Output

Contents

class `openstaadpy.os_analytical.osoutput.OSOutput` [source]

Bases: `object`

`AreResultsAvailable()` [source]

Check if analysis results are available or not.

Returns:

True if results are available, False otherwise.

Return type:

bool

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> are_results_available = staad_obj.Output.AreResultsAvailable()
```

`GetAllPlateCenterForces(plateNo: int, loadCaseNo: int)` [source]

Get the plate center stresses (Shear & Membrane) for the specified plate for specified load case.

Parameters:

- **plateNo** (*int*) – The plate number.

- **loadCaseNo** (*int*) – The load case number.

Returns:

list of plate center forces organized in following order:

List Index	Variable	Load Type
0	SQX	Shear stress on the local X face in the Z direction
1	SQY	Shear stress on the local Y face in the Z direction
2	SX	Axial stress in the local X direction
3	SY	Axial stress in the local Y direction
4	SXY	Shear stress in the local XY plane

Note :

- For additional information, please refer to Section: “Sign Convention of Plate Element Stresses and Moments” and Section 5.42 of the Technical Reference manual.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plateList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plateCenterForces = staad_obj.Output.GetAllPlateCenterForces(plateL
```

`GetAllPlateCenterMoments(plateNo: int, loadCaseNo: int)` [source]

Get the plate center stresses (Shear & Membrane) for the specified plate for specified load case.

Parameters:

- **plateNo** (*int*) – The plate number.
- **loadCaseNo** (*int*) – The load case number.

Returns:

list of plate center moments organized in following order:

List Index	Variable	Load Type
0	MX	Moment per unit width about the local X face
1	MY	Moment per unit width about the local Y face
2	MXY	Torsional Moment per unit width in the local X-Y plane

Note :

- For additional information, please refer to Section: “Sign Convention of Plate Element Stresses and Moments” and Section 5.42 of the Technical Reference manual.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plateList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plateCenterMoments = staad_obj.Output.GetAllPlateCenterMoments(plat
```

GetAllPlateCenterPrincipalStressesAndAngles(*plateNo: int*, *loadCaseNo: int*) [source]

Get all plate center principal stresses and angles.

Parameters:

- **plateNo** (*int*) – Plate number ID.
- **loadCaseNo** (*int*) – Case reference ID.

Returns:

List of float values orgainized according to below table:

Variable	Description
pdStresses[0]	Top-Maximum in-plane Principal stress
pdStresses[1]	Top-Minimum in-plane Principal stress
pdStresses[2]	Top-Maximum in-plane Shear stress
pdStresses[3]	Bottom-Maximum in-plane Principal stress
pdStresses[4]	Bottom-Minimum in-plane Principal stress
pdStresses[5]	Bottom-Maximum in-plane Shear stress
pdStresses[6]	Top-Angle which determines direction of maximum principal stress with respect to local X axis
pdStresses[7]	Bottom-Angle which determines direction of maximum principal stress with respect to local X axis

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plateList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plateCenterPrincipalStressesAndAngles = staad_obj.Output.GetAllPlat
```

GetAllPlateCenterPrincipalStressesAndAnglesEx(*plateNo: int, loadCaseNo: int*) [source]

Get all plate center principal stresses and angles (extended).

Parameters:

- **plateNo** (*int*) – Plate number ID.
- **loadCaseNo** (*int*) – Load case number ID.

Returns:

Tuple of principal stresses values list and angle value list. They are organized following way

Principal Stresses list :

Variable	Description
pdStresses[0]	Top-Maximum in-plane Principal stress
pdStresses[1]	Top-Minimum in-plane Principal stress
pdStresses[2]	Top-Maximum in-plane Shear stress
pdStresses[3]	Bottom-Maximum in-plane Principal stress
pdStresses[4]	Bottom-Minimum in-plane Principal stress
pdStresses[5]	Bottom-Maximum in-plane Shear stress

Angles List :

Variable	Description
pdAngles[0]	Top-Angle which determines direction of maximum principal stress with respect to local X axis
pdAngles[1]	Bottom-Angle which determines direction of maximum principal stress with respect to local X axis

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> principal_stress_list, angle_list = staad_obj.Output.GetAllPlateCen
```

GetAllPlateCenterStressesAndMoments(*plateNo: int*, *loadCaseNo: int*) [source]

Gets plate center stresses and moments for the specified plate for specified load case.

Parameters:

- **plateNo** (*int*) – Plate number.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

list of plate center stresses and moments organized in following order:

List Index	Variable	Load Type
0	SQX	Shear stress on the local X face in the Z direction
1	SQY	Shear stress on the local Y face in the Z direction
2	MX	Moment per unit width about the local X face
3	MY	Moment per unit width about the local Y face
4	MXY	Torsional Moment per unit width in the local X-Y plane
5	SX	Axial stress in the local X direction
6	SY	Axial stress in the local Y direction
7	SXY	Shear stress in the local XY plane

Note :

- For additional information, please refer to Section: “Sign Convention of Plate Element Stresses and Moments” and Section 5.42 of the Technical Reference manual.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plateList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plateCenterStressesAndMoments = staad_obj.Output.GetAllPlateCenterS
```

GetAllSolidNormalStresses(*nSolidNo: int, nCorner: int, loadCaseNo: int*) [source]

Gets all solid normal stresses.

Parameters:

- **nSolidNo** (*int*) – Solid number ID.
- **nCorner** (*int*) – Corner of the solid.
- **loadCaseNo** (*int*) – The load case number.

Returns:

list with solid normal stresses SXX, SYY and SZZ (in order).

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> solidList = staad_obj.Geometry.GetSolidList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> solidNormalStresses = staad_obj.Output.GetAllSolidNormalStresses(so
```

GetAllSolidPrincipalStresses(*nSolidNo: int, nCorner: int, loadCaseNo: int*) [source]

Get all solid principal stresses.

Parameters:

- **nSolidNo** (*int*) – Solid number ID.
- **nCorner** (*int*) – Corner of the solid.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of principal stresses S_1, S_2, S_3 in same order.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> solidList = staad_obj.Geometry.GetSolidList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> principalStresses = staad_obj.Output.GetAllSolidPrincipalStresses(s
```

GetAllSolidShearStresses(*nSolidNo: int, nCorner: int, loadCaseNo: int*) [source]

Get all solid shear stresses.

Parameters:

- **nSolidNo** (*int*) – Solid number ID.
- **nCorner** (*int*) – Corner of the solid.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of shear stresses SXY, SYZ, SZX in same order.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> solidList = staad_obj.Geometry.GetSolidList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> shearStresses = staad_obj.Output.GetAllSolidShearStresses(solidList
```

`GetAllSolidVonMisesStresses(nSolidNo: int, nCorner: int, loadCaseNo: int)` [source]

Get all solid Von Mises stresses.

Parameters:

- **nSolidNo** (*int*) – Solid number ID.
- **nCorner** (*int*) – Corner of the solid.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of Von Mises stresses.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> solidList = staad_obj.Geometry.GetSolidList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> vonMisesStresses = staad_obj.Output.GetAllSolidVonMisesStresses(sol
```

GetBasePressures(*loadCaseNo: int, nodelist: list*) [source]

Gets base presure in X, Y and Z direction using Base Presure command.

Parameters:

- **loadCaseNo** (*int*) – Load Case reference ID.
- **nodelist** (*list*) – List of node numbers.

Returns:

Tuple of list of base pressures in X, Y and Z direction per load case reference ID.

Return type:

tuple of list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> base_pressures_x, base_pressures_y, base_pressures_z = staad_obj.Ou
```

GetBucklingFactor(*buckling_mode_no: int*) [source]

Gets the buckling factor for a specified buckling mode.

Parameters:

buckling_mode_no (*int*) – Buckling mode number

Returns:

Buckling factor.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> buckling_factor = staad_obj.Output.GetBucklingFactor(1)
```

GetBucklingModeDisplacementAtNode(*buckling_mode_no: int, node_no: int*) [source]

Gets the modal displacement at a specified node number and mode.

Parameters:

- **buckling_mode_no** (*int*) – Buckling mode number.
- **node_no** (*int*) – Node number at which buckling analysis result is to be extracted.

Returns:

List of buckling mode displacements.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> buckling_mode_displacement_at_node = staad_obj.Output.GetBucklingMo
```

GetIntermediateDeflectionAtDistance(*memberNo: int, distance: int, loadCaseNo: int*) [source]

Get the intermediate section deflections for specified member number and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **distance** (*float*) – Distance from the starting end of the member.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

Tuple of displacement in Y & Z direction respectively.

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beam_list = staad_obj.Geometry.GetBeamList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> deflections = staad_obj.Output.GetIntermediateDeflectionAtDistance(
```

GetIntermediateMemberAbsTransDisplacements(*memberNo: int, distance: float, loadCaseNo: int*) [source]

Gets section displacement (or relative displacements) of a beam section for specified member number, distance, and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **distance** (*float*) – Distance from starting end in terms of member length.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of relative displacements at specified section in terms of LOCAL X, Y, Z coordinates (in order).

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> beam_list = staad_obj.Geometry.GetBeamList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> intermediate_member_abs_trans_displacements = staad_obj.Output.GetI
```

GetIntermediateMemberForcesAtDistance(*memberNo: int, distance: float, loadCaseNo: int*) [source]

Gets sectional forces and moments for specified member number, distance, and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **distance** (*float*) – Distance from the starting end of the member.

- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of Section axial force, Shear force in LOCAL Y & Z direction, Torsion and Bending moment in Local MY & MZ direction respectively.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> memberForces = staad_obj.Output.GetIntermediateMemberForcesAtDistan
```

`GetIntermediateMemberTransDisplacements`(*memberNo: int*, *distance: int*, *loadCaseNo: int*) [source]

Get section displacement (or relative displacements) of a beam section for specified member number, distance, and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **distance** (*int*) – Distance from starting end in terms of member length.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of relative displacements at specified section in terms of LOCAL X, Y, Z coordinates (in order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> intermediateMemberTransDisplacements = staad_obj.Output.GetIntermed
```

GetMatInfluenceAreas(*nodelist: list*) [source]

Gets the mat influence areas for nodes supported using ELASTIC MAT command.

Parameters:

nodelist (*list*) – List of node numbers.

Returns:

Tuple of lists containing influence areas in YZ, ZX & XY place ordered per node number in node list .

Return type:

tuple of list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> mat_influence_areas = staad_obj.Output.GetMatInfluenceAreas(node_li
```

GetMaxBeamStresses(*beamNo: int*, *loadCaseNo: int*) [source]

Gets the maximum beam Stresses for Beam.

Parameters:

- **beamNo** (*int*) – Beam number ID.
- **loadCaseNo** (*int*) – The load case number.

Returns:

tuple with value of maximum compressive stress, integer value (0 for non-corner-section material (eg. prismatic circle and pipe), 1 to 4 for section corner where maximum compressive stress was found), value of maximum tensile stress and integer value (0 for non-corner-section material (eg. prismatic circle and pipe), 1 to 4 for section corner where maximum tensile stress was found) (in order).

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> maxBeamStresses = staad_obj.Output.GetMaxBeamStresses(beamList[0],
```

GetMaxSectionDisplacement(*memberNo: int, dir: str, loadCaseNo: int*) [source]

Gets the maximum section displacements for specified member number, direction, and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **dir** (*str*) – Direction in GLOBAL: X, Y or Z
- **loadCaseNo** (*int*) – The load case number.

Returns:

tuple with maximum section displacement in specified direction and the location along the length of the member where the maximum section displacement is located (in order).

Return type:

tuple of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> maxSectionDisplacement = staad_obj.Output.GetMaxSectionDisplacement
```

GetMemberDesignSectionName(*beamNo: int*) [source]

Get the design section name for specified member.

Parameters:

beamNo (*int*) – Beam number ID.

Returns:

Returns design section name for the specified member. Returns empty string if not found.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beam_list = staad_obj.Geometry.GetBeamList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> design_section_name = staad_obj.Output.GetMemberDesignSectionName(b
```

GetMemberEndDisplacements(*memberNo: int, end: int, loadCaseNo: int*) [source]

Get the end displacements for a given member.

Parameters:

- **memberNo** (*int*) – The member number.
- **end** (*int*) – The end number (0 for starting and 1 for ending).
- **loadCaseNo** (*int*) – The load case number.

Returns:

List with end displacements of Member End in terms of X, Y, Z (in order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> memberEndDisplacements = staad_obj.Output.GetMemberEndDisplacements
```

GetMemberEndForces(*memberNo: int, end: int, loadCaseNo: int, LocalOrGlobal: int*) [source]

Get the end forces for a given member.

Parameters:

- **memberNo** (*int*) – The member number.
- **end** (*int*) – The end number (0 for starting and 1 for ending).
- **loadCaseNo** (*int*) – The load case number.
- **LocalOrGlobal** (*int*) – Local Or Global direction (0 for Local and 1 for Global).

Returns:

list with end force values FX, FY, FZ, MX, MY and MZ (in order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> memberEndForces = staad_obj.Output.GetMemberEndForces(beamList[0],
```

GetMemberSteelDesignMaxFailureRatio()

[source]

Gets the maximum failure ratio across all beams in the model.

Returns:

The maximum failure ratio.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> member_steel_design_max_failure_ratio = staad_obj.Output.GetMemberS
```

GetMemberSteelDesignMinFailureRatio()

[source]

Gets the minimum failure ratio across all beams in the model.

Returns:

The minimum failure ratio.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> member_steel_design_min_failure_ratio = staad_obj.Output.GetMemberS
```

`GetMemberSteelDesignRatio(beamNo: int)` [source]

Gets the critical steel design ratio for a steel member. This method will return the results from the last parameter block for which the beam has been designed.

Parameters:

beamNo (*int*) – The beam number ID.

Returns:

Returns the critical steel design ratio. Returns -999 if analysis is performed but the member is not designed. Returns -1 if analysis is not performed.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> memberSteelDesignRatio = staad_obj.Output.GetMemberSteelDesignRatio
```

`GetMemberSteelDesignResults(beamNo: int)` [source]

Gets steel design results for the specified member. This method will return the results from the last parameter block for which the beam has been designed.

Parameters:

beamNo (*int*) – Id of the member for which design results should be retrieved.

Returns:

Tuple of steel design results containing the design code name, the design status (pass or fail will be returned), the design utilization ratio, the allowable design utilization ratio, the critical design load case number, the critical section location from the start of member in current base unit, the critical design clause, the design section name, array of size 3 and type double to hold critical section forces. returns fx, my and mz values at 0, 1, & 2 index respectively in current base units and kl/r ratio of the specified member. (in same order)

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> beam_list = staad_obj.Geometry.GetBeamList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> design_code_name, design_status, critical_ratio, allowable_ratio, c
```

GetMinMaxAxialForce(*memberNo: int, loadCaseNo: int*) [source]

Gets the maximum and minimum axial forces and their locations for specified member number and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **loadCaseNo** (*int*) – The load case number.

Returns:

tuple with minimum axial force, the location along the length of the member where the minimum axial force is located, maximum axial force and the location along the length of the member where the maximum axial force is located (in order).

Return type:

tuple of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> minMaxAxialForce = staad_obj.Output.GetMinMaxAxialForce(beamList[0]
```

GetMinMaxBendingMoment(*memberNo: int, dir: str, loadCaseNo: int*) [source]

Gets the maximum and minimum bending moments and their locations for specified member number, load case, and bending direction.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **dir** (*str*) – Bending direction in LOCAL coordinate: MY or MZ.
- **loadCaseNo** (*int*) – The load case number.

Returns:

tuple with minimum bending moment, the location along the length of the member where the minimum bending moment is located, maximum bending moment and the location along the length of the member where the maximum bending moment is located (in order).

Return type:

tuple of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> minMaxBendingMoment = staad_obj.Output.GetMinMaxBendingMoment(beamL
```

`GetMinMaxShearForce(memberNo: int, dir: str, loadCaseNo: int)`

Gets the maximum and minimum shear forces and their locations for specified member number, load case, and force direction. [source]

Parameters:

- **memberNo** (*int*) – Member number ID.
- **dir** (*str*) – Force direction in LOCAL coordinate: FY or FZ.
- **loadCaseNo** (*int*) – The load case number.

Returns:

tuple with minimum bending shear, the location along the length of the member where the minimum bending shear is located, maximum bending shear and the location along the length of the member where the maximum bending shear is located (in order).

Return type:

tuple of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetBeamList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> minMaxShearForce = staad_obj.Output.GetMinMaxShearForce(beamList[0]
```

GetModalDisplacementAtNode(*modeNo: int*, *nodeNo: int*) [source]

Gets the modal displacement at a specified node number and mode.

Parameters:

- **modeNo** (*int*) – Mode number.
- **nodeNo** (*int*) – Node number ID.

Returns:

List of nodal displacements.

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> mode_displacement_at_node = staad_obj.Output.GetModalDisplacementAt
```

GetModalMassParticipationFactors(*modeNo: int*) [source]

Gets the modal participation factors for a specified mode number.

Parameters:

modeNo (*int*) – Mode number.

Returns:

Tuple of modal mass participation factor for X, Y & Z direction respectively.

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> mode_mass_participation_factors = staad_obj.Output.GetModalMassPart
```

GetModeFrequency(*modeNo: int*) [source]

Get the natural frequency (Hz) for a specified mode.

Parameters:

modeNo (*int*) – The mode number.

Returns:

Natural Frequency value for a specified mode.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> mode_frequency = staad_obj.Output.GetModeFrequency(4)
```

GetMultipleMemberSteelDesignMaxRatio(*beamNo: int*) [source]

Gets the maximum critical steel design ratio across all parameter blocks for a steel member.

Parameters:

beamNo (*int*) – Beam number ID.

Returns:

Returns the maximum critical steel design ratio.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> beam_list = staad_obj.Geometry.GetBeamList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> multiple_member_steel_design_max_ratio = staad_obj.Output.GetMultip
```

`GetMultipleMemberSteelDesignRatio(param_blk_name: str, beam_no: int)` [source]

Gets the critical steel design ratio for a steel member. This function is for AISC 360-16 code only.

Parameters:

- **param_blk_name** (*str*) – Steel design parameter block name.
- **beam_no** (*int*) – Beam number ID.

Returns:

Returns the critical steel design ratio.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> countOfParamBlocks = staad_obj.Output.GetSteelDesignParameterBlockC
>>> if (countOfParamBlocks > 0) :
>>>     param_blk_name = staad_obj.Output.GetSteelDesignParameterBlockN
>>>     beam_list = staad_obj.Geometry.GetBeamList()
>>>     staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>>     multiple_member_steel_design_ratio = staad_obj.Output.GetMultip
```

GetMultipleMemberSteelDesignResults(*param_blk_name: str, beam_no: int*) [source]

Gets the critical steel design result information for a steel member.

Parameters:

- **param_blk_name** (*str*) – Steel design parameter block name.
- **beam_no** (*int*) – Beam number ID.

Returns:

Tuple consisting of the design code name, the design status (pass or fail will be returned), the design utilization ratio., the allowable design utilization ratio., the critical design load case number., the critical design clause.and the design section name. (in same order)

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> countOfParamBlocks = staad_obj.Output.GetSteelDesignParameterBlockC
>>> if (countOfParamBlocks > 0) :
>>>     param_blk_name = staad_obj.Output.GetSteelDesignParameterBlockN
>>>     beam_list = staad_obj.Geometry.GetBeamList()
>>>     staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>>     multiple_member_steel_design_results = staad_obj.Output.GetMult
```

GetNLLoadStep(*LoadCaseNo: int*) [source]

Gets the Load Step value used for nonlinear analysis for specified Load Case.

Parameters:

loadCaseNo (*int*) – Load Case reference ID.

Returns:

Returns Load Step value. Returns -1 General error. Returns -8002 Load Case nLC not found. Returns -9911 Nonlinear result set is not available.

Return type:

int

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> load_step_value = staad_obj.Output.GetNLLoadStep(load_cases[0])
```

GetNLNodeDisplacements(*nodeNo: int, LoadCaseNo: int, LoadStep: int*) [source]

Get the Load Level value and nodal displacements for specified node, Load Case and Load Step.

Parameters:

- **nodeNo** (*int*) – Node number ID.
- **loadCaseNo** (*int*) – Load Case reference ID.
- **loadStep** (*int*) – Load step number.

Returns:

Tuple containing value of load level and list of nodal displacements in Global (X, Y, Z, rX, rY, rZ) in same order.

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> nl_node_displacement = staad_obj.Output.GetNLNodeDisplacements(node
```

`GetNoOfBucklingFactors()` [source]

Gets the number of buckling factors computed.

Returns:

The number of buckling factor(s) extracted by the eigen analysis.

Return type:

int

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> number_of_buckling_factors = staad_obj.Output.GetNoOfBucklingFactor
```

`GetNoOfModesExtracted()` [source]

Gets the number of modes extracted by a dynamic analysis.

Returns:

Number of modes extracted by a dynamic analysis.

Return type:

int

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> no_of_modes_extracted = staad_obj.Output.GetNoOfModesExtracted()
```

`GetNodeDisplacements(nodeNo: int, loadCaseNo: int)` [source]

Get the displacements for a given node.

Parameters:

- **nodeNo** (*int*) – The node number.
- **loadCaseNo** (*int*) – The load case number.

Returns:

List with nodal translational displacements in X, Y and Z directions, rotation about X, Y and Z directions, respectively

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> nodeList = staad_obj.Geometry.GetNodeList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 1, 1)
>>>
>>> displacements = staad_obj.Output.GetNodeDisplacements(nodeList[3],
```

`GetOutputUnitForDensity()` [source]

Get the output unit for density.

Returns:

The unit for density.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForDensity()
```

`GetOutputUnitForDimension()` [source]

Get the output unit for dimension.

Returns:

The unit for dimension.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForDimension()
```

`GetOutputUnitForDisplacement()` [source]

Get the output unit for displacement.

Returns:

The unit for displacement.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForDisplacement()
```

GetOutputUnitForDistForce() [source]

Get the output unit for distributed force.

Returns:

The unit for distributed force.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForDistForce()
```

GetOutputUnitForDistMoment() [source]

Get the output unit for distributed moment.

Returns:

The unit for distributed moment.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForDistMoment()
```

GetOutputUnitForForce() [source]

Get the output unit for force.

Returns:

The unit for force.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForForce()
```

GetOutputUnitForMoment() [source]

Get the output unit for moment.

Returns:

The unit for moment.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForMoment()
```

GetOutputUnitForRotation() [source]

Get the output unit for rotation.

Returns:

The unit for rotation.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForRotation()
```

GetOutputUnitForSectArea() [source]

Get the output unit for section area.

Returns:

The unit for section area.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForSectArea()
```

GetOutputUnitForSectDimension() [source]

Get the output unit for section dimension.

Returns:

The unit for section dimension.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForSectDimension()
```

GetOutputUnitForSectInertia() [source]

Get the output unit for section inertia.

Returns:

The unit for section inertia.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForSectInertia()
```

GetOutputUnitForSectModulus() [source]

Get the output unit for section modulus.

Returns:

The unit for section modulus.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForSectModulus()
```

GetOutputUnitForStress() [source]

Get the output unit for stress.

Returns:

The unit for stress.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> unit = staad_obj.Output.GetOutputUnitForStress()
```

GetPMemberEndForces(*memberNo: int, end: int, loadCaseNo: int, LocalOrGlobal: int*) [source]

Gets member end forces for specified physical member number, member end and load case.

Parameters:

- **memberNo** (*int*) – Member number ID.
- **end** (*int*) – Member End (0 for starting and 1 for ending).
- **loadCaseNo** (*int*) – Load Case reference ID.
- **LocalOrGlobal** (*int*) – Results returned in either local or global axes. 0= Local, 1= Global direction.

Returns:

List of force of Member End in LOCAL coordinates in terms of FX, FY, FZ, MX, MY and MZ (in order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> p_member_end_forces = staad_obj.Output.GetPMemberEndForces(plate_li
```

GetPMemberIntermediateForcesAtDistance(*memberNo: int, distance: int, loadCaseNo: int*) [source]

Gets sectional forces and moments for specified physical member number, distance, and load case.

Parameters:

- **memberNo** (*int*) – Physical Member number ID.
- **distance** (*int*) – Distance from the starting end of the member.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

List of 6 elements consisting of Section axial force, Shear force in LOCAL Y & Z direction, Torsion and Bending moment in Local MY & MZ direction.

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> p_member_end_forces = staad_obj.Output.GetPMemberIntermediateForces
```

GetPlateCenterNormalPrincipalStresses(*plateNo: int, loadCaseNo: int*) [source]

Get principal stresses of specified plate.

Parameters:

- **plateNo** (*int*) – The plate number.
- **loadCaseNo** (*int*) – The load case number.

Returns:

Tuple containing maximum in-plane Principal Stress at Top surface of plate, minimum in plane Principal Stress at Top surface of plate, maximum in-plane Principal Stress at Bottom surface of plate and minimum in-plane Principal Stress at Bottom surface of plate respectively.

Return type:

Tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> beamList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plateCenterNormalPrincipalStresses = staad_obj.Output.GetPlateCente
```

GetPlateCenterVonMisesStresses(*plateNo: int*, *loadCaseNo: int*)

Gets Von Mises stresses at center of specified plate for specified load case. [source]

Parameters:

- **plateNo** (*int*) – Plate number ID.
- **loadCaseNo** (*int*) – Load Case reference ID.

Returns:

Tuple of Von Mises stress on the top surface of the plate and Von Mises stress on the bottom surface of the plate respectively

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plateList = staad_obj.Geometry.GetPlateList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> vonMisesStresses = staad_obj.Output.GetPlateCenterVonMisesStresses(
```

GetPlateCornerForces(*plateNo: int*, *cornerCode: int*, *loadCaseNo: int*) [source]

Get nodal forces at 4 corners of specified plate at load case.

Parameters:

- **plateNo** (*int*) – Plate number ID.
- **cornerCode** (*int*) – Corner Node No.
- **loadCaseNo** (*int*) – Load case number.

Returns:

List of nodal forces at 4 corners of specified plate at load case.

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> forces = staad_obj.Output.GetPlateCornerForces(plate_list[0], 1, lo
```

`GetPlateStressAtPoint(plateNo: int, loadNo: int, stressPoint: list, facingPoint: int)` [source]

Get stresses values at a point on a specified plate.

Parameters:

- **plateNo** (*int*) – Number of the plate.
- **loadNo** (*int*) – Load number for which stress is requested
- **stressPoint** (*list*) – The coordinate at which the stress is required in global axes as an array size of 3 doubles. The values of x, y, z values of the point defined in the array index (0), (1) and (2).
- **facingPoint** (*int*) – x, y, z values of the facing node at indexes 0, 1 and 2. API always expects an array size of 3 doubles. Definition of facingPoint: It is the node which sits on the tip of a vector which is orthogonal to the vector stressPoint -> facingPoint and lies in the same plane as that of the plates through which the cut line passes.

Returns:

List of stresses values at a point on a specified plate containing following 32 values:

Array Index	stress Type
0	None
1	MaxAbs
2	TopMax
3	TopMin
4	TopTauMax
5	BotMax
6	BotMin
7	BotTauMax
8	MaxVM
9	VMTopMax
10	VMBotMax
11	MaxTresca
12	TopTresca
13	BotTresca
14	FX
15	FY
16	FXY
17	MX
18	MY
19	MZ
20	QX

Array Index	stress Type
21	QY
22	Global
23	GlobalMembraneStresses
24	GlobalshearStresses
25	BasePres
26	CombXTop
27	CombYTop
28	CombXYTop
29	CombXBot
30	CombYBot
31	CombXYBot

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> surface_name = "WALL 1"
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> plate_stress_at_point = staad_obj.Output.GetPlateStressAtPoint(plat
```

GetResultantForceAlongLineForParametricSurface(*parametricSurfaceName: str, nplates: int, loadId: int, startNode: list, endNode: list, facingNode: list, isTransformForceToGlobal: int, firstNode: int, secondNode: int, thirdNode: int*) [source]

Gets forces and moments along the cut line for a particular load case.

Parameters:

- **parametricSurfaceName** (*str*) – Name of the parametric surface.
- **nplates** (*int*) – Number of plates in plateList.
- **loadId** (*int*) – The load case for plate analysis.
- **startNode** (*list*) – List of x, y, z values of the start node at indexes 0, 1 and 2.
- **endNode** (*list*) – List of x, y, z values of the end node at indexes 0, 1 and 2.
- **facingNode** (*list*) – List of x, y, z values of the facing node at indexes 0, 1 and 2.
- **isTransformForceToGlobal** (*int*) – 1: return force in Global System, 0: return forces in local system of cut line.
- **firstNode** (*int*) – Node no representing the origin point for building the surface axis system.
- **secondNode** (*int*) – Node no representing the second point for building the surface axis system.
- **thirdNode** (*int*) – Node no representing the third point for building the surface axis system.

Returns:

List of resultant forces consisting Fx, Fy, Fz, Mx, My, Mz (in same order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> surface_name = "WALL 1"
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> result = staad_obj.Output.GetResultantForceAlongLineForParametricSu
```

GetResultantForceAlongLineForPlateList(*plateList: list, nplates: int, loadIdList: list, startNode: list, endNode: list, isTransformForceToGlobal: int, firstNode: int, secondNode: int, thirdNode: int*) [source]

Gets forces and moments along the cut line for a particular load case.

Parameters:

- **plateList** (*list*) – List of plates IDs. a) All plates in model, b) plates through which the cut line crosses (both would work but ‘a’ is computationally expensive)
- **nplates** (*int*) – No of plates in plateList
- **loadIdList** (*list*) – The load cases for plate analysis
- **startNode** (*list*) – List of x, y, z values of the start node at indexes 0, 1 and 2.
- **endNode** (*list*) – List of x, y, z values of the end node at indexes 0, 1 and 2.
- **isTransformForceToGlobal** (*int*) – 1: return force in Global System 0: return forces in local system of cut line
- **firstNode** (*int*) – Node no representing the origin point for building the surface axis system
- **secondNode** (*int*) – Node no representing the second point for building the surface axis system
- **thirdNode** (*int*) – Node no representing the third point for building the surface axis system

Returns:

List of resultant forces consisting Fx, Fy, Fz, Mx, My, Mz (in same order).

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> plate_list = staad_obj.Geometry.GetPlateList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> resultant_force_along_line_for_plate_list = staad_obj.Output.GetRes
```

GetStaticCheckResult(*LoadCaseNo: int*) [source]

Gets the statics check result containing loads and reactions for specified load case.

Parameters:

loadCaseNo (*int*) – Load Case reference ID.

Returns:

Tuple of list of loads (FX, FY, FZ, MZ, MY, MZ (in same order)) and list of reactions (FX, FY, FZ, MZ, MY, MZ (in same order)).

Return type:

tuple of list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> static_check_result = staad_obj.Output.GetStaticCheckResult(load_ca
```

GetSteelDesignParameterBlockCount() [source]

Gets the count of steel design parameter blocks in the model. This function is for AISC 360-16 code only.

Returns:

Returns the count of steel design parameter blocks.

Return type:

int

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> steel_design_parameter_block_count = staad_obj.Output.GetSteelDesig
```

GetSteelDesignParameterBlockNameByIndex(*index: int*) [source]

Gets steel design parameter name at the specified index. This function is for AISC 360-16 code only. :param index: The index value of steel design parameter block list. Note, the index is zero based. :type index: int

Returns:

Steel design parameter block name.

Return type:

str

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> countOfParamBlocks = staad_obj.Output.GetSteelDesignParameterBlockC
>>> if (countOfParamBlocks > 0) :
>>>     param_blk_name = staad_obj.Output.GetSteelDesignParameterBlockN
```

`GetSupportReactions(nodeNo: int, loadCaseNo: int)` [source]

Get the support reactions for a given support.

Parameters:

- **nodeNo** (*int*) – The node number.
- **loadCaseNo** (*int*) – The load case number.

Returns:

List with Reaction in GLOBAL direction:[FX, FY, FZ, MX, MY, MZ]

Return type:

list

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> nodeList = staad_obj.Geometry.GetNodeList()
>>> loadCases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> supportReactions = staad_obj.Output.GetSupportReactions(nodeList[3]
```

`GetTimeHistoryIntegrationStepInfo()` [source]

Gets the time-step (secs) used for time-history integration.

Returns:

Time step used for the integration in seconds.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>>
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> delta, nsteps = staad_obj.Output.GetTimeHistoryIntegrationStepInfo(
```

GetTimeHistoryResponse(*load_case: int*, *node_no: int*, *dof_no: int*, *response_type: int*) [source]

Gets the time-history responses of DOF at specified node in the VARIANT array responses

Parameters:

- **load_case** (*int*) – Load case number for future use. Use 0 at present.
- **node_no** (*int*) – Node number where the response is sought.
- **dof_no** (*int*) –

  Degrees of freedom define as DegreesOfFreedom enum:

Value	Degrees Of Freedom
Fx = 1	DegreesOfFreedom.Fx
Fy = 2	DegreesOfFreedom.Fy
Fz = 3	DegreesOfFreedom.Fz
Mx = 4	DegreesOfFreedom.Mx
My = 5	DegreesOfFreedom.My
Mz = 6	DegreesOfFreedom.Mz

- **response_type** (*int*) –

  Response type, i.e., displacement, velocity, or acceleration defined in TimeHistoryResponseType enum:

Value	Response Type
displacement = 0	TimeHistoryResponseType.dispResponse
velocity = 1	TimeHistoryResponseType.velResponse
acceleration = 2	TimeHistoryResponseType.acclResponse

Returns:

Response value.

Return type:

float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> time_history_response = staad_obj.Output.GetTimeHistoryResponse(loa
```

GetTimeHistoryResponseAtTime(*load_case: int, node_no: int, dof_no: int, response_type: int, at_time: float*) [source]

Gets the response at a specific time within the integration time span at a specified node for a given DOF.

Parameters:

- **load_case** (*int*) – Load case number for future use. Use 0 at present.
- **node_no** (*int*) – Node number where the response is sought.
- **dof_no** (*int*) –

  Degrees of freedom define as DegreesOfFreedom enum:

Value	Degrees Of Freedom
Fx = 1	DegreesOfFreedom.Fx
Fy = 2	DegreesOfFreedom.Fy
Fz = 3	DegreesOfFreedom.Fz
Mx = 4	DegreesOfFreedom.Mx
My = 5	DegreesOfFreedom.My
Mz = 6	DegreesOfFreedom.Mz

- **response_type** (*int*) –

  Response type, i.e., displacement, velocity, or acceleration defined in TimeHistoryResponseType enum:

Value	Response Type
displacement = 0	TimeHistoryResponseType.dispResponse
velocity = 1	TimeHistoryResponseType.velResponse
acceleration = 2	TimeHistoryResponseType.acclResponse

- **at_time** (*int*) – Time in seconds for which the response is sought.

Returns:

List returning the responses at the integration steps. The size of the list is (no of integration steps), the first location is for the response at time = 0.0.

Return type:

list of float

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> load_cases = staad_obj.Load.GetPrimaryLoadCaseNumbers()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> time_history_response_at_time = staad_obj.Output.GetTimeHistoryResp
```

GetTimeHistoryResponseMinMax(*load_case: int, node_no: int, dof_no: int, response_type: int*) [source]

Gets the min/max time-history responses of DOF at specified node.

Parameters:

- **load_case** (*int*) – Use 0 at present.
- **node_no** (*int*) – Node number where the response is sought.
- **dof_no** (*int*) –

  Degrees of freedom define as DegreesOfFreedom enum:

Value	Degrees Of Freedom
Fx = 1	DegreesOfFreedom.Fx
Fy = 2	DegreesOfFreedom.Fy
Fz = 3	DegreesOfFreedom.Fz

- **response_type** (*int*) –

  Response type, i.e., displacement, velocity, or acceleration defined in TimeHistoryResponseType enum:

Value	Response Type
displacement = 0	TimeHistoryResponseType.dispResponse
velocity = 1	TimeHistoryResponseType.velResponse
acceleration = 2	TimeHistoryResponseType.acclResponse

Returns:

Tuple constisiting of maximum response, the time when the maximum response occurs, minimum response and the time when the minimum response occurs (in same order).

Return type:

tuple

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> node_list = staad_obj.Geometry.GetNodeList()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> time_min_max = staad_obj.Output.GetTimeHistoryResponseMinMax(0, nod
```

`IsBucklingAnalysisResultsAvailable()` [source]

Determines whether buckling results are available

Returns:

True if buckling analysis results are available, False otherwise.

Return type:

bool

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> buckling_analysis_results_available = staad_obj.Output.IsBucklingAn
```

IsMultipleMemberSteelDesignResultsAvailable() [source]

Checks whether steel design results from multiple design block can be extracted or not. If true, then relevant multiple steel design parameters like GetMultipleMemberSteelDesignRatio or GetMultipleMemberSteelDesignResults can be used. Currently, this facility is limited to AISC 360-16 code only. For further details, please check RR 22.02.00-4.2 of STAAD.Pro Help manual.

Returns:

returns true (for boolean variable, for long variable, return value is 1) if result extraction from multiple steel design block is possible (i.e. aisc 360-16 code is used). else the return value will be false (0 for long variable).

Return type:

bool

Example

```
>>> from openstaadpy import os_analytical
>>>
>>> staad_obj = os_analytical.connect()
>>> staad_obj.AnalyzeEx(1, 0, 1)
>>>
>>> is_multiple_member_steel_design_results_available = staad_obj.Outpu
```

__init__(*staad_obj*) [source]