**Proceedings of the ASME 2022**
**International Manufacturing Science and Engineering Conference**
**MSEC2022**
**June 27 - July 1, 2022, IN, USA**

**MSEC2022-85981**

# ISTRUCTURE: AN OPEN SOURCE STRUCTURAL DESIGN FRAMEWORK

**Juan David Argüello Plata, Octavio Andrés González Estrada**
Universidad Industrial de Santander

## ABSTRACT

Artificial Intelligence (AI) can be defined as a developed technology which has come to supplant cyclic tasks in different work environments. It allows the optimization of operational costs and productivity. Thanks to it, there have been published several studies in different areas: as an evaluation procedure to define the probability of guilty in legal processes; in clinical studies to develop and simulate new drug composition, and to predict electrocardiogram behaviour of heart problem patients; in the automotive industry, in the automation of production systems and tasks, due to the product variants and complexity, leading to simplify manufacturing processes;and in the design of photovoltaic systems in modelling, sizing, control and diagnosis stages. One of the most affected, and challenging, industries around the world is the civil industry, principally at the design stage. To optimize the process of design, structural and cost analysis, we created the iStructure project, which is a manufacturing tool that automate the design of structural members. It is a python-based framework library that, in its first version, can automate the design procedure of Mezzanine Floor Racking Systems. It allows the user to: specify the cross-section geometries of the structural members and predicts its geometric properties by the Finite Element Method – FEM; define the modular area distribution per floor; select the minimum cost cross-sections of the structural members which can resist the specified load conditions; create an automatic report, in LaTeX/PDF, which illustrates the design procedure (according to ANSI MH16.1, AISI S100 and ASCE 7-16 North American standards) and the costs of the structure; and elaborates automatic 3D CAD plans in DXF format.

Keywords: Artificial Intelligence, Neural Networks, FEM, structural design, Mezzanine Floor Racking Systems, Open Source.

## INTRODUCTION

Automation in engineering is widely applied to save both time and money. The construction industry is falling behind others in terms of making productivity gains [**?**]. Various reasons have been invoked to explain negative productivity trends, such as shifts within construction, increases in land-use regulation and the use of questionable deflators [?].

Unlike concrete buildings and bridges, metallic structures does not require special architectural design because its purpose is basically to increase the storage area of goods and raw materials of different industries in warehouses. Thanks to it, there is an innovation opportunity in the automation of the design stage of metallic structures. The present work focus on Mezzanine Floor Racking Systems and in the explanation of the logic behind a structural design framework, made by the authors, in Python - Jupyter, which starts by asking the user the area of the structure and load conditions. Then, it selects the structural members (beams, columns, and joists), evaluated by the FEM and selected by a defined design procedure, based on AISI S-100-16, ANSI MH16.1 and ASCE 7-16 North American standards and NSR-10 Colombian standard for seismic evaluation of structures installed in Colombia. Finally, the results are an economic report, an engineering design report and the CAD draws, 2D and 3D, of the structure.

 

## 1. The framework

iStructure is an *Open Source* framework that has the module organization shown on Figure 1.

```
iStructure
├── Architect
│   ├── CADrawings
│   │   └── Mezzanine.py
│   └── Schemes
│       ├── Areas.py
│       └── CrossSection.py
├── CrossSection
│   ├── GenericMaterials.py
│   └── MechProp.py
├── Reports
│   └── Mezzanine.py
└── StructuralDesign
    ├── Mezzanine
    │   ├── Members.py
    │   └── Results.py
    └── StructuralMember.py
```

**FIGURA 1**: iStructure modules.

The project is divided by the modules: Architect, CrossSection, Reports and StructuralDesign. As mentioned before, this first version is focused on Mezzanine Floor Racking Systems.

## 2. The structure

Mezzanine Floor Racking Systems are modular structures which consist of beams, columns and joists, as shown on Figure 2.

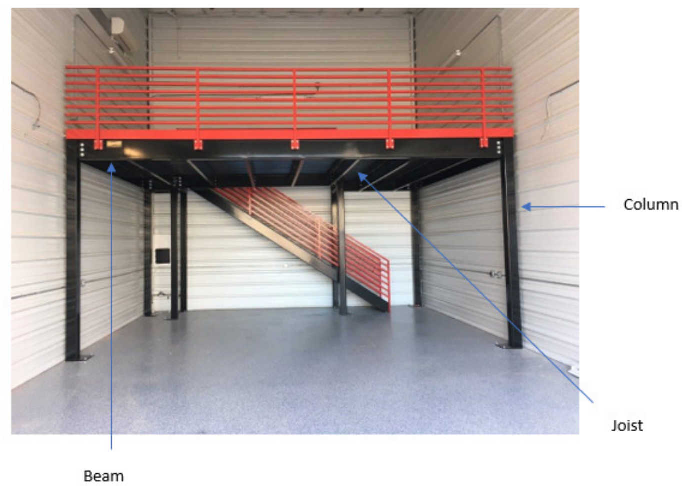


**FIGURA 2**: Mezzanine parts. Source: http://www.firststorage.co.za

The resistance of a structural member rely on the material, geometry of the section and the length of the member. By default, the framework is programmed to work with steel ASTM A1011 SS Grade 36/2, which has the properties shown on Table 1.

| Parameter | Value |
|---|---|
| Density $[kg/m^3]$ | 7850 |
| Young Modulus $[GPa]$ | 200 |
| Coefficient of Poisson | 0.3 |
| Yield strength $[MPa]$ | 248.25 |
| Ultimate strength $[Mpa]$ | 440 |

**CUADRO 1**: Strength properties.

Other materials can be defined and selected by the user, even composite cross sections can be implemented.

## 3. Cross Sections

To predict the cross section properties, iStructure implements the *section properties* Open Source library [?]. With the material of the structural members chosen, the next step should be to establish a database of structural profiles. By default, the software use square sections for columns, I section for beams and C section for joists. Other kinds of sections are also supported, including user custom sections. For example, for a joist, Cee section, you simply specify the dimensions, as shown on Listing 1.

```python
from sectionproperties.pre.sections import
    CeeSection
cee = CeeSection(d=120, b=60, l=15, t=4, r_out
    =5.4, n_r=8)
cee.plot_geometry()
```

Listing 1: Cee section definition

Cee cross section scheme can be appreciated on figure 3.

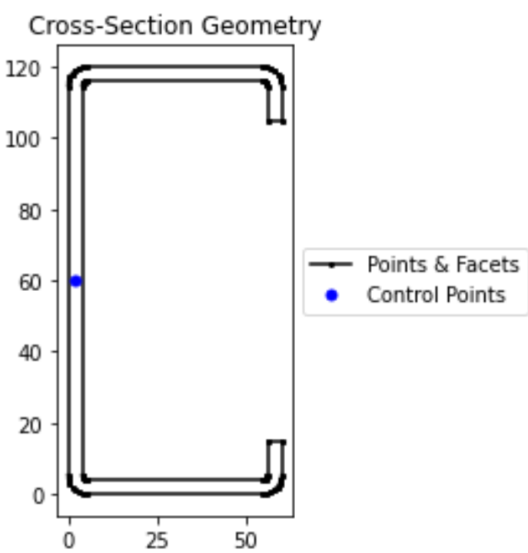


**FIGURA 3**: Cee section.

Geometric, plastic and warping analysis is developed by a Finite Element Method simulation of the cross section. We start by applying the discretization of the domain.

```python
from iStructure.CrossSection.MechProp import
    CrossSection
from iStructure.CrossSection.GenericMaterials
    import steel
mesh = cee.create_mesh(mesh_sizes=[3.0])
ceeSection = CrossSection(cee, mesh, [steel])
ceeSection.display_mesh_info()
```

Listing 2: Cee section definition

With this configuration, a regular tetrahedral mesh, with elements of $3[mm]$ has been developed and can be appreciated next.

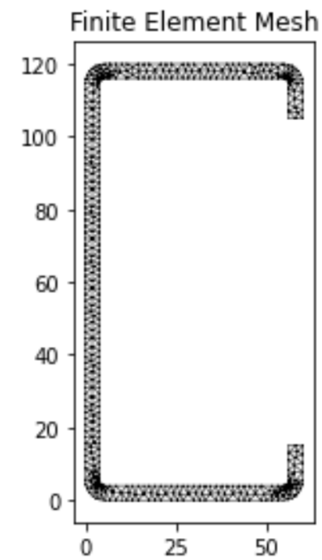


**FIGURA 4**: Cee section mesh.

## 3.1. Cross section properties and FEM formulation

**3.1.1. Problem statement** The geometrical and wrap properties are calculated by the Finite Element Method. For the 2D linear elasticity problem, the unknown displacement field $u$, taking values in $\Omega \subset \mathbb{R}^2$, is the solution of the boundary value problem, given by:

$$-\nabla \cdot \sigma(u) = b \qquad \text{in } \Omega \tag{1}$$

$$\sigma(u) \cdot n = t \qquad \text{on } \Gamma_N \tag{2}$$

$$u = 0 \qquad \text{on } \Gamma_D \tag{3}$$

Where $\Gamma_N$ and $\Gamma_D$ denote the Neumann and Dirichlet boundaries with $\partial\Omega = \Gamma_N \cup \Gamma_D$ and $\Gamma_N \cap \Gamma_D = \emptyset$. The Dirichlet boundary condition in (3) is assumed to be homogeneous. The weak form of the problem reads: Find $u \in V$ sucha that

$$\forall v \in V \qquad a(u,v) = l(v) \tag{4}$$

where V is the standard test space for the elasticity problem such that $V = v|v$, and

$$a(u,v) = \int_{\Omega} \epsilon(u)^T D \epsilon(v) d\Omega = \int_{\Omega} \sigma(u)^T D^{-1} \sigma(v) d\Omega \quad (5)$$

$$l(v) := \int_{\Omega} b^T v d\Omega + \int_{\Gamma_N} t^T v d\Gamma \quad (6)$$

where $D$ is the elasticity matrix of the constitutive relation $\sigma = D\epsilon$, $\sigma$ and $\epsilon$ denote the stress and strain operators.

#### 3.1.2. Finite element formulation

Let $u^h$ be a finite element approximation to $u$. The solution lies in a functional space $V^h \subset V$ associated with a mesh of isoparametric finite elements of characteristic size h, which is defined by equation (3).

Using a variational formulation of the problem (1-3) and a finite element approximation $u^h = N u^e$, where $N$ denotes the shape functions of order $p$, we obtain a system of linear equations to solve the displacements at nodes $u^e$:

$$KU = f \quad (7)$$

where $K$ is the stiffness matrix, $U$ is the vector of nodal displacements and $f$ is the load vector.

### 3.2. NN mesh validation

For mesh validation error, an internal neural network algorithm has been implemented.

## 4. Area distribution

The modular distribution of the structure is automated by an algorithm from which the user can interact directly by telling the software the general dimensions (width, height and length), separation between joists, modular subdivisions and load distribution. Moreover, it was designed to be flexible enough to adapt for complex geometries. For example, lets suppose that a client has a warehouse of six by six meters of area and wants a structure with three meters of height which has to support 700 $[kg/m^2]$, but the building has a column in the middle of it with square dimensions of 0,5$[m]$. A possible solution for this problem can be appreciated on Figure 5.

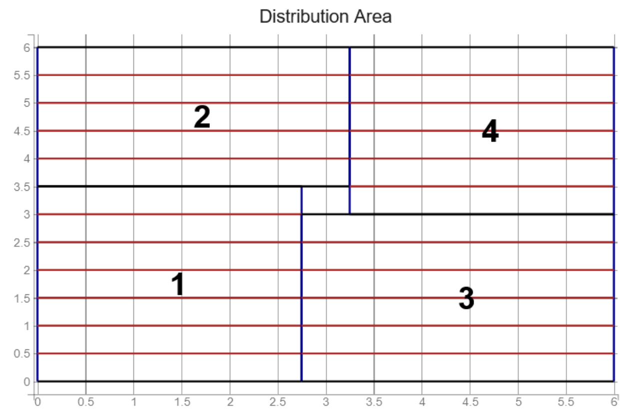


**FIGURA 5**: Area distribution of study.

As seen on Figure 6, an algorithm of *Object Oriented Programming* (OOP) has been implemented to the automatic distribution of the area, using RAM memory[1] to store the dimensions of each module, in this case: 1, 2, 3 and 4. An object of the module was created which uses $x$ and $y$ position to know its position in the space, $L$ and $W$ parameters referring to *length* and *width*, respectively, and a 'Delete' checkbox to erase the module (not used in this example).

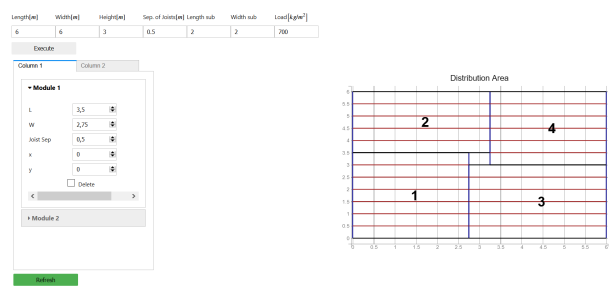

**FIGURA 6**: User interface of the area distribution.

## 5. Structural members selection procedure

To start the structural members selection, it is needed to be known the longest dimensions of each member. It can be achieved by implementing an algorithm which compares all the dimensions of each module. All of this information has been stored in a Python dictionary.

[1]Off course, this can be changed to be stored on the HDD memory rather than RAM memory, if needed.

## 5.1. Load study

As defined on AISI S100-16 standard, load distribution can be segmented in *dead*, *live*, *product* and *combined* loads.

**5.1.1. Dead load** It consists of the weight of the structural members. It depends on the densisty of the material, the cross sectional area and the length of the member.

$$D = A\rho L g \tag{8}$$

From Equation (8), $D$ correspond to the value of the dead load, $A$ to the cross sectional area, $L$ to the length of the member and $g$ to gravity's acceleration.

**5.1.2. Live load** The distirbuted live load for the food traffic on pick module walkways should be, at least, of $293\left[kg/m^2\right]$, according to the standard ANSI MH16, section 8.4.2. The live load can be calculated using Equation. (9).

$$L = L_{dist} A g \tag{9}$$

Where: $L$ is the live load, $L_{dist}$ is the distributed load, $A$ correspond to the area of the biggest module and $g$ to the gravity.

**5.1.3. Product load** The product load correspond to the difference between the design distributed load, the live load and the dead load.

**5.1.4. Combined load** The combined load is used to determine the stresses of the structural members. There was used a Load and Resistance Factor Design (LRFD, ANSI MH16.1 section 2.1) as seen on Equation (10).

$$CL = 1{,}2D + 1{,}4P + 1{,}6L \tag{10}$$

Where: $CL$ is the combined load, $D$ is the dead load (Eq. (8)), $P$ is the product load and $L$ is the live load (Equation (9)).

## 5.2. Joists

Structural design of joists is evaluated based on the maximum momentum compared *nominal resistance momentum* (section 3.3.2.2.2 of AISI S100-16 standard).

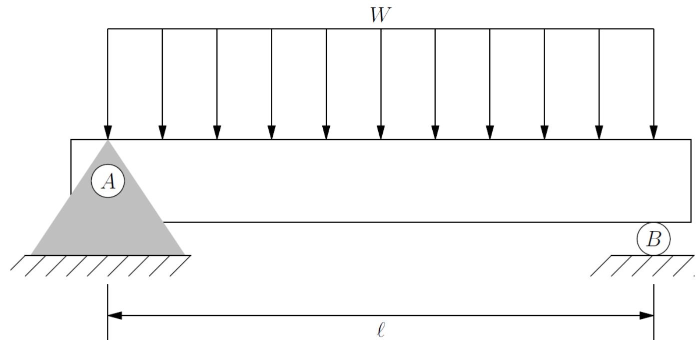


**FIGURA 7**: Load distribution over a joist.

As it can be appreciated on Figure 7, the calculation proccess starts by the supposition of a simply supported beam. The distributed load $W$ correspond to the combined load, from Equation (10). The momentum behaviour over the joist is defined by Equation (11).

$$M(x) = \frac{Wx(L-x)}{2} \tag{11}$$

According to the Equation (11), the maximum momentum is:

$$M = \frac{WL^2}{8} \tag{12}$$

**5.2.1. Selection** Joist selection is developed by each of the structural profile in the database. For each of them is evaluated and compared both maximum momentum and nominal resistance momentum. The final selection is to the member with the lowest cross sectional area, for this example the summary table is shown on Figure 8.

From Table 8, the selected member is 'MZVA23414 (2.0 mm)'.

## 5.3. Beams

The structural evaluation procedure is the same as joists (section 3.2).

## 5.4. Columns

Unlike beams and joists, columns are multiaxial load members. When the column of a structure fails it is mainly because of flexural-torsional buckling stresses ('large' columns) or axial stresses ('short' columns). The procedure starts by evaluating compression load (axial stress) and

```
[9]: from App.Mezzanine.Calculus.Members import Selection

JoistSelec = Selection(
    Calculus['Combined_Loads'],
    Calculus['Init'], Cross_Sections, "Joists")
Calculus['Selection'] = JoistSelec()
df = JoistSelec.win
df.style.apply(lambda x: ['background: lightblue' if x.name == JoistSelec.winner[1] else '' for i in x], axis=1)
```

[9]:

| | Reference | Satisfactory? |
|---|---|---|
| 0 | MZVA20412 (2.5mm) | O |
| 1 | MZVA23412 (2.5mm) | O |
| 2 | MZVA23414 (2.0mm) | O |
| 3 | MZVA26012 (2.5mm) | O |
| 4 | MZVA36012 (2.5mm) | O |
| 5 | MZVA36014 (2.0mm) | O |
| 6 | MZVA20414 (2.0mm) | X |
| 7 | MZVA20416 (1.5mm) | X |
| 8 | MZVA23416 (1.5mm) | X |

**FIGURA 8**: Selection table.

```
[20]: BeamSelec = Selection(
    Calculus['Combined_Loads'],
    Calculus['Init'], Cross_Sections, "Beams")
Calculus['Selection'] = BeamSelec()
df = BeamSelec.win
df.style.apply(lambda x: ['background: lightgreen' if x.name == BeamSelec.winner[1] else '' for i in x], axis=1)
```

[20]:

| | Reference | Satisfactory? |
|---|---|---|
| 0 | MZV40012 (2.5mm) | O |
| 1 | MZV26612 (2.5mm) | X |
| 2 | MZV26614 (2.0mm) | X |
| 3 | MZV29612 (2.5mm) | X |
| 4 | MZV29616 (1.5mm) | X |
| 5 | MZV35012( 2.5mm) | X |
| 6 | MZV35014 (2.0mm) | X |
| 7 | MZV40014 (2.0mm) | X |

**FIGURA 9**: Selection table for beams.

compare it with the nominal load (result of the defined procedure given by the AISI S100-16 standard) for each option of the database. If the nominal load has a bigger value than the calculated compression load, then it can be concluded that axial stress is not critical so the momentum due to flexural-torsional buckling should be compared with nominal momentum (defined by AISI S100-16 standard) to discard failure by buckling stresses. The results for the given example is shown on Figure ??.

From Figure 10, the column of reference ‘MZC10214’ has been selected.

## ACKNOWLEDGMENT

This creation was supported by Universidad Industrial de Santander (UIS). We thank our collegues from UIS who provided insight and expertise that greatly assisted this work.

[23]:

| | Reference | Satisfactory? |
|---|---|---|
| 0 | MZC10212 (2.5mm) | O |
| 1 | MZC10214 (2.0mm) | O |
| 2 | MZC13612 (2.5mm) | O |
| 3 | MZC13614 (2.0mm) | O |
| 4 | MZC17212 (2.5mm) | O |
| 5 | MZC18012 (2.5mm) | O |
| 6 | MZC18014 (2.0mm) | O |
| 7 | MZC10216 (1.5mm) | X |
| 8 | MZC13616 (1.5mm) | X |

**FIGURA 10**: Column selection table.