Módulo TQSMODEL

O modelo estrutural, criado através do Modelador em modo interativo, pode ser totalmente lido e gravado pelo módulo TQSModel. Existem inúmeras possibilidades para uso do TQSModel, entre criar um modelo novo, modificar um modelo existente e extrair dados de modelos. Podemos testar várias soluções estruturais diferentes de um único edifício, com geração e processamento automático, combinando diferentes critérios, dimensões e cargas. Basta combinar este módulo com os módulos TQSBuild e TQSExec.

Este módulo é maior que os demais vistos. Os diversos elementos estruturais do Modelador são representados por objetos. Considerando apenas 15 objetos do Modelador definidos inicialmente (de um total de 54), foram definidas cerca de 100 classes e 1200 funções.

Modelo Estrutural

Fica mais fácil entender o funcionamento do TQSModel, se entendermos como são relacionados os dados do edifício e os diversos objetos do Modelo Estrutural. É ideal que o programador tenha experiência no uso do TQS e do Modelador Estrutural. Como sempre, um ótimo meio de iniciar o uso deste módulo é examinando e copiando partes dos exemplos distribuídos (TSTMODELn.PY).

A forma de criar e alterar objetos através do TQSModel reflete o funcionamento do TQS – por exemplo, a planta atual e a criação de elementos estruturais usando dados atuais. Muitas estruturas de dados correspondem aproximadamente à abas de edição de dados de elementos estruturais.

Dados do Edifício e Modelo Estrutural

No TQS, o modelo estrutural completo é dividido em duas partes: uma chamada de Dados do Edifício e outra Modelo Estrutural. A separação não tem a ver com o modelo em si, mas como o TQS foi desenvolvido ao longo do tempo.

Os Dados do Edifício contém a definição dos pavimentos e suas cotas, casos de carregamento, materiais, cobrimentos e critérios. O modelo estrutural contém os objetos estruturais como vigas, pilares, lajes, fundações, cargas e outros elementos, por planta, assim como pilares e elementos inclinados entre plantas diferentes. Antes da criação ou edição de um modelo estrutural, é necessário definir os dados do edifício. Isto é feito interativamente ou pelo módulo TQSBuild mostrado neste manual. Desde a V24, é possível editar os dados do edifício dentro do Modelador. Esta função não está disponível no módulo TQSModel.

Ao criar os dados do edifício, é criado um conjunto de pastas para trabalho. O Modelador precisa ter a pasta atual como uma das pastas de trabalho do edifício. Estas pastas são chamadas de Contexto do Edifício.

Organização de dados do Modelo Estrutural

Os dados estão distribuídos por classes, que são instanciadas em objetos. Os objetos são o que se espera do modelo: vigas, pilares, lajes, fundações, etc. Os objetos são armazenados em listas de dois tipos:

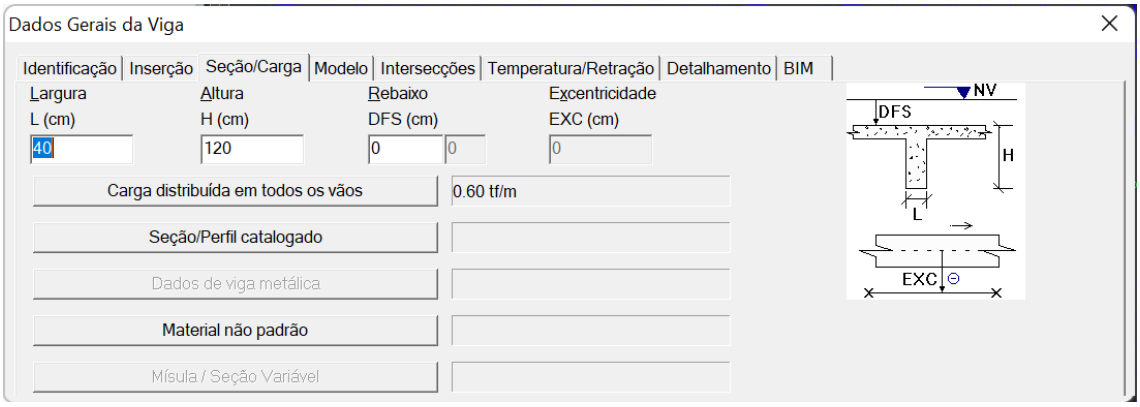
Lista	Conteúdo
`Pavimentos`	Uma lista de objetos por pavimento. Vigas, lajes, contornos, cargas, cotagens, dados por pavimento, etc
`Lista espacial`	Objetos que passam por mais de uma planta – pilares, elementos inclinados, dados globais.

Para facilitar o acesso, certos objetos espaciais aparecem na lista de objetos do pavimento, conforme o contexto. Por exemplo, se estamos trabalhando com o pavimento térreo e um pilar passa por ele, você pode acessar os dados do

pilar neste pavimento. Se ele morre antes do térreo, não aparecerá nesta lista.

Criando elementos novos e dados atuais

Seguindo a lógica do Modelador Estrutural, a criação de um elemento estrutural tem poucos dados. A maioria deles é pré-definida em Dados Atuais. Por exemplo, os dados atuais de vigas:



Temos então duas maneiras de definir elementos estruturais:

Preenchemos os dados atuais e criamos o elemento. Esta é a maneira de criar elementos no Modelador;
ou, criamos um elemento sem verificar os dados atuais, e depois modificamos o elemento criado.
Na criação de cada elemento, mostraremos onde estão os dados atuais usados na criação, e as variáveis de acesso correspondentes no elemento criado.

Objeto TQSModel.Model

Este objeto representa todo o modelo estrutural, e todos os demais objetos serão acessados começando através ele. O acesso a um modelo estrutural é sempre sobre um edifício existente, com dados do edifício já criados. A pasta atual para abrir um modelo necessariamente é uma das pastas do edifício.

O módulo TQSBuild tem uma função para entrar na pasta raiz do edifício. Para abrir um modelo existente de nome TESTE, podemos fazer:

```
from TQS import TQSBuild, TQSModel

....

build = TQSBuild.Building ()

istat = build.RootFolder ("TESTE")

ifistat == 0:

model = TQSModel.Model ()
```

Neste exemplo, criamos um objeto tipo TQSBuild.Building e entramos na pasta raiz do edifício TESTE (ex: C:\TQS\TESTE). Então, abrimos o modelo estrutural. O modelo tem 3 subobjetos:

Objeto	Uso
`file`	Leitura e salvamento do modelo
`type`	Lista os tipos de objeto do Modelador

floors	Acesso aos pavimentos e objetos FloorsList
current	Dados atuais de uso global CurGlobalData (parte dos dados atuais é definida também por pavimento)
visData	Modos de visualização do Modelador VisData

Leitura e gravação de modelos com o Model.file

O nome do edifício não é fornecido com as funções de leitura e salvamento. Ele é deduzido pela pasta atual do edifício, ou contexto do edifício.

```
file.OpenModel ()
```

Abre modelo existente ou cria um novo

Retorna: istat!=0 Se erro

```
file.OpenNewModel ()
```

Abre um modelo novo, apaga o velho se já existir

Retorna: istat!=0 Se erro

```
file.Save ()
```

Salva o modelo estrutural

Retorna: istat!=0 Se erro

```
file. Close ()
```

Fecha modelo estrutural aberto

Lista de tipos de objeto com Model.type

O Model.type lista os tipos de objeto do Modelador. Os tipos são numerados sequencialmente. Nem todos os tipos tem tratamento previsto dentro do TQSModel.

```
type.GetNumTypes()
```

Retorna o número de tipos de objetos do Modelador

```
type.GetTypeClass(itype)
```

Retorna o nome de classe de um tipo de objeto (classe C++, não do TQSModel)

`itype` TYPE_xxxx

Retorna

Nomeclasse(string) Nome da classe

```
type.GetTypeDescription(itype)
```

Retorna a descrição de um tipo de objeto

```
itype  TYPE_xxxx
```

Retorna

Nomeobjeto(string) Nome do objeto

Os tipos descritos pelas constantes TYPE_xxx são os seguintes:

Constante	Tipo
TQSModel.TYPE_VIGAS	Vigas
TQSModel.TYPE_PILARES	Pilares
TQSModel.TYPE_LAJES	Lajes
TQSModel.TYPE_FUNDAC	Fundações
TQSModel.TYPE_FUROS	Furos
TQSModel.TYPE_CAPITEIS	Capiteis
TQSModel.TYPE_FORNER	Formas de lajes nervuradas
TQSModel.TYPE_CTRAUX	Contorno auxiliar
TQSModel.TYPE_EIXOS	Eixos rotulados
TQSModel.TYPE_COTAGENS	Cotagens
TQSModel.TYPE_CORTES	Cortes
TQSModel.TYPE_DADPAV	Dados comuns do pavimento
TQSModel.TYPE_DADEDI	Dados comuns do edifício
TQSModel.TYPE_SECPIL	Seções de pilares
TQSModel.TYPE_LISMAL	Lista de malhas de lajes
TQSModel.TYPE_CARCON	Cargas concentradas
TQSModel.TYPE_CARLIN	Cargas distribuídas lineares
TQSModel.TYPE_CARARE	Carga distribuída por área
TQSModel.TYPE_BARICEN	Tabela de baricentros de pilares
TQSModel.TYPE_CARADI	Carga adicional em laje
TQSModel.TYPE_RAMPAS	Laje inclinada
TQSModel.TYPE_ORIEIX	Origem do sistema de eixos
TQSModel.TYPE_TABNP	Tabela de seções não padrão

TQSModel.TYPE_IDCLI	Identificação do cliente
TQSModel.TYPE_CONSBIB	Biblioteca de consolos
TQSModel.TYPE_FUROVIG	Furo em viga
TQSModel.TYPE_ELMLJP	Elemento de laje pré-moldada
TQSModel.TYPE_TABELEM	Tabela com lista de elementos
TQSModel.TYPE_COTDIA	Cotagem de raio/diâmetro
TQSModel.TYPE_COTANG	Cotagem angular
TQSModel.TYPE_COTNOT	Cotagem de notas
TQSModel.TYPE_VENDST	Tabela de distribuição de vento
TQSModel.TYPE_CENTOR	Centro de torção - túnel de vento
TQSModel.TYPE_CONSIND	Consolos Independentes
TQSModel.TYPE_CAREMP	Cargas de empuxo
TQSModel.TYPE_TABNIVPAV	Tabela de níveis de pavimentos
TQSModel.TYPE_LEGDESNIV	Tabela de legenda de desníveis
TQSModel.TYPE_SIMDESNIV	Símbolo de desnível de pavimento
TQSModel.TYPE_ELM3D	Elemento 3D
TQSModel.TYPE_INSRBLO	Biblioteca de insertos
TQSModel.TYPE_INSRPOS	Inserto
TQSModel.TYPE_FACHADA	Fachada de painel pré-moldado
TQSModel.TYPE_TUBOS	Tubos
TQSModel.TYPE_FUROPIL	Furo em pilar
TQSModel.TYPE_LAJESSOVOL	Lajes somente volume
TQSModel.TYPE_ESTACRAD	Estaca sob Radier
TQSModel.TYPE_VIGASINC	Viga inclinada
TQSModel.TYPE_PILARETES	Pilaretes
TQSModel.TYPE_ESCADAS	Escada

TQSModel.TYPE_PATAMARES	Patamar
TQSModel.TYPE_ORIBARI	Origem de baricentros
TQSModel.TYPE_EDIEDI	Dados do edifício
TQSModel.TYPE_PTBAS	Ponto base do projeto
TQSModel.TYPE_PTLVT	Ponto de levantamento topográfico
TQSModel.TYPE_CERCATOR	Cerca de torre (separa torre e periferia)

Acesso aos pavimentos via Model.floors

Permite o acesso aos pavimentos do edifício, seja por enumeração, seja por nome. Os índices utilizados para acesso se iniciam em (1).

```
floors.GetFloor(nompla)
```

Define a planta atual e retorna um objeto Floor() de planta

`nompla(string)` Nome da planta

Retorna:

`floor` Objeto Floor(), com dados de uma planta

```
floors.GetNumFloors()
```

Retorna o número de plantas do edifício

```
floors.GetFloorName(indpla)
```

Nome de uma planta

`indpla` Índice da planta 1..GetNumFloors()

Retorna:

`nompla` Nome da planta

```
floors.GetFloorIndex (nompla)
```

Índice da planta

`nompla` Nome da planta

Retorna:

`indpla` Índice da planta 1..GetNumFloors()ou zero se não achar

Dados globais Model.current

Esta variável retorna um objeto da classe CurGlobalData(), que contém subobjetos com dados atuais, usados na criação de diversos tipos de objetos.

Subobjetos	Conteúdo
current.globalAxisData	Eixos
current.globalCutData	Cortes
current.globalSoilpress	Carga de empuxo
current.globalRefxData	Critérios de referências externas
current.globalBeamData	Vigas
current.globalSlabData	Lajes
current.globalColumnData	Pilares, pilaretes e fundações
current.globalColumnOpening	Furos em pilares
current.globalSlabFPile	Estaca de laje radier
current.globalPrecast	Pré-moldados
current.globalBimData	BIM

O detalhamento da classe de cada um deles será feito neste capítulo, junto a cada objeto onde são aplicáveis. Nem todos os dados atuais são armazenados globalmente; alguns são definidos por pavimento.

Modos de visualização Model.visData

Modos de visualização são definidos para o modelo na tela do Modelador e diversos desenhos de formas. O objeto que define os modos depende então de onde devem ser aplicados:

```
current.visData.GetVisModes(imode)
```

Objeto de modos de visualização VisModes()

`imode` Aplicar modos em VISMODE_xxxx

Retorna:

`Vismodes` Objeto VisModes() para controle de visualização

Os modos possíveis são:

Constante	Uso
TQSModel.VISMODE_MATUAL	Modo de visualização atual
TQSModel.VISMODE_FORMAS	Desenho de formas FORnnnn.DWG

TQSModel.VISMODE_VERIFI	Modo de verificação
TQSModel.VISMODE_MODELO	Desenho MODELO.DWG
TQSModel.VISMODE_GRUPOA	Grupo A de visualização do usuário
TQSModel.VISMODE_GRUPOB	Grupo B de visualização do usuário

O objeto VisMode() retornado será detalhado adiante.

Objeto do pavimento: Floor()

O Floor() é responsável pela criação de todos os objetos do Modelador, assim como a iteração por eles. As tarefas são divididas nos seguintes subobjetos:

Objeto	Uso
current	Dados atuais por pavimento, CurrentFloorData()
create	Cria objetos na planta. Dependendo do objeto, ele pode ir para a lista espacial.
iterator	Itera pelos objetos da planta
util	Utilidades diversas

Dados atuais por pavimento Floor.current

Os dados atuais por pavimento (floor.current), junto com os dados atuais globais, são usados na criação de elementos novos. A lógica é semelhante à criação de objetos novos dentro do Modelador. São eles:

Objeto	Uso
floorDrawingData	Dados de desenho
floorLoadData	Cargas usadas na planta
floorBeamData	Dados de vigas
floorSlabData	Dados de lajes
floorColumnData	Dados de pilares
floorCutData	Dados de cortes
floorAxisData	Dados de eixos

Detalharemos cada um destes objetos junto dos respectivos elementos onde se aplicam.

Criação de objetos com Floor.create

Este subobjeto de Floor permite a criação de todos os elementos estruturais. O objeto Floor mantém a lista de todos os objetos planos do pavimento, mas repassa a criação de pilares e objetos inclinados para uma lista espacial. Veja os exemplos TSTMODEL1.PY e TSTMODEL2.PY que criam novos modelos estruturais. A seguir, mostraremos as funções para criação de objetos.

O objeto de pilares é usado para representar 3 tipos de objetos estruturais: Pilares, Pilaretes e Fundações. O objeto efetivamente definido depende dos dados de pilares definidos.

```
create.CreateColumn(xins, yins)
```

Criação de um pilar

`xins` Ponto de inserção X

`yins` Ponto de inserção Y

Retorna:

`Objeto` tipo Column()

Por exemplo, considerando o modelo em model, o pavimento atual em floor e as coordenadas de inserção xins,yins, podemos construir um pilar retangular 20x50 cm assim:

```
columnData = model.current.globalColumnData.columnData
columnData.columnStarts = TQSModel.COLUMNSTART_NASCEPIFU
columnGeometry = columnData.columnGeometry
columnGeometry.sectionType = TQSModel.COLUMNTYPE_R
columnGeometry.sectionB1 = 20.
columnGeometry.sectionH1 = 50.
column = floor.create.CreateColumn (xins, yins)
```

A variável columnData contém dados atuais para a criação de pilares. Na criação de pilaretes e de fundações, usamos as variáveis shortColumnData e foundationData no lugar de columnData. Ainda no exemplo acima, a variável columnStarts recebeu uma constante que diz que o pilar nasce na fundação, e a variável sectionType, o formato do pilar retangular.

Nos pilares, assim como nos demais objetos do Modelador, detalharemos os dados atuais e os dados de cada objeto junto da definição de cada um deles.

A definição de pilaretes e fundações é semelhante, e gera um objeto armazenado na estrutura de dados de pilares:

```
create.CreateShortColumn(xins, yins)
```

Criação de um pilarete

`xins` Ponto de inserção X

`yins` Ponto de inserção Y

Retorna:

`Objeto` tipo Column()

```
create.CreateFoundation(xins, yins)
```

Criação de uma fundação

`xins` Ponto de inserção X

`yins` Ponto de inserção Y

Retorna:

`Objeto` tipo Column()

Os demais objetos que podem ser criados são:

```
create.CreateBeam(xy)
```

Cria viga no plano

xy ((x1,y1),(x2,y2)...) coordenadas da viga

Retorna:

`Objeto` tipo Beam()

```
create.CreateBeamOpening(beam, xins, yins)
```

Cria um furo horizontal em viga

`beam` Objeto da viga

`xins,yins` Centro do furo em planta no eixo da viga

Retorna:

`Objeto` tipo BeamOpening()

```
create.CreateSlabContour(x1, y1, x2, y2)
```

Cria contorno auxiliar de laje - bordo livre

`x1,y1,x2,y2` Coordenadas da linha cm

Retorna:

`Objeto` tipo SlabContour()

```
create.CreateSlab(xins, yins, angpri)
```

Cria laje dentro de contorno formado por vigas, pilares ou contorno auxiliar

`xins` Ponto X dentro da laje

`yins` Ponto Y dentro da laje

`angpri` Ângulo principal, graus

Retorna:

`Objeto` tipo Slab()

```
create.CreateDropPanel(polig)
```

Cria capitel

`polig` Poligonal Polygon() fechada que define o capitel

Retorna:

`Objeto` tipo DropPanel()

```
create.CreateSlabMould(slab, xnerv, ynerv)
```

Cria uma forma/cubeta de laje nevurada

`slab` Laje Slab() onde vai a forma

`xnerv` Posição do centro da forma

`ynerv` Posição do centro da forma

Retorna:

`Objeto` tipo SlabMould()

```
create.CreateSlabOpening(polig, irecorte, isobrecapa)
```

Cria capitel

`polig` Poligonal fechada que define o furo em planta

`irecorte` (0) Furo (respeita nervuras) (1) Recorte

`isobrecapa` (0) Normal (1) Só na capa de laje nervurada

Retorna:

`Objeto` tipo SlabOpening()

```
create.CreateSoilPressureLoad(xini, yini, xfin, yfin)
```

Cria uma carga de empuxo

`xini` Ponto inicial X

`yini` Ponto inicial Y

`xfin` Ponto final X

`yfin` Ponto final X

Retorna:

`Objeto` tipo SoilPressureLoad()

```
create.CreateAdditionalLoad(xins, yins)
```

Cria carga adicional em laje

`xins` Ponto X na laje com o símbolo da carga

`yins` Ponto Y

Retorna:

`Objeto` tipo AdditionalLoad()

```
create.CreateAreaDistributedLoad(polig)
```

Cria distribuída em laje dentro de uma poligonal fechada tf/m2

`polig` Poligonal Polygon() onde a área é aplicada

Retorna:

`Objeto` tipo AreaDistributedLoad()

```
create.CreateConcentratedLoad(xins, yins)
```

Cria carga concentrada tf

`xins` Ponto X na laje com o símbolo da carga

`yins` Ponto Y

Retorna:

`Objeto` tipo ConcentratedLoad()

```
create.CreateLinearyDistributedLoad(xins1, yins1, xins2, yins2)
```

Cria linear tf/m sobre viga ou laje

`xins1` Ponto inicial X na laje com o símbolo da carga

`yins1` Ponto inicial Y

`xins2` Ponto final X na laje com o símbolo da carga

`yins2` Ponto final Y

Retorna:

`Objeto` tipo LinearyDistributedLoad()

Leitura do pavimento com Floor.iterator

O subobjetos iterator são do tipo SMObject e permitem a leitura de todos os objetos de um pavimento. Você pode

listar e editar os objetos lidos.

```
iterator.GetNumObjects(itype)
```

Retorna o número de objetos de uma lista de uma planta

`itype` TQSModel.TYPE_xxxx

Retorna:

Número de objetos deste tipo

```
iterator.GetObject(itype, iobject)
```

Retorna um objeto de uma lista

`itype` TQSModel.TYPE_xxxx

`iobject` 0..GetNumObjects()-1

Retorna:

`smobject` Objeto derivado de SMObject ou None

```
iterator.FindObject(identobj)
```

Acha um objeto dado seu identificador único

`identobj` Identificador único de um objeto

Retorna:

`objme` Objeto ou None se não achar

`itype` Tipo do objeto TQSModel.TYPE_xxxx

`index` Índice de acesso

`istat` (0) Ok (1) Não achou

Utilitários do pavimento: Floor.util

São utilitários que agem sobre os objetos do pavimento.

```
util.DoIntersections()
```

Refaz as intersecções de vigas, pilares e lajes em uma planta. Necessário para reconhecimento de intersecções entre vigas e pilares, além do reconhecimento do contorno das lajes. É uma operação lenta, assim deve-se deixar para executar quando for necessário o conhecimento das intersecções de vigas e pilares, ou o contorno das vigas para a criação de lajes.

util.DistributeSlabMould(slab, slabMould)

Distribui formas de laje nervurada em uma laje, a partir de uma fornecida

`slab` Laje Slab() onde será feita a distribuição

`slabMould` Forma de nervura SlabMould() como semente

Identificação de elementos - SMObjectIdent()

O objeto SMObjectIdent() é usado em diversos elementos do Modelador, e contém dados de identificação e posição de um elemento.

`objectNumber`

Número do elemento

`objectTitle`

Título alfanumérico ou "" se não definido

`textPosition`

Retorna objeto TextPosition() com posição de texto editável

`renumerable`

Elemento renumeravel (0) Não (1) Sim

Identificação de pré-moldados PreCastGroup()

Uma das características principais das peças pré-moldadas é sua repetição, o que facilita a fabricação. Além do número dos elementos atribuídos pelo Modelador, os elementos pré-moldados são nomeados por grupos, que podem ser gerados pelo Modelador. O acesso aos nomes usados nos grupos é pelos objetos tipo PreCastGroup().

`floorPlanGroup`

Nome do grupo de formas

`floorPlanNumber`

Número único de elemento do grupo

`reinforcementGroup`

Nome do grupo de armação

`beamLoad`

Carga distribuída para vigas com grupo de armação renumerado (string)

Objeto Polygon

O objeto Polygon() é usado por alguns tipos de objeto, como pilares, capitéis, furos em lajes, cargas distribuídas por área, e outros. Trata-se de uma poligonal com pontos definida em 2D, aberta ou fechada, conforme o uso.

`Clear()`

Esvazia uma poligonal

`NumPts()`

Retorna o número de pontos

`GetPt(ipt)`

Lê um ponto

`ipt` Índice do ponto 0..NumPts()-1

Retorna:

`x, y` Coordenadas do ponto

`GetPts()`

Lê a poligonal inteira como uma matriz

Retorna:

`x[], y[]`

`Enter(x, y)`

Entra um ponto no final da poligonal

`x, y` Coordenadas do ponto

`Set(ipt, x, y)`

Define as coordenadas de um ponto na posição ipt

`ipt` Índice do ponto 0..NumPts()-1

`x, y` Coordenadas do ponto

`Insert(ipt, x, y)`

Entra as coordenadas de um ponto na posição ipt e empurra os demais para frente

`ipt` Índice do ponto 0..NumPts()-1

`x, y` Coordenadas do ponto

Carga básica Load()

Este objeto representa uma força ou momento aplicado em um elemento estrutural. Ele é usado para compor cargas concentradas, lineares e distribuídas por área em objetos do Modelador.

`SetLoad(itype)`

Define o tipo de carga

`itype` Tipo LOADTYPE_xxxxx

Tipo	Uso
`TQSModel.LOADTYPE_DISARE`	Distribuída por área
`TQSModel.LOADTYPE_DISLIN`	Distribuída linear
`TQSModel.LOADTYPE_CONCEN`	Concentrada
`TQSModel.LOADTYPE_MOMCON`	Momento concentrado

`SetAlpha(icase, ialpha)`

Define modo de carga numérico ou alfanumérico

`icase` Caso de carga 1..número de casos de grelha do pavimento

`ialfa` Carga (0) Numérica (1) Alfanumérica da tabela de tipos de cargas

`SetLoadName(icase, loadname)`

Para cargas alfanuméricas, define o nome da carga na tabela de cargas

`icase` Caso de carga 1..número de casos de grelha do pavimento

`loadname` Nome da carga

`SetMainLoad(icase, loadval)`

Valor da carga permanente

`icase` Caso de carga 1..número de casos de grelha do pavimento

`loadval` Valor da carga em unidades coerentes

`SetLiveLoad(icase, loadval)`

Valor da carga variável

`icase` Caso de carga 1..número de casos de grelha do pavimento

`loadval` Valor da carga em unidades coerentes

`SetWallHeight(icase, iwall, wallheight)`

Define o uso da altura da parede. Paredes importadas ficam à disposição no modelo para visualização em 3D.

`icase` Caso de carga 1..número de casos de grelha do pavimento

`iwall` Usa parede para carga por área (0) Não (1) Sim

`wallheight` Altura da parede (m) ou (0) Para PD da planta

Superclasse SMObject dos objetos do Modelador

Os objetos do Modelador são todos derivados da classe SMObject. Assim, tem os mesmos atributos abaixo.

`type`

Tipo do objeto (TQSModel.TYPE_xxxx)

`identobj`

Identificador único do objeto

`number`

Número do objeto

`title`

Título opcional do objeto

`formattedTitle`

Título considerando número e título opcional

A rotina que retorna o GUID de um objeto (identificador universal único em forma de string) exige o fornecimento de parâmetros adicionais. Isto porque a estrutura de dados usada no Modelador pode ser diferente do programa para onde os objetos serão exportados. Por exemplo, uma única viga com dois vãos no Modelador pode ser dois objetos diferentes em um programa BIM. Também um piso com repetição, pode ter que ser exportado como pisos separados, e os elementos em cada piso precisarão de um GUID diferente.

`GetGuid(iadic, ipiso, itrecho)`

`iadic` Número adicional para diferenciar pisos e trechos

`ipiso` Piso

`itrecho` Trecho opcional

Retorna:

Identificador tipo GUID (string)

Pilar - Column(SMObject)

Pilares são armazenados na lista espacial do edifício. Apesar da lista especial, a criação do pilar é feita a partir do objeto do pavimento atual. Um pilar é composto por seções de pilar, independentes uma da outra. Um pilar pode ter mais de uma seção, de maneira que pode variar as dimensões de uma seção para outra, assim como variar a posição em planta. Em um pilar inclinado, também cada piso tem uma seção diferente, pois sua posição muda.

Todo pilar nasce em um determinado pavimento (não pode ser pavimento com repetição). A definição do pavimento

final é dada na estrutura de dados de cada seção.

Pilaretes são um caso especial de pilar, com apenas uma seção que nasce e/ou morre em um piso auxiliar.

Fundações, por questões históricas, são armazenadas na mesma estrutura de dados de pilares.

Dados atuais de pilares

São os dados usados na criação de um pilar novo, que valem para todos os pavimentos. Você também pode criar um novo com quaisquer dados atuais, e altera-los posteriormente diretamente na estrutura de dados do pilar. Nas descrições abaixo, model é o objeto do modelo estrutural obtido como em:

```
model = TQSModel.Model ()
```

A maior parte dos dados atuais é mantida globalmente. Entretanto alguns dados atuais são definidos por pavimento, e neste caso usa-se o pavimento atual em uma variável como floor:

```
floor = model.floors.GetFloor ("nome-da-planta")
```

Os dados atuais globais são estes:

```
model.current.globalBimData.userAttrib.GetUserAttrib(itypeattrib)
```

Retorna um objeto tipo UserAttrib(), com atributos definidos pelo usuário. Os tipos definidos em itypeattrib são do tipo BIM_USRATRTP_xxxxx. Esta chamada obtém dados BIM de pilares, fundações, vigas, lajes e elementos 3D:

Constante	Uso
`TQSModel.BIM_USRATRTP_PILAR`	Pilar
`TQSModel.BIM_USRATRTP_FUNDC`	Fundações
`TQSModel.BIM_USRATRTP_VIGAS`	Vigas
`TQSModel.BIM_USRATRTP_LAJES`	Lajes
`TQSModel.BIM_USRATRTP_ELM3D`	Elementos 3D

```
model.current.globalBimData.globalAttrib
```

Retorna um objeto tipo GlobalAttrib() com variáveis BIM definidas pelo usuário e que valem para todos os elementos exportados

```
model.current.globalColumnData.foundationData
```

Retorna dados que definem uma fundação - FoundationData().

```
model.current.globalColumnData.spreadFootingSideWalls
```

Valor do colarinho de uma sapata, cm

```
model.current.globalColumnData.foundationsTopFaceRecess
```

Fundações: Valor do rebaixo cm

```
model.current.globalColumnData.firstPileNumber
```

Fundações: Número de 1a estaca

```
model.current.globalColumnData.lastPileNumber
```

Fundações: Número da última estaca

```
columnData = model.current.globalColumnData.columnData
columnData = model.current.globalColumnData.shortColumnData
columnData = model.current.globalColumnData.foundationColumnData
```

Estas chamadas retornam um objeto da classe GlobalColumnData(), que contém dados do mesmo tipo respectivamente para pilares, pilaretes e fundações. Este objeto tem as seguintes variáveis a serem definidas:

```
columnData.columnDetailing
```

Dados de detalhamento de pilar – objeto ColumnDetailing()

```
columnData.columnGeometry
```

Dados de geometria de pilar - objeto ColumnGeometry()

```
columnData.columnInsertion
```

Dados de inserção de pilar - objeto ColumnInsertion()

```
columnData.columnPolygon
```

Contorno de pilar polygonal – objeto Polygon()

```
columnData.columnCoil
```

Mola do pilar no pórtico especial – Objeto ColumnCoil()

```
columnData.columnIdent
```

Identificação do pilar – Objeto SMObjectIdent()

`columnData.columnPrecastData`

Dados de pilar pré-moldados – Objeto ColumnPrecastData()

`columnData.columnStarts`

Pilar nasce em COLUMNSTART_xxxx

Constante	Uso
`TQSModel.COLUMNSTART_NASCEDIRT`	Nasce no solo (sem fundação definida)
`TQSModel.COLUMNSTART_NASCEVIGA`	Nasce em viga
`TQSModel.COLUMNSTART_NASCEPIFU`	Nasce em pilar ou fundação
`TQSModel.COLUMNSTART_NASCELAJE`	Nasce em laje

`columnData.columnModel`

Modelo de pilar: COLUMNUSE_xxxxxx

Constante	Uso
`TQSModel.COLUMNUSE_COMPRESSAO`	Padrão - forças de compressão
`TQSModel.COLUMNUSE_TRACAO`	Principalmente tração, pode compressão
`TQSModel.COLUMNUSE_COMPAT`	Pilar de compatibilização
`TQSModel.COLUMNUSE_ESCORA`	Escora (só compressão)
`TQSModel.COLUMNUSE_TIRANTE`	Tirante (só tração)

`columnData.columnWindSupport`

Suporte a vento: (0) não (1) sim

`columnData.columnSectionMode`

Dados de condições de contorno: (0) da seção (1) do pavimento

`columnData.columnInterference`

Verificação de interferências (0) não (1) sim

`columnData.columnSloped`

Pilar inclinado (0) não (1) sim

`columnData.columnIsAFoundation`

O objeto de pilar representa uma fundação (0) não (1) sim

`columnData.columnHinges`

Pilar articulado em (0)CONTPOR.DAT (1)base/topo (2)base (3)topo

`columnData.columnExport`

Pilar exportável para o 3D (0) não (1) sim

`columnData.columnNonLinearity`

Coeficientes de não linearidade (0)Pilar (1)Não fissurado (2)Fissurado

`columnData.columnBucklingX`

Coeficiente de flambagem X

`columnData.columnBucklingY`

Coeficiente de flambagem X

`columnData.columnDoubleStoryX`

Pé-direito duplo X (0) Geometria (1) Não (2) Sim

`columnData.columnDoubleStoryY`

Pé-direito duplo X (0) Geometria (1) Não (2) Sim

`columnData.columnConcreteFc`

Fck diferenciado de pilar (string)

`floor.current.column.columnFloorNames`

Retorna a lista de plantas de pilares ColumnFloorNames()

`columnData.column.columnBoundaryCond`

Retorna objeto de condições de contorno de grelha – ColumnBoundaryCond()

`columnData.column.columnCover`

Cobrimento do pilar na planta, cm

`columnData.column.columnExposure`

Pilar em contato com o solo (0) não (1) sim

`floor.current.column.columnDynHorLoad`

Carga dinâmica de veiculo em pilar (0) não (1) sim

Dados de pilares

Parte dos dados é comum aos dados atuais.

`hasFixedPoint`

(1) Se o pilar tem um ponto fixo

`fixedPointX`

X do ponto fixo do pilar

`fixedPointY`

Y do ponto fixo do pilar

`columnDetailing`

Dados de detalhamento de pilar – objeto ColumnDetailing()

`columnIdent`

Identificação do pilar – objeto SMObjectIdent()

`columnStarts`

Pilar nasce em COLUMNSTART_xxxx

Constante	Uso
`TQSModel.COLUMNSTART_NASCEDIRT`	Nasce no solo (sem fundação definida)
`TQSModel.COLUMNSTART_NASCEVIGA`	Nasce em viga

TQSModel.COLUMNSTART_NASCEPIFU	Nasce em pilar ou fundação
TQSModel.COLUMNSTART_NASCELAJE	Nasce em laje

columnData.columnModel

Modelo de pilar: COLUMNUSE_xxxxxx

Constante	Uso
TQSModel.COLUMNUSE_COMPRESSAO	Normal - forças de compressão
TQSModel.COLUMNUSE_TRACAO	Principalmente tração, pode compressão
TQSModel.COLUMNUSE_COMPAT	Pilar de compatibilização
TQSModel.COLUMNUSE_ESCORA	Escora (só compressão)
TQSModel.COLUMNUSE_TIRANTE	Tirante (só tração)

columnWindSupport

Suporte a vento: (0) não (1) sim

columnInterference

Verificação de interferências (0) não (1) sim

columnSloped

Inclinação do pilar (0) não (1) sim

columnCoil

Mola do pilar no pórtico espacial - ColumnCoil()

columnIsAFoundation

O objeto de pilar representa uma fundação (0) não (1) sim

foundationData

Dados de fundação - FoundationData()

foundationDefined

Fundação definida (0) não (1) sim

`isShortColumn`

O pilar é (0) normal (1) pilarete

`ShortColumnAuxiliaryFloor`

Piso auxiliar de base do pilarete

`columnPrecastData`

Dados de pilar pré-moldados - ColumnPrecastData()

`columnExport`

Pilar exportável para o 3D (0) não (1) sim

`GetFoudation()`

Retorna pilar ou fundação que servem de apoio ao pilar atual ou None

`columnNonLinearity`

Coeficientes de não linearidade (0)Pilar (1)Não fissurado (2)Fissurado

`userAttrib`

Atributos de usuário - UserAttrib ()

`ColumnNumSections()`

Retorna o número de seções ColumnSection() de pilar

`ColumnGetSection(isec)`

Retorna uma seção

`isec` 0..ColumnNumSections()-1

Retorna:

`objeto` ColumnSection()

`ColumnCreateSection(isec, lastfloor)`

Insere nova seção de pilar. Empurra demais seções

A primeira seção é criada junto com o pilar

Cria com a geometria da seção anterior

`isec` 0..ColumnNumSections()-1

`lastfloor` Última planta desta seção (string)

Retorna:

`objeto` ColumnSection()

`ColumnSectionUpdateGeometry(columnSection)`

Atualiza uma seção de pilar após alteração de geometria

`columnSection` Objeto ColumnSection

Seção de pilar – ColumnSection()

Um pilar pode ter uma ou mais seções de pilares. A seção contém dados de geometria. Com múltiplas seções, podemos definir um pilar que varia ao longo dos lances, assim como pilares inclinados.

`column`

Retorna pilar Column() correspondente à seção

`insertionX`

Ponto de inserção X cm

`insertionY`

Ponto de inserção Y cm

`GetColumnFloorData(floor)`

Retorna objeto com Informações por planta - ColumnFloorData()

Retorna um objeto com informações por planta

`floor` Pavimento Floor()

Retorna:

`objeto` ColumnFloorData()

`GetColumnBoundaryCond(floor)`

Retorna objeto de condições de contorno de grelha

`floor` Pavimento Floor()

Retorna:

`objeto` ColumnBoundaryCond()

`SetColumnBoundaryMode(floor, idefcontpil)`

Define se as condições de contorno são por seção de pilar ou planta

`floor` Pavimento Floor()

`idefcontpil` Condições de contorno por (0) seção (1) planta

`GetShearWallStripNum()`

Retorna o número de lâminas de discretização de pilares parede

`GetShearWallStripNumPts(ilam)`

Retorna o número de pontos de uma lâmina

`ilam` Índice da lâmina 0..GetShearWallStripNum()-1

`GetShearWallStripPoint(ilam, ipt)`

Retorna o número de pontos de uma lâmina

ilamÍndice da lâmina 0..GetShearWallStripNum()-1

`ipt` Índice do ponto 0..GetShearWallStripNumPts()-1

Retorna:

`x, y` Um ponto da lâmina

`columnGeometry`

Dados de geometria de pilar - ColumnGeometry()

`sectionPolig`

Seção transversal - Polygon()

`sectionLastFloor()`

Última planta da seção (string)

`columnInsertion`

Dados de inserção de pilar - ColumnInsertion()

Geometria de pilares – ColumnGeometry()

Contém dados de seções padrão e seções catalogadas. A poligonal é definida por sectionPolig.

`sectionType()`

Tipo de seção COLUMNTYPE_xxx

Constante	Uso
TQSModel.COLUMNTYPE_R	Seção Retangular
TQSModel.COLUMNTYPE_L	Seção L
TQSModel.COLUMNTYPE_U	Seção U
TQSModel.COLUMNTYPE_C	Seção Circular
TQSModel.COLUMNTYPE_P	Seção Poligonal
TQSModel.COLUMNTYPE_N	Outras não padrão

`sectionB1`

Dimensão B1 de seções R/L/U/C

`sectionH1`

Dimensão H1 de seções R/L/U

`sectionB2`

Dimensão B2 de seções L/U

`sectionH2`

Dimensão H2 de seções L/U

`sectionAngle`

Ângulo de inserção da seção em graus

`sectionMaterial`

Título do material diferente do concreto (não padrão)

`sectionName`

Nome de seção não padrão – catalogada

Lista de planta de pilares - ColumnFloorNames()

Fora a planta onde um pilar nasce, cada planta desta lista representa um limite para uma seção de pilar.

`floorNumber()`

Número de plantas

`Clear()`

Limpa a lista de plantas e cria duas vazias: a primeira e a última do edifício

`EnterFloor(iplanta, nompla)`

Entra uma planta em posição fixa e empurra as demais plantas

`iplanta` Planta 0.. floorNumber()-1

`nompla` Nome da planta

`GetFloor(iplanta)`

Lê uma planta do pilar

`iplanta` Planta 0.. floorNumber()-1

Retorna:

`nompla` Nome da planta

Informações de pilar por seção ou pavimento - ColumnFloorData()

Certas informações de pilares podem ser definidas para todos os lances de mesma seção, ou para todos os lances na repetição de um pavimento.

`infoByFloor`

Informações de pilares por (0) Seção de pilar (1) Planta

`columnBoundaryCond`

Condições de contorno de pilares – objeto ColumnBoundaryCond()

`columnBucklingX`

Coeficiente de flambagem X

`columnBucklingY`

Coeficiente de flambagem Y

`columnDoubleStoryX`

Pé-direito duplo X (0) Geometria (1) Não (2) Sim

`columnDoubleStoryY`

Pé-direito duplo X (0) Geometria (1) Não (2) Sim

`columnConcreteFc`

Fck diferenciado de pilar (string)

`columnCover`

Cobrimento do pilar na planta, cm

`columnExposure`

Pilar em contato com o solo (0) não (1) sim

`columnHinges`

Pilar articulado em (0)CONTPOR.DAT (1)base/topo (2)base (3)topo

`columnDynHorLoad`

Carga dinâmica de veiculo em pilar (0) não (1) sim

`columnTransferBlock`

Dados de Bloco de transição – objeto ColumnTransferBlock()

Inserção interativa de pilares - ColumnInsertion()

São dados usados na inserção interativa de um pilar.

`insertionType`

Inserção de pilar por (0) Centro (1) Canto

`insertionCorner`

Numero do canto (0..) para insertionType == 1

`sectionCover`

Revestimento cm

`mediumPointInsertion`

Inserção por ponto intermediário (0) Não (1) Sim

Condições de contorno ColumnBoundaryCond()

São condições de contorno para o Modelo IV na discretização em grelha de cada piso.

`bearingModel`

Tipo de apoio em grelha COLUMNBEARING_xxxxx

Constante	Uso
`TQSModel.COLUMNBEARING_DEFAULT`	Default
`TQSModel.COLUMNBEARING_ARTCONT`	Articulado contínuo
`TQSModel.COLUMNBEARING_ARTINDP`	Articulado independente
`TQSModel.COLUMNBEARING_ELACONT`	Elástico continuo
`TQSModel.COLUMNBEARING_ELAINDP`	Elástico independente

`GetSpringValue(icoef)`

Retorna o valor da mola de rotação ou translação do apoio em pilar nas grelhas

`icoef` Coeficiente COLUMNSPRING_xxxx

Retorna:

coefmola Coeficiente de mola em tfm/rad (rotação) ou tf/m (translação)

Constante	Uso
`TQSModel.COLUMNSPRING_IRX`	Rx
`TQSModel.COLUMNSPRING_IRY`	Ry
`TQSModel.COLUMNSPRING_ITZ`	Tz
`TQSModel.COLUMNSPRING_ITX`	Tx
`TQSModel.COLUMNSPRING_ITY`	Ty

`SetSpringValue(icoef, coefmola)`

Define o valor da mola de rotação ou translação do apoio em pilar nas grelhas

icoefCoeficiente COLUMNSPRING_xxxx

`coefmola` Coeficiente de mola em tfm/rad (rotação) ou tf/m (translação)

madatoryPunchingShear

Armadura de punção obrigatória (0) Não (1) Sim

shearWallDiscretization

Pilar parede discretizado (0) Não (1) Sim, estimado ou discretizado

GetGap(igap)

Valor do gap (m) de restrição de apoio

igap Gap TIPO COLUMNCOIL_IGAP_xxxx

Retorna:

gap Valor do gap (m)

Constante	Uso
TQSModel.COLUMNCOIL_IGAP_XPOS	Gap Tx+
TQSModel.COLUMNCOIL_IGAP_XNEG	Gap Tx-
TQSModel.COLUMNCOIL_IGAP_YPOS	Gap Ty+
TQSModel.COLUMNCOIL_IGAP_YNEG	Gap Ty-
TQSModel.COLUMNCOIL_IGAP_ZPOS	Gap Tz+
TQSModel.COLUMNCOIL_IGAP_ZNEG	Gap Tz-

SetGap(igap, gap)

Valor do gap (m) de restrição de apoio

igap Gap TIPO COLUMNCOIL_IGAP_xxxx

gap Valor do gap (m)

Detalhamento de pilar – ColumnDetailing()

São informações usadas para detalhamento de pilares.

foundationHeight

Altura da fundação, cm

baseRecess

Rebaixo da base, cm

`topRecess`

Rebaixo no topo, cm

`isDetailable`

Pilar detalhável (0) não (1) sim

`isCurtain`

Pilar simula cortina (0) não (1) sim

`canStartOutOfFoundation`

Pode nascer fora do piso fundação (0) não (1) sim

`isFictitious`

Pilar fictício (0) não (1) sim

`isShearWall`

Pilar parede para o BIM

Molas de pilar no portico especial - ColumnCoil()

São as molas sob o pilar ou fundação usados no portico especial.

`GetSupportType(ires)`

Tipo de apoio de pilar

`ires` Restrição a considerar COLUMNCOIL_IRES_xxxx

Retorna:

`iapopor` Tipo de apoio COLUMNCOIL_IAPOPOR_xxxx

Constante	Uso
`TQSModel.COLUMNCOIL_IRES_IRX`	Rotação X
`TQSModel.COLUMNCOIL_IRES_IRY`	Rotação Y
`TQSModel.COLUMNCOIL_IRES_IRZ`	Rotação Z
`TQSModel.COLUMNCOIL_IRES_ITX`	Translação X

TQSModel.COLUMNCOIL_IRES_ITY	Translação Y
TQSModel.COLUMNCOIL_IRES_ITZ	Translação Z

Constante	Uso
TQSModel.COLUMNCOIL_IAPOPOR_DEFAULT	Engaste ou padrão
TQSModel.COLUMNCOIL_IAPOPOR_ARTICULADO	Articulado
TQSModel.COLUMNCOIL_IAPOPOR_ELASTICO	Elástico
TQSModel.COLUMNCOIL_IAPOPOR_RECALQUE	Recalque

```
SetSupportType(ires, iapopor)
```

Tipo de apoio de pilar

ires Restrição a considerar COLUMNCOIL_IRES_xxxx

iapopor Tipo de apoio COLUMNCOIL_IAPOPOR_xxxx

```
GetCoil(ires)
```

Coeficiente de mola

ires Restrição a considerar COLUMNCOIL_IRES_xxxx

Retorna:

pormola Coeficiente de mola à Rotação tfm/rad Translação tf/m

```
SetCoil(ires, pormola)
```

Coeficiente de mola

ires Restrição a considerar COLUMNCOIL_IRES_xxxx

pormola Coeficiente de mola à Rotação tfm/rad Translação tf/m

```
GetGap(igap)
```

Valor do gap (m) de restrição de apoio

igap Gap TIPO COLUMNCOIL_IGAP_xxxx

Retorna:

gap Valor do gap (m)

Constante	Uso
TQSModel.COLUMNCOIL_IGAP_XPOS	Gap Tx+
TQSModel.COLUMNCOIL_IGAP_XNEG	Gap Tx-
TQSModel.COLUMNCOIL_IGAP_YPOS	Gap Ty+
TQSModel.COLUMNCOIL_IGAP_YNEG	Gap Ty-
TQSModel.COLUMNCOIL_IGAP_ZPOS	Gap Tz+
TQSModel.COLUMNCOIL_IGAP_ZNEG	Gap Tz-

SetGap(igap, gap)

Valor do gap (m) de restrição de apoio

igap Gap TIPO COLUMNCOIL_IGAP_xxxx

gap Valor do gap (m)

Dados de fundações - FoundationData()

Os objetos desta classe podem representar fundações em sapatas, blocos, tubulões e blocos sobre tubulões.

type

Tipo de fundação FOUNDATION_ITIPOFUNDAC_xxxx

Constante	Uso
TQSModel.FOUNDATION_ITIPOFUNDAC_SAPATA	Sapata
TQSModel.FOUNDATION_ITIPOFUNDAC_BLOCO	Bloco sobre estacas
TQSModel.FOUNDATION_ITIPOFUNDAC_TUBULAO	Tubulão

ficticiousColumn

Pilar fictício definido (0) não (1) sim

ficticiousColumnXDim

Dimensão X de pilar fictício cm

ficticiousColumnYDim

Dimensão Y de pilar fictício cm

beamSupport

Vigas apoiam na fundação (0) não (1) sim

footingXDim

Sapatas, dimensão X da base cm

footingYDim

Sapatas, dimensão Y da base cm

footingTopXDim

Sapatas, dimensão X do topo cm

footingTopYDim

Sapatas, dimensão Y do topo cm

footingTopXExc

Sapatas, excentricidade X do topo cm

footingTopYExc

Sapatas, eycentricidade Y do topo cm

footingTopExcLeft

Sapata de divisa à esquerda (0) não (1) sim

footingTopExcAbove

Sapata de divisa acima (0) não (1) sim

footingTopExcRight

Sapata de divisa à direita (0) não (1) sim

footingTopExcUnder

Sapata de divisa abaixo (0) não (1) sim

footingHeight

Altura total cm

`footingXBaseBoard`

Altura X do rodapé cm

`footingYBaseBoard`

Altura Y do rodapé cm

`pileCapX`

Base X do bloco, cm

`pileCapY`

Base Y do bloco, cm

`pileCapHeight`

Altura do bloco, cm

`pileCapPiles`

Número de estacas do bloco

`pileCapFormat`

Formato (0..) Varia com o número de estacas

`pileCapDiameter`

Diâmetro da estaca cm

`pileCapHeightW`

Altura da estaca dentro do bloco cm

`pileCapPileXDist`

Distância X entre eixos de estacas cm

`pileCapPileYDist`

Distância Y entre eiyos de estacas cm

pileCapPileFaceDist

Distância do eixo da estaca à face do bloco cm

pileCapDistMode

Cálculo da dimensão do bloco (0) dimensões fornecidas (1) distância entre estacas

pileCapPileHeight

Altura das estacas cm

pileCapOverPiers

Bloco sobre tubulões (0) não (1) sim

pierShaftDiameter

Tubulão - diâmetro do fuste cm

pierShaftHeight

Tubulão - altura do fuste cm

pierBaseboardDiameter

Tubulão - diâmetro do rodapé cm

pierBaseboardHeight

Tubulão - altura do rodapé cm

pierBaseboardCone

Tubulão - altura do cone cm

pierFalseEllipse

Tubulão com base alargada (0) não (1) sim

pierFalseEllipseWidth

Tubulão - alargamento da base cm

pierFalseEllipseAngle

Tubulão - Rotação da base, sistema global, graus

`pocketFoundation`

Fundação em cálice (0) não (1) sim

`pocketFoundationFormat`

Cálice: Formato FOUNDATION_FORM_xxxxx

Constante	Uso
`TQSModel.FOUNDATION_FORM_RETANGULAR`	Cálice: Formato retangular
`TQSModel.FOUNDATION_FORM_CIRCULAR`	Cálice: Formato circular

`pocketFoundationRoughness`

Cálice: Rugosidade FOUNDATION_RUGO_xxxx

Constante	Uso
`TQSModel.FOUNDATION_RUGO_PADRAO`	Cálice: Rugosidade conforme critérios
`TQSModel.FOUNDATION_RUGO_SULISA`	Cálice: Superfície lisa
`TQSModel.FOUNDATION_RUGO_RUGOSA`	Cálice: Superfíce rugosa, pilar também
`TQSModel.FOUNDATION_RUGO_CHAVECIS`	Cálice: Chave de cisalhamento

`pocketFoundationXDim`

Cálice: dimensão X em planta cm (D circular)

`pocketFoundationYDim`

Cálice: dimensão Y em planta cm (D circular)

`pocketFoundationXWall`

Cálice: dimensão parede X cm

`pocketFoundationYWall`

Cálice: dimensão parede Y cm

`pocketFoundationXExc`

Cálice: Excentricidade X em relação ao centro da fundação cm

`pocketFoundationYExc`

Cálice: Excentricidade Y em relação ao centro da fundação cm

`pocketFoundationXInc`

Cálice: Caimento X planta parede interna cm

`pocketFoundationYInc`

Cálice: Caimento Y planta parede interna cm

`pocketFoundationHeight`

Cálice: Altura cm

`pocketFoundationRecess`

Cálice: Rebaixo em relação ao nível da fundação cm

Dados de pré-moldados - ColumnPrecastData()

Estes são dados específicos para pilares pré-moldados.

`precastRegion`

Pilar pré-moldado: região construtiva

`isPrecast`

Pilar pré-moldado: (0) Não (1) Sim

`rebarBundling`

Pilar pré-moldado: Alojamento de armadura (0)Critérios (1)TQS Pilar (2)Feixe

`rainWaterPipe`

Tubo de água pluvial - objeto RainWaterPipe()

`liftingAnchors`

Alça de içamento de pilares pré-moldados – objeto LiftingAnchors()

`liftOpenning`

Furos de levantamento de pilares pré-moldados – objeto LiftOpenning()

`preCastGroup`

Grupo de pré-moldados – objeto PreCastGroup()

Pré-moldados: tubo de água pluvial RainWaterPipe()

É um tubo de água pluvial que pode ser definido dentro de um pilar pré-moldado.

`pipeDiameter`

Diâmetro do tubo cm

`minFunnelWidth`

Diâmetro do funil menor cm

`maxFunnelWidth`

Diâmetro do funil maior cm

`rotation`

Ângulo adicional em relação ao pilar em graus

`baseDistance`

Distância de saída da base cm (0.) Adota default

`tubeIdentification`

Identificador do tubo

Pré-moldados: alças de içamento LiftingAnchors()

São as alças pelas quais as peças são suspenas e transportadas na fábrica.

`GetNumLiftingAnchors()`

Retorna o número de alças/rebaixos definidos

`GetLiftingRecess(ialca)`

Retorna um rebaixo de alça\n

`ialca` 0..GetNumLifting()-1

Retorna:

`dfs` Rebaixo de alça cm

`SetLiftingRecess(ialca, dfs)`

Define um rebaixo de alça

`ialca` 0..GetNumLifting()-1

`dfs` Rebaixo de alça cm

`EnterLiftingRecess(dfs)`

Entra um rebaixo de alça no final da lista

`dfs` Rebaixo de alça cm

`EraseLiftingAnchor(ialca)`

Apaga um rebaixo de alça

`ialca` 0..GetNumLifting()-1

`SortLiftingAnchors()`

Classifica as alças por ordem de rebaixo

`liftingAnchorId`

Identificador de alças (string)

`liftingPosition`

Alças em posição padrão (0) Não (1) Sim

`liftingAnchorsDefined`

Alças definidas (0) Não (1) Sim

`topLiftingAngle`

Ângulo da alça de topo em graus

Pré-moldados: furos de levantamento LiftOpenning()

São os furos por ondem passam cabos para levantar as peças pré-moldadas na montagem.

`GetNumLiftingOpennings()`

Retorna o número de furos de içamento definidos

`GetLiftingRecess(ifur)`

Retorna um rebaixo de furo

`ifur` 0..GetNumLiftingOpennings()-1

Retorna:

`dfs` Rebaixo de furo cm

`SetLiftingRecess(ifur, dfs)`

Define um rebaixo de furo

`ifur` 0..GetNumLiftingOpennings()-1

`dfs` Rebaixo de furo cm

`EnterLiftingRecess(dfs)`

Entra um rebaixo de furo no final da lista

`dfs` Rebaixo de furo cm

`EraseLiftingOpenning(ifur)`

Apaga um rebaixo de furo

`ifur` 0..GetNumLifting()-1

`SortLiftingOpennings()`

Classifica as furos por ordem de rebaixo

`liftingDiameter`

Diâmetro do furo, cm

`liftingEccentricity`

Excentricidade do furo (cm) em relação ao CG do pilar

`liftingPosition`

Furos em posição padrão (0) Não (1) Sim

`liftOpenningDefined`

Furos de içamento definidos (0) Não (1) Sim

Viga - Beam(SMObject)

Vigas são objetos compostos por trechos, definidos por nós de vigas. Um nó de viga ao mesmo tempo define as coordenadas do nó, e as propriedades do trecho até o próximo nó (o último nó não tem trecho definido).

Você não define necessariamente todos os nós de uma viga. Você define os nós originais, em geral o primeiro e último nó, e intermediários se houver mudança de direção. Os demais nós são gerados automaticamente após atualização das intersecções com outras vigas e pilares do pavimento.

Alguns dados de geometria e inércia, podem ser definidos especificamente para um trecho. Assim, temos dados que valem para toda a viga e dados que são específicos por trecho.

Dados atuais de vigas

São dados usados na criação de uma viga, antes dela existir. Você também pode criar uma viga sem considerar os dados atuais, e modifica-la depois de criada. Considerando o modelo e pavimento:

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")
```

temos os seguintes dados atuais:

```
model.current.globalBeamData.beamRestraint
```

Objeto de articulação de vigas BeamRestraint()

```
model.current.globalBeamData.firstSlopedBeamNumber
```

Número da primeira viga inclinada

```
model.current.globalBeamData.slopedBeamIdent
```

Identificação da viga inclinada

```
model.current.globalBimData.userAttrib
```

Dados BIM de usuário – Objeto UserAtrib()

```
floor.current.floorBeamData.beamGeometry
```

Geometria da viga (todos os trechos) – Objeto BeamGeometry()

```
floor.current.floorBeamData.beamInertia
```

Inércia de viga (todos os trechos) – Objeto BeamInertia()

```
floor.current.floorBeamData.beamBond
```

Vinculações e outros dados de viga – objeto BeamBond()

`floor.current.floorBeamData.beamInsertion`

Dados de inserção de vigas – objeto BeamInsertion()

`floor.current.floorBeamData.beamTemperShrink`

Temperatura / retração de vigas – objeto TemperatureShrink()

`floor.current.floorBeamData.beamDetailing`

Detalhamento de vigas – objeto BeamDetailing()

`floor.current.floorBeamData.beamExport`

(1) Se viga exportável para o 3D/BIM

`floor.current.floorLoadData.load`

Objeto Load() com carga distribuída em toda a viga

Dados de vigas

São dados de uma viga existente, e podem ser editados.

`beamIdent`

Identificação da viga – Objeto SMObjectIdent()

`NumNodes()`

Número de nós de uma viga (objetos BeamNode())

`GetBeamNode(ino)`

Retorna um nó da viga na forma de um objeto BeamNode()

`GetBeamGeometry()`

Geometria de viga - válida nos trechos sem definição específica (SetSegmentBeamGeometryDefined) – Objeto BeamGeometry()

`GetSegmentBeamGeometry(ino)`

Geometria de viga - do trecho ou geral (conforme SetSegmentBeamGeometryDefined) – Objeto BeamGeometry()

`GetSegmentBeamGeometryDefined(ino)`

Retorna (1) se a geometria do trecho ino é específica ou (0) se é geral

`SetSegmentBeamGeometryDefined(ino, idefgeovig)`

Define (1) se a geometria do trecho ino é específica ou (0) se é geral

`GetBeamInertia()`

Inércia de viga - válida nos trechos sem definição específica (SetSegmentBeamInertiaDefined) – Objeto BeamInertia()

`GetSegmentBeamInertia(ino)`

Inércia de viga - Do trecho ou geral, conforme SetSegmentBeamInertiaDefined – Objeto BeamInertia()

`SetSegmentBeamInertiaDefined(ino, idefinervig)`

Define (1) se a inércia do trecho ino é específica ou (0) se é geral

`beamBond`

Vinculações e outros dados de viga – objeto BeamBond()

`beamInsertion`

Dados de inserção de vigas – objeto BeamInsertion()

`GetLoad(floor)`

Retorna objeto Load() com carga distribuída em toda a viga

`floor` Objeto Floor(), com dados do pavimento atual

`temperShrink`

Temperatura / retração de vigas – objeto TemperatureShrink()

`beamDetailing`

Detalhamento de vigas – objeto BeamDetailing()

`auxiliaryFloor`

Piso auxiliar

`userAttrib`

Atributos de usuário – objeto UserAttrib()

`beamExport`

(1) Se viga exportável para o 3D/BIM

Nó de viga - BeamNode()

Um nó de viga representa o nó, original ou gerado por intersecções, e as propriedades do trecho em seguida. O último nó não representa o trecho, que não existe.

`nodeX`

X do nó da viga

`nodeY`

Y do nó da viga

`crossingType`

Tipo de vinculação do nó da viga BEAMCROSSING_xxxx

Constante	Uso
`TQSModel.BEAMCROSSING_INDEFINIDO`	Indefinido
`TQSModel.BEAMCROSSING_RECEBE`	Recebe carga
`TQSModel.BEAMCROSSING_CRUZAMENTO`	Cruzamento sem apoio identificado
`TQSModel.BEAMCROSSING_APOIAVIGA`	Apoia em viga
`TQSModel.BEAMCROSSING_APOIAPILAR`	Apoia em pilar
`TQSModel.BEAMCROSSING_N`	Neutro

`GetSlabConnection(idireito)`

Retorna engastamento de um lado da viga com a laje

`idireito` (0) Esquerdo (1) Direito

Retorna:

`iengast` Tipo de engaste BEAMCONNECTION_XXX

`fatengast` Fator de engaste 0..1 para iengast==BEAMCONNECTION_FATEMGS

Constante	Uso
TQSModel.BEAMCONNECTION_DEFAULT	Default
TQSModel.BEAMCONNECTION_ENLIVRE	Livre
TQSModel.BEAMCONNECTION_ARTCMXY	Articulado mx,my
TQSModel.BEAMCONNECTION_ARTMXYF	Articulado mx,my,fx
TQSModel.BEAMCONNECTION_FATEMGS	Engastado com fatengast

SetSlabConnection(idireito, iengast, fatengast)

Define engastamento de um lado da viga com a laje

idireito (0) Esquerdo (1) Direito

iengast Tipo de engaste BEAMCONNECTION_XXX

fatengast Fator de engaste 0..1 para iengast==BEAMCONNECTION_FATEMGS

GetRestraint(ifinal)

Retorna objeto de engastamento no início ou fim de um trecho

ifinal (0) Inicial (1) Final

Retorna:

Objeto BeamRestraint()

arcSegment()

Retorna objeto ArcSegment() com dados de arco no trecho atual

Geometria de viga - BeamGeometry()

Contém dados de geometria da viga, que podem valer para toda a viga ou somente para um trecho.

width

Largura de viga retangular cm

depth

Altura de viga retangular cm

eccentricity

Excentricidade lateral cm

`recess`

Rebaixo em cm, positivo para baixo.

`GetFloorRecess(floor, beam)`

Rebaixo em cm, considerando piso auxiliar

`floor` Objeto Floor(), com dados de um pavimento

`beam` Objeto Beam() que contém esta geometria

Retorna:

`dfspiso` Rebaixo da viga, cm, considerando piso auxiliar

`sectionName`

Nome de seção não padrão – catalogada

`sectionRotation`

Ângulo de rotação da seção, graus

`sectionMaterial`

Título do material diferente do concreto (não padrão)

`variableSection`

Retorna objeto VariableSection() para a definição de mísula

`precastRegion`

Pré-moldados: Região construtiva

`beamPrecastData`

Retorna objeto BeamPrecastData() para a definição de viga pré-moldadas

`beamMetalSection`

Retorna objeto BeamMetalSection() para a definição de viga com seção metálica

Inércia de viga - BeamInertia()

São dados que afetam o cálculo de inércia da viga no modelo estrutural.

`effectiveFlange`

Mesa colaborante (0) Não (1) Sim

`maxEffectiveFlange`

Mesa colaborante máxima cm

`fixedEffectiveFlange`

Mesa colaborante fixa cm

`torsionalMomentMode`

Divisor de inércia à torção definido em (0) Critérios (1) torsionalMomentDivider ()

`torsionalMomentDivider`

Divisor de inércia à flexão, conforme torsionalMomentMode

`inertiaMomentDivider`

Divisor de inércia à flexão, se (!=0)

`notIntercectLeftSlab`

Viga em relação à laje à esquerda (0) intercepta (1) não intercepta

`notIntercectRightSlab`

Viga em relação à laje à direita (0) intercepta (1) não intercepta

`plasticAdaptability`

Capacidade de adaptação plástica à torção (0) Não (1) Sim

Vinculações de viga - BeamBond()

Definição de vinculações e outros dados de vigas.

`wideBeam`

Viga faixa (0) Não (1) Sim

`fixAtTheBegin`

Engaste inicial (0) Não (1) Sim

`fixAtTheEnd`

Engaste final (0) Não (1) Sim

`disableSelfWeight`

Desabilitar peso próprio (0) Não (1) Sim

`bearColumnByTheFaces`

Intersececção de pilar com faces (0) Não (1) Sim

`adjustEdges`

Ajuste automático das pontas (0) Não (1) Sim

`interceptOthersBeams`

Intersececção com outras vigas (0) Não (1) Sim

`startStrapBeam`

Alavanca inicial (0) Não (1) Sim

`endStrapBeam`

Alavanca final (0) Não (1) Sim

`transferGirder`

Transição (0) pela geometria (1) Sim (2) Não

`workAs`

Trabalha como (0) Viga (1) Tirante (2) Escora

Inserção interativa de viga - BeamInsertion()

Usados quando uma viga é inserida ou editada.

`insertBy`

Alinhamento (0) Esquerda (1) Direita (2) Eixo

`covering`

Revestimento das faces, cm

Detalhamento de viga - BeamDetailing()

Dados para detalhamento de vigas.

`detailable`

Detalhamento da viga (0) Normal (1) Não (2) De compatibilização

`cover`

Cobrimento diferenciado em cm

`exposure`

Em contato com o solo (0) Não (1) Sim (2) Exposta ao ambiente

`restrainColumns`

A viga trava pilares (0) Não (1) Sim

`simulatesCurtain`

Simula uma cortina (0) Não (1) Sim

`prestressed`

Protendida (0) Não (1) Sim

`vproInterface`

Interface com o VPRO (0) Não (1) Sim

Articulação de vigas – BeamRestraint()

Articulação nas extremidades da viga.

`GetRestraintCoef(irdir)`

Coeficientes de articulação na extremidade da viga

`Irdir` Direção (0) RX (1) RY (2) RZ (3) FZ

Retorna:

`iartic` Articulação definida (0) Não (1) Sim

`coef` Coeficiente de rotação ou translação

`SetRestraint(irdir, iartic, coef)`

Coeficientes de articulação na extremidade da viga

`irdir` Direção (0) RX (1) RY (2) RZ (3) FZ

`iartic` Articulação definida (0) Não (1) Sim

`coef` Coeficiente de rotação ou translação

`GetPlasticMoments(itipopl, imompos)`

Coeficientes de plastificação impostos

Valor máximo de momento fletor em uma ligação

`itipopl` Definição (0) My (1) Mz

`imompos` Definição (0) Máximo negativo (1) Positivo

Retorna:

`isfmax` (0) Conforme critérios (1) Não definido (2) em esfmax

`esfmax` Momento máximo tfm

`SetPlasticMoments(itipopl, imompos, isfmax, esfmax)`

Coeficientes de plastificação impostos

Valor máximo de momento fletor em uma ligação

`itipopl` Definição (0) My (1) Mz

`imompos` Definição (0) Máximo negativo (1) Positivo

`isfmax` Conforme critérios (1) Não definido (2) em esfmax

`esfmax` Momento máximo tfm

Segmento de arco de viga - ArcSegment()

Descreve o próximo trecho da viga como um segmento de arco.

`startNode`

Nó inicial da viga 0..NumNodes()-1

`endNode`

Nó final da viga 0..NumNodes()-1

`signal`

Sinal do arco (1) anti-horário (-1) horário

`centerX`

X do centro do arco cm

centerY

Y do centro do arco cm

Mísula e seção variável em viga - VariableSection()

Mísulas e seções variáveis não prismáticas em vigas não são calculadas no TQS. A sua definição permite construir um modelo 3D mais realista.

corbelDefined

Mísula definida (0) Não (1) Sim

startBottomHeight

Mísula: Altura inferior inicial cm

endBottomHeight

Mísula: Altura inferior final cm

startTopHeight

Mísula: Altura superior inicial cm

endTopHeight

Mísula: Altura superior final cm

startLeftWidth

Mísula: Largura esquerda inicial cm

endLeftWidth

Mísula: Largura esquerda final cm

startRightWidth

Mísula: Largura direita inicial cm

endRightWidth

Mísula: Largura direita final cm

variableSectionDefined

Seção variável definida (0) Não (1) Sim

```
startWidth
```

Seção variável: Largura inicial cm

```
startHeight
```

Seção variável: Altura inicial cm

```
startRecess
```

Seção variável: Rebaixo inicial cm

```
endWidth
```

Seção variável: Largura final cm

```
endHeight
```

Seção variável: Altura final cm

```
endRecess
```

Seção variável: Rebaixo final cm

Dados de viga pré-moldada - BeamPrecastData()

Em desenvolvimento.

Seção metálica - BeamMetalSection()

Em desenvolvimento.

Laje - Slab(SMObject)

Lajes planas no pavimento são definidas pelas coordenadas do título da laje. O contorno é obtido automaticamente a partir de uma malha de vigas, pilares e fechamentos de bordo. O contorno de uma laje é aquele onde está seu título. Laje sem contorno definido é uma condição de erro.

Em lajes sem vigas, é comum a condição de contornos externos com contornos internos (como por exemplo no núcleo do edifício). Para as laje externas, os contornos internos se comportam como furos.

Dados atuais de laje

São dados usados na criação de uma laje – existem antes da laje. Você pode também criar uma laje sem considerar estes dados, e modifica-la posteriormente.

Considerando o modelo e pavimento:

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")
```

temos os seguintes dados atuais:

```
floor.current.floorSlabData.slabIdent
```

Objeto SMObjectIdent() com identificação da laje

```
floor.current.floorSlabData.slabFirstNumber
```

Número da primeira laje

```
floor.current.floorLoadData.GetLoad(TQSModel.TPLOAD_CARLAJ)
```

Objeto Load() com carga distribuída em toda laje

```
floor.current.floorSlabData.slabGeometry
```

Objeto SlabGeometry() com dados de geometria de laje

```
floor.current.floorSlabData.slabGrid
```

Objeto SlabGrid() com dados para discretização de laje por grelha

```
floor.current.floorSlabData.stairTitle
```

Título de escada

```
floor.current.floorSlabData.slabExport
```

Exportar a laje para visualização 3D (0) não (1) sim

```
floor.current.floorSlabData.slabDetailing
```

Objeto SlabDetailing() com dados de detalhamento de lajes

```
model.current.globalSlabData.stairCase
```

Objeto StairCase() com dados de escadas

```
model.current.globalSlabData.firstRampNumber
```

Número da primeira rampa

Dados de laje

São dados de uma laje existente, podem ser editados.

`slabIdent`

Objeto SMObjectIdent() de identificação da laje

`GetLoad(floor)`

Objeto Load() com carga distribuída em toda a laje tf/m2

`slabGeometry`

Objeto SlabGeometry() com dados de geometria de laje

`slabGrid`

Objeto SlabGrid() com dados para discretização de laje por grelha

`mainAngle`

Ângulo principal em graus

`slabTemperShrink`

Objeto de temperatura / retração de lajes - TemperatureShrink()

`slabDetailing`

Objeto de detalhamento de lajes - SlabDetailing()

`auxiliaryFloor`

Piso auxiliar

`isAStair`

Lance de escada (0) Não (1) Sim

`stairCase`

Objeto StairCase() com dados de escada

`isALanding`

Patamar de escada (0) Não (1) Sim

`stairIdent`

Título de escadas

`isVolumeOnly`

Laje somente de volume (0) Não (1) Sim

`volumeOnlySlab`

Objeto de laje somente de volume - VolumeOnlySlab()

`userAttrib`

Atributos de usuário - UserAttrib()

`slabExport`

Laje exportável para 3D/BIM (0) Não (1) Sim

Dados de geometria da laje - SlabGeometry()

`type`

Tipo de laje SLABTYPE_xxxxx

Constante	Uso
`TQSModel.SLABTYPE_MACICA`	Maciça
`TQSModel.SLABTYPE_NERVUR`	Nervurada retangular
`TQSModel.SLABTYPE_NERVUT`	Nervurada trapezoidal
`TQSModel.SLABTYPE_VIGOTA`	Vigotas treliçadas
`TQSModel.SLABTYPE_TRELIC`	Treliçada
`TQSModel.SLABTYPE_PREFAB`	Pré-fabricada
`TQSModel.SLABTYPE_MISNER`	Mista nervurada (ex: TUPER)

`noSelfWeight`

Desconsiderar peso proprio (0) Não (1) Sim

`GetSelfWeightLoad(floor)`

Retorna carga fixa de peso próprio para lajes preo

`floor` Objeto Floor(), com dados de uma planta

Retorna:

LoadObjeto Load(), com carga de peso próprio

`thickness`

Altura da laje maciça em cm

`recess`

Rebaixo em cm

`GetFloorRecess(floor, slab)`

Rebaixo em cm, considerando piso auxiliar

`floor` Objeto Floor(), com dados de uma planta

`slab` Objeto Slab() que contém esta geometria

Retorna:

`dfspiso` Rebaixo da laje, cm, considerando piso auxiliar

`additionalTopRecess`

Rebaixo adicional superior em cm

`additionalBottomRecess`

Rebaixo adicional inferior em cm

`ribbedTopping`

Laje nervurada: capa, cm

`ribbedBottomTopping`

Laje nervurada: capa inferior, cm

`ribHeight`

Laje nervurada: altura de nervura cm

`ribbedFill`

Laje nervurada: peso de enchimento tf/m3

`averageSizeX`

Laje nervurada: distância média horizontal entre nervuras cm

averageSizeY

Laje nervurada: distância média vertical entre nervuras cm

topSpacingX

Laje nervurada: largura superior de nervura horizontal cm

topSpacingY

Laje nervurada: largura superior de nervura vertical cm

bottomSpacingX

Laje nervurada: largura inferior de nervura horizontal cm

bottomSpacingY

Laje nervurada: largura inferior de nervura vertical cm

inertiaX

Laje nervurada: inércia opcional horizontal cm4

inertiaY

Laje nervurada: inércia opcional vertical cm4

formworkVolume

Laje nervurada: volume cubeta trapezoidal cm3

mouldManufacturer

Laje nervurada: nome do fabricante de enchimentos

mouldID

Laje nervurada: nome do enchimento

latticeTraverseRibEvery

Laje treliçada: nervura transversal a cada N blocos

latticeTraverseWidth

Laje treliçada: largura da nervura secundária cm

`latticeJoistWidth`

Laje treliçada: largura da vigota cm

`latticeJoistHeight`

Laje treliçada: altura da vigota cm

`latticeMiniPanel`

Laje treliçada: minipainél (0) Não (1) Sim

`compositeHeight`

Laje mista: altura do perfil, cm

`compositeWidth`

Laje mista: largura do perfil, cm

`compositeFlange`

Laje mista: largura da aba, cm

`precastRegion`

Pré-moldados: Região construtiva

`sectionName`

Nome de seção não padrão - catalogada

`prestressedReinforcement`

Nome da configuração de armaduras protendidas

`LoadManufacturer(nomfab, nomenc)`

Carrega os dados de moldes de lajes nervuradas. Retorna (0) Se carregou

`nomfab` Nome do fabricante

`nomenc` Nome do enchimento

Discretização de laje por grelha - SlabGrid()

Dados para discretização de laje por grelha.

`asGrid`

Discretizar esta laje em grelha (0) Não (1) Sim

`forceAsGrid`

Forçar a discretização em laje de escadas (0) Não (1) Sim

`plastSupports`

Plastificação (0) conforme critérios (1) Sim (2) Não

`elstBase`

Laje sobre base elástica (0) Não (1) Sim

`elstBaseTxSpring`

Mola de translação Tx por nó tf/m

`elstBaseTySpring`

Mola de translação Ty por nó tf/m

`elstBaseTzSpring`

Mola de translação Tz por nó tf/m

`elstBaseGapZPlus`

Gap de translação Z+ (m)

`elstBaseGapZMinus`

Gap de translação Z- (m)

Detalhamento de lajes - SlabDetailing()

`detailable`

Detalhamento da laje (0) Não (1) Sim

`rigidDiafragm`

Diafragma rígido (0) Não (1) Sim

`prestressed`

Protendida (0) Não (1) Sim

`cover`

Cobrimento diferenciado em cm

`exposure`

Em contato com o solo (0) Não (1) Sim (2) Exposta ao ambiente

`cantileverFactor`

Majorador para laje em balanço ou (0.)

`cantilever`

Verificação de dimensões: Balanço (0) Estimado, geometria (1) Não (2) Sim

`estimatedSpan`

Vão estimado ou (0.) cm

`roofSlab`

Verificação de dimensões: Laje de cobertura (0) Não (1) Sim

`flatSlab`

Verificação de dimensões: Laje plana (0) Geometria (1) Não (2) Sim (3) C/capitéis

`bottomReinforcement`

Direção de armação: (0) Padrão (1) uma direção (2) duas direções

Dados de escada - StairCase()

Para lajes inclinadas que são lances de escada.

`tread`

Passo cm

`riser`

Espelho cm

`adjustment`

Ajuste inicial ou final cm conforme adjustmentMode

`adjustmentMode`

Ajuste (0) Inicial (1) Final

`fixedSteps`

Número de degraus fixos

`winderStair`

Representar como escada plissada (0) Não (1) Sim

Laje somente de volume - VolumeOnlySlab()

Lajes somente de volume são elementos que não tem função no modelo estrutural, mas são representados e exportados em 3D. São uma maneira simples de exportar lajes inclinadas e escadas sem a complexidade do modelo estrutural.

`insertionX`

Ponto de inserção X de laje retangular

`insertionY`

Ponto de inserção Y de laje retangular

`fixedSpace`

Fixar comprimento da laje (0) Não (1) Sim

`space`

Comprimento de laje retangular cm

`fixedWidth`

Fixar largura da laje (0) Não (1) Sim

`width`

Largura de laje retangular cm

`fixedAngle`

Fixar ângulo da laje (0) Não (1) Sim

`angle`

Direção da laje retangular em graus

`recess`

Rebaixo inicial em cm

`zHeight`

Altura Z total em cm

Fechamento de bordo SlabContour(SMObject)

Linhas de fechamento ou contorno auxiliar, são contornos de laje em bordo livre.

Dados de fechamento de bordo

`startX`

X inicial cm

`startY`

Y inicial cm

`endX`

X final cm

`endY`

Y final cm

`3`

Piso auxiliar

Capitel - DropPanel(SMObject)

Um capitel é uma região dentro da laje delimitada por um polígono, e com altura independente da laje.

Dados atuais de um capitel

Usados na criação de um capitel.

`floor.current.floorSlabData.dropPanelThickness`

Altura de capitel cm

`floor.current.floorSlabData.dropPanelDiv`

Divisor de flexão de capitel

Dados de um capitel

`dropPanelPolygon`

Poligonal do capitel - Poygon()

`dropPanelThickness`

Altura do capitel, cm

`dropPanelDiv`

Divisor de inércia à flexão do capitel

`auxiliaryFloor`

Piso auxiliar

Forma de laje nervurada SlabMould(SMObject)

São moldes usados na construção de uma laje nervurada.

Dados atuais de forma de laje nervurada

São usados na criação de uma nova forma.

`floor.current.floorSlabData.slabMouldXSize`

Tamanho X da forma de nervura cm

`floor.current.floorSlabData.slabMouldYSize`

Tamanho Y da forma de nervura cm

`floor.current.floorSlabData.slabMouldXSpace`

Espaçamento X de forma de nervura cm

`floor.current.floorSlabData.slabMouldYSpace`

Espaçamento Y de forma de nervura cm

`floor.current.floorSlabData.slabMouldBasePoint`

Ponto base de inserção de nervura (0..5)

Dados de forma de laje nervurada

São dados de uma forma existente, podem ser editados.

insX

X de inserção do molde

insY

Y de inserção do molde

averageSizeX

Tamanho X médio cm

averageSizeY

Tamanho Y médio cm

rotationAngle

Ângulo de rotação em graus

sizeXDelta

Variação de espessura trapezoidal X cm

sizeYDelta

Variação de espessura trapezoidal Y cm

auxiliaryFloor

Piso auxiliar

Furo em laje SlabOpening(SMObject)

Furo em laje são aberturas definidas por poligonais.

Dados de furo em laje

slabOpeningPolygon

Poligonal do furo em laje - Polygon()

isCut

Tipo de furo (0) Shaft (respeita nervuras) (1) Corte

No tipo Shaft, na geração de lajes nervuradas, o programa considera o contorno do furo como um alinhamento, e tenta passar barras de grelha no contorno.

```
inConcretTopping
```

Somente na capa da laje nervurada (0) Não (1) Sim

```
auxiliaryFloor
```

Piso auxiliar

Furo em pilar - ColumnOpening(SMObject)

Em desenvolvimento.

Estaca para laje radier SlabFoundationPile (SMObject)

Em desenvolvimento

Carga adicional em laje AdditionalLoad(SMObject)

É uma carga que atua sobre toda a laje, além da declarada nos dados da laje.

Dados atuais de carga adicional

Usados na criação de uma carga nova.

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")
```

```
floor.current.floorLoadData.GetLoad (TPLOAD_CARADI)
```

Objeto Load() com a carga distribuída.

Dados de carga adicional

```
load
```

Valor da carga - Objeto Load ()

```
insertionX
```

Ponto de inserção X

```
insertionY
```

Ponto de inserção Y

```
auxiliaryFloor
```

Piso auxiliar

Carga distribuída por área AreaDistributedLoad(SMObject)

É uma carga distribuída definida por uma poligonal fechada. Se a carga se sobrepor a vários elementos estruturais, o

programa quebra a carga e faz o lançamento em partes.

Dados atuais de carga adicional

Usados na criação de uma carga nova.

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")

floor.current.floorLoadData.GetLoad (TPLOAD_CARADI)
```

Objeto Load() com a carga distribuída.

Dados de carga adicional

```
load
```

Valor da carga - Objeto Load()

```
polygon
```

Poligonal fechada onde a polig é distribuída - Polygon()

```
auxiliaryFloor
```

Piso auxiliar

Carga concentrada ConcentratedLoad(SMObject)

Carga pontual sobre laje, viga ou pilar.

Dados atuais de carga concentrada

São usados na criação de uma nova carga

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")

floor.current.floorLoadData.GetLoad (tipo)
```

Objeto Load() com a carga concentrada. Tipo é uma das direções de força/momento nas constantes TQSModel.TPLOAD_FORxxx.

Dados de carga concentrada

```
loadFx
```

Força Fx - tf - Somente pilar - Objeto Load()

```
loadFy
```

Força Fy - tf - Somente pilar - Objeto Load()

```
loadFz
```

Força Fz - tf - Pilar, viga ou laje - Objeto Load()

```
loadMx
```

Força Mx - tfm - Somente pilar - Objeto Load()

```
loadMy
```

Força My - tfm - Somente pilar - Objeto Load()

```
insertionX
```

Ponto de inserção X

```
insertionY
```

Ponto de inserção Y

```
auxiliaryFloor
```

Piso auxiliar

Carga distribuída linearmente LinearyDistributedLoad(SMObject)

É uma carga distribuída por metro, entre dois pontos. Se a carga estiver sobre mais de uma laje e/ou viga, ela será quebrada para cada elemento.

Dados atuais de carga distribuída linearmente

```
model = TQSModel.Model ()
floor = model.floors.GetFloor ("nome-da-planta")

floor.current.floorLoadData.GetLoad (TPLOAD_CARLIN)
```

Objeto Load() com a carga distribuída lineramente.

Dados de carga distribuída linearmente

```
load
```

Valor da carga - Objeto Load()

```
startPointX
```

Ponto inicial X

```
startPointY
```

Ponto inicial Y

```
endPointX
```

Ponto final X

```
endPointY
```

Ponto final Y

```
auxiliaryFloor
```

Piso auxiliar

```
wall
```

Parede 3D importada do BIM - Objeto Wall() ou None

Carga de empuxo SoilPressureLoad(SMObject)

Define a carga de empuxo entre dois pontos, seguindo a convenção interativa do Modelador. Uma carga de empuxo é um objeto espacial, com definição de pavimento inicial e final de atuação.

Dados atuais de carga de empuxo

São definidos por um objeto tipo SoilPressureLoad() atual.

```
model = TQSModel.Model ()
```

```
model.current.globalSoilpress.soilPressureLoad
```

Objeto do tipo SoilPressureLoad() com carga de empuxo

Dados de carga de empuxo

```
topLoad
```

Carga de empuxo no topo - Objeto Load()

```
baseLoad
```

Carga de empuxo na base - Objeto Load()

topFloorName

Nome da planta do topo da carga de empuxo

baseFloorName

Nome da planta da base da carga de empuxo

startPointX

Ponto inicial X

startPointY

Ponto inicial Y

endPointX

Ponto final X

endPointY

Ponto final Y

Temperatura e retração de vigas e lajes - TemperatureShrink()

Dados de temperatura e retração de vigas, se habilitados nos dados do edifício.

temperatureDefined

Definição de variação de temperatura ao longo da viga (0) Não (1) Sim

uniformVariation

Variação longitudinal de temperatura em graus Celcius

uniformVariation2

Variação longitudinal de temperatura em graus Celcius - caso 2

transverseVariation

Variação transversal de temperatura em graus Celcius

`transverseVariation2`

Variação transversal de temperatura em graus Celcius - caso 2

`shrinkageDefined`

Definição de retração ao longo da viga (0) Não (1) Sim

`uniformShrinkage`

Variação longitudinal de temperatura equivalente à retração em graus Celcius

Modos de visualização - VisModes()

Os modos de visualização se aplicam a diversos tipos de desenho. O objeto que define os modos é obtido a partir do modelo estrutural:

```
model = TQSModel.Model ()
```

```
model.current.visData.GetVisModes(imode)
```

Objeto VisModes() com m odos de visualização

Onde imode pode ser:

Constante	Uso
`TQSModel.VISMODE_MATUAL`	Modo de visualização atual
`TQSModel.VISMODE_FORMAS`	Desenho de formas FORnnnn.DWG
`TQSModel.VISMODE_VERIFI`	Modo de verificação
`TQSModel.VISMODE_MODELO`	Desenho MODELO.DWG
`TQSModel.VISMODE_GRUPOA`	Grupo A de visualização do usuário
`TQSModel.VISMODE_GRUPOB`	Grupo B de visualização do usuário

Os modos de visualização são separados nos seguintes subobjetos:

Subobjeto	Tipo	Uso
`columns`	VisModeColumns()	Pilares

beams	VisModeBeams()	Vigas
slabs	VisModeSlabs()	Lajes
bim	VisModeBim()	BIM
loads	VisModeLoads()	Cargas
foundations	VisModeFoundations()	Fundações
precast	VisModePrecast()	Pré-moldados
details	VisModeDetails()	Outros

Modos de visualização de pilares

`columns.columns`

Pilares (0) Não (1) Sim

`columns.name`

Título de pilar (0) Não (1) Sim

`columns.dimensions`

Dimensões de pilares (0) Não (1) Sim

`columns.overlapUpper`

Sobreposição de pilar atual e superior (0) Não (1) Sim

`columns.overlapBottom`

Sobreposição de pilar atual e inferior(0) Não (1) Sim

`columns.gridSupport`

Condições de apoio em grelha (0) Não (1) Sim

`columns.axes`

Eixos de pilares (0) Não (1) Sim

`columns.details`

Outros dados de detalhamento de pilares (0) Não (1) Sim

`columns.circle`

Pilares circulares como círculos (0) Não (1) Sim

`columns.shearStrip`

Mostra lâmina de pilares parede (0) Não (1) Sim

`columns.shearWallDiscret`

Discretização de pilar parede (0) Não (1) Sim

`columns.fixedPoint`

Ponto fixo do pilar (0) Não (1) Sim

`columns.imposedRestraint`

Travamento imposto (0) Não (1) Sim

`columns.ficticiousSupport`

Apoios fictícios (0) Não (1) Sim

`columns.horizontalOpenings`

Furos em pilares (0) Não (1) Sim

`columns.metalCheck`

Resultados de pilares no MetalCheck (0) Não (1) Sim

Modos de visualização de vigas

`beams.beams`

Vigas (0) Não (1) Sim

`beams.name`

Título de vigas (0) Não (1) Sim

`beams.dimensions`

Título de vigas (0) Não (1) Sim

beams.axes

Eixos de vigas (0) Não (1) Sim

beams.connections

Vinculações de vigas (0) Não (1) Sim

beams.nodes

Nós de vigas (0) Não (1) Sim

beams.details

Outros dados de detalhamento de vigas (0) Não (1) Sim

beams.sectionT

Seção T e B colaborante (0) Não (1) Sim

beams.cantilever

Pintar vigas em balanço (0) Não (1) Sim

beams.openings

Furos horizontais (0) Não (1) Sim

beams.transferBeam

Viga de transição (0) Não (1) Sim

beams.metalCheck

Vigas no MetalCheck (0) Não (1) Sim

beams.precast

Pintar vigas protendidas (0) Não (1) Sim

Modos de visualização de lajes

slabs.slabs

Lajes (0) Não (1) Sim

slabs.name

Título de lajes (0) Não (1) Sim

slabs.dimensions

Dimensões de lajes (0) Não (1) Sim

slabs.mainDirection

Símbolo de ângulo principal da laje (0) Não (1) Sim

slabs.contour

Contorno da laje (0) Não (1) Sim

slabs.details

Outros dados de detalhamento de lajes (0) Não (1) Sim

slabs.ribs

Nervuras em lajes (0) Não (1) Sim

slabs.openings

Furos em lajes (0) Não (1) Sim

slabs.caps

Capitéis (0) Não (1) Sim

slabs.trapezoidalRibs

Bordas de nervuras trapezoidais (0) Não (1) Sim

slabs.ficticiusRibs

Nervuras em maciços (0) Não (1) Sim

slabs.prefabDir

Direção de nervuras (0) Não (1) Sim

slabs.cantilever

Laje em balanço (0) Não (1) Sim

Modos de visualização de BIM

`bim.importStatus`

Status de importação do BIM (0) Não (1) Sim

`bim.notExported`

Elementos não exportados para o BIM (0) Não (1) Sim

`bim.importedPipes`

Tubos importados (0) Não (1) Sim

`bim.pipeView`

Visualizar tubos (0) Só na vista 3D (1) Na vista 3D e 2D

`bim.importedWalls`

Paredes importadas (0) Não (1) Sim

`bim.objects3D`

Objetos 3D (0) Não (1) Sim

`bim.plan2D`

Planta 2D como referência para o 3D (0) Não (1) Sim

`bim.reference3D`

Referência externa 3D (0) Abaixo (1) Acima (2) Ambos

`bim.basePoint`

Ponto base de projeto (0) Não (1) Sim

`bim.surveyPoint`

Ponto de levantamento topográfico (0) Não (1) Sim

Modos de visualização de cargas

`loads.beamDistributedLoad`

Cargas distribuídas nas vigas (0) Não (1) Sim

loads.slabDistributedLoad

Cargas distribuídas nas lajes (0) Não (1) Sim

loads.concentratedLoad

Cargas concentradas (0) Não (1) Sim

loads.linearLoad

Cargas concentradas (0) Não (1) Sim

loads.areaDistributedLoad

Carga distribuída por área (0) Não (1) Sim

loads.soilPressureLoad

Cargas de empuxo (0) Não (1) Sim

loads.showCaseOnlyLoad

Restringir ao caso (0)=todos

loads.manualWindDistribution

Distribuição manual de vento (0) Não (1) Sim

loads.manualWindPrefix

Prefixo do caso de vento para distribuição manual (""=Todos)

loads.loadViewMode

Mostrar cargas em (0) 2D (1) 3D (2) 2D e 3D

Modos de visualização de fundações

foundations.foundations

Fundações (0) Não (1) Sim

foundations.name

Títulos de fundações (0) Não (1) Sim

`foundations.dimensions`

Dimensões de fundações (0) Não (1) Sim

`foundations.details`

Outros dados de detalhamento de fundações (0) Não (1) Sim

`foundations.piles`

Estacas de blocos (0) Não (1) Sim

`foundations.cutOffLevel`

Cotas de arrasamento (0) Não (1) Sim

`foundations.pierDimension`

Dimensão da base do tubulão (0) Não (1) Sim

`foundations.pileQuantity`

Quantidade e bitola de estacas (0) Não (1) Sim

`foundations.pileIdent`

Número da estaca (0) Não (1) Sim

`foundations.pileDimensions`

Diâmetro da estaca (0) Não (1) Sim

Modos de visualização de pré-moldados

`precast.constructionRegions`

Regiões pré-moldadas (0) Não (1) Sim

`precast.floorPlanGroups`

Título dos grupos de formas (0) Não (1) Sim

`precast.reinforcementGroups`

Título dos grupos de armação (0) Não (1) Sim

`precast.corbelsName`

Título de consolos (0) Não (1) Sim

precast.corbels

Consolos (0) Não (1) Sim

precast.corbelsData

Dados de consolos (0) Não (1) Sim

precast.accessories

Acessórios (0) Não (1) Sim

precast.slabsReport

Resultados do cálculo de lajes pré-moldadas, se disponível (0) Não (1) Sim

precast.showSlabsReport

Mostrar o relatório após o cálculo de lajes pré-moldadas (0) Não (1) Sim

precast.beamsSectionTitle

Título de seções catalogadas vigas (0) Não (1) Sim

precast.sectionsLines

Linhas de contorno das seções não padrão (0) Não (1) Sim

precast.inserts

Insertos (0) Não (1) Sim

precast.facades

Fachadas (0) Não (1) Sim

Outros modos de visualização

details.crossSections

Cortes da planta de formas (0) Não (1) Sim

details.dimensions

Cotas (0) Não (1) Sim

`details.controlPoints`

Pontos de controle de cotagem (0) Não (1) Sim

`details.formworkArea`

Áreas de vigas, pilares, lajes (0) Não (1) Sim

`details.axes`

Eixos (0) Não (1) Sim

`details.centroidsTable`

Tabela de baricentros (0) Não (1) Sim

`details.overlapAuxFloors`

Sobrepõe pisos auxiliares (0) Não (1) Sim

`details.elementsTable`

Tabela de elementos (0) Não (1) Sim

`details.tunnelReference`

Centro de torção - referência de túnel de vento (0) Não (1) Sim

`details.linesOutOfXY`

Linhas fora da direção X/Y global (0) Não (1) Sim

`details.disablePlotingHatches`

Inibe display de hachuras plotadas (0) Não (1) Sim

`details.hatchedUnevenness`

Legenda e hachura de desníveis (0) Não (1) Sim

`details.floorLevels`

Cota pavimentos em desnível (0) Não (1) Sim

`details.floorLevelsTable`

Tabela de nível dos pavimentos (0) Não (1) Sim

`details.towerFence`

Cerca da torre (0) Não (1) Sim

Atributos BIM de objeto - UserAttrib()

É um grupo de atributos na forma chave=valor, que podem ser associados a elementos do modelo estrutural.

`usratribClear()`

Limpa atributos do usuário

`usratribNum()`

Retorna o número de atributos

`usratribInsert(chave, sval)`

Insere novo atributo

`chave` Chave

`sval` Valor

`usratribErase(chave)`

Apaga atributo existente

`chave` Chave do atributo a apagar

`usratribBegin()`

Prepara para ler todos os atributos

`usratribReadNext()`

Lê próximo atributo

Retorna:

`chave` Chave

`sval` Valor

`istat` (0) Ok (!=0) Acabou

`usratribFind(chave)`

Lê próximo atributo

`chave` Chave a pesquisar

Retorna:

`sval` Valor

`istat` (0) Ok (!=0) Acabou

Atributos BIM globais - GlobalAttrib()

Consistem em grupos de atributos comuns gravados para todos os elementos exportados para o BIM. Cada grupo de atributos contém um ou mais objetos tipo UserAttrib().

`Clear()`

Limpa atributos globais

`SetGroup(nomgrupo, iabrang, ielementos, ipiso, nompla)`

Define novo grupo. Se já existir, acessa grupo existente

`nomgrupo` Nome do grupo

`iabrang` Abrangência: BIM_GLBA_xxxx

`ielementos` Elementos definidos TYPE_xxxx (TYPE_INDEF=todos)

`ipiso` Piso para iabrang==BIM_GLBA_UMPISO

`nompla` Pavimento para iabrang==BIM_GLBA_PLANTA

Retorna:

`objeto` UserAttrib() de atributos do grupo

Constante	Uso
`TQSMoldel.BIM_GLBA_GLOBAL`	Todos do edifício
`TQSMoldel.BIM_GLBA_PLANTA`	Todos de uma planta
`TQSMoldel.BIM_GLBA_UMPISO`	Todos de um piso

`NumGroups()`

Retorna o número de grupos de atributos

`EraseGroup(nomgrupo)`

Apaga atributo existente

`ReadBegin()`

Prepara para ler todos os atributos

`ReadNext()`

Lê próximo atributo

Retorna:

`nomgrupo` Nome do grupo

`iabrang` Abrangência: TQSMoldel.BIM_GLBA_xxxx

`ielementos` Elementos definidos TQSMoldel.TYPE_xxxx (TYPE_INDEF=todos)

`ipiso` Piso para iabrang== TQSMoldel.BIM_GLBA_UMPISO

`nompla` Pavimento para iabrang== TQSMoldel.BIM_GLBA_PLANTA

`UserAttrib`() Objeto de atributos do grupo

`istat` (!=0) quando acabar atributos

`ReadGroup(nomgrupo)`

Lê atributos dado nome do grupo

`nomgrupo` Nome do grupo

Retorna:

`iabrang` Abrangência: TQSMoldel.BIM_GLBA_xxxx

`ielementos` Elementos definidos TQSMoldel.TYPE_xxxx (TYPE_INDEF=todos)

`ipiso` Piso para iabrang== TQSMoldel.BIM_GLBA_UMPISO

`nompla` Pavimento para iabrang== TQSMoldel.BIM_GLBA_PLANTA

`UserAttrib()` Objeto de atributos do grupo

`istat` (!=0) o grupo não existe

`Sort()`

Classifica grupos por nome

Parede 3D importada do BIM - Wall()

Em desenvolvimento

Testes do TQSModel.PY

Os seguintes programas mostram o funcionamento do TQSModel.

Programa	Finalidade

TSTMODEL1.PY	Mostra a criação de um edifício simples e dos principais objetos do modelo estrututral: pilares, vigas, cruzamento de vigas com definição de quem apoia, furos em vigas, lajes, capiteis, formas de nervuras, fechamentos de bordo, furos em lajes, cargas concentradas, lineares, distribuídas, adicionais e empuxo, fundações em blocos e sapatas.
TSTMODEL2.PY	Mostra a criação de um edifício com pilares helicoidais, trabalhosos para criar manualmente, mas simples por programa. Demonstra a distribuição automática de formas de nervuras em batch para qualquer formato de laje.
TSTMODEL3.PY	Lista elementos estruturais do edifício "Mod-Padrão" distribuído para teste de instalação. Leitura de atributos de vigas, pilares, lajes e fundações.

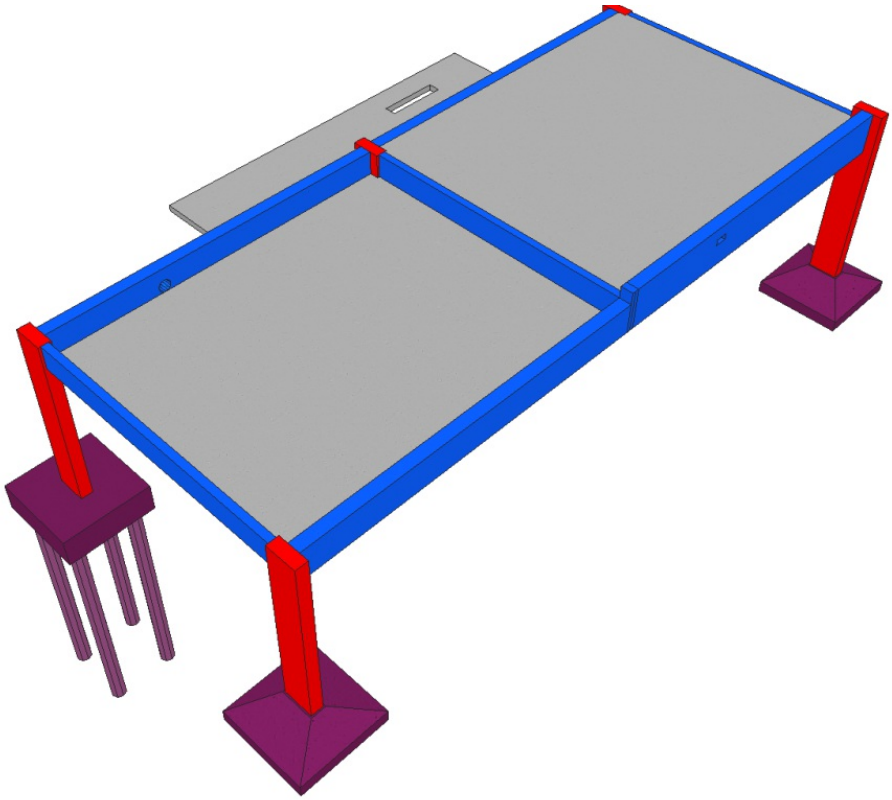
Programa TSTMODEL1.PY

Este programa gera um edifício novo de nome TSTMODEL1. O modelo estrutural é aberto sempre no contexto do edifício. Então este programa primeiro usa a classe TQSBuild para criar um edifício novo e depois torna atual a pasta raiz, daonde o modelo estrutural pode ser aberto para gravação:

```
build = CriarDadosDoEdificio ()
build.RootFolder ("TSTMODEL1")
model = TQSModel.Model ()
istat = model.file.OpenNewModel ()
```

Os elementos estruturais são criados sempre usando um pavimento como base:

```
floorTerreo = model.floors.GetFloor ("Terreo ")
floorFundacao = model.floors.GetFloor ("Fundacao")
CriarPilares (model, floorTerreo)
CriarVigasTerreo (model, floorTerreo)
CriarLajesTerreo (model, floorTerreo)
CriarCargasTerreo (model, floorTerreo)
CriarFundacoes (model, floorFundacao)
```



Programa TSTMODEL2.PY

O programa para este modelo TSTMODEL2 é mais simples que o anterior porque gera menos tipos de elementos, mas a estrutura final é mais complexa, com 20 pisos com pilares em formato helicoidal, laje plana nervurada e vigas no contorno que acompanham os pilares. Os pilares circulares helicoidais são marcados como inclinados, e um loop gera todos os pilares de um piso e gira as coordenadas de piso a piso:

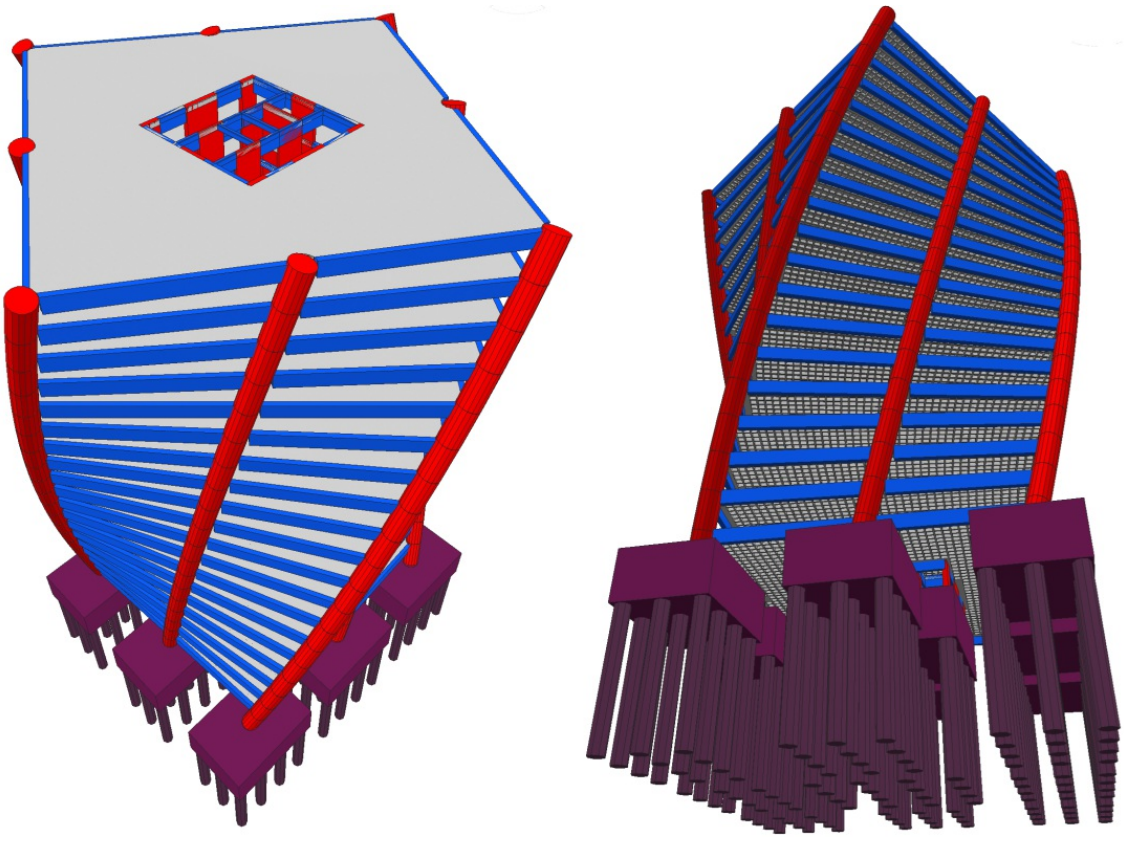
```
xy = ((-1316., 1274.),....)
for coords in xy:
column = floor.create.CreateColumn (coords [0], coords [1])
for ipla in range (1, 20):
nompla = PegarNomePlanta (ipla)
isec = ipla
columnSection = column.ColumnCreateSection (isec, nompla)
xins = columnSection.insertionX
yins = columnSection.insertionY
xins, yins = TransformarCoords (1., xins, yins)
columnSection.insertionX = xins
columnSection.insertionY = yins
column.ColumnSectionUpdateGeometry (columnSection)
```

As lajes em cada piso recebem o mesmo ângulo principal do giro dos pilares, assim como as formas de nervuras distribuídas. Para especificar a geometria da laje nervurada, carregamos os dados de fabricante e cubetas Atex®:

```
nompla= PegarNomePlanta (ipla)
floor = model.floors.GetFloor (nompla)
slabGeometry= floor.current.floorSlabData.slabGeometry
slabGeometry.type= TQSModel.SLABTYPE_NERVUT
istat=slabGeometry.LoadManufacturer("Atex","Atex 650 Capa 7.5 H 35")
```

Depois criamos a laje e uma cubeta, que mandamos distribuir:

```
angpri = 3.0*float (ipla)
xins = -609.0
yins = 101.7
slab = floor.create.CreateSlab (xins, yins, angpri)
slabMould = floor.create.CreateSlabMould (slab, xins, yins)
floor.util.DistributeSlabMould (slab, slabMould)
```



O núcleo do edifício tem coordenadas fixas, e é formado por um pilar genérico e outros retangulares, ligados por vigas. A distribuição de formas de nervuras reconhece o contorno do núcleo.

Programa TSTMODEL3.PY

Este programa demonstra que é possível ler e listar um modelo. Esta lógica também serve para modificar modelos –

basta iterar, carregar elementos do modelo e modifica-los.

O programa começa abrindo um modelo existente – "Mod-Padrão", que vem na pasta de testes \TQSW\USUARIO\TESTE com o nome de MODPLA.TQS e deve estar descompactado.

```
GLB_NOMEDIF = "Mod-Padrão"
build= TQSBuild.Building ()
if (build.RootFolder (GLB_NOMEDIF) != 0):
TQSUtil.writef ("O edifício [%s] não existe" % GLB_NOMEDIF)
return
model = TQSModel.Model ()
if (model.file.OpenModel () != 0):
TQSUtil.writef ("Não abri o edifício [%s]" % GLB_NOMEDIF)
```

Um loop pelas plantas do edifício faz a extração de cada uma delas:

```
floorslist = model.floors
numpla = floorslist.GetNumFloors ()
for indpla in range (1, numpla+1):
nompla = floorslist.GetFloorName (indpla)
floor = floorslist.GetFloor (nompla)
ExtrairPlanta (model, nompla, floor)
```

Em cada planta, existe uma lista separada de objetos por tipo de elemento estrutural. O loop pelos tipos de elemento é este:

```
numtypes = model.type.GetNumTypes ()
for itype in range (0, numtypes):
ExtrairTipo (model, floor, itype)
```

Nem todos os tipos de elementos são tratados atualmente pela interface, somente os listados neste manual. Listamos como exemplo as vigas, pilares e lajes:

```
numobjetos= floor.iterator.GetNumObjects (itype)
for iobjeto in range (0, numobjetos):
smobject = floor.iterator.GetObject (itype, iobjeto)
if (smobject is None):
```

```
continue
if (itype == TQSModel.TYPE_VIGAS):
ListarViga (floor, smobject)
elif (itype == TQSModel.TYPE_SECPIL):
ListarPilar (model, floor, smobject)
elif (itype == TQSModel.TYPE_LAJES):
ListarLaje (model, floor, smobject)
```

O objeto retornado por GetObject é derivado do tipo SMObject, é um dos objetos do Modelador. A rotina ListarViga por exemplo, recebe este objeto como tipo Beam():

```
def ListarViga (floor, beam):
```

A listagem emitida é um exemplo do tipo de informação que pode ser obtida na leitura de um modelo.

Programa TSTMODEL4.PY

O TSTMODEL4.PY demonstra o uso da interface Python para testar múltiplas soluções estruturais, variando elementos de um modelo. Após duplicar e modificar um edifício, o módulo TQSExec é usado para fazer um processamento global. No final, os edifícios resultantes poderão ser comparados.

Assim como em TSTMODEL3.PY, abrimos o modelo "Mod-Padrão", que deve estar descompactado. Este edifício é copiado três vezes com sufixos diferentes ("A", "B" e "C") e em cada cópia são alterados quatro pilares, mais duas vigas do pavimento "Tipo". A lista edifícios contém as alterações, como abaixo:

```
edificios=
(
("A",
(
("Fundacao",
(
(GLB_PILAR, 5, 3, -295.5, 544.5, 40., 19.),
(GLB_PILAR, 8, 2, -295.5, 315.5, 40., 19.),
(GLB_PILAR, 6, 3, -000.5, 544.5, 35., 14.),
(GLB_PILAR, 9, 2, -000.5, 315.5, 35., 14.),
)
),
("Tipo",
```

```
(
(GLB_VIGA, 2, 0, 000.0, 000.0, 14., 35.),
(GLB_VIGA, 3, 0, 000.0, 000.0, 14., 35.),
),
)
)
),
("B",
.......................
```

Temos uma lista de 3 edifícios "A", "B" e "C", cada edifício com alteração em pavimentos "Fundacao" e "Tipo", e cada pavimento com elementos a alterar – vigas ou pilares. Os parâmetros de cada lista se referem no caso de pilares ao número do pilar a ser alterado, o ponto de inserção, e a nova dimensão de pilar retangular. No caso das vigas, o seu número e sua nova dimensão.

Cada versão do edifício primeiro é copiada:

```
GLB_NOMEDIF = "Mod-Padrão"
for edificio in edificios:
nomedi = GLB_NOMEDIF + "-" + edificio [0]
building.file.SuppressMessages (1)
building.file.SaveAs (nomedi)
```

Depois, para cada elemento a modificar, a rotina ModificarModelo é chamada para iterar por todos os elementos do mesmo tipo até achar o elemento com o número correto. A modificação do pilar redefine o seu ponto de inserção, para garantir que com o tamanho diferente continue na posição correta:

```
columnSection = pilar.ColumnGetSection (0)
columnSection.insertionX = xins
columnSection.insertionY = yins
columnSection.columnInsertion.insertionType = 1
columnSection.columnInsertion.insertionCorner = icanto
columnSection.columnGeometry.sectionB1 = dimb
columnSection.columnGeometry.sectionH1 = dimh
pilar.ColumnSectionUpdateGeometry (columnSection)
```

Mas como a alteração na geometria provoca uma excentricidade na fundação, vamos à fundação sob o pilar e definimos um pilar fictício para não causar erro de extração:

```
fundacao = pilar.GetFoudation ()
if (fundacao != None):
if (fundacao.columnIsAFoundation != 0 and \
fundacao.foundationDefined != 0):
fundacao.foundationData.ficticiousColumn = 1
fundacao.foundationData.ficticiousColumnXDim = dimb
fundacao.foundationData.ficticiousColumnYDim = dimh
```

No caso das vigas alteradas, tínhamos no modelo original uma mudança de seção. Alteramos a geometria para que todas as seções sejam iguais:

```
numnodes = viga.NumNodes ()
for inode in range (0, numnodes-1):
viga.SetSegmentBeamGeometryDefined (inode, 0)
beamGeometry = viga.GetBeamGeometry ()
beamGeometry.width = dimb
beamGeometry.depth = dimh
```

As alterações de contorno de lajes causada pela alteração na dimensão dos pilares faz com que o contorno das lajes fique desatualizado no final. No final, varremos todas as plantas e mandamos refazer as intersecções:

```
numfloors = model.floors.GetNumFloors ()
for ifloor in range (1, numfloors+1):
nompav = model.floors.GetFloorName (ifloor)
floor = model.floors.GetFloor (nompav)
floor.util.DoIntersections ()
```

O processamento global de cada edifício é feito o TQSExec, onde a partir da pasta Espacial:

```
job = TQSExec.Job ()
job.EnterTask (TQSExec.TaskFolder (nomedi, \
```

```
TQSExec.TaskFolder.FOLDER_FRAMES))
job.EnterTask (TQSExec.TaskGlobalProc ( \
floorDraw = 1,slabs = 2, beams = 3, \
columns = 2, foundations = 1, stairs = 1))
job.Execute ()
```