indication of its structural properties when comparing different cross sections (see Table 3).

The global element size (mesh size) was set to 18 for all simulations.

Box Profile vs Slanted Box Profile

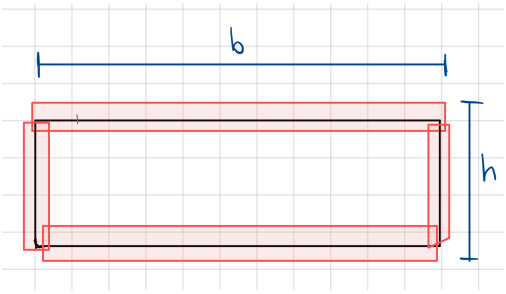


Figure 6, box profile without spars

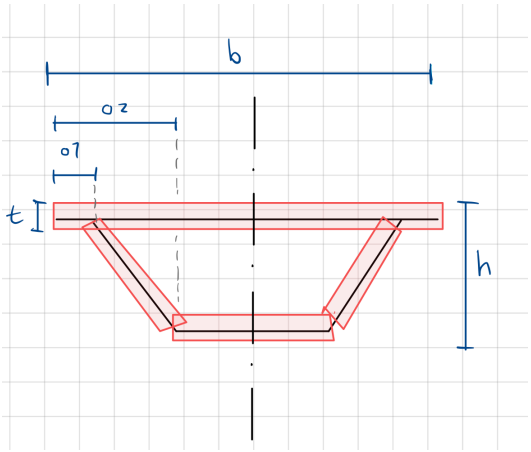


Figure 7, slanted box profile

Dimensions according to Figure 6 and Figure 7:

- b = 200 mm
- h = 75 mm
- o1 = 10 mm
- o2 = 40 mm
- t is dynamically set to match weight between cases

The length of the bridge (in plane) is set to 1800 mm, which is half the size of the full bridge (due to reductions from symmetry).

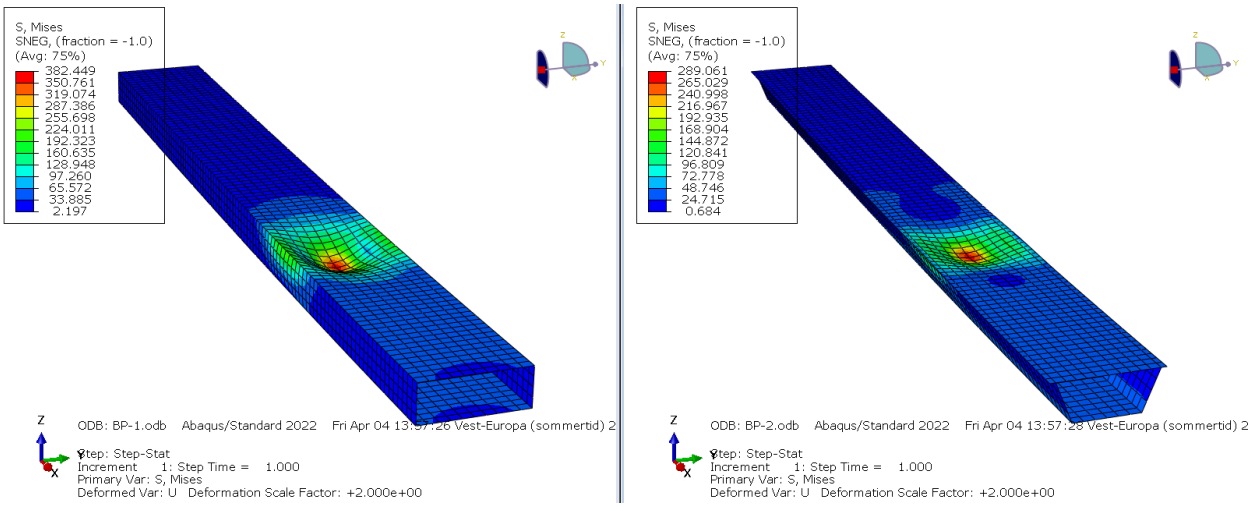


Figure 8, Mises stresses from a box profile, and a slanted box profile.

Figure 8 shows a 24.4% reduction in maximum Mises stress when transitioning from a rectangular box profile to one with slanted sides. This reduction is likely due to the decreased volume, which redistributes more material to the upper plate. The slanted profile is the preferred option when considering shear stresses in the yz-plane. Although this study does not explicitly account for shear effects in the yz-plane, we will proceed with the slanted profile.

Redistributing Material to Spars

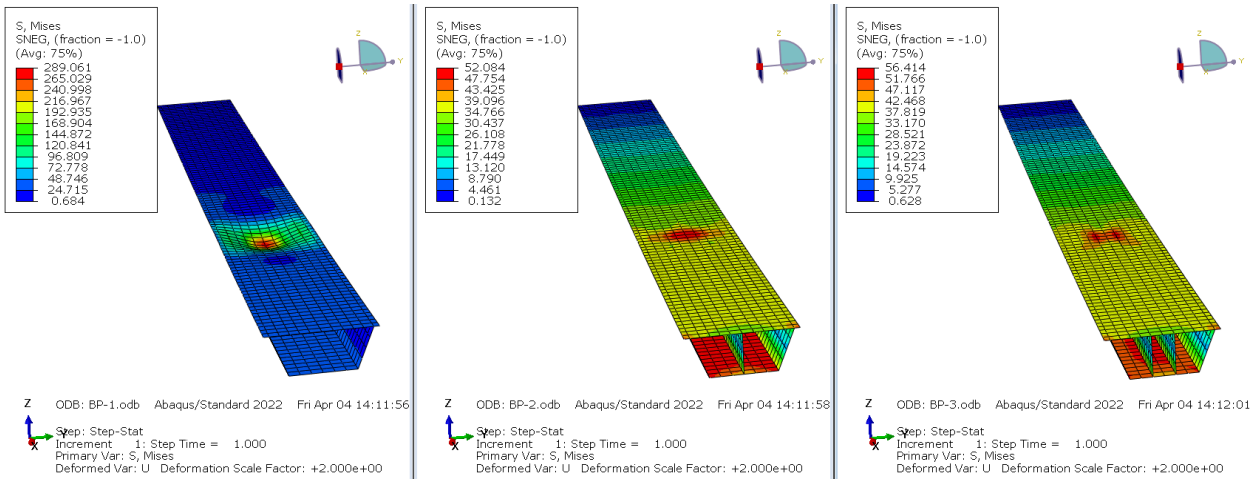


Figure 9, effects on Mises stresses when redistributing material to spars

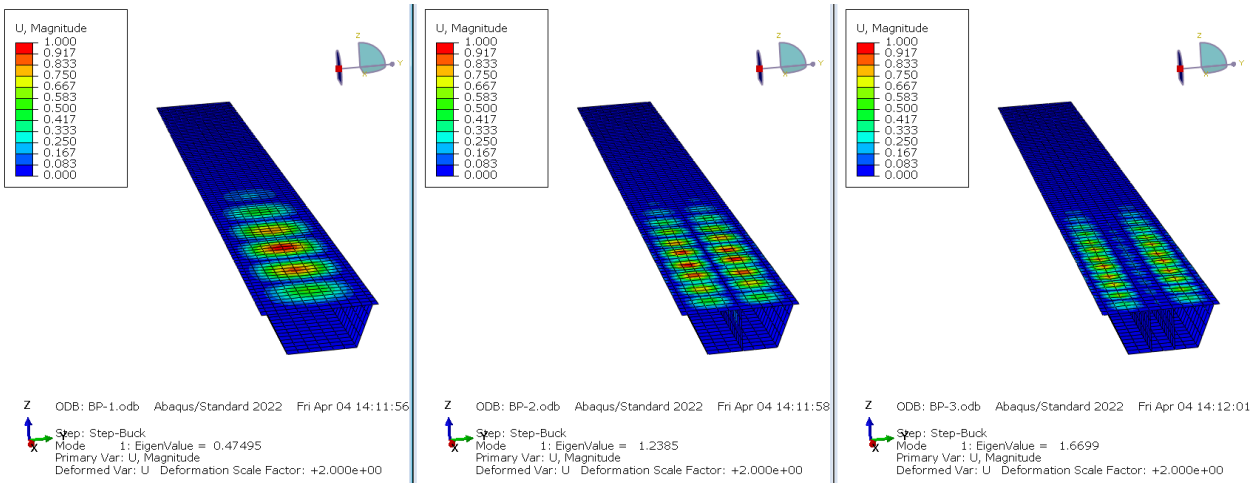


Figure 10, effects on buckling modes when redistributing material to spars

As seen from Figure 10, the eigenvalue is increased by 160.7 % by adding a single spar, and increased by 251.6 % by adding two spars. This is a significant increase in the critical buckling load. Figure 9 shows a large reduction in max Mises stresses when adding a spar. To keep the design simple, we will go further with using a single spar for the composite designs.

Composite Design

Various laminate layups were tested and optimized to achieve the lowest possible weight, while maintaining a minimum safety factor of 2. This is based on the first positive eigenvalue of the buckling analysis (eig. >= 2.0), and the Tsai-Wu exposure factor (fE <= 0.5).
The final design uses the slanted profile with a single spar, using the following layups:

Top sheet:

		Ply Name	Region	Material	Thickness	CSYS	Rotation Angle	Integration Points
1	✔	Ply-5	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3
2	✔	Ply-6	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
3	✔	Ply-7	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3
4	✔	Ply-7-Copy2	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
5	✔	Ply-7-Copy1	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
6	✔	Ply-2	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3
7	✔	Ply-3-Copy1	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
8	✔	Ply-4	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3

Bottom sheet:

		Ply Name	Region	Material	Thickness	CSYS	Rotation Angle	Integration Points
1	✔	Ply-1	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3
2	✔	Ply-2	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
3	✔	Ply-3	(Picked)	S-glass/Epoxy	0.25	<Layup>	90	3
4	✔	Ply-4	(Picked)	S-glass/Epoxy	0.25	<Layup>	0	3

Slanted sheet (one for each side):

		Ply Name	Region	Material	Thickness	CSYS	Rotation Angle	Integration Points
1	✔	Ply-1	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
2	✔	Ply-2	(Picked)	S-glass/Epoxy	0.25	<Layup>	-45	3
3	✔	Ply-3	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
4	✔	Ply-6	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
5	✔	Ply-7	(Picked)	S-glass/Epoxy	0.25	<Layup>	-45	3
6	✔	Ply-8	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3

Spar sheet:

		Ply Name	Region	Material	Thickness	CSYS	Rotation Angle	Integration Points
1	✔	Ply-1	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
2	✔	Ply-2	(Picked)	S-glass/Epoxy	0.25	<Layup>	-45	3
3	✔	Ply-3	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
4	✔	Ply-6	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3
5	✔	Ply-7	(Picked)	S-glass/Epoxy	0.25	<Layup>	-45	3
6	✔	Ply-8	(Picked)	S-glass/Epoxy	0.25	<Layup>	45	3

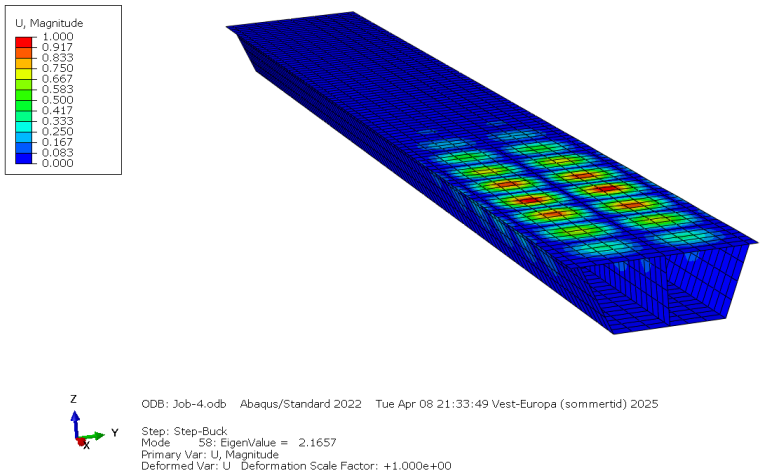


Figure 11, First positive eigenvalue of the buckling step (Eig. = 2.1657)

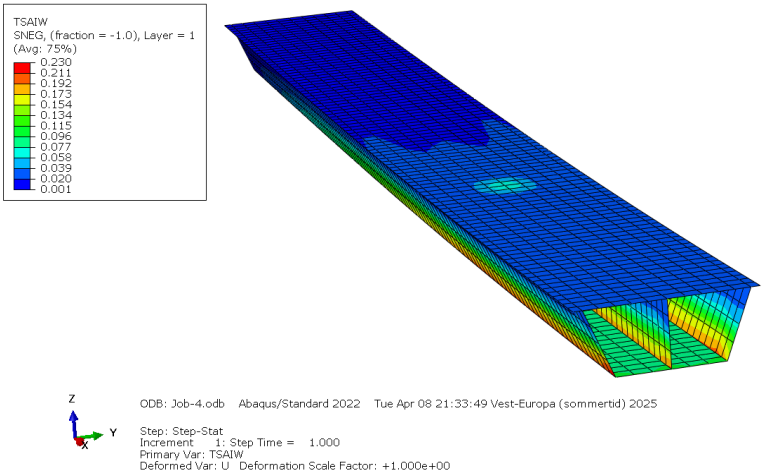


Figure 12, Tsai-Wu exposure factor

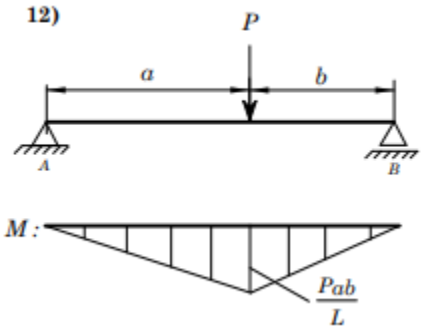
Analytical Calculations of Bending Stiffness

A Python script was made for calculating the effective bending stiffness of a composite cross-sectional profile made of multiple material segments, including slanted (rotated) and layered laminates (see Appendix). The analytical model helps identify modeling errors, validate assumptions, and offers a way of verifying the accuracy and reliability of the FEA simulation.

The script uses a class based approach to define the geometry and material properties of each section of a composite profile. The class **MaterialSection** defines a rectangular section with:

- Location of its centroid
- Width and height
- Elastic modulus
- Rotation angle (optional)

The class **Profile** is used for assembling and calculating its neutral axis and bending stiffness.



$$w(x) = \frac{Pbx}{6EI \cdot L}(L^2 - b^2 - x^2) \ , \ x \le a$$

$$w(x) = \frac{Pa(L-x)}{6EI \cdot L}(2Lx - a^2 - x^2) \ , \ x \ge a$$

The resulting displacement was calculated using Formula 12 from the Mekanikk 2 formula sheet.

Key Results

Material	Bucking Eigenvalue	Critical Tsai-Wu	Weight [kg] per component	True Disp. [mm] (U3)	Analytical Disp. [mm]
S-glass/ Epoxy	**2.1657**	**0.230**	**6.3**	**45.697**	**43.997**

Discussion

The proposed portable bridge system demonstrates promising feasibility for enabling off-road wheelchair mobility. Preliminary FEA results for the slanted aluminium box profile show a 24.4% reduction in maximum von Mises stress compared to a rectangular box profile of equivalent mass. This improvement is due to several related mechanisms. One of these is that the slanted walls redistribute material towards the compression flanges, where stresses due to bending are highest. Additionally, the angled walls contribute to geometric stiffness allowing the profile to better resist transverse shear forces. The result is a more structurally efficient profile, maintaining the same weight as the rectangular profile.

The buckling load factor increased significantly when internal spars were introduced. The load factor increased 160.7 % for a single spar, and 251.6 % for dual spars. Spars break up wide flat surfaces into smaller segments that prevent buckling of the wall while under compression. With the flat panels now braced internally, the structure behaves more like a series of shorter, stiffer components that are better supported against deformation. This leads to a higher critical load before buckling occurs, without the need to increase the profile's mass.
These gains are further enhanced by transitioning to S-glass/epoxy laminates, which offer substantial weight reduction and customizable mechanical properties. Composite materials like S-glass/epoxy allow us to modify stiffness and strength by adjusting the fiber orientation. A [0/±45/90]s layup was used in our design. 0 plies carry the main bending loads along the bridge span, providing high axial stiffness ($E_1 \approx 48$ GPa). ±45 plies handle shear and torsional stresses ($G_{12} \approx 4.2$ GPa), which are especially important given the bridge's open-section geometry and slanted walls. 90 plies stabilize the flanges against local buckling and provide transverse stiffness ($E_2 \approx 11$ GPa).

The multi-directional fiber layup helps maintain most of the bending stiffness you would get from a unidirectional laminate, but with a much better ability to handle shear stresses.The Balanced layup is especially important for practical durability. While using only 0 fibers would give higher stiffness in theory, such a design would create high shear stresses near the spar connections potentially leading to failure. By including ±45 plies, the laminate will be better at handling shear stresses, avoiding overstressing of any single direction.

The strength of the [0/±45/90]s layup is supported by the Tsai-Wu failure analysis (Figure 12), which shows a maximum exposure factor of just 0.23. This indicates that all plies are well within the material's strength limits, with a significant safety margin for real-world loading variations. However, the buckling analysis reveals that the first positive eigenvalue is 2.17 (Figure 11), making buckling the governing failure mode. Although it exceeds the target safety factor of 2.0, it is more critical than material failure due to its lower margin. The associated mode shape indicates global bending rather than local flange wrinkling, confirming that the spar layout and fiber orientations contribute effectively to overall stability.

It is important to recognize that this is a simplified feasibility study. For example, the joints have not been modeled in detail. If bolts are used, metal inserts would be needed to prevent crushing or peeling of the composite. Adhesive bonding could work as a lighter option especially beneficial for the composite profile, but we don't yet know how it would hold up under repeated use and wear in the field. Another important aspect to consider for further development is the interaction at the bottom contact surface of the ends of the bridge. Imperfections in the ground (e.g. rocks) can lead to high localized compressive stresses. A potential solution is to reinforce the contact area at the ends with a ductile material such as aluminium or rubber, with the intention of transferring an evenly distributed load to the brittle composite material. Updated simulations and weight assessments would then be necessary.

Conclusion

This study shows that a lightweight portable bridge for off-road wheelchairs is feasible. By using a slanted composite profile with an internal spar and a balanced fiber layup, the design meets both the strength and the buckling requirements while keeping the weight low (6.3 kg). Simulations confirm a safety factor above 2.0, and a low stress exposure factor of 0.23 (Tsai-Wu).

The concept is promising but still early-stage. Joints, adhesive bonding, and real-world interactions such as the surface contact at the ends need further work. With further refinement, the Granny-Bridge can safely provide access to rough terrain without breaking Tim's back. There are also plans in the pipeline of expanding the product range to include the Gramps-bridge, a beefier version of the Granny-bridge.

References

Dassault Systèmes. (n.d.). *Eigenvalue buckling prediction*. ABAQUS Analysis User's Manual (v6.6). Retrieved April 3, 2025, from ttps://classes.engineering.wustl.edu/2009/spring/mase5513/abaqus/docs/v6.6/books/usb/default.htm?startat=pt03ch06s02at02.html

Wikipedia. (n.d.). *Buckling*. Wikipedia. Retrieved April 3, 2025, from https://en.m.wikipedia.org/wiki/Buckling

Wikipedia. (n.d.). *7050 aluminium alloy.* Wikipedia. Retrieved April 10, 2025, from https://en.wikipedia.org/wiki/7050_aluminium_alloy

Appendix

Attached code:

- Abaqus_Script.py
- Analytical_calculations.ipynb

Abaqus Script

In [ ]:
```
from abaqus import *
from abaqusConstants import *
import numpy as np

mcfrp = {
    "name": "S-glass/Epoxy", "units": "MPa-mm-Mg", "type": "UD", "fiber": "S-glass",
    "Vf": 0.55, "rho": 2000E-12,
    "description": "Typical UD S-glass/Epoxy from TMM4175",
    "E1": 48000, "E2": 11000, "E3": 11000,
    "v12": 0.3, "v13": 0.3, "v23": 0.4,
    "G12": 4200, "G13": 4200, "G23": 3600,
    "a1": 4e-06, "a2": 2.0e-05, "a3": 2.0e-05,
    "XT": 1300, "YT": 40, "ZT": 40,
    "XC": 800, "YC": 140, "ZC": 140,
    "S12": 70, "S13": 70, "S23": 40,
    "f12":-0.5, "f13":-0.5, "f23":-0.5
}


def matlib(modelname):
    mod = mdb.models[modelname]

    # Definerer material
    mat = mod.Material(mcfrp['name'])
    mat.Density(table=((mcfrp['rho'], ), ))
    mat.Elastic(type=ENGINEERING_CONSTANTS,
                table=((mcfrp['E1'], mcfrp['E2'], mcfrp['E3'],
                        mcfrp['v12'], mcfrp['v13'], mcfrp['v23'],
                        mcfrp['G12'], mcfrp['G13'], mcfrp['G23']), ))
    mat.elastic.FailStress(table=((mcfrp['XT'], mcfrp['XC'], mcfrp['YT'], mcfrp['YC'], mcfrp['S12'], mcfrp['f12'], 0.0), ))


def boxpro(modelname, L, b, h, t, n_spars, esize, applied_mass, profile=1):

    # Define length from edge to point load
    load_pos = (2.0 / 3.0) * L
    # Correct h and b (due to offsetType = MIDDLE_SURFACE)
    b -= t
    h -= t

    # Create model
    mod = mdb.Model(name=modelname, modelType=STANDARD_EXPLICIT)
    matlib(modelname)

    # region Sketch
    if profile == 1:
        # ------- Outer Sketch ----------
        ske = mod.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        ske.rectangle(point1=(-b/2.0, -h/2.0), point2=(b/2.0, h/2.0))


        # ------- Inner Sketch ----------

        # Vertical Spars
        if n_spars > 0:
            spacing = b / (n_spars + 1)
            x_pos = -b/2.0 + spacing
            for _ in range(n_spars):
                ske.Line(point1=(x_pos, h/2.0), point2=(x_pos, -h/2.0))
                x_pos += spacing


        # Calculate total length
        total_length = 2*b + 2*h + n_spars*h
        mass_factor = 550 # factor that will give simple box profile t=1
        t = mass_factor / total_length # update t

    elif profile == 2:
        ske = mod.ConstrainedSketch(name='__profile__', sheetSize=200.0)
        ske.Line(point1=(-b/2.0, h/2.0), point2=(b/2.0, h/2.0))

        o1 = 10.0
        o2 = 40.0

        ske.Line(point1=(-b/2.0 + o1, h/2.0), point2=(-b/2.0 + o2, -h/2.0))
        ske.Line(point1=(b/2.0 - o1, h/2.0), point2=(b/2.0 - o2, -h/2.0))
        ske.Line(point1=(-b/2.0 + o2, -h/2.0), point2=(b/2.0 - o2, -h/2.0))

        # Vertical Spars
        if n_spars > 0:
            spacing = (b - 2*o2) / (n_spars + 1)
            x_pos = -b/2.0 + o2 + spacing
            for _ in range(n_spars):
                ske.Line(point1=(x_pos, h/2.0), point2=(x_pos, -h/2.0))
                x_pos += spacing

        # Calculate total length
        total_length = b + (b-2*o2) + 2*np.sqrt((o2-o1)**2 + h**2) + n_spars*h
```

```
    mass_factor = 550 # factor that will give simple box profile t=1
    t = mass_factor / total_length # update t

elif profile == 3:
    pass # Legg til profil her


# Create part from sketch
prt = mod.Part(name='Box', dimensionality=THREE_D, type=DEFORMABLE_BODY)
prt.BaseShellExtrude(sketch=ske, depth=L)
del mod.sketches['__profile__']


# Partition for load surface
wheel_width = 0.1 * b + 0.1 # added small value
wheel_length = wheel_width * 2
id = prt.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=(wheel_width)).id
prt.PartitionFaceByDatumPlane(datumPlane=prt.datums[id], faces=prt.faces)
id = prt.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=(-wheel_width)).id
prt.PartitionFaceByDatumPlane(datumPlane=prt.datums[id], faces=prt.faces)

id = prt.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=(load_pos + wheel_length)).id
prt.PartitionFaceByDatumPlane(datumPlane=prt.datums[id], faces=prt.faces)
id = prt.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset=(load_pos - wheel_length)).id
prt.PartitionFaceByDatumPlane(datumPlane=prt.datums[id], faces=prt.faces)

"""
# Material and section
mat = mod.Material(name='Alu')
mat.Density(table=((2.7E-9, ), ))
mat.Elastic(table=((70000.0, 0.33), ))
mod.HomogeneousShellSection(name='Section-shell',
    preIntegrate=OFF, material='Alu', thicknessType=UNIFORM, thickness=t,
    thicknessField='', nodalThicknessField='',
    idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF,
    integrationRule=SIMPSON, numIntPts=5)
region = prt.Set(faces=prt.faces, name='faces-all')
prt.SectionAssignment(region=region, sectionName='Section-shell', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='', thicknessAssignment=FROM_SECTION)

"""

# Mesh
prt.setMeshControls(regions=prt.faces, elemShape=QUAD, technique=STRUCTURED)
prt.seedPart(size=esize, deviationFactor=0.1, minSizeFactor=0.1)
prt.generateMesh()


# Assembly and constraints
ass = mod.rootAssembly
ass.DatumCsysByDefault(CARTESIAN)
ins = ass.Instance(name='Box', part=prt, dependent=ON)
ass.rotate(instanceList=('Box', ), axisPoint=(0.0, 0.0, 0.0), axisDirection=(0.0, 1.0, 0.0), angle=90.0)
ass.rotate(instanceList=('Box', ), axisPoint=(0.0, 0.0, 0.0), axisDirection=(1.0, 0.0, 0.0), angle=90.0)
rf1id = ass.ReferencePoint(point=(0.0, 0.0, 0.0)).id
rf2id = ass.ReferencePoint(point=(L, 0.0, 0.0)).id
regionRF1=ass.Set(referencePoints=(ass.referencePoints[rf1id],), name='RF1')
regionRF2=ass.Set(referencePoints=(ass.referencePoints[rf2id],), name='RF2')
edges1=ins.edges.getByBoundingBox(xMax=0.0)
region1 = ass.Set(edges=edges1, name = 'EDGES1')
edges2=ins.edges.getByBoundingBox(xMin=L)
region2 = ass.Set(edges=edges2, name = 'EDGES2')
mod.MultipointConstraint(name='Constraint-1',
    controlPoint=regionRF1, surface=region1, mpcType=BEAM_MPC,
    userMode=DOF_MODE_MPC, userType=0, csys=None)
mod.MultipointConstraint(name='Constraint-2',
    controlPoint=regionRF2, surface=region2, mpcType=BEAM_MPC,
    userMode=DOF_MODE_MPC, userType=0, csys=None)

# Steps, BC and loading
bc1 = mod.DisplacementBC(name='BC1', createStepName='Initial',
    region=regionRF1, u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET, ur3=SET)
bc2 = mod.DisplacementBC(name='BC2', createStepName='Initial',
    region=regionRF2, u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET, ur3=SET)

mod.BuckleStep(name='Step-Buck', previous='Initial', numEigen=2, vectors=4, maxIterations=700)
mod.StaticStep(name='Step-Stat', previous='Step-Buck')


bc1.setValuesInStep(stepName='Step-Buck',
    ur2=FREED, buckleCase=PERTURBATION_AND_BUCKLING)
bc1.setValuesInStep(stepName='Step-Stat',
    ur2=FREED)
bc2.setValuesInStep(stepName='Step-Buck',
    u1=FREED, u3=FREED, buckleCase=PERTURBATION_AND_BUCKLING)
bc2.setValuesInStep(stepName='Step-Stat',
    u1=FREED, u3=FREED)

# Create load surface
faces = ins.faces.getByBoundingBox(
    xMin=load_pos-wheel_length, xMax=load_pos+wheel_length,
    zMin=h/2.0,
    yMax=wheel_width, yMin=-wheel_width
```

```
    )
    region3 = ass.Set(name='CONTACT-SURFACE', faces=faces)

    # Add inertia to load surface
    ass.engineeringFeatures.NonstructuralMass(
        name='Inertia-1', region=region3, units=TOTAL_MASS, magnitude=applied_mass,
        distribution=MASS_PROPORTIONAL)

    # Add gravity
    mod.Gravity(name='Load-1', createStepName='Step-Buck',
        comp3=-9810.0, distributionType=UNIFORM, field='')
    mod.Gravity(name='Load-2', createStepName='Step-Stat',
        comp3=-9810.0, distributionType=UNIFORM, field='')


    # Job:
    # job = mdb.Job(name=modelname, model=modelname)
    # job.submit()


# boxpro(modelname='BP-1', L=1800, b=200, h=75, t=0.5, n_spars=0, esize=18, applied_mass=0.05, profile=2)
boxpro(modelname='BP-2', L=1800, b=200, h=75, t=0.5, n_spars=1, esize=18, applied_mass=0.05, profile=2)
# boxpro(modelname='BP-3', L=1800, b=200, h=75, t=0.5, n_spars=2, esize=18, applied_mass=0.05, profile=2)
```

In [1]:
```python
import numpy as np

wheel_base = 1200       # [mm]
L = 3600                # [mm]
M = 200                 # [kg]
h = 75                  # [mm]
g = 9.81                # [m/s^2]
F_point = M * g / 4.0   # [N] (per wheel)

a = (L - wheel_base) / 2.0 # [mm] distance frome edge to point load (wheel)
```

Deformation functions

In [2]:
```python
def disp(x, P, a, L, EI):
    """ Fra formelsamling i mek 2. Bjelkeformel 12 """
    b = L - a
    if x <= a:
        return P*b*x/(6*EI*L) * (L**2 - b**2 - x**2)
    else:
        return P*a*(L-x)/(6*EI*L) * (2*L*x - a**2 - x**2)


def calculate_critical_disp(wheel_base, F1, F2, L, BendingStiffness):

    a1 = (L - wheel_base) / 2.0
    a2 = a1 + wheel_base

    max_disp_pos = L/2.0

    # Calculate deflection caused by each point load
    w = disp(max_disp_pos, F1, a1, L, BendingStiffness) + disp(max_disp_pos, F2, a2, L, BendingStiffness)

    print(f"Max displacement: {w}")
```

Deviations of second moment of area caused by mid plane offset in Abaqus

In [3]:
```python
t = 1
PROFILE_WIDTH = 200

h = 75
max_spars = 8

I_approx_list = np.zeros(max_spars+1)
I_analytical_list = np.zeros(max_spars+1)

# Calculate box profile
I_horizontal_approx = (1.0/12.0) * (PROFILE_WIDTH - t) * t**3 + ((h - t)/2.0)**2 * (PROFILE_WIDTH - t) * t # Steiners sats bunn eller topp
I_vertical_approx = (1.0/12.0) * (t) * (h - t)**3 # Steiners sats høyre eller venstre side
I_box_approx = 2 * I_horizontal_approx + 2 * I_vertical_approx
I_approx_list[0] = I_box_approx

# Calculate analytical
I_box_analytical = (1.0/12.0) * (PROFILE_WIDTH * h**3 - (PROFILE_WIDTH-2*t) * (h-2*t)**3)
I_analytical_list[0] = I_box_analytical
I_spar_analytical = (1.0/12.0) * t * (h-2*t)**3

# add spars
for i in range(max_spars):
    # add spar contribution to I and append to list
    I_box_approx += I_vertical_approx
    I_approx_list[i+1] = I_box_approx

    I_box_analytical += I_spar_analytical
    I_analytical_list[i+1] = I_box_analytical


print(np.round(I_approx_list/I_analytical_list * 100, 2))
```

```
[ 99.99 100.2  100.39 100.57 100.72 100.87 101.   101.12 101.24]
```

Calculate composite bending stiffness

The class `Profile` consists of rectangular sections from the class `MaterialSection`. `Profile` calculates the bending stiffness of a composite cross section.

In [ ]:
```python
import math

def rotated_inertia(b, h, theta_deg):
    """ Transformasjonsformler fra Mekanikk 2 Formelhefte side 16 """
    theta = math.radians(theta_deg)
    # For a Rectangle
    Iy = (b * h**3) / 12
    Iz = (h * b**3) / 12
    Iyz = 0
    Iyp = 0.5 * (Iy + Iz) + 0.5 * (Iy - Iz) * math.cos(2 * theta) - Iyz * math.sin(2 * theta)
    return Iyp
```

```python
class MaterialSection:
    def __init__(self, z_center, b, h, E_modulus, rotation_deg=0.0):
        self.z_center = z_center
        self.b = b
        self.h = h
        self.area = b*h
        self.E_modulus = E_modulus
        self.rotation_deg = rotation_deg

    @property
    def EA(self):
        return self.E_modulus * self.area

    @property
    def I_local(self):
        theta = math.radians(self.rotation_deg)
        # For a Rectangle
        Iy = (self.b * self.h**3) / 12
        Iz = (self.h * self.b**3) / 12
        Iyz = 0
        Iyp = 0.5 * (Iy + Iz) + 0.5 * (Iy - Iz) * math.cos(2 * theta) - Iyz * math.sin(2 * theta)
        return Iyp

    def __repr__(self):
        return f"MaterialSection(z_center={self.z_center}, area={self.area}, E_modulus={self.E_modulus})"


class Profile:
    def __init__(self):
        self.sections = []

    def add_section(self, z_center, b, h, E_modulus, rotation_deg=0.0):
        section = MaterialSection(z_center, b, h, E_modulus, rotation_deg=rotation_deg)
        self.sections.append(section)

    def compute_neutral_axis(self):
        """
        Beregn nøytralaksen (z-coordinate) for en kompositttbjelkeprofil, tilsvarende formel i mekanikk 2 formelhefte.
        Nøytralaksen er definert ved:

            z_neutral = sum(EA * z_center) / sum(EA)

        Der EA = E_modulus * area for hver seksjon.
        """
        total_EA = sum(section.EA for section in self.sections)
        if total_EA == 0:
            raise ValueError("Total EA kan ikke være 0")
        numerator = sum(section.EA * section.z_center for section in self.sections)
        neutral_axis =  numerator / total_EA
        self.neutral_axis = neutral_axis
        return neutral_axis


    def compute_bending_stiffness(self):
        # First, compute the neutral axis
        self.compute_neutral_axis()

        EI = 0
        for section in self.sections:
            EI += section.E_modulus * (section.I_local + section.area * (section.z_center - self.neutral_axis)**2)
        self.bending_stiffness = EI
        return EI

    def calculate_cross_section_ares(self):
        cs_area = 0
        for section in self.sections:
            cs_area += section.area
        return cs_area


    def __repr__(self):
        sections_str = "\n".join(str(section) for section in self.sections)
        return f"Profile med {len(self.sections)} seksjoner:\n{sections_str}"
```

Inlcude laminates

In [5]:
```python
from laminatelib import laminateStiffnessMatrix, laminateThickness, repeatLayers, symmetricLayup
import matlib

def effective_laminate_Ex(layup):
    ABD = laminateStiffnessMatrix(layup)
    h=laminateThickness(layup)
    Ex = (1/h)*( ABD[0,0] - (ABD[0,1]**2)/ABD[1,1]  )
    return Ex
```

In [6]:
```python
def print_weigh_estimate(cross_section_area):
    rho_s_glass = 2.0E-6 # [kg/mm^3] Density of S-glass

    L = 3600

    # Total weight
```

```
    total_weight = rho_s_glass * cross_section_area * L
    print(f"Total weight per bridge component: {total_weight:.2f} kg")
```

Slanted Profile with 1 spar (Test Profile)

In [7]:
```
m1 = matlib.get('S-glass/Epoxy')

# Define top Layup
layup_top = [
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25}
]
repeatLayers(layup_top, 2)
symmetricLayup(layup_top)

# Define bottom Layup
layup_bottom = layup_top

# Define side layups
layup_sides = [
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori': -45  , 'thi':0.25}
]
repeatLayers(layup_sides, 2)
symmetricLayup(layup_sides)

# Get effective E-modulus
E_eff_top = effective_laminate_Ex(layup_top)
E_eff_bottom = effective_laminate_Ex(layup_bottom)
E_eff_sides = effective_laminate_Ex(layup_sides)

# Define custom profile
outer_height = 75
outer_width = 200
o1 = 10
o2 = 40


profile1 = Profile()

# TOP
t_top = laminateThickness(layup_top)
profile1.add_section(z_center=(outer_height-0.5*t_top),
                     b=outer_width,
                     h=t_top,
                     E_modulus=E_eff_top)
# BOTTOM
t_bottom = laminateThickness(layup_bottom)
profile1.add_section(z_center=(0.5*t_bottom),
                     b=(outer_width-2*o2),
                     h=t_bottom,
                     E_modulus=E_eff_bottom)

# SLANTED SIDE 1
t_side = laminateThickness(layup_sides)
length_slanted_side = np.sqrt((o2-o1)**2 + outer_height**2)
angle = math.degrees(np.arctan((o2-o1) / outer_height))
profile1.add_section(z_center=0.5*outer_height,
                     b=t_side,
                     h=length_slanted_side,
                     E_modulus=E_eff_sides,
                     rotation_deg=angle)

# SLANTED SIDE 2
profile1.add_section(z_center=0.5*outer_height,
                     b=t_side,
                     h=length_slanted_side,
                     E_modulus=E_eff_sides,
                     rotation_deg=angle)

# SPAR
t_spar = laminateThickness(layup_sides)
profile1.add_section(z_center=0.5*outer_height,
                     b=t_spar,
                     h=outer_height,
                     E_modulus=E_eff_sides)

# EFFECTIVE BENDING STIFFNESS
Db = profile1.compute_bending_stiffness()

calculate_critical_disp(wheel_base=1200, F1=(50*9.81), F2=(50*9.81), L=3600 , BendingStiffness=Db)
print_weigh_estimate(cross_section_area=profile1.calculate_cross_section_ares())
```

```
Max displacement: 20.427895342741124
Total weight per bridge component: 12.02 kg
```

Design 1

In [8]:
```
m1 = matlib.get('S-glass/Epoxy')
```

```python
# Define top layup
layup_top = [
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
]

# Define bottom layup
layup_bottom = [
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori': 90  , 'thi':0.25},
    {'mat':m1 , 'ori':  0  , 'thi':0.25},
]

# Define side layups
layup_sides = [
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori': -45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori': -45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
]

# Define spar layup
layup_spar = [
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori': -45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
    {'mat':m1 , 'ori': -45  , 'thi':0.25},
    {'mat':m1 , 'ori':  45  , 'thi':0.25},
]

# Get effective E-modulus
E_eff_top = effective_laminate_Ex(layup_top)
E_eff_bottom = effective_laminate_Ex(layup_bottom)
E_eff_sides = effective_laminate_Ex(layup_sides)
E_eff_spar = effective_laminate_Ex(layup_spar)

# Define custom profile
outer_height = 75
outer_width = 200
o1 = 10
o2 = 40


profile1 = Profile()

# TOP
t_top = laminateThickness(layup_top)
profile1.add_section(z_center=(outer_height-0.5*t_top),
                     b=outer_width,
                     h=t_top,
                     E_modulus=E_eff_top)
# BOTTOM
t_bottom = laminateThickness(layup_bottom)
profile1.add_section(z_center=(0.5*t_bottom),
                     b=(outer_width-2*o2),
                     h=t_bottom,
                     E_modulus=E_eff_bottom)

# SLANTED SIDE 1
t_side = laminateThickness(layup_sides)
length_slanted_side = np.sqrt((o2-o1)**2 + outer_height**2)
angle = math.degrees(np.arctan((o2-o1) / outer_height))
profile1.add_section(z_center=0.5*outer_height,
                     b=t_side,
                     h=length_slanted_side,
                     E_modulus=E_eff_sides,
                     rotation_deg=angle)

# SLANTED SIDE 2
profile1.add_section(z_center=0.5*outer_height,
                     b=t_side,
                     h=length_slanted_side,
                     E_modulus=E_eff_sides,
                     rotation_deg=angle)

# SPAR
t_spar = laminateThickness(layup_sides)
profile1.add_section(z_center=0.5*outer_height,
                     b=t_spar,
                     h=outer_height,
                     E_modulus=E_eff_spar)

# EFFECTIVE BENDING STIFFNESS
Db = profile1.compute_bending_stiffness()
```

```
calculate_critical_disp(wheel_base=1200, F1=(50*9.81), F2=(50*9.81), L=3600 , BendingStiffness=Db)
print_weigh_estimate(cross_section_area=profile1.calculate_cross_section_ares())
```

```
Max displacement: 43.99739524011364
Total weight per bridge component: 6.30 kg
```