Command: Analysis Commands

Command

Functions

afx_msg void	**OSCommandsUI::PerformAnalysis** (const VARIANT FAR &PrintOption) Assigns the commands required to perform a FIRST ORDER ELASTIC ANALYSIS of the current structure with specified print option.
afx_msg void	**OSCommandsUI::PerformPDeltaAnalysisNoConverge** (const VARIANT FAR &NoOfIterations, const VARIANT FAR &PrintOption) Creates a command for performing PDELTA ANALYSIS with SmallDelta option.
afx_msg void	**OSCommandsUI::PerformPDeltaAnalysisEx** (const VARIANT FAR &NoOfIterations, const VARIANT FAR &PrintOption, const VARIANT FAR &bSmallDelta, const VARIANT FAR &bKG) Creates a command for performing PDELTA ANALYSIS.
afx_msg void	**OSCommandsUI::PerformPDeltaAnalysisConverge** (const VARIANT FAR &NoOfIterations, const VARIANT FAR &PrintOption) This API has been deprecated. Please use the APIs **OSCommandsUI::PerformPDeltaAnalysisNoConverge** or **OSCommandsUI::PerformPDeltaAnalysisEx** instead.
afx_msg void	**OSCommandsUI::PerformCableAnalysis** (const VARIANT FAR &iNoOfIterations, const VARIANT FAR &iPrintOption) Creates commands required to perform a CABLE ANALYSIS. This requires the presence of cable members in the structure. Advanced algorithm will be used only if Advanced license is enabled, else basic algorithm will be used. For further details, please check TR.37.3 of STAAD.Pro Help manual.
afx_msg VARIANT	**OSCommandsUI::PerformCableAnalysisEx** (const VARIANT FAR &iAdvancedCableAnalysis, const VARIANT FAR &varAdvOptions, const VARIANT FAR &varParams, const VARIANT FAR &PrintOption) Assigns the commands required to perform a CABLE ANALYSIS on the model. This requires the presence of nonlinear cables in the structure.
afx_msg void	**OSCommandsUI::PerformBucklingAnalysis** (const VARIANT FAR &iMaxSteps, const VARIANT FAR &iPrintOption) Creates commands required to perform BUCKLING ANALYSIS.
afx_msg VARIANT	**OSCommandsUI::PerformBucklingAnalysisEx** (const VARIANT FAR &Method, const VARIANT FAR &MaxSteps, const VARIANT FAR &PrintOption) Assigns the commands required to perform a BUCKLING ANALYSIS on the model. This requires the presence of nonlinear cables in the structure.
afx_msg VARIANT	**OSCommandsUI::PerformDirectAnalysis** (const VARIANT FAR &Option, const VARIANT FAR &..., const VARIANT FAR &varAddOptions, const VARIANT FAR &PrintOption) Assigns the commands required to perform a DIRECT ANALYSIS for AISC on the model.

afx_msg VARIANT	**OSCommandsUI::PerformNonlinearAnalysisEx** (const VARIANT FAR &PrintOption, const VARIANT FAR &Arclength, const VARIANT FAR &NoOfIterations, const VARIANT FAR &Tolerance, const VARIANT FAR &Steps, const VARIANT FAR &Rebuild, const VARIANT FAR &KG, const VARIANT FAR &varDispLimitData) Assigns the commands required to perform NONLINEAR ANALYSIS of the current structure with specified print option.
afx_msg VARIANT	**OSCommandsUI::SetFloorDiaphragmBaseCommand** (const VARIANT FAR &varValue) Add or update Base command for Floor Diaphragms.
afx_msg VARIANT	**OSCommandsUI::DeleteFloorDiaphragmBaseCommand** () Delete existing Base command for Floor Diaphragms.
afx_msg VARIANT	**OSCommandsUI::SetCheckSoftStoryCommand** (const VARIANT FAR &varDesignCode) Add or update Check Soft Story command.
afx_msg VARIANT	**OSCommandsUI::DeleteCheckSoftStoryCommand** () Delete existing Check Soft Story command.
afx_msg VARIANT	**OSCommandsUI::SetCheckIrregularitiesCommand** (const VARIANT FAR &varDesignCode) Add or update Check Irregularities command. This will work when Floor Diaphragm data is available in the model.
afx_msg VARIANT	**OSCommandsUI::DeleteCheckIrregularitiesCommand** () Delete existing Check Irregularities command.
afx_msg VARIANT	**OSCommandsUI::DeleteAllAnalysisCommands** () Delete all existing analysis commands.

Detailed Description

These functions are related to Analysis commamnds.

Function Documentation

◆ DeleteAllAnalysisCommands()

VARIANT OSCommandsUI::DeleteAllAnalysisCommands ( )

protected

Delete all existing analysis commands.

Return values

1 OK.

0 Perform analysis commands not present

C++ Syntax

```
// Delete all existing analysis commands.
long RetVal = OSCommandsUI::DeleteAllAnalysisCommands();
```

C# Syntax

```
// Delete all existing analysis commands.
int RetVal = OSCommandsUI::DeleteAllAnalysisCommands();
```

VBA Syntax

```
' Delete all existing analysis commands.
Dim bResult As VARIANT
bResult = OSCommandsUI.DeleteAllAnalysisCommands()
```

◆ DeleteCheckIrregularitiesCommand()

Loading [MathJax]/extensions/MathZoom.js

VARIANT OSCommandsUI::DeleteCheckIrregularitiesCommand ( ) protected

Delete existing Check Irregularities command.

Return values

1 OK.

0 Failed to delete.

C++ Syntax

```
// Delete existing Check Irregularities command.
long RetVal = OSCommandsUI::DeleteCheckIrregularitiesCommand();
```

VBA Syntax

```
' Delete existing Check Irregularities command.
Dim bResult As VARIANT
bResult = OSCommandsUI.DeleteCheckIrregularitiesCommand()
```

See also

OSCommandsUI::SetCheckIrregularitiesCommand

OSCommandsUI::SetFloorDiaphragmBaseCommand

OSCommandsUI::DeleteFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckSoftStoryCommand

OSCommandsUI::DeleteCheckSoftStoryCommand

◆ DeleteCheckSoftStoryCommand()

VARIANT OSCommandsUI::DeleteCheckSoftStoryCommand ( ) protected

Delete existing Check Soft Story command.

Return values

1 OK.

0 Failed to delete.

C++ Syntax

```
// Delete existing Check Soft Story command.
long RetVal = OSCommandsUI::DeleteCheckSoftStoryCommand();
```

VBA Syntax

```
' Delete existing Check Soft Story command.
Dim bResult As VARIANT
bResult = OSCommandsUI.DeleteCheckSoftStoryCommand()
```

See also

OSCommandsUI::SetCheckSoftStoryCommand

OSCommandsUI::SetFloorDiaphragmBaseCommand

OSCommandsUI::DeleteFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckIrregularitiesCommand

OSCommandsUI::DeleteCheckIrregularitiesCommand

◆ DeleteFloorDiaphragmBaseCommand()

VARIANT OSCommandsUI::DeleteFloorDiaphragmBaseCommand ( ) protected

Delete existing Base command for Floor Diaphragms.

Return values

1 OK.

0 Failed to delete.

C++ Syntax

```
// Delete existing Base command.
long RetVal = OSCommandsUI::DeleteFloorDiaphragmBaseCommand();
```

VBA Syntax

```
' Delete existing Base command.
Dim bResult As VARIANT = OSCommandsUI.DeleteFloorDiaphragmBaseCommand()
```

See also

OSCommandsUI::SetFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckSoftStoryCommand

OSCommandsUI::DeleteCheckSoftStoryCommand

OSCommandsUI::SetCheckIrregularitiesCommand

OSCommandsUI::DeleteCheckIrregularitiesCommand

◆ PerformAnalysis()

void OSCommandsUI::PerformAnalysis (const VARIANT FAR & PrintOption) protected

Assigns the commands required to perform a FIRST ORDER ELASTIC ANALYSIS of the current structure with specified print option.

Parameters

[in] **iPrintOption** Option index for specifying the print option:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Print Mode Shapes
5	Print Both
6	Print All
0	No Print

C++ Syntax

```
// Perform FIRST ORDER ELASTIC ANALYSIS and print ALL.
OSCommandsUI::PerformAnalysis(6);
```

◆ PerformBucklingAnalysis()

void OSCommandsUI::PerformBucklingAnalysis (const VARIANT FAR & iMaxSteps,
const VARIANT FAR & iPrintOption) protected

Creates commands required to perform BUCKLING ANALYSIS.

Parameters

[in] **iNoOfIterations** Maximum no. of iterations desired

[in] **iPrintOption** Option for specifying the print option. Option should be specified in accordance with below table:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
5	Print Both
6	Print All
0	No Print

C++ Syntax

```
//Create Buckling Analysis command, with MAXSTEP 20 and PRINT ALL option.
OSCommandsUI::PerformBucklingAnalysis(20, 6);
```

VBA Syntax

```
'Create Buckling Analysis command, with MAXSTEP 20 and PRINT ALL option.
objOpenStaad.Command.PerformBucklingAnalysis(20, 6)
```

◆ PerformBucklingAnalysisEx()

VARIANT OSCommandsUI::PerformBucklingAnalysisEx (const VARIANT FAR & Method,
const VARIANT FAR & MaxSteps,
const VARIANT FAR & PrintOption) protected

Assigns the commands required to perform a BUCKLING ANALYSIS on the model. This requires the presence of nonlinear cables in the structure.

Parameters

[in] **method** Buckling Analysis method using enum type BucklingAnalysisMethod.

Value	Method
Iterative = 0	BucklingAnalysisMethod.Iterative
Eigen = 1	BucklingAnalysisMethod.Eigen

[in] **iMaxSteps** Maximum no. of iterations desired (default value of n = 10). 15 is recommended. Used for Basic Solver, only.

[in] **iPrintOption** Option index for specifying the print option:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Reserved, do not use
5	Print Both
6	Print All
0	No Print

Return values

TRUE(1) OK.

FALSE(0) Failed to add or update.

C++ Syntax

```
// Perform BUCKLING ANALYSIS (with 15 iterations and printing static check) command.
long RetVal = OSCommandsUI::PerformBucklingAnalysisEx(BucklingAnalysisMethod.Iterative,
       15, 2);
```

C# Syntax

```
// Perform BUCKLING ANALYSIS (with 15 iterations and printing static check) command.
long RetVal = OSCommandsUI::PerformBucklingAnalysisEx(BucklingAnalysisMethod.Iterative,
       15, 2);
```

Loading [MathJax]/extensions/MathZoom.js

VB Syntax

```
' Perform BUCKLING ANALYSIS (with 15 iterations and printing static check) command.
Dim method As Integer
Dim iPrintOption As Integer
Dim iteration As Integer
Dim ret As Integer
method = 0
iteration = 15
iPrintOption = 2

ret = objOpenStaad.Command.PerformBucklingAnalysisEx(method, iteration, iPrintOption)
```

◆ PerformCableAnalysis()

void OSCommandsUI::PerformCableAnalysis (const VARIANT FAR & iNoOfIterations,
const VARIANT FAR & iPrintOption) protected

Creates commands required to perform a CABLE ANALYSIS. This requires the presence of cable members in the structure. Advanced algorithm will be used only if Advanced license is enabled, else basic algorithm will be used. For further details, please check TR.37.3 of STAAD.Pro Help manual.

Parameters

[in] **iNoOfIterations** Desired number of iterations.

[in] **iPrintOption** Option for specifying the print option. Option should be specified in accordance with below table:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
5	Print Both
6	Print All
0	No Print

C++ Syntax

```
//Create Cable Analysis command, with 20 iterations and PRINT ALL option.
OSCommandsUI::PerformCableAnalysis(20, 6);
```

VBA Syntax

```
'Create Cable Analysis command, with 20 iterations and PRINT ALL option.
objOpenStaad.Command.PerformCableAnalysis(20, 6)
```

- PerformCableAnalysisEx()

VARIANT OSCommandsUI::PerformCableAnalysisEx (const VARIANT FAR & iAdvancedCableAnalysis,
const VARIANT FAR & varAdvOptions,
const VARIANT FAR & varParams,
const VARIANT FAR & PrintOption) protected

Assigns the commands required to perform a CABLE ANALYSIS on the model. This requires the presence of nonlinear cables in the structure.

Parameters

[in] **iAdvancedCableAnalysis** Is Advanced Cable Analysis to perform (= **TRUE/FALSE**)

[in] **varAdvOptions** The additional options for Advanced Cable Analysis stored in a BOOL/short/int array object.

Array Index	Default	STAAD Command	Options
0	1	REFORM f11	Use Full Newton-Raphson method? (0 = **FALSE**; 1 = **TRUE**)
1	0	KGEOM f12	Use Geometric Matrix (Kg)? (0 = **FALSE**; 1 = **TRUE**)

User can supply NULL or an array of any size. For Advanced Cable Analysis,

- if NULL is supplied both these values will be assumed as FALSE
- If array of size=1 is supplied the second value will be assumed as FALSE
- If array of size>2 is supplied all the values except the first two will be ignored

[in] **varParams** The additional parameters required for Cable Analysis stored in a double array object. Some of these values are used to represent integer value.

- For basic cable analysis: -

Array Index	Default value	STAAD Command	Parameters
0	145	STEPS f1	The number of load steps. The applied loads will be applied gradually in this many steps. Each step will be iterated to convergence. Should be in the range 5 to 145, with 145 as the default number.

1	300	EQITERATIONS f2	Maximum number of iterations permitted in each load step. Should be in the range of 10 to 500.
2	1.0E-4	EQTOLERANCE f3	The convergence tolerance for the above iterations.
3	0.0	SAGMINIMUM f4	Sag Minimum. Default is 0.0.
4	1.0	STABILITY f5	A stiffness matrix value to be added to the global matrix at each translational direction for joints connected to cables and nonlinear trusses for the first (f6) Load Steps. Should be within the range of 0.0 to 1000.0.
5	1	f6	The number of load steps over which the Stability stiffness matrix(f5) is gradually applied.
6	0.0	KSMALL f7	A stiffness matrix value to be added to the global matrix at each translational direction for joints connected to cables and nonlinear trusses for every load step. Should be within the range of 0.0 and 1.0.

- For Advanced cable analysis: -

Array Index	Default value	STAAD Command	Parameters
0	1	STEPS f1	Used to divide the loading into small increments. Any positive integer is valid.

			Generally, a value larger than one (1) may be beneficial for solution convergence, because the loading is applied gradually. However, with cable elements having fixed prestressing, a single step typical converges faster. Default is 1.
1	300	EQITERATIONS f2	Maximum number of iterations permitted in each load step. During each loading increment, the analysis will iterate to find the converging solution. Any positive integer is valid. Values too small may prevent the solution to achieve convergence. However, excessively large values may cost unnecessary running time when the problem diverges.
2	1.0E-6	EQTOLERANCE f3	The convergence tolerance for the iterations specified in EQITERATIONS. The residual force norm is used for this convergence. This error threshold indicates when the nonlinear solver will stop iterating and consider the ongoing step as converged once the computed error is equal or smaller than this value. A value that is too small may prevent

			the solution to be considered as converged. However, value that is too larger may result in inaccurate results.

[in] **iPrintOption** Option index for specifying the print option:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
5	Print Both
6	Print All
0	No Print

Return values

TRUE(1) OK.

FALSE(0) Failed to add or update.

C++ Syntax

```
// Perform CABLE ANALYSIS (with 10 iterations and printing all) command.
double arrParameters[7];
arrParameters[0] = 125;          // load steps
arrParameters[1] = 350;          // Max iterations
arrParameters[2] = 0.000001;     // Convergence tollerance
arrParameters[3] = 0.84;         // Sag minimum
arrParameters[4] = 560.0;        // Stability
arrParameters[5] = 3;            // Load steps on which stability stiffness matrix is
       applied
arrParameters[6] = 0.6;          // k-small
object objParameters = arrParameters as object;

int rValue = OSCommandsUI::PerformCableAnalysisEx(Convert.ToInt16(false), null,
       objParameters, AnalysisPrintOption.PrintAll);
```

C# Syntax

```
// Perform CABLE ANALYSIS (with 10 iterations and printing all) command.
double[] arrParameters = new double[7];
arrParameters[0] = 125;          // load steps
arrParameters[1] = 350;          // Max iterations
arrParameters[2] = 0.000001;     // Convergence tollerance
arrParameters[3] = 0.84;         // Sag minimum
arrParameters[4] = 560.0;        // Stability
arrParameters[5] = 3;            // Load steps on which stability stiffness matrix is
       applied
arrParameters[6] = 0.6;          // k-small
object objParameters = arrParameters as object;
```

```
int rValue = OSCommandsUI::PerformCableAnalysisEx(Convert.ToInt16(false), null,
        objParameters, AnalysisPrintOption.PrintAll);
```

VB Syntax

```
' Perform CABLE ANALYSIS (with 10 iterations and printing all) command.
Dim iAdvanced As Integer
Dim arrAdvOptions() As Boolean
Dim arrParameters() As Double
Dim iPrintOption As Integer
Dim ret As Integer
iAdvanced = 0
ReDim arrAdvOptions(0)
ReDim arrParameters(6)
arrParameters(0) = 125          'load steps
arrParameters(1) = 350          'Max iterations
arrParameters(2) = 0.000001     'Convergence tollerance
arrParameters(3) = 0.84         'Sag minimum
arrParameters(4) = 560.0        'Stability
arrParameters(5) = 3            'Load steps on which stability stiffness matrix is
        applied
arrParameters(6) = 0.6          'k-small
iPrintOption = 6

ret = objOpenStaad.Command.PerformCableAnalysisEx(iAdvanced, arrAdvOptions,
        arrParameters, iPrintOption)
```

See also

OSCommandsUI::CreateNewCableAnalysisBlock

◆ PerformDirectAnalysis()

```
VARIANT OSCommandsUI::PerformDirectAnalysis ( const VARIANT FAR & Option,
                                              const VARIANT FAR & varParams,
                                              const VARIANT FAR & varAddOptions,
                                              const VARIANT FAR & PrintOption )
```

protected

Assigns the commands required to perform a DIRECT ANALYSIS for AISC on the model.

Parameters

`[in]` **option** Direct Analysis option (LRFD/ASD) using enum type DirectAnalysisOption. Default is 1 (LRFD).

Value	Method
LRFD = 1	DirectAnalysisOption.LRFD
ASD = 2	DirectAnalysisOption.ASD

`[in]` **varParams** The additional parameters required for Direct Analysis stored in a double array object. Some of these values are used to represent integer value.

Array Index	Default value	STAAD Command	Parameters
0	0.01	TAUTOL f1	Tau-b tolerance. Is normally 0.001 to 1.0.
1	0.01	DISPTOL f2	Displacement tolerance - • 0.01 in (displacement) • 0.01 radians (rotation) The value should not be too tight.
2	1	ITERDIRECT i3	Limits the number of iterations. A value between 1 to 10 is typically sufficient.
3	15	PDiter i5	The number of iterations, used in the iterative PDelta with SmallDelta analysis procedure within Direct Analysis. 5 to 15 iterations is the normal range. Any value more than 15 will reset to 15.

`[in]` **varAddOptions** The additional options for Direct Analysis stored in a BOOL/short/int array object.

Array Index	Default	STAAD Command	Options
0	0	REDUCEDEI i4	whether to use the reduced EI (Tau-b * 0.8 * EI) for member section moment and section displacement? (0 = **FALSE** (full EI); 1 = **TRUE** (reduced EI))
1	0	TBITER	Iterate Tau-b? (0 = **FALSE**; 1 = **TRUE**)

an supply NULL or an array of any size.

- if NULL is supplied both these values will be assumed as FALSE

- If array of size=1 is supplied the second value will be assumed as FALSE
- If array of size>2 is supplied all the values except the first two will be ignored

`[in]` **iPrintOption** Option index for specifying the print option:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Print Mode Shapes
5	Print Both
6	Print All
0	No Print

Return values

TRUE(1) OK.

FALSE(0) Failed to add or update.

C++ Syntax

```
// Perform DIRECT ANALYSIS (with printing all) command.
double arrParameters[4];
arrParameters[0] = 0.015;       // Tau-b tolerance
arrParameters[1] = 0.02;        // Disp iterations
arrParameters[2] = 7;           // Iteration limit
arrParameters[3] = 13;          // Iteration for P-SmallDelta
object objParameters = arrParameters as object;

short arrAddOptions[2];
arrAddOptions[0] = Convert.ToInt16(false);    // Use Reduced EI?
arrAddOptions[1] = Convert.ToInt16(false);    // Iterate Tau-b?
object objAddOptions = arrAddOptions as object;

long RetVal = COSCommand::PerformDirectAnalysis(DirectAnalysisOption.LRFD, objParameters,
       objAddOptions, AnalysisPrintOption.PrintAll);
```

C# Syntax

```
// Perform DIRECT ANALYSIS (with printing all) command.
double[] arrParameters = new double[4];
arrParameters[0] = 0.015;       // Tau-b tolerance
arrParameters[1] = 0.02;        // Disp iterations
arrParameters[2] = 7;           // Iteration limit
arrParameters[3] = 13;          // Iteration for P-SmallDelta
object objParameters = arrParameters as object;

short[] arrAddOptions = new short[2];
arrAddOptions[0] = Convert.ToInt16(false);    // Use Reduced EI?
arrAddOptions[1] = Convert.ToInt16(false);    // Iterate Tau-b?
object objAddOptions = arrAddOptions as object;
```

```
long RetVal = COSCommand::PerformDirectAnalysis(DirectAnalysisOption.LRFD, objParameters,
       objAddOptions, AnalysisPrintOption.PrintAll);
```

VB Syntax

```
' Perform DIRECT ANALYSIS (with printing all) command.
Dim daOption As Integer
Dim arrParameters() As Integer
Dim arrAddOptions() As Double
Dim iPrintOption As Integer
Dim iteration As Integer
Dim ret As Integer
ReDim arrParameters(3)
ReDim arrAddOptions(1)
daOption = 1
arrParameters(0) = 0.015        ' Tau-b tolerance
arrParameters(1) = 0.02         ' Disp iterations
arrParameters(2) = 7            ' Iteration limit
arrParameters(3) = 13           ' Iteration for P-SmallDelta
arrAddOptions(0) = False
arrAddOptions(1) = False
iPrintOption = 6

ret = objOpenStaad.Command.PerformDirectAnalysis(daOption, arrParameters, arrAddOptions,
       iPrintOption)
```

◆ PerformNonlinearAnalysisEx()

```
VARIANT OSCommandsUI::PerformNonlinearAnalysisEx ( const VARIANT FAR & PrintOption,
                                                   const VARIANT FAR & Arclength,
                                                   const VARIANT FAR & NoOfIterations,
                                                   const VARIANT FAR & Tolerance,
                                                   const VARIANT FAR & Steps,
                                                   const VARIANT FAR & Rebuild,
                                                   const VARIANT FAR & KG,
                                                   const VARIANT FAR & varDispLimitData )
```
protected

Assigns the commands required to perform NONLINEAR ANALYSIS of the current structure with specified print option.

Parameters

[in] **PrintOption** Option index for specifying the print option:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Print Mode Shapes
5	Print Both
6	Print All
0	No Print

[in] **Arclength** Absolute displacement limit (float/double) for the first analysis step for Displacement Control (= 0 for no Displacement Control).

[in] **NoOfIterations** Upper limit for the times of iteration(s) to achieve equilibrium in the deformed position to the tolerance specified.

[in] **Tolerance** Tolerance value (float/double) for determining convergence.

[in] **Steps** Number of Load Step(s): load is applied in stage(s).

[in] **Rebuild** Frequency of rebuilds of the Tangent or Stiffness Matrix(K) per load step & iteration.
0 = once per load step
1 = every load step & iteration

[in] **KG** Add the geometric stiffness(K_g) to the Stiffness Matrix(K): **TRUE** or **FALSE**.

[in] **varDispLimitData** The displacement limit data to specify target displacement.

Array Index	Default	STAAD Command	Options
0	1	JOINT_TARGET i1	Joint being monitored in a displacement target analysis

1	1	i2	Global degree of freedom (1 through 6): • 1:Global X • 2:Global Y • 3:Global Z • 4:Moment about Global X • 5:Moment about Global Y • 6:Moment about Global Z
2	0	f1	Displacement target value in current length units.

Return values

TRUE(1) OK.

FALSE(0) Failed to add or update.

C++ Syntax

```
// Perform NONLINEAR ANALYSIS with iterations for no more than 5 times, tolerance of 1e-
       9, Load Steps 10, update K once per load step, consider Geometric Stiffness,
       joint 2 for displacement limit in Global Z with target displacement value 0.01
       in, and print ALL.
double arrDispLim[3];
arrDispLim[0] = 2;        // target joint
arrDispLim[1] = 3;        // DOF
arrDispLim[2] = 0.01;     // target displacement
object objDispLim = arrDispLim as object;
long RetVal = COSCommand::PerformNonlinearAnalysisEx(6, 0, 5, 1e-9, 10, 0, TRUE,
       objDispLim);
```

 # Syntax

```
// Perform NONLINEAR ANALYSIS with iterations for no more than 5 times, tolerance of 1e-
       9, Load Steps 10, update K once per load step, consider Geometric Stiffness,
       joint 2 for displacement limit in Global Z with target displacement value 0.01
       in, and print ALL.
double[] arrDispLim = new double[3];
arrDispLim[0] = 2;        // target joint
arrDispLim[1] = 3;        // DOF
arrDispLim[2] = 0.01;     // target displacement
object objDispLim = arrDispLim as object;
long RetVal = COSCommand::PerformNonlinearAnalysisEx(6, 0, 5, 1e-9, 10, 0, TRUE,
       objDispLim);
```

VB Syntax

```
' Perform NONLINEAR ANALYSIS with iterations for no more than 5 times, tolerance of 1e-9,
       Load Steps 10, update K once per load step, consider Geometric Stiffness, joint 2
       for displacement limit in Global Z with target displacement value 0.01 in, and
       print ALL.
Dim arrDispLim() as Double
ReDim arrDispLim(2)
```

```
arrDispLim(0) = 2          ' target joint
arrDispLim(1) = 3          ' DOF
arrDispLim(2) = 0.01       ' target displacement
ret = objOpenStaad.Command.PerformNonlinearAnalysisEx(6, 0, 5, 1e-9, 10, 0, TRUE,
       arrDispLim)
```

@ sa COSCommand::CreateNewNonlinearAnalysisBlock

◆ PerformPDeltaAnalysisEx()

void OSCommandsUI::PerformPDeltaAnalysisEx (const VARIANT FAR & NoOfIterations,
const VARIANT FAR & PrintOption,
const VARIANT FAR & bSmallDelta,
const VARIANT FAR & bKG) protected

Creates a command for performing PDELTA ANALYSIS.

Parameters

[in] **NoOfIterations** Desired number of iterations.

[in] **PrintOption** Option for specifying the print option. Option should be specified in accordance with below table:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Print Mode Shapes
5	Print Both
6	Print All
0	No Print

[in] **bSmallDelta** 1 = Use P-SMALL-Delta effect, 0 = Use P-LARGE-Delta effect.

[in] **bKG** Add geometric stiffness($\mathbf{K_g}$) to the Stiffness Matrix(**K**): 1 = TRUE or 0 = FALSE.

C++ Syntax

```
//Create PDelta Analysis command, with 20 iterations with SmallDelta option and without
       KG option.
OSCommandsUI::PerformPDeltaAnalysisEx(20, 5, 1, 0);
```

VBA Syntax

```
'Create PDelta Analysis command, with 20 iterations without SmallDelta option and with KG
       option.
objOpenStaad.Command.PerformPDeltaAnalysisEx(20, 5, 0, 1)
```

See also

OSCommandsUI::PerformPDeltaAnalysisNoConverge

◆ PerformPDeltaAnalysisNoConverge()

void OSCommandsUI::PerformPDeltaAnalysisNoConverge (const VARIANT FAR & NoOfIterations, const VARIANT FAR & PrintOption) protected

Creates a command for performing PDELTA ANALYSIS with SmallDelta option.

Parameters

[in] **NoOfIterations** Desired number of iterations.

[in] **PrintOption** Option for specifying the print option. Option should be specified in accordance with below table:

Value	Print Option
1	Print Load Data
2	Print Statics Check
3	Print Statics Load
4	Print Mode Shapes
5	Print Both
6	Print All
0	No Print

C++ Syntax

```
//Create PDelta Analysis command, with 20 iterations and PRINT ALL option.
OSCommandsUI::PerformPDeltaAnalysisNoConverge(20, 6);
```

VBA Syntax

```
'Create PDelta Analysis command, with 20 iterations and PRINT ALL option.
objOpenStaad.Command.PerformPDeltaAnalysisNoConverge(20, 6)
```

See also

OSCommandsUI::PerformPDeltaAnalysisEx

◆ SetCheckIrregularitiesCommand()

VARIANT OSCommandsUI::SetCheckIrregularitiesCommand (const VARIANT FAR & varDesignCode) protected

Add or update Check Irregularities command. This will work when Floor Diaphragm data is available in the model.

Parameters

varDesignCode Option index for specifying code to be used:

Value	Code
2	ASCE 7 2016
3	IS1893 2016

Return values

1 OK.

0 Failed to add or update.

C++ Syntax

```
// Add or update Check Irregularities command.
long RetVal = OSCommandsUI::SetCheckIrregularitiesCommand(3);
```

VBA Syntax

```
' Add or update Check Irregularities command.
Dim bResult As VARIANT
bResult= OSCommandsUI.SetCheckIrregularitiesCommand(3)
```

See also

OSCommandsUI::DeleteCheckIrregularitiesCommand

OSCommandsUI::SetFloorDiaphragmBaseCommand

OSCommandsUI::DeleteFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckSoftStoryCommand

OSCommandsUI::DeleteCheckSoftStoryCommand

◆ SetCheckSoftStoryCommand()

VARIANT OSCommandsUI::SetCheckSoftStoryCommand (const VARIANT FAR & varDesignCode) protected

Add or update Check Soft Story command.

Parameters

varDesignCode Option index for specifying code to be used:

Value	Code
1	IS1893 2002
2	ASCE7 05/10/16
3	IS1893 2016

Return values

1 OK.

0 Failed to add or update.

C++ Syntax

```
// Add or update Check Soft Story command.
long RetVal = OSCommandsUI::SetCheckSoftStoryCommand(3);
```

VBA Syntax

```
' Add or update Check Soft Story command.
Dim bResult As VARIANT
bResult= OSCommandsUI.SetCheckSoftStoryCommand(3)
```

See also

OSCommandsUI::DeleteCheckSoftStoryCommand

OSCommandsUI::SetFloorDiaphragmBaseCommand

OSCommandsUI::DeleteFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckIrregularitiesCommand

OSCommandsUI::DeleteCheckIrregularitiesCommand

◆ SetFloorDiaphragmBaseCommand()

VARIANT OSCommandsUI::SetFloorDiaphragmBaseCommand (const VARIANT FAR & varValue) protected

Add or update Base command for Floor Diaphragms.

Parameters

[in] **varValue** Base elevation (Unit: Length)

Return values

1 OK.

0 Failed to add or update.

C++ Syntax

```
// Add or update Base command with base elevation 2 units.
long RetVal = OSCommandsUI::SetFloorDiaphragmBaseCommand(2);
```

VBA Syntax

```
' Add or update Base command with base elevation 2 units.
Dim bResult As VARIANT = OSCommandsUI.SetFloorDiaphragmBaseCommand(2)
```

See also

OSCommandsUI::DeleteFloorDiaphragmBaseCommand

OSCommandsUI::SetCheckSoftStoryCommand

OSCommandsUI::DeleteCheckSoftStoryCommand

OSCommandsUI::SetCheckIrregularitiesCommand

OSCommandsUI::DeleteCheckIrregularitiesCommand