UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTRUTURAS E CONSTRUÇÃO CIVIL

PROGRAMA GRÁFICO LIVRE PARA A ANÁLISE DE LAJES DE EDIFICAÇÕES DE CONCRETO ARMADO USANDO O MODELO DE GRELHA EQUIVALENTE

ANDREW JOHN RICHTER CASS

Dissertação apresentada ao Programa de Pós Graduação em Construção Civil da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Estruturas e Construção Civil.

Área de Concentração: Sistemas Construtivos.

Orientador: Prof. Dr. Roberto Chust Carvalho

SÃO CARLOS
2015

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C343pg

Cass, Andrew John Richter.
Programa gráfico livre para a análise de lajes de edificações de concreto armado usando o modelo de grelha equivalente / Andrew John Richter Cass. -- São Carlos : UFSCar, 2015.
185 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2015.

1. Concreto armado. 2. Lajes. 3. Grelhas (Engenharia de estruturas). 4. Programa gráfico. I. Título.

CDD: 624.18341 (20ª)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Estruturas e Construção Civil

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Andrew John Richter Cass, realizada em 05/05/2015:

Prof. Dr. Roberto Chust Carvalho
UFSCar

Prof. Dr. Libânio Miranda Pinheiro
USP

Prof. Dr. Luiz Fernando Campos Ramos Martha
PUC-RJ

AGRADECIMENTOS

Todo meu agradecimento ao Prof. Dr. Roberto Chust Carvalho pela oportunidade, a confiança e a inabalável paciência em sua orientação, permitindo-me atingir os objetivos necessários para o término deste trabalho, que rapidamente tornou-se uma incrível aventura e alegria.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABNT: Associação Brasileira de Normas Técnicas

API: Application Program Interface

BIM: Building Information Model

CAD: Computer Aided Design

CG: Computer Graphics

CAM: Computer Aided Manufactoring

DXF: Design Exchange Format

DWG: Drawing Binary Format

FGV: Fundação Getúlio Vargas

GDI: Graphic Device Manager

IDE: Integrated Development Environment

LISP: List Processing Language

OOP: Object Oriented Programming

OSS: Open Source Software

OS: Operating System

$M_{f:}$ Momento Fletor

M_t: Momento Torçor

Mx: Momento no Eixo x

My: Momento no Eixo y

Mz: Momento no Eixo z

Mtr: Momento Torços de Fissuração

Vx: Força Cortante em x

Vy: Força Cortante em y

Vz: Força Cortante em z

I_f: Inércia à Flexão

I_t: Inércia à Torção

E_C: Módulo de Elasticidade do Concreto

G_C: Módulo de Elasticidade Transversal do Concreto

fck: Resistência Característica do Concreto à Compressão

ca: Carga Acidental

cp: Carga Permanente

cl: Carga Nodal

cd: Carga Distribuída

pp: Peso Próprio

ν: Coeficiente de Poisson

q: Sobrecarga de Utilização

p: Sobrecarga Total

Resumo

PROGRAMA GRÁFICO LIVRE PARA A ANÁLISE DE LAJES DE CONCRETO USANDO O MODELO DE GRELHA EQUIVALENTE

Apresenta-se neste trabalho um programa gráfico que permite a análise (cálculo de esforços e deslocamentos) de lajes de concreto pelo modelo da grelha equivalente. O pavimento é considerado como monolítico sem a tradicional discretização da estrutura, ou seja, a separação dos diversos elementos estruturais (vigas, lajes e pilares). Apenas os pilares são considerados como indeslocáveis na direção do seu eixo principal. O ambiente gráfico idealizado para tal é pioneiro, desenvolvido por meio da programação orientada a objetos e será livre (gratuito e com listagem disponível). No sistema é possível lançar a estrutura como se desenham as fôrmas de uma estrutura, permitindo assim, de forma fácil, a caracterização das diferentes propriedades dos materiais e a definição das ações atuantes na estrutura. O programa estabelece uma interface para o módulo de cálculo adaptado que, inicialmente, foi desenvolvido por Igor Stayanov Cotta em 2006.e assim permite a visualização dos esforços, dos deslocamentos e dos gráficos de tensões na tela do monitor.

Palavras Chave: Grelha Equivalente, Concreto Armado, Estruturas, Análise Estrutural, Método da Rigidez, Programa Livre, Pascal, Lisp.

Abstract

FREE GRAPHIC PROGRAM FOR REINFORCED CONCRETE SLABS ANALYSIS USING THE EQUIVALENT GRID MODEL

It’s presented in this work a graphic program that allows the analysis (calculation of forces and displacements) of concrete slabs by the equivalent grid model. The surface is considered as monolithic without the traditional discretization of the structure, i.e., the separation of the various structural members (beams, columns and slabs). Only the pillars are considered as unmovable in the direction of its principal axis. The graphical environment designed for this is pioneering, developed by object-oriented programming and will be distributed freely and with listings available. The system allows the designing of the structure as a model, allowing an easier perception of its different structural components and the definition of the different stresses on it. The program establishes an interface adapted to the calculation module, that initially was developed by Igor Stayanov Cotta in 2006,.and thus allows the graphic view of the efforts, displacements and stresses acting over the model.

Key words: Equivalent Grid, Reinforced Concrete, Structures, Structural Analysis, Stiffness Method, Free Program, Pascal, Lisp.

1 INTRODUÇÃO

1 INTRODUÇÃO

1.1 GENERALIDADES

É uma tarefa difícil a geração de malhas, barras e nós que além de manterem certa uniformidade possam representar as vigas, pilares, aberturas e possam ainda ter aumentada a quantidade de elementos e pontos de concentração de esforços. Outro problema é a criação de arquivos de transferência de dados que representam as características dos diversos elementos e também das ações atuantes. A questão da edição é crítica e complexa. Alterar arquivos, recompô-los e reorganizá-los em informações correlacionadas nos arquivos já existentes não são tarefas triviais. Os detalhes e características de cada particularidade da estrutura devem ser mantidos coerentes e interligados, para que no processo de cálculo a influência que cada elemento produz nos demais seja considerada com exatidão.

Outro desafio é o desenvolvimento de uma ferramenta que não tenha limitações quanto ao número de nós, que podem chegar facilmente aos milhares.

Tão importantes quanto gerarem nós, elementos estruturais e montar arquivos de entrada, são a representação gráfica dos diagramas dos esforços na estrutura e a grelha equivalente deformada, que serão produzidos por meio da análise numérica dos resultados.

Espera-se que com tal ferramenta possa-se fazer análises mais complexas das estruturas de concreto armado, considerando-se os efeitos das deformações relativas entre lajes e vigas, na continuidade entre lajes, nos apoios de vigas, nas cargas verticais de qualquer tipo (inclusive lineares no meio da laje) e na torção de vigas. Efeitos, estes, que deveriam ser calculados analisando-se o pavimento como um todo.

1.2 OBJETIVOS

O objetivo principal deste trabalho são o de se desenvolver um programa de modelagem gráfica de grelhas. O módulo de cálculo escolhido foi o idealizado, inicialmente, por (Cotta, 2006) mestre e pertencente ao grupo de pesquisa da UFSCar. A plataforma gráfica será gratuita e disponibilizada na internet.

Essa plataforma permitirá a geração de arquivos completos com os elementos necessários para o cálculo de grelhas. Terá a capacidade de fornecer informações completas a respeito dos componentes estruturais, permitirá a navegação visual ampla por todo o projeto e permitirá a exportação das informações gráficas em formatos de intercâmbio de dados gráficos consagrados, como os formatos DXF e DWG.

Como objetivos complementares têm-se:

1. A remodelagem da plataforma de cálculo para que esta possa processar um número ilimitado de elementos estruturais.
2. O desenvolvimento de um pós-processador de resultados (esforços e deslocamentos) que facilite o entendimento do comportamento das estruturas sob o efeito de um conjunto de ações.
3. Melhorar a plataforma de cálculo de forma que sejam aumentados a velocidade de processamento, o limite de tamanho de grelha e a capacidade de resolução de estruturas com diversas combinações de carregamentos simultâneos.
4. Implementar a consideração das particularidades das estruturas de concreto armado tais como o monolitismo.
5. Auxiliar no ensino de estruturas nas universidades, facilitando o entendimento e aprendizado, pelos discentes, do comportamento das estruturas sob o efeito de esforços.
6. Facilitar o serviço dos profissionais que, eventualmente, não têm a possibilidade de adquirir programas comerciais caros.

1.3 JUSTIFICATIVAS

1.3.1 Modernização e Rapidez

Nestes últimos anos tem-se observado que o uso de computadores tem aumentado de forma exponencial, com previsão pela FGV, para o Brasil, de um computador por brasileiro até 2016[1]. Portanto o computador vem influenciando cada vez mais a forma como organizamos as tarefas, principalmente aquelas que, faz alguns anos, eram longas, tediosas ou pouco precisas. Como a régua de cálculo foi substituída pela calculadora, o computador pessoal vem ganhando cada vez mais espaço no rol de ferramentas dos projetistas em engenharia. Atualmente além dos processadores de texto, têm-se as planilhas eletrônicas de cálculo, inúmeras linguagens de programação e equipamentos que a cada ano vêm se beneficiando de novas atualizações e oferecem espaço a uma nova geração de máquinas com ainda maior capacidade de processamento. Neste cenário tem-se uma clara evolução dos conceitos referentes às exigências com relação às tarefas de engenharia, que se tornam cada vez mais convergentes às ferramentas numéricas computacionais. A capacidade de processamento dos equipamentos da atualidade permite que algoritmos complexos de cálculo estrutural, que apesar de alguns existirem desde a década de setenta, possam ser verificados e colocados em prática com sucesso. Numa relação de causa e efeito, o mercado, na busca de maior precisão de cálculo e de satisfazer prazos de execução de obras cada vez mais estreitos, vem inflacionando a taxa de uso de sistemas computacionais direcionados ao uso de profissionais de engenharia. Esse quadro potencializa ainda mais a demanda por estudos proporcionalmente mais aprimorados desses sistemas e de profissionais capazes de utilizá-los, deferindo-lhes crescente confiabilidade e utilidade. Os

[1] Portal Terra Tecnologia, http://tecnologia.terra.com.br/pais-tera-um-computador-por-abitante-em-2017-diz-pesquisa,0838138d3b35b310VgnCLD200000bbcceb0aRCRD.html, 18 Out. 2012

métodos manuais de cálculo estrutural geralmente carregam em suas formulações várias imprecisões oriundas de aproximações e simplificações. Geralmente esses métodos demandam um tempo excessivamente grande de análises, tornando-os inviáveis para os padrões atuais. Portanto o uso de ferramentas numéricas de cálculo estrutural, principalmente no caso de estruturas reticuladas[2], como os métodos das forças, ou da rigidez, igualmente conhecido como método dos deslocamentos, é ostensivamente utilizado nos programas de cálculo estrutural por barras reticuladas.

A interface gráfica para a parametrização de dados se insere naturalmente neste contexto de sistemas numéricos de cálculo estrutural, pois permitem que as estruturas possam ser idealizadas de forma mais elegante, precisa, intuitiva e natural. Percebe-se uma clara tendência do mercado em investir em sistemas que possam trabalhar dessa forma, como por exemplo, o EBERICK da Alto QI, o CYPECAD da Cype, o TQS da TQS informática, o STRAP da SAE Sistemas etc.

O AutoCAD da Autodesk, que foi inicialmente desenvolvido para o desenho técnico, hoje ainda é muito utilizado como ferramenta auxiliar para sistemas de cálculo em engenharia. Salienta-se que a versão gráfica original do programa de grelha utilizado pelo grupo de pesquisa foi desenvolvida em AUTOLISP por (Raymundo, 2008). Infelizmente, plataformas, como a da Autodesk, são excessivamente caras e frequentemente demonstram ineficiência na capacidade de programação de rotinas modulares de cálculo e têm baixa interoperabilidade quando é necessário interagir com outros sistemas computacionais (Müller, 2011) [3]. No Autocad é tradicionalmente utilizado o AutoLisp como linguagem de programação de comandos (macros), que é

[2] Estruturas formadas por elementos de barras e nós.

[3] Marina F. Muller estudou a questão da interoperabilidade geral entre sistemas CAD e os sistemas BIM para as estruturas, particularmente, de concreto armado. As dificuldades relativas à produção de tais arquivos é amplamente discutida em sua dissertação de mestrado.

uma ferramenta notoriamente limitada e lenta[4]. Essas condições aliadas ao preço do sistema AutoCAD são bastante desestimuladoras para o desenvolvimento de novas ferramentas para esse tipo de plataforma.

É frente a essa conjuntura que se justifica naturalmente este estudo, pois se apresentarão alternativas viáveis para se criar uma ferramenta completa de cálculo estrutural, como neste caso, de cálculo de grelhas de concreto armado com uma interface gráfica (API) dedicada.

1.3.2 Precisão de Cálculo

Hoje em dia o cálculo dos esforços solicitantes, deformações e deslocamentos nas estruturas de concreto são feitas usualmente por meio de programas de computador. Geralmente estes processam a estrutura ou por barra ou pelo método dos elementos finitos. Mesmo os que usam o método dos deslocamentos (barras), que são mais simples, exigem a capacidade de resolução de sistemas de equações de grande envergadura. Dessa forma, mesmo uma estrutura simples como a indicada na Figura 1(a), que por várias décadas foi resolvida por meio de técnicas de cálculo de discretização, como indicado na Figura 1(b), ou seja, reduzindo a estrutura em elementos unitários (lajes, vigas, pilares etc), hoje pode ser resolvida como um todo.

[4] Nota-se aqui um agravante que é o fato de programas LISP escritos para uma determinada versão do Autocad raramente podem ser processados corretamente em versões posteriores do sistema, muito menos em CAD's de outras empresas.

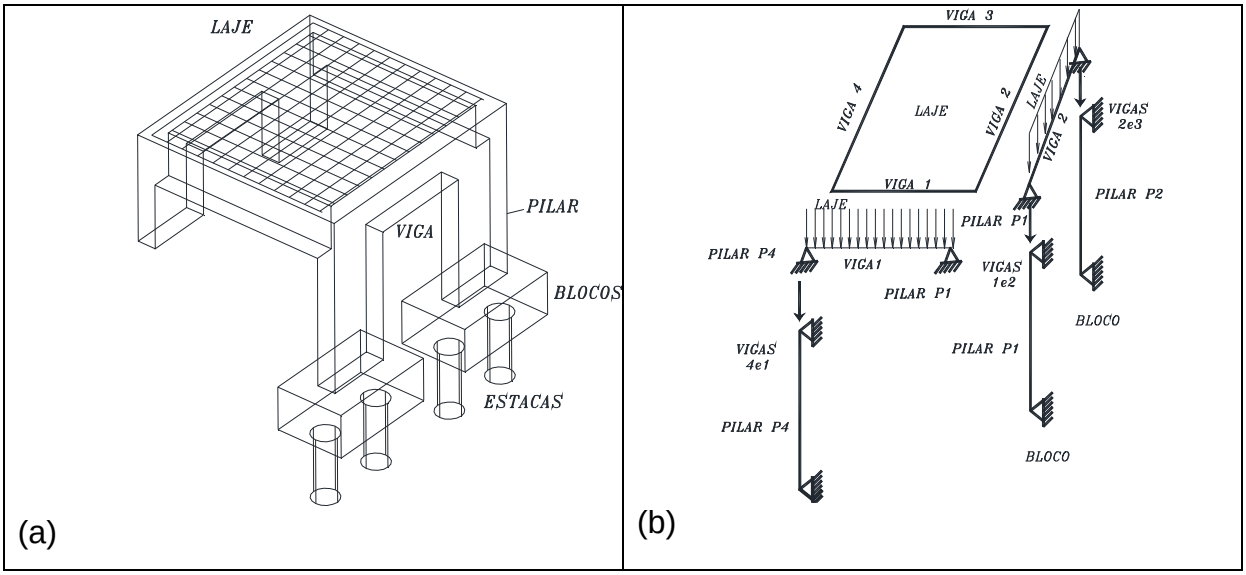


Figura 1: (a) Esquema monolítico da estrutura de concreto armado, (b) estrutura dividida em elementos discretos.

Pode-se perceber que ao configurar, como indicado na Figura 2, a estrutura em um pavimento (laje, pilar e vigas) em um pórtico tridimensional equivalente, para a resolução desta estrutura seria necessário resolver um sistema muito grande de equações lineares. Essa passa ser a principal justificativa para se empregar as técnicas computacionais na resolução de pórticos, ou de qualquer outra estrutura, com uma conformação minimamente complexa. É inviável a análise de estruturas de grelhas, mesmo simples, por meios tradicionais.

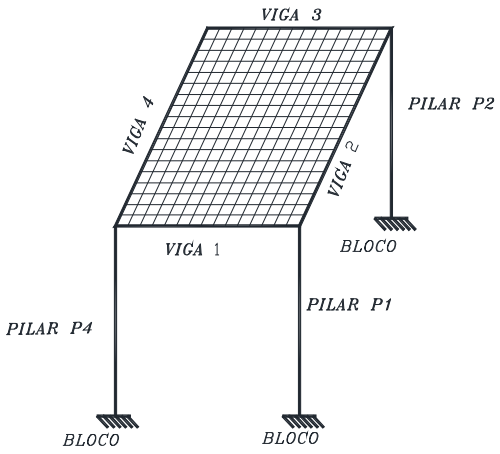


Figura 2: Esquema de modelo de barras prismáticas.

1.3.3 Monolitismo e Economia

As estruturas de concreto armado ou protendido podem trabalhar de forma monolítica, ou seja, como um todo. Considerar as características monolíticas do concreto no cálculo permite considerar as estruturas como mais rígidas e com maior hiperestaticidade e, portanto, com maiores reservas diante a condição de colapso, o que permite o dimensionamento de elementos estruturais mais simples e econômicos. O monolitismo só pode ser considerado com razoável precisão usando uma metodologia de cálculo não discretizante, que não despreze as ações intrínsecas de interação dos elementos estruturais. É importante salientar que as estruturas de concreto armado têm características peculiares que precisam ser consideradas, como por exemplo, a fissuração do concreto à flexão e à torção. Assim os melhores programas de análise de estruturas são aqueles em que são considerados estes efeitos. Programas comuns podem não fornecer resultados adequados para a elaboração de projetos dentro das normas atuais, por não considerarem esses efeitos em toda sua magnitude.

1.3.4 Ferramentas Livres de Ensino

Atualmente a maior parte dos escritórios de engenharia beneficia-se dos programas do tipo CAD ou outros específicos de cálculo estrutural, que tratam a estrutura como um todo. No entanto nas universidades ainda continuam sendo lecionados os métodos simplificados de cálculo. Justamente pela falta de ferramentas desse tipo disponíveis para o ensino. Os programas comerciais, além de terem um alto custo, são fechados e nada mostram a respeito dos algoritmos utilizados, o que os torna pouco úteis como ferramenta “*Strictu Sensu*” de ensino em estruturas.

Apesar de mais de trinta anos terem-se passado desde as primeiras publicações a respeito da análise de estruturas reticuladas (Gere, et al., 1987), ainda não é

possível encontrar programas gratuitos de grelha, muito menos livres nos moldes da OSS[5] "*Open Source Software*".

Tais sistemas são de grande complexidade para serem produzidos e normalmente é necessária uma equipe interdisciplinar de profissionais das áreas de computação e engenharia para a produção dos mesmos. O custo de produção é alto para os investidores, mas o maior problema é que essas empresas passam a ditar o comportamento e a forma de como as estruturas devem ser calculadas pelos usuários desses sistemas. Assim passam as empresas a serem monopolizadoras de tecnologia e "*know-how*". O ensino dos métodos computacionais de análise estrutural com o paralelo aprendizado de uma linguagem de programação básica, porém bem estruturada, podem reverter esse quadro, pois assim permitiria que os discentes começassem a produzir suas próprias ferramentas de trabalho, solidificando, inclusive, seus conhecimentos das teorias numéricas de cálculo estrutural.

1.3.5 Pioneirismo da Interface Gráfica

Finalmente é preciso deixar claro que sem uma interface gráfica é quase impossível de se analisarem grelhas, mesmo as de pequenas dimensões. É importante, para ter-se um resultado adequado, que o sistema seja complementado com uma plataforma gráfica de edição, de tratamento de dados e apresentação de resultados.

O sistema como um todo deve ser projetado pautando-se nas disciplinas de análise de estruturas, estruturas de concreto, cálculo numérico e computação gráfica, pois

[5] **O Software de Código Aberto** é um programa de computador com o seu código fonte disponibilizado e licenciado, com uma licença de código aberto, no qual o direito autoral permite de estudar, modificar e distribuir o software de graça para qualquer finalidade. Esse tipo de Software, geralmente, é desenvolvido de forma colaborativa e pública. Um relatório do Standish Group de 2008 afirma que a adoção do modelo OSS para softwares resultou numa economia direta de cerca de 60 bilhões de dólares ao ano aos consumidores.

sem essas áreas de conhecimento o programa certamente não poderá ser produzido.

Atualmente a computação gráfica está diretamente relacionada à evolução dos computadores ("*hardware e software*")[6]. De fato atualmente alguns computadores têm performance compatível com alguns tipos de estações de trabalho, e no mercado encontra-se uma grande variedade de dispositivos gráficos de alto desempenho. Essa evolução vem permitindo a criação de aplicativos gráficos com interação em outras áreas da ciência (Figura 3), interação caracterizada por uma via de dois sentidos, onde a computação gráfica tanto recebe subsídios quanto fornece.

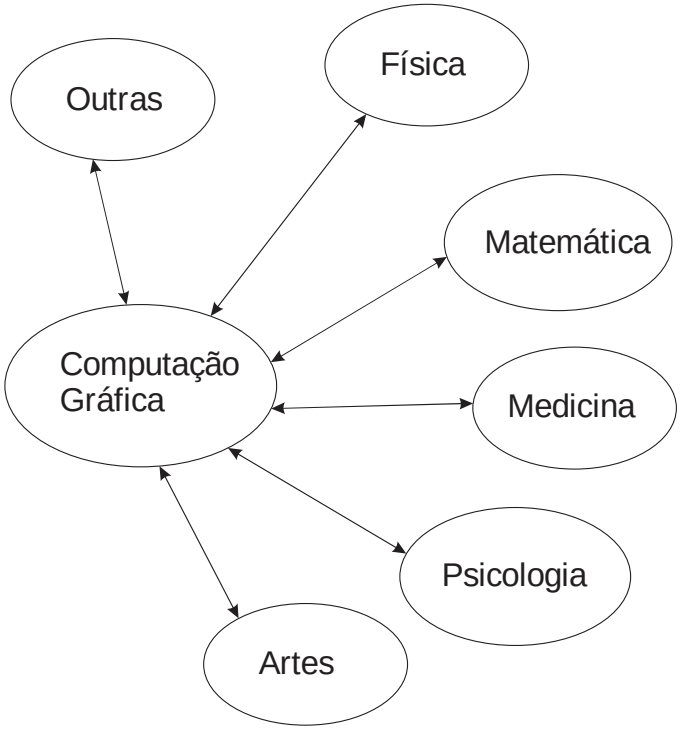


Figura 3: Interação computação gráfica e demais ciências.

[6] Em um sistema de processamento de dados existem dois componentes principais: O Hardware que é o conjunto de toda a parte física e o software que é toda a parte lógica do computador.

Tanto na física, na matemática quanto na engenharia, a computação gráfica recebe modelos numéricos de análise técnico-científica, e por sua vez fornece as ferramentas de visualização que permitem codificar as informações e dados experimentais de forma gráfica e intuitiva.

Essa via de mão dupla vem criando um novo modelo de ensino nas universidades, que vêm integrando cada vez mais a ciência da computação aos seus currículos. Notadamente a PUC-Rio incluiu a computação gráfica como linha de pesquisa na área de concentração em estruturas do Departamento de Engenharia Civil[7].

1.3.6 Precisão dos Modelos Gerados

Segundo Chuang (Chuang, 2006), o uso de um sistema gráfico parametrizado permite a geração de um modelo da estrutura muito mais preciso e eficiente.

É um consenso que uma estrutura, que é representada de forma gráfica, tem seu processo de conceituação estrutural facilitado, pois esta toma a forma de um objeto físico. Esse processo é visualmente muito mais intuitivo, do ponto de vista do projetista.

1.3.7 Facilidade na Verificação dos Modelos Gerados

Uma vez o modelo estrutural conceituado como um objeto, tem-se o acesso facilitado aos elementos constituintes da estrutura e assim obtêm-se mais facilmente os dados relativos a esses elementos. Portanto modelos que apresentam resultados insatisfatórios podem ser facilmente refeitos e redesenhados com novos parâmetros de cálculo.

[7] Site: http://www.puc-rio.br/ensinopesq/ccpg/progciv.html#linhas e o site da Tecgraf de Luiz Fernando Martha: http://www.tecgraf.puc-rio.br/~lfm/

Sendo a verificação dos modelos fácil e intuitiva, há uma drástica redução dos riscos de erros de projeto, sendo possível a realização de inúmeras simulações sem grandes perdas de tempo.

1.4 METODOLOGIA

Inicialmente houve uma grande preocupação em entender o problema apresentado, não somente as questões relativas ao cálculo estrutural em si, mas principalmente as questões de computação gráfica que formam uma lista de problemas que são atípicos do ensino de engenharia civil, como os algoritmos de geração de malhas "*Numerical Grid Generation* "(Thompson, 1998), a organização dos elementos gráficos numa tela de computador "*Viewport Interface Management*", a composição das primitivas gráficas, a manutenção dos arquivos de banco de dados e a maneira como as informações são interligadas e indexadas.

Todas essas questões foram devidamente estudadas e analisadas de forma a se produzir soluções que fossem aplicáveis no escopo desta dissertação. Inclusive apresentar-se-ão mais adiante, detalhadamente, os algoritmos e fluxogramas mais relevantes dessas soluções.

Para efeito comparativo utilizaram-se alguns programas de cálculo estrutural consagrados no mercado, que serviram para a verificação dos resultados.

Finalmente admitiu-se a necessidade da produção de um programa que permitisse ao usuário o lançamento da estrutura por meio de uma ferramenta gráfica do tipo CAD "*Computer Aided Design*", que fosse capaz de transmitir para o módulo de cálculo todos os dados necessários para a análise de grelha, por meio de arquivos de troca com formatos preestabelecidos, que lesse os resultados gerados e que, finalmente, gerasse uma saída gráfica desses resultados.

1.5 ORGANIZAÇÃO DO TRABALHO

No segundo capítulo é estudado: como são resolvidos os pavimentos com grelha equivalente, as características das estruturas de concreto,. como são discretizados os pavimentos com grelhas, como são vistas algumas ponderações a respeito das

malhas e, finalmente, são feitas algumas ponderações a respeito das ações nas estruturas.

No terceiro capítulo são apresentados: a forma de resolução das estruturas reticuladas usando o método dos deslocamentos, os princípios inerentes a elas, os métodos da rigidez e o uso da análise matricial, os fluxogramas de um programa de grelha, o modelo de entrada de dados "*Input*" e de saída de dados "*Output*".

No quarto capítulo vê-se o modelo do sistema gráfico para a análise de estruturas, uma introdução aos princípios de programação, os diferentes paradigmas de OOP "*Object Oriented Programming*"[8], os princípios de herança e polimorfismo, os fluxogramas que compõem o sistema e finalmente a interface gráfica em si.

No quinto capítulo são vistas as rotinas dos componentes do programa e a interdependência deles, os modelos adotados, as linguagens estudadas e as diferentes IDE's "*Integrated Development Interface*"[9] disponíveis para a realização do trabalho.

No sexto capítulo são vistos exemplos e comparam-se os resultados obtidos com aqueles oriundos dos sistemas comerciais, é analisado o desempenho geral dos sistemas envolvidos e suas respectivas limitações.

[8] Programação Orientada a Objetos é um modelo de análise, projeto e programação de sistemas de "*software*" baseado na composição e interação entre diversas unidades de programas chamados de objetos. O paradigma da orientação a objeto tem bases conceituais no campo de estudo da cognição, que influenciou a área da inteligência artificial e da linguística digital.

[9] IDE, do inglês "*Integrated Development Environment*" -Ambiente Integrado de Desenvolvimento-, é um programa de computador que reúne ferramentas de apoio ao desenvolvimento de software que tem como objetivo o de agilizar o processo de desenvolvimento.

2 PAVIMENTOS DE CONCRETO ARMADO E USO DE GRELHA EQUIVALENTE

2 PAVIMENTOS DE CONCRETO ARMADO E USO DE GRELHA EQUIVALENTE

2.1 CARACTERÍSTICAS DAS ESTRUTURAS DE CONCRETO

2.1.1 Particularidade das estruturas de concreto

As estruturas de concreto armado devem ser estudadas de forma diferenciada, apesar do fato que os comportamentos gerais dos sistemas estruturais não dependem muito do material de que são feitos. De maneira geral, os materiais estruturais devem apresentar grande capacidade portante, ou seja, os materiais utilizados devem ter boa capacidade resistente. No caso específico do concreto armado tem-se alta resistência à compressão e baixa resistência à tração, necessitando assim de armadura de aço para suportar as tensões de tração, principalmente as oriundas da flexão. Ainda segundo Carvalho (Carvalho, et al., 2013), para se calcular e detalhar corretamente estruturas de concreto armado, algumas características do material precisam ser conhecidas, como por exemplo:

1. Condições de monolitismo;
2. O papel de diafragma desempenhado em lajes horizontais;
3. O fenômeno de fluência e retração em lajes horizontais;
4. Surgimento de fissuração devido à torção;
5. Condição de não linearidade física de peças comprimidas;

2.1.2 Monolitismo

O concreto armado é um material moldável que permite a realização de estruturas monolíticas. Considera-se como monolitismo a propriedade decorrente da capacidade que um concreto novo tem de aderir-se ao resto da estrutura. Uma estrutura executada como a vista na Figura 4 trabalha como se tivesse sido moldada de uma única vez, ou seja, de forma monolítica. Vê-se na figura que na primeira etapa moldam-se os pilares, concretando até a cota (A). Na segunda etapa concreta-se a viga, formando assim o pórtico definitivo. Finalmente, no esquema estrutural indicado à direita dos demais esquemas, não há distinção das regiões próximas ao ponto K e as demais regiões. Consequentemente, em torno ao ponto (K), as

deformações são compatíveis, ou seja, na seção do pilar e a imediatamente perpendicular na viga obtém-se a mesma rotação.

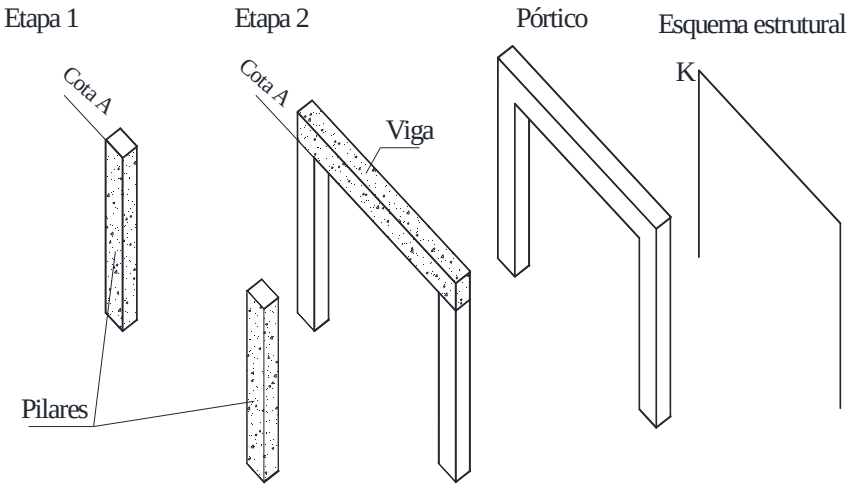


Figura 4: Execução de um pórtico de concreto armado feita "*in loco*".

Nas estruturas de concreto pré-moldado, mesmo em concreto armado, como no exemplo da Figura 5, diferentemente do caso anterior, pode haver um giro relativo entre o pilar e a viga.

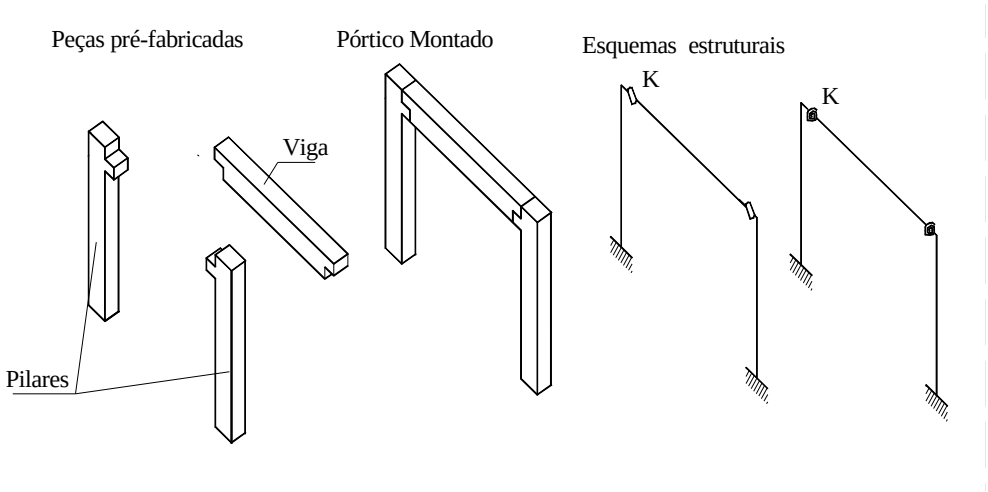


Figura 5: Estrutura de pórtico feito com elementos pré-moldados. O esquema estrutural indica que a ligação entre a viga e o pilar tem rotação totalmente livre de girar ou é parcialmente impedida.

Esse giro pode ter qualquer magnitude caso a ligação seja executada para que haja uma rótula, como mostrado no primeiro esquema da Figura 6. Já no caso de uma ligação semirrígida à flexão, como mostrado no segundo esquema da mesma figura,

haverá um impedimento parcial da rotação. Esse impedimento parcial é denominado de Mola.

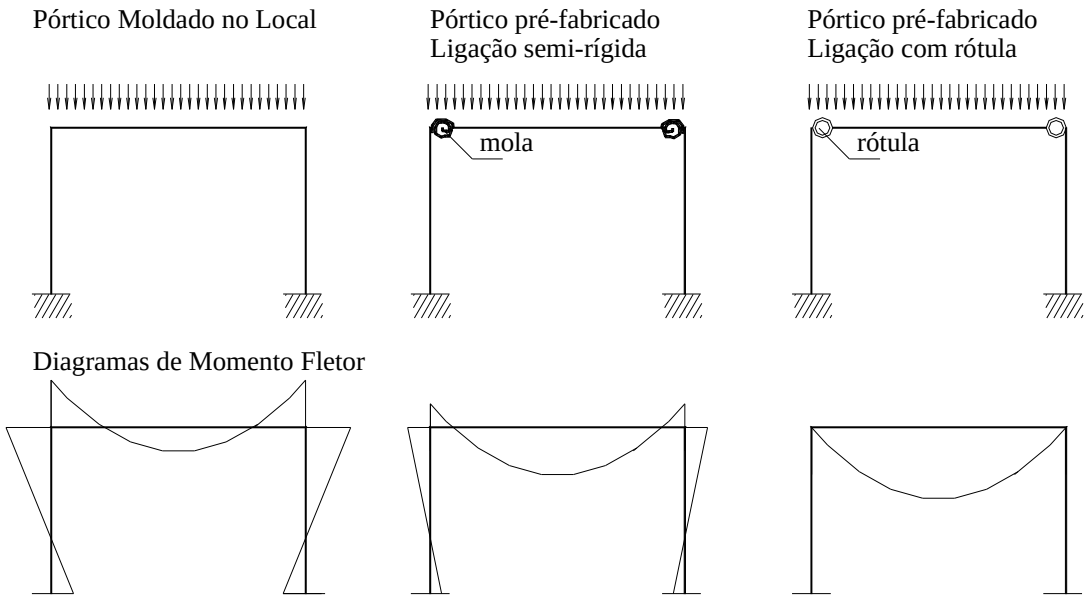


Figura 6: Esquema de pórticos com elementos pré-moldados.

Desta forma, para uma ação uniformemente distribuída, no pórtico considerado, o diagrama de momento fletor ocorre, de forma esquemática, como mostrado na segunda linha de esquemas da Figura 6. Fica claro que, com o pórtico moldado no local, tem-se um momento fletor no pilar maior (em módulo) que nos outros dois casos.

Há também o monolitismo entre a laje e suas vigas de contorno, ou seja, que embora no modelo clássico considera-se a laje girando em torno das vigas sem transmitir momentos a elas, no modelo de grelhas é possível considerar a influência da rigidez à torção das vigas nas lajes.

2.1.3 Lajes horizontais desempenhando o papel de diafragma rígido

Outra característica oriunda do monolitismo do concreto é o efeito de diafragma rígido que as lajes horizontais passam a exercer sobre a estrutura.

Considerem-se duas estruturas, como as mostradas na Figura 7, que podem perfeitamente serem comparadas com uma mesa com tampa e outra sem.

Naturalmente sabe-se que o primeiro caso é mais resistente às deformações que o segundo.

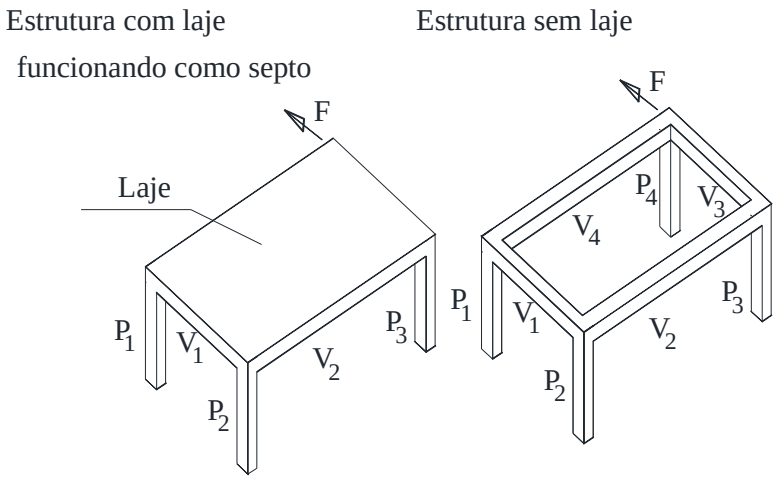


Figura 7: Estrutura com pórtico e laje e apenas com pórtico.

Devido ao fato da primeira estrutura ter a laje maciça, que é concretada junto com as vigas, a ação da força (F) provocará um movimento de corpo rígido de todos os pontos contidos na superfície da laje, inclusive dos pontos centrais e os das extremidades dos pilares. Na segunda estrutura as deformações do pórtico P3, V3 e P4 serão maiores que as do pórtico P1, V1 e P2, pois as vigas V2 e V4 não conseguem transmitir muito esforço para o pórtico. Assim não há um movimento de corpo rígido e as distâncias entre os centros das extremidades superiores dos pilares terão uma deformação diferente que a do caso anterior.

No primeiro caso, devido ao fato da laje fazer o papel de diafragma rígido, todas as vigas V1, V2, V3 e V4 podem ser consideradas com uma inércia transversal muito grande, ou melhor, podem ter desprezados os valores dos momentos fletores transversais e as cortantes correspondentes.

2.1.4 Não linearidade devido à fissuração do concreto

Uma das características importantes do concreto armado é a fissuração à flexão, devida sua baixa resistência à tração. Na Figura 8 é mostrado o resultado obtido da

flecha (máximo deslocamento de um ponto) no ensaio à flexão de uma nervura de concreto armado sob o efeito de carga concentrada.

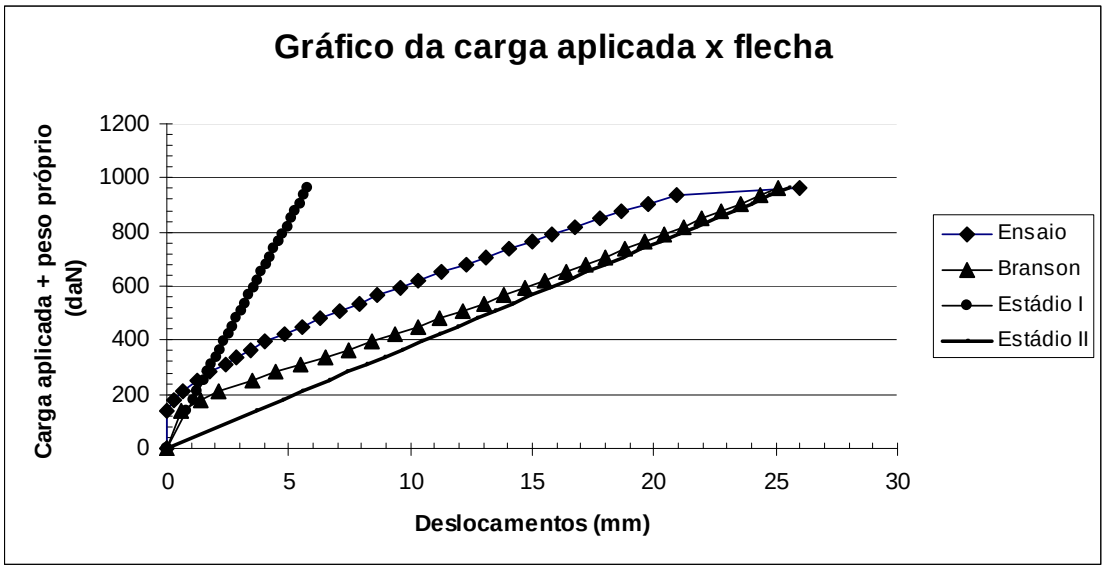


Figura 8: Gráfico de ensaio de nervura de concreto armado de laje pré-moldada (Carvalho, et al., 2013).

A curva mais à esquerda (pontos com pequenos círculos) corresponde à flecha teórica obtida considerando-se que toda a seção da peça está no estádio I. A mais à direita (traço contínuo) representa a variação da flecha, considerando-se que toda a seção está trabalhando no estádio II.

Percebe-se que a curva de ensaio (pontos com losangos) está mais próxima da curva da expressão de BRANSON (pontos com triângulos).

A viga, portanto, tem um comportamento de não linearidade física. Percebe-se esse fenômeno mais claramente na Figura 9, onde numa viga simplesmente apoiada observam-se regiões funcionando no estádio I e outras no estádio II[10].

[10] A definição desses estádios é mostrada no capítulo 3 de CARVALHO e FIGUEIREDO FILHO (2004).

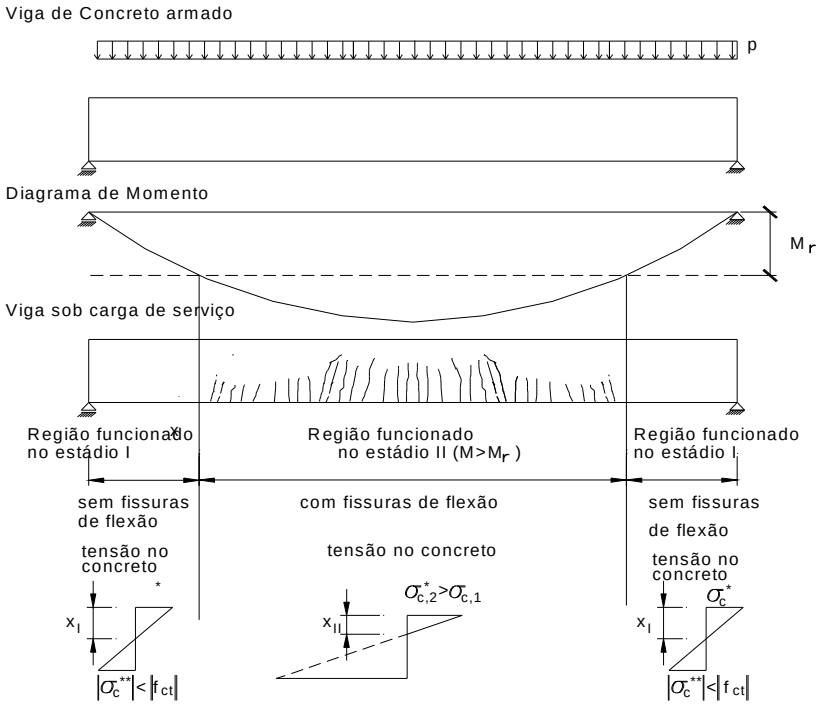


Figura 9: Viga de concreto armado simplesmente apoiada sob ações de serviço.

Para considerar esse efeito é possível resolver a estrutura através de carregamentos incrementais, onde em cada etapa considera-se o grau de fissuração do trecho calculado. Se o momento atuante for maior que o momento de fissuração, altera-se a inércia no trecho pela inércia de BRANSON.

Um programa com as características acima foi desenvolvido pelo autor durante o Estágio Supervisionado de Capacitação Docente – Curso de Noções de Análise Matricial de Estruturas (Tabela 31 – Anexo 2).

2.1.5 Fissuração devida à torção

Além da fissuração do concreto à flexão, há também a fissuração devida à ação da força cortante e da torção. A consideração da torção é primordial, principalmente para elementos prismáticos que têm seção transversal constante e de pequena espessura. O comportamento de uma seção transversal retangular de concreto armado sujeita a um momento torçor está indicada na Figura 10, com duas situações. A primeira em que o momento aplicado (M_x) é inferior à do momento limite de fissuração à torção (M_{tr}), a segunda na qual o momento é superior a este valor. Se este momento torçor aplicado na seção tiver uma intensidade inferior ao

momento limite de fissuração (M_{tr}) a distribuição das tensões de cisalhamento na seção será linear (em relação a um traçado de centro), como considerado na resistência dos materiais (ver detalhe da situação 1). Agora, quando o momento aplicado ultrapassa o momento de torção à fissuração (M_{tr}), então há uma fissuração contínua na seção, isolando a região central da peça (cria-se um núcleo). Vê-se essa condição na situação 2 da figura: a faixa entorno do núcleo tem espessura (t) e a tensão de cisalhamento atuante nessa faixa passa a ser constante. Nesta situação a rigidez da seção transversal diminui sobremaneira.

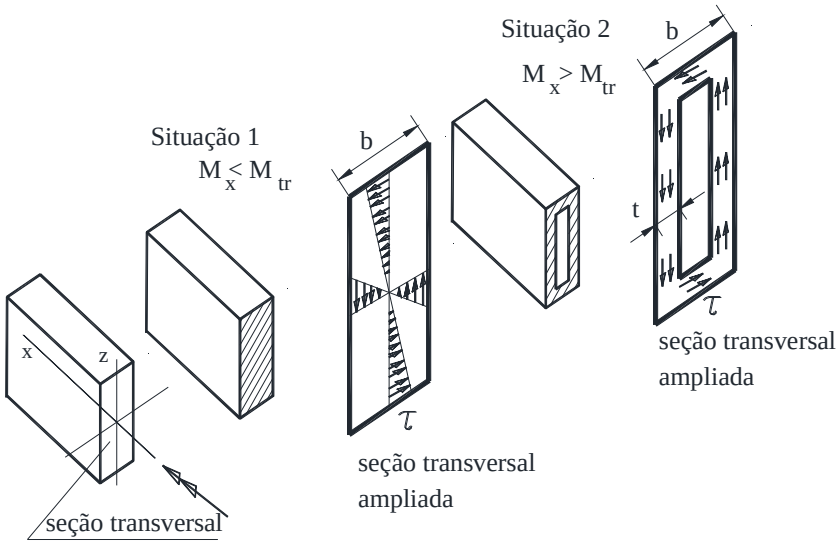


Figura 10: Seção sujeita a esforço torçor.

Na situação o valor da inércia transversal equivale a:

$$I_t = \frac{h.b^3}{\eta} \quad (1)$$

Onde:

h: altura da viga;

b: largura da viga;

η: parâmetro que depende da relação entre h e b (Geralmente usa-se 3);

No estádio II a inércia pode ser tomada como sendo igual a um décimo do valor anterior.

Assim, o projetista de estruturas de concreto, desde que não seja importante para o equilíbrio, pode em diversas situações desprezar a inércia à torção no estádio I. Este é, por exemplo, o caso do possível engastamento de lajes em vigas periféricas, ou seja, a consideração de que as vigas periféricas podem impedir a rotação das lajes.
Na Figura 11 são mostradas situações extremas. Na primeira as vigas têm grande rigidez à torção impedindo assim a rotação nas extremidades da laje, tornando-a engastada à flexão. Na segunda situação imagina-se uma rigidez tão baixa das vigas que a laje praticamente trabalha como simplesmente apoiada nos contornos. Na prática, o que ocorre é uma situação intermediária (rotação parcialmente impedida), que depende fundamentalmente da consideração da rigidez da viga à torção.

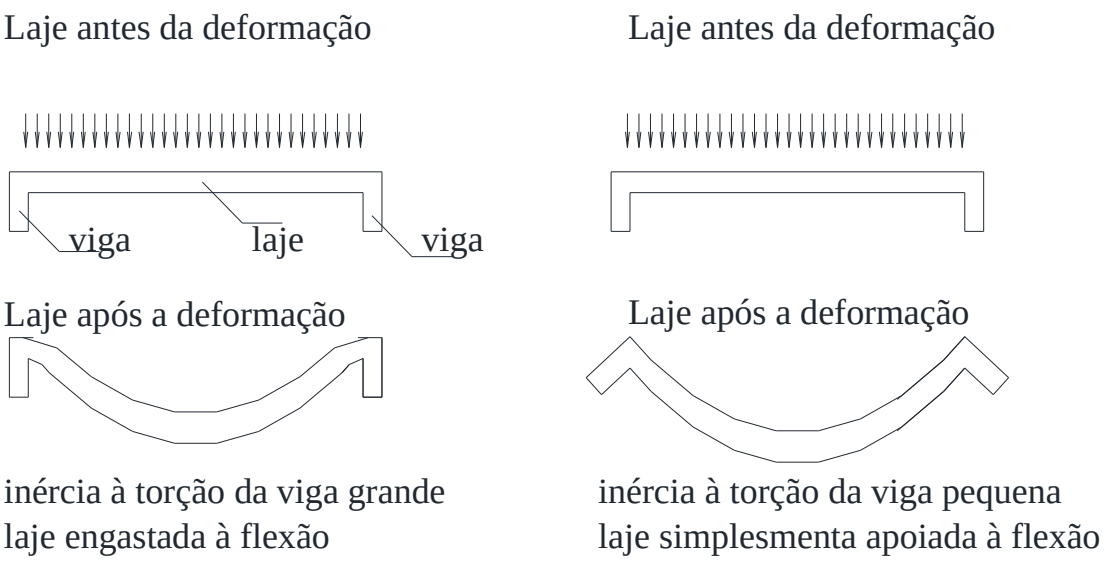


Figura 11: seção transversal de uma laje apoiada em vigas.

Mais adiante é mostrado que a inércia à torção de elementos de concreto de uma grelha equivalente, que representa uma laje, tem um tratamento distinto do que é tratado aqui.

2.2 CONSIDERAÇÃO DAS AÇÕES

O desenvolvimento de projetos de edificações de concreto armado, graças à evolução dos computadores e o amplo acesso a eles, já pode ser feito considerando a estrutura como um todo e em todas as suas três dimensões. Inclusive já é possível a análise de não linearidade geométrica, considerando um cálculo iterativo por análise matricial. Uma dessas técnicas consiste em usar a matriz de rigidez secante, que é a soma de outras três matrizes, onde uma delas é a matriz de rigidez elástica usada na análise matricial comum. Detalhes deste procedimento e a teoria envolvida são amplamente discutidos em (Corrêa, 1987).

O trabalho de Cotta (Cotta, 2006) que aborda o assunto já está disponível na biblioteca virtual da UFSCar. Desta forma, a estrutura da edificação apresentada na Figura 1, que normalmente seria discretizada em lajes, vigas, pilares, blocos e estacas, pode ser considerada como mostrado na Figura 2. O que se observa é a modelagem da laje na forma de grelha equivalente, constituída por barras prismáticas paralelas às vigas. Tal grelha pode ser calculada separadamente do pórtico espacial principal composto por vigas e pilares. A consideração de um pórtico tridimensional é feita por meio das coordenadas globais, como mostrado na Figura 12.

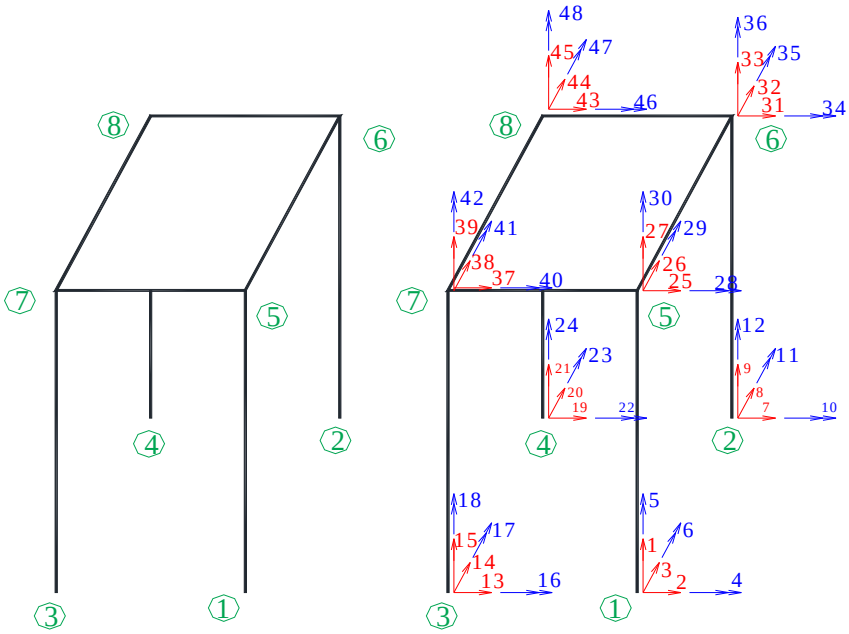


Figura 12: Coordenadas a serem consideradas quando se faz a análise como pórtico tridimensional.

Uma seção transversal de uma estrutura de um edifício, como é tridimensional e normalmente submetida a ações de esforços, apresenta valores de solicitações de todos os tipos possíveis: Momento fletor, momento fletor transversal, cortante, cortante transversal, torçor e normal, como indicado na Figura 13

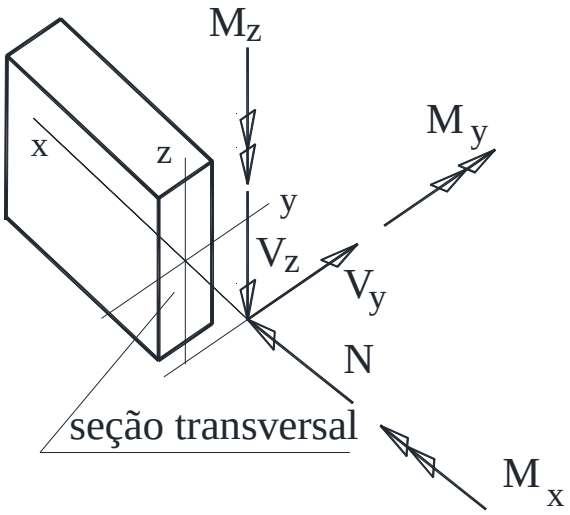


Figura 13: esquema de esforços solicitantes possíveis em uma seção transversal de uma barra.

Em algumas situações e regiões da edificação, como é o caso das vigas travadas por lajes de concreto, o momento fletor (M_z) e a cortante transversal (V_y) podem ser desprezados.

Além dos esforços solicitantes, o conhecimento do comportamento da estrutura na deformação fornece muitas informações ao projetista. Na verdade é através da medição das deformações e deslocamentos por meio de ensaios em protótipos que se pode calcular com razoável precisão os esforços atuantes em uma dada seção ou mesmo as tensões em uma determinada região da estrutura.

2.3 DISCRETIZAÇÃO DO PAVIMENTO COM GRELHA

2.3.1 Processo de Analogia de Grelha

O processo basicamente baseia-se na substituição de um pavimento qualquer por uma grelha equivalente, onde as barras da grelha representam os diferentes elementos estruturais do pavimento (lajes e vigas). Graças a esse processo é possível reproduzir o comportamento estrutural de pavimentos com, praticamente, qualquer geometria, seja ele composto por lajes de concreto armado maciças, com ou sem vigas, ou de lajes nervuradas. Assim deve-se dividir a laje num número adequado de faixas, as quais têm dimensões dependentes da geometria do pavimento. Essas faixas são então substituídas por elementos de barras, que formam uma grelha equivalente representando o pavimento.

As cargas distribuídas no pavimento são divididas entre as barras da grelha de acordo com suas áreas de influência. As cargas podem ser consideradas como uniformemente distribuídas ao longo das barras ou concentradas nos nós. As características das barras são basicamente divididas em dois tipos: as do elemento de placa (laje) e as da união viga-placa (viga-laje). O cálculo da inércia dos elementos à flexão é feito considerando-se uma faixa de largura (b), a qual é dada pela soma da metade dos espaços entre elementos vizinhos, e altura (h) (espessura da placa). A rigidez à torção (I_t), no estádio I, é o dobro da rigidez à flexão (I_f) (Hambly, 1976).

Assim para um elemento de placa pode-se escrever:

$$I_f = \frac{h.b^3}{12} \quad (2)$$

$$I_t = 2.I_f = \frac{h.b^3}{6} \quad (3)$$

Para elementos de viga-placa, na flexão, pode-se considerar uma parte da placa trabalhando como mesa-viga que, dependendo da posição, configura uma viga de seção “T”. Uma vez determinada a largura colaborante, a inércia a flexão da seção resultante pode ser calculada supondo a peça trabalhando tanto no estádio I como no estádio II. A inércia à torção do elemento de viga no estádio I, de forma simplificada e admitindo que a viga é retangular com altura (h) e largura (b) e sem considerar a contribuição da laje adjacente, é:

$$I_t = \frac{h.b^3}{3} \quad (4)$$

Como indicado por Carvalho (Carvalho, 1994) , pode-se considerar a inércia à torção do elemento viga no estádio II como sendo igual a 10% daquele dado pela resistência dos materiais.

Portanto tem-se:

$$I_t = \frac{h.b^3}{30} \quad (5)$$

Os valores do módulo de deformação longitudinal à compressão do concreto (Ec), do módulo de deformação transversal do concreto (Gc) e o coeficiente de Poisson (ν) relativo às deformações elásticas, podem ser determinados a partir das recomendações da NBR 6118-2014.

O programa Grelha realiza por meio de análise matricial o cálculo dos deslocamentos, reações de vínculos e esforços internos solicitantes das estruturas de grelhas. Após a realização dessa análise, chamada de análise linear, é possível executar a análise não linear da estrutura. Essa segunda análise consiste na verificação da não linearidade física, através da comparação do momento limite de fissuração, calculado para um determinado elemento da estrutura, com o momento atuante no elemento. Para tanto, a técnica utilizada é a de carregamentos incrementais onde a referida comparação é feita para cada soma de parcela de carregamento. Se em algum instante o momento de fissuração for superado pelo momento atuante, o valor do momento de inércia é substituído pela inércia proposta pela expressão de Branson (Branson, 1968).

$$I_e = \left(\frac{M_r}{M_{max}}\right)^3 . I_0 + \left[1 - \left(\frac{M_r}{M_{max}}\right)^3\right] . I_n \leq I_0 \quad (6)$$

2.3.2 Metodologia

Para se considerarem as características citadas anteriormente, o trabalho aqui proposto se valerá dos algoritmos do programa 'Grelha 98', que é destinado a realizar a análise de pavimentos. Ao contrário dos programas existentes no mercado, este terá seu código fonte aberto e disponível, permitindo assim que seja aprimorado por profissionais da área.

Para o desenvolvimento deste trabalho, optou-se por um compilador compatível com a linguagem PASCAL. A escolha do PASCAL como linguagem de desenvolvimento para o trabalho aqui proposto deve-se aos seguintes fatores:

a) Capacidade de programação: O PASCAL é uma linguagem estruturada, orientada a objetos e eventos. Essas condições são mínimas para a se criar um programa moderno;

b) Heranças: Algumas bibliotecas de análise estrutural disponíveis no departamento já foram feitas em linguagem PASCAL. Evita-se dessa forma a tradução dessas bibliotecas para outra linguagem, o que nem sempre é uma tarefa trivial;

c) Aprendizado: O PASCAL é uma linguagem relativamente fácil de aprender, que tem uma curva de aprendizado bastante acentuada. A linguagem foi desenvolvida com o intuito de ser acadêmica[11] e com um controle extremamente rígido de tipos, o que faz com que um programa em PASCAL não possa ser executado se contiver erros;

[11] A linguagem PASCAL recebeu este nome em homenagem ao matemático Blaise Pascal e foi criada em 1970 pelo engenheiro suíço Niklaus Wirth. A linguagem foi criada para serem ensinados os padrões da programação estruturada. O PASCAL tornou-se muito popular após a criação do TURBO PASCAL da Borland disponível para os computadores de arquitetura 8086.

d) Modernidade: Existem várias IDE's disponíveis baseadas nessa linguagem, notadamente o "*Lazarus Free PASCAL*" que é gratuita e mantida por uma comunidade universitária distribuída pelo mundo;

e) Rapidez: Programas compilados em PASCAL geram um código executável nativo bem compacto e que não necessita de bibliotecas adicionais para serem executados. Essa condição lança uma enorme vantagem com relação aos programas baseados em BASIC ou similares pos esses programas são compilados em "*Pseudo-Code*", ou seja, endereços que devem ser associados às rotinas pré-definidas nas bibliotecas do núcleo do compilador. Os programas resultantes são muito maiores e mais lentos. Tendo-se em mente que neste trabalho tratam-se matrizes de grandes dimensões e de tamanho variável, o PASCAL certamente é uma opção adequada;

f) Multiplataforma: No caso do "*Lazarus*", os programas podem ser compilados tanto para Windows como Linux, OS, IOS ou Android;

g) Acesso a "*Heap*": O PASCAL como as linguagens de alto desempenho tem acesso a "*Heap*"[12] (bloco de memória principal) e não se limita ao "*Stack*" (pilha de dados que geralmente tem tamanho de 64k). Isso permite a manipulação de grandes quantidades de dados na forma de vetores e matrizes dinâmicos;

h) Comunidade acadêmica: A linguagem PASCAL é muito difundida no meio acadêmico, uma grande quantidade de bibliotecas está disponível na internet, em bibliotecas, numéricas, estatísticas, de controle de produção, bancos de dados, controles gráficos etc;

[12] "*Stack"* são regiões da memória alocadas na forma de pilha de dados. As pilhas podem ser do tipo FILO (*First In Last Out*) ou FIFO (*First In First Out*). Essas pilhas são uma forma muito eficiente de acesso às variáveis, no entanto essas regiões costumam ter um tamanho bem limitado. Quando queremos lidar com vetores ou matrizes muito grandes é comum o erro de "*Stack Overflow*" que significa estouro de pilha. Programas baseados em memória "*Heap*" têm a vantagem de não ter limites de memória, no entanto a manutenção da memória fica por conta do programador, o que não é uma tarefa simples.

Existe hoje uma grande variedade de linguagens de programação, algumas mais direcionadas aos aplicativos comerciais outras para o tratamento científico de dados. Algumas, como o FORTRAN, entraram em desuso, outras como o JAVA, tornam-se extremamente populares principalmente devido à internet e a multiplicidade de plataformas de SO (sistemas operacionais).

Algumas linguagens geram códigos extremamente rápidos de serem executados, como o C, C++ que têm a perigosa liberdade de permitir quase qualquer tipo de código. A escolha de uma linguagem ou outra deve seguir uma mistura de praticidade, conhecimento das capacidades da linguagem e disponibilidade de compiladores.

2.3.3 Considerações sobre Malhas

O sistema operacional tem suas bibliotecas, como a API e GDI[13] e nelas existe uma grande quantidade de funções e procedimentos que podem ser acessados por qualquer aplicativo do sistema. Este é um ponto fundamental quando se tem como parte dos objetivos o barateamento do custo geral sistema. De fato, baseando-se nessa conjuntura, não é interessante criar-se funções que já são existentes ao sistema operacional. Além da GDI, podem-se usar os “drivers” de alto desempenho como, por exemplo, o OpenGL, DirectX.

Drivers desse tipo têm alto desempenho na execução de procedimentos gráficos, como a geração de superfícies tridimensionais, malhas, nurbs, renderizações e etc. Os fatores limitadores desse tipo de “driver” são que eles precisam ser instalados no sistema separadamente e com antecedência. O maior fator limitante desse modelo gráfico é a necessidade de se acessar as bibliotecas por meio de códigos de máquina escritos em Assembler, que é uma linguagem altamente complexa e pouco

[13] API: “*Application Program Interface*”, GDI “*Graphic Device Interface*”, essa duas bibliotecas são responsáveis pela capacidade do sistema operacional de gerar uma interface amigável que seja capaz de ser executada em qualquer configuração de computador.

amigável. Algumas bibliotecas em PASCAL foram desenvolvidas para dar acesso a essas bibliotecas: um projeto bastante conhecido e difundido no meio acadêmico o GLScene[14] (Lischke, 2006).

Como visto anteriormente, mesmo para uma estrutura simples, o número de elementos empregados para representá-la é muito grande. A geração desses dados, se for feita manualmente, torna o trabalho enfadonho e sujeito a imprecisões. Como as estruturas, na maioria dos casos, são regulares, há a possibilidade de se fazer a geração dos dados necessários para resolvê-las por meio de rotinas de cálculo ou leis de progressão e montagem. Conjuntamente a essas rotinas, são necessárias outras capazes de realizar os cálculos das demais características dos elementos, tais como as características geométricas das barras (área, inércia à flexão etc.), seus comprimentos, cossenos diretores etc. que devem ser acopladas às bibliotecas do programa principal.

Importante também é se projetar a entrada de dados segundo uma metodologia simples e intuitiva. Como visto nas justificativas deste trabalho, a maneira ideal é por meio de um editor gráfico semelhante aos sistemas CAD. Obviamente a vantagem de um editor profissional está no emprego dos vastos recursos nele disponíveis, fruto de vários anos de investimentos contínuos, como no caso da Autodesk. No entanto a maior desvantagem reside no fato do acoplamento dos recursos de um sistema livre a outro comercial e de alto custo. Sabe-se que muitos desses sistemas gráficos são distribuídos com descontos ou até mesmo de graça (por tempo limitado ou com restrições) para as instituições de ensino. Mesmo assim não é uma tarefa fácil o desenvolvimento de estruturas de comunicação para esses programas, que têm uma interface de comunicação extremamente limitada, e muitas vezes após uma atualização, tornam-se incompatíveis com os sistemas de terceiros acoplados a eles.

[14] http://glscene.sourceforge.net/wikka/HomePage

O potencial do sistema de geração de dados e representação de resultados já foi mostrado pelo grupo com o uso do CALCO-Grelhas, que é um programa em LISP compatível com o AutoCAD 2007.

O sistema aqui apresentado, provavelmente, não terá problemas para ser expandido a uma solução de edifício tridimensional e também, noutro momento, para a geração de plantas não simétricas e retangulares. Poderá ser expandido também, para os planos inclinados como o visto na Figura 14, onde se vê uma malha representando o patamar e o plano inclinado de uma escada.

Assim, o desenvolvimento de um programa que gere de forma analítica a malha[15] de nós e barras é o passo inicial para se criarem rotinas de desenho na tela na forma de estruturas. São essas bibliotecas gráficas que permitem a visualização da estrutura e transforma as informações nela contida em dados relevantes para o sistema de cálculo estrutural.

Lajes planas, como a representada na Figura 15, podem ser associadas a uma estrutura equivalente de pórtico espacial como a representada na Figura 16.

[15] Pode-se recorrer a uma extensa literatura a respeito de geração de malhas. Referencia-se aqui: "*Grid Generation*" (Thompson, 1998), "*Mesh User's Guide*" (Bank, 2012), Regular Grid (Colaborative Commons, 2009).

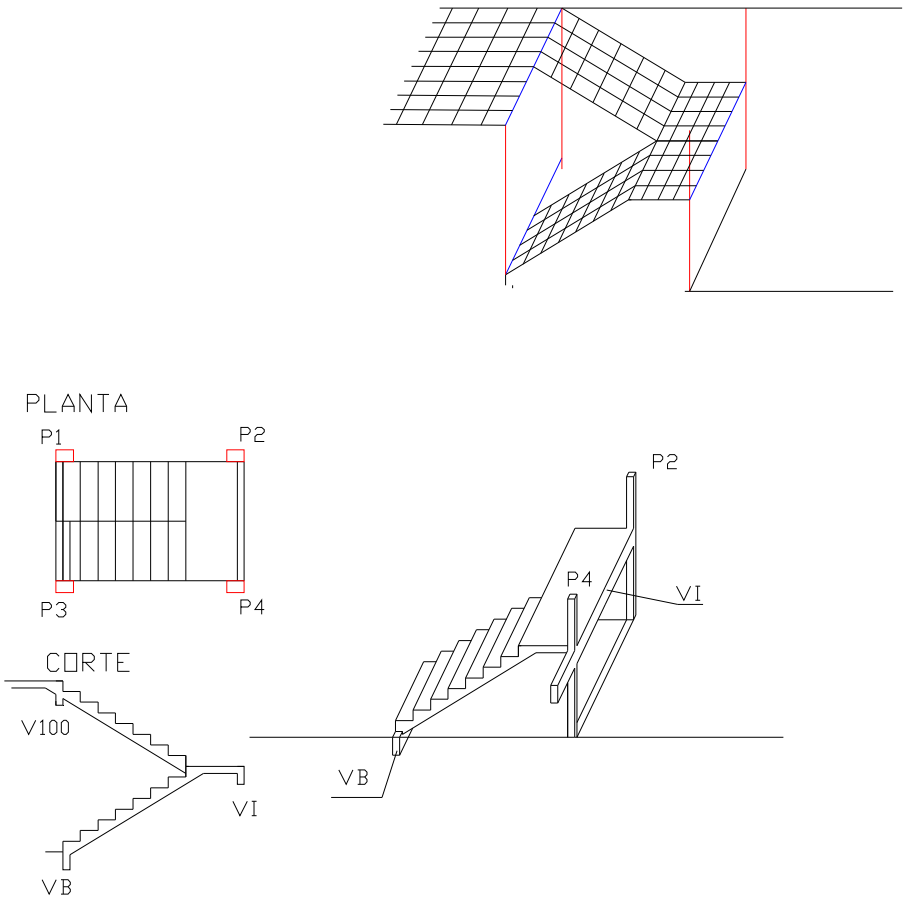


Figura 14: Geração de malhas em planos inclinados.

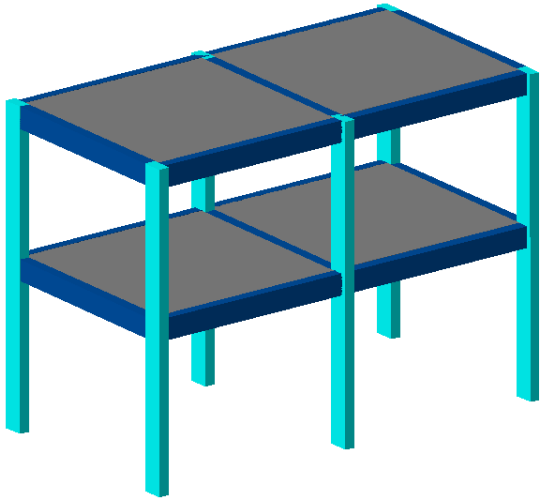


Figura 15: Estrutura com lajes representada espacialmente.

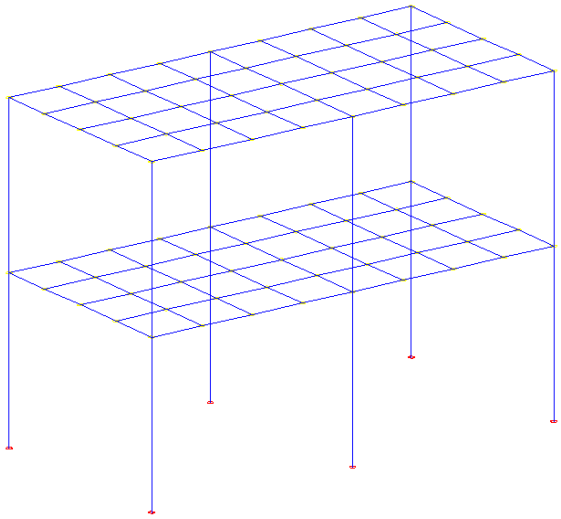


Figura 16: Pórtico espacial equivalente.

2.4 CONSIDERAÇÃO DO PAVIMENTO DE PRÉDIO ISOLADO COMO UM ÚNICO ELEMENTO ATRAVÉS DA GRELHA

Como visto anteriormente, o pavimento pode ser considerado como um elemento estrutural isolado. Para o projeto de um pavimento pode-se valer de sistemas com vigas, lajes maciças Figura 17 ou lajes nervuradas, como as da Figura 19, unidirecionais ou bidirecionais, todas moldadas "*In Loco*". Com vigas pode-se ainda usar lajes pré-fabricadas do tipo alveolar ou duplo "T" e, finalmente, ainda é possível o uso de lajes lisas, ou seja, sem vigas.

Todos esses tipos de lajes podem ser discretizadas por meio de um sistema de grelhas e tratadas isoladamente dos pilares, vigas e dos demais andares. Quanto à ação lateral do vento ou outras ações, estas podem ser consideradas em um pórtico tridimensional e em seguida consideradas nas vigas.

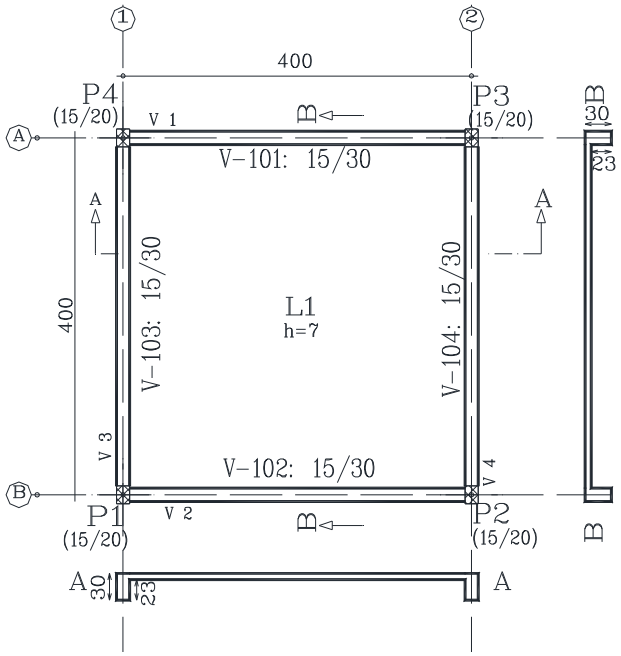


Figura 17: Fôrma de pavimento com laje maciça.

Na Figura 18 mostra-se a perspectiva esquemática da fôrma do pavimento com laje maciça, vigas, pilares e o esquema de grelha equivalente usada na modelagem. Em princípio tanto as vigas como as lajes são modeladas por barras. A diferença está nos valores da inércia à flexão e da inércia à torção aplicadas a cada barra. As vigas costumam ter baixa inércia à torção e alta inércia à flexão.

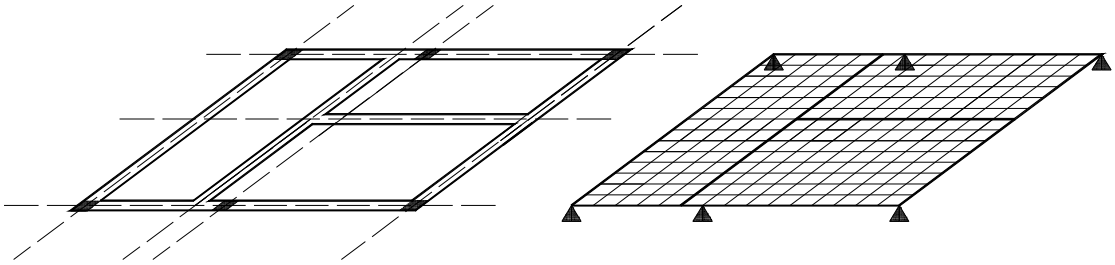


Figura 18: Perspectiva esquemática de fôrma de pavimento com laje maciça e o esquema de grelha equivalente usada na sua modelagem.

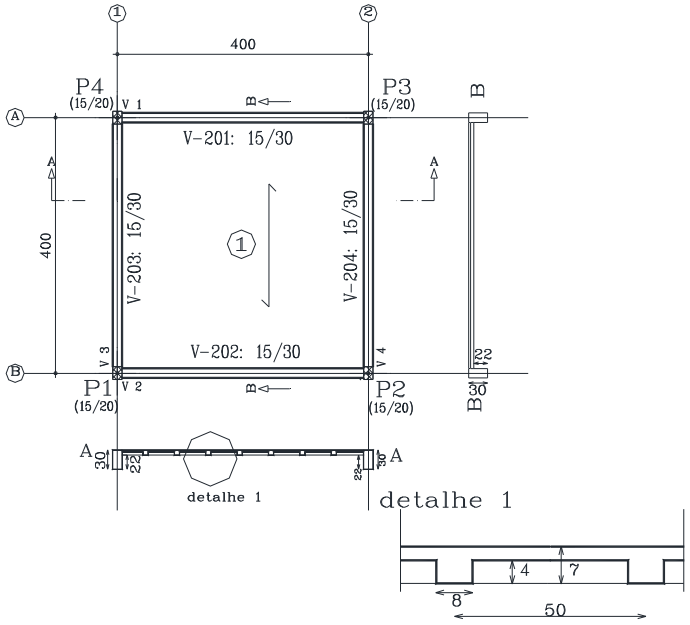


Figura 19: Fôrma de laje unidirecional.

No caso da laje maciça a modelagem deve considerar barras em duas direções. Os pilares são considerados apoios indeslocáveis na vertical. O impedimento de rotação das vigas pelos pilares pode ser considerado colocando-se uma mola nos mesmos. Essa mola só faz sentido quando colocada nos pilares das extremidades. Nos pilares internos, com razoável simetria de cargas e geometria, o momento absorvido é pequeno, mas mesmo assim a mola não conseguirá representar todo o efeito do pórtico espacial que os pilares e vigas dos diversos andares formam.

No caso de pavimento com lajes nervuradas, como o da Figura 20, pode-se igualmente usar uma grelha equivalente para a sua modelagem. Torna-se interessante neste caso que as barras usadas nas lajes coincidam com as nervuras da estrutura. Neste caso a inércia à torção desses elementos (barras que representam as nervuras) será menor que a usada nas barras que representam os elementos de laje maciça.

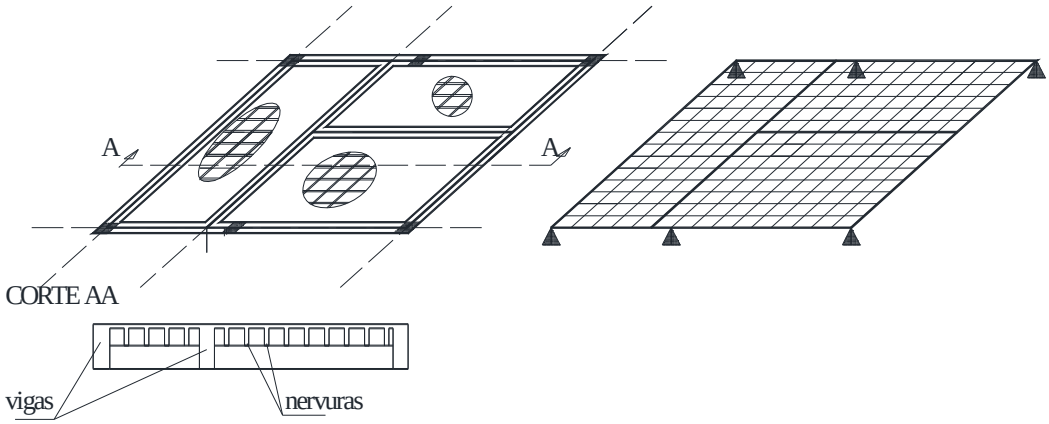


Figura 20: Esquema em perspectiva de fôrma de pavimento com laje nervurada bidirecional e o esquema de grelha equivalente usada na sua modelagem.

Na Figura 21 é mostrado um esquema de grelha equivalente usada para modelar o pavimento com lajes nervuradas unidirecionais. Note-se que as barras representam as nervuras e, portanto, para cada trecho de laje só há barras em uma direção (Carvalho, et al., 2013) e (Flòrio, et al., 2003).

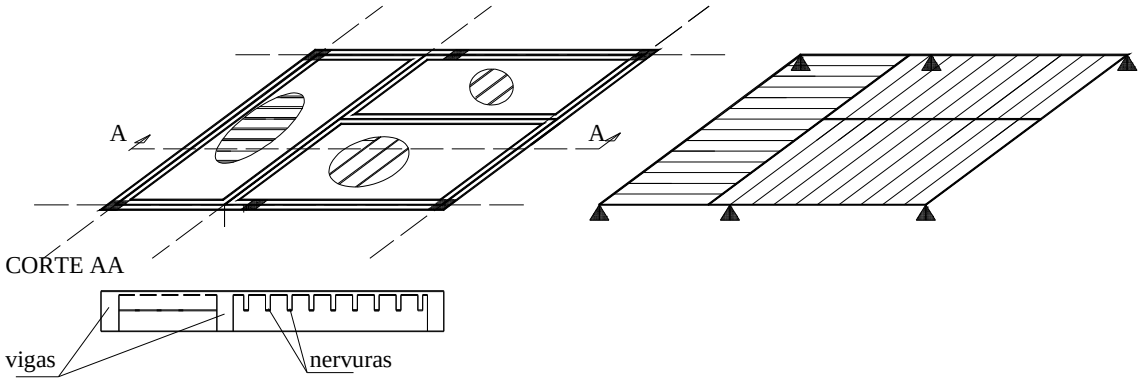


Figura 21: Perspectiva esquemática de fôrma de pavimento com laje nervurada unidirecional e esquema de grelha equivalente usada na sua modelagem.

3 RESOLUÇÃO DE ESTRUTURAS RETICULADAS USANDO MÉTODOS MATRICIAIS

3 RESOLUÇÃO DE ESTRUTURAS RETICULADAS USANDO MÉTODOS MATRICIAIS

3.1 PRINCÍPIOS

3.1.1 Introdução

A teoria da análise de estruturas reticuladas (cálculo dos esforços e deslocamentos) é amplamente detalhada por vários autores, dos quais podem ser citados: (Martha, 2010), (Gere, et al., 1987), (Mukhin, et al., 1983) e (Darkov, et al., 1983). Neste capítulo procura-se, de forma sucinta e simplificada, a introduzir alguns conceitos de forma a encaminhar o leitor para a compreensão mais rápida do restante do processo considerado. Também é ressaltado e focado o método matricial de se resolver as estruturas, sempre considerando o processo dos deslocamentos. Para maiores detalhes e fundamentação teórica, aconselha-se aos leitores, entre outras obras, as citadas anteriormente.

A análise estrutural é a primeira etapa de um projeto, e para iniciá-la é preciso definir o sistema construtivo, o sistema estrutural, o material a ser utilizado, as características geométricas do projeto e as solicitações de serviço.

O objetivo é de, a partir de uma determinada estrutura com características geométricas e mecânicas conhecidas e submetidas a ações (cargas ou deformações impostas), ser possível a determinação dos deslocamentos (translações e/ou rotações) de todos os nós, os esforços internos às barras e as reações de apoio.

Classifica-se a análise estrutural como sendo linear (quando a estrutura tem comportamento linear, ou seja, tem uma relação ação-deslocamento linear, lei de Hooke) ou não linear (quando a estrutura tem um comportamento não linear, geralmente devido a uma conformação geométrica ou de material).

O comportamento linear é típico de estruturas que sofrem deformações ou deslocamentos pequenos e cujo material deve ser elástico[16] e linear, o que permite inclusive a aplicação do “princípio da superposição de efeitos[17]”.

3.1.2 Análise Matricial de Estruturas

Na análise matricial de estruturas, as equações que regem o problema são formuladas matricialmente, sejam estas equações de equilíbrio, de forças ou de compatibilidade de deformações. Existem dois métodos adequados à análise matricial, o método dos esforços ou o método dos deslocamentos, sendo este último mais adequado para a modelação computacional.

3.1.3 Definições

1- **Graus de liberdade:** são as variáveis envolvidas no processo de análise de uma estrutura. Quando se trata do método dos deslocamentos, os graus de liberdade são os deslocamentos (ou rotações) dos nós da estrutura;
2- **Sistemas Contínuos:** são aqueles que possuem uma infinita quantidade de pontos de referência e que, portanto, possuem um número infinito de graus de liberdade. São sistemas indeterminados por natureza;
3- **Sistemas Discretos:** são aqueles que possuem um número finito de pontos de referência e que, portanto, têm um número finito de graus de liberdade. Sistemas contínuos, para serem resolvidos, devem ser convertidos para um sistema discreto equivalente;

[16] O termo elasticidade designa a propriedade mecânica de certos materiais de sofrer deformações reversíveis. Deformações são elásticas quando a ação de forças exteriores provoca deformações que são anuladas se essas forças exteriores se eliminam.

[17] Nos materiais dentro de um regime linear, o efeito da combinação de várias solicitações produz um efeito igual ao produzido por uma única solicitação, equivalente à somatória das ações separadas.

4- **Barras Prismáticas:** considera-se uma estrutura como formada por barras prismáticas aquela cujos elementos são sólidos lineares, ou seja, que apresentam duas dimensões, muito menores que a terceira (da ordem de dez vezes) e que apresentam seção transversal constante, como mostrado na Figura 22. Por se admitir que a seção é constante, admite-se também que as características geométricas (área, inércia etc.) são constantes ao longo do comprimento dos elementos;
5- **Eixos:** é o conjunto dos diversos centros de massa das seções transversais que formam o eixo da peça, representado na forma de uma reta central. Elementos curvos podem ser discretizados por um conjunto de segmentos lineares;
6- **Nós:** os eixos se interceptam nos pontos nodais da estrutura. Para efeito de análise estrutural, podem ser considerados como nós os pontos de apoio da estrutura, as extremidades livres ou qualquer outro ponto pertencente ao eixo de um elemento. É interessante notar que os nós podem ser inseridos arbitrariamente na estrutura, criando-se pontos de controle ou de análise adicional;
7- **Nós de apoio:** os nós de apoio são os pontos onde se apoia a estrutura. Esses nós podem ser indeslocáveis em três, duas ou uma direção. Engastes impedem deslocamentos horizontais, verticais e rotações. Articulações impedem os deslocamentos verticais e horizontais, mas permitem as rotações. O apoio deslizante permite somente um dos deslocamentos, vertical ou horizontal;
8- **Cargas:** são consideradas cargas em uma estrutura as cargas concentradas, distribuídas ou binárias, que submeterão a estrutura a um estado de deformação;
9- **Deslocamentos ou deformação:** é a translação ou rotação dx, dy, dz, d∂ (translações em x, y, z e rotação ∂) de algum ponto nodal da estrutura. Por deformação entende-se que houve pequenas mudanças na configuração dos elementos da estrutura devido a um carregamento. Os deslocamentos em uma estrutura são causados pelo acúmulo de deformações sofrido pelos seus elementos;

10-Grelha: é uma estrutura plana composta de elementos contínuos que têm uma configuração cruzada. O deslocamento em um nó de grelha pode ter três tipos de configuração: uma rotação devido à flexão, uma rotação devida à torção e uma translação em z (deslocamento perpendicular ao plano da grelha);

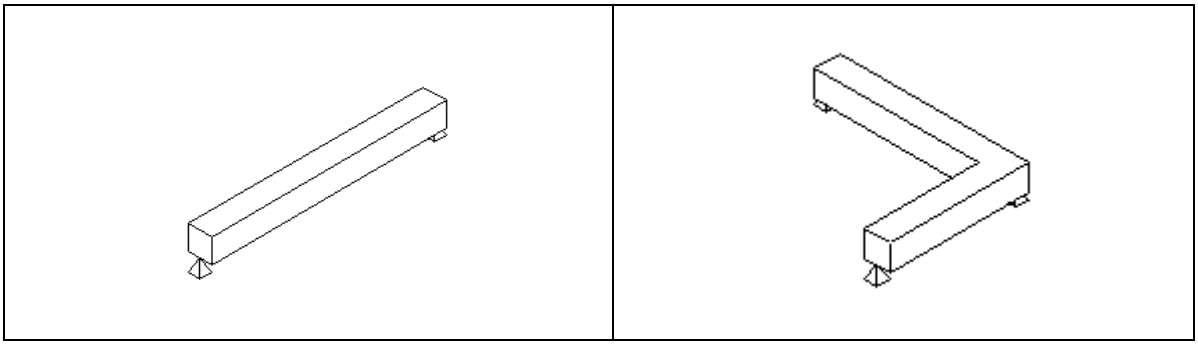


Figura 22: exemplos de barras prismáticas.

3.1.4 Idealização Estrutural

As estruturas podem ser discretizadas em um conjunto de elementos estruturais conectados entre si, representados por barras e nós. O passo mais importante na análise matricial é a correta formulação do problema estrutural, onde os elementos equivalentes devem representar a estrutura contínua real. Tal modelo é imprescindível para se obter um sistema de equações com um número finito de variáveis (graus de liberdade). Tal formulação é a idealização estrutural.

Na Figura 23 vê-se o exemplo da discretização de um pórtico simples.

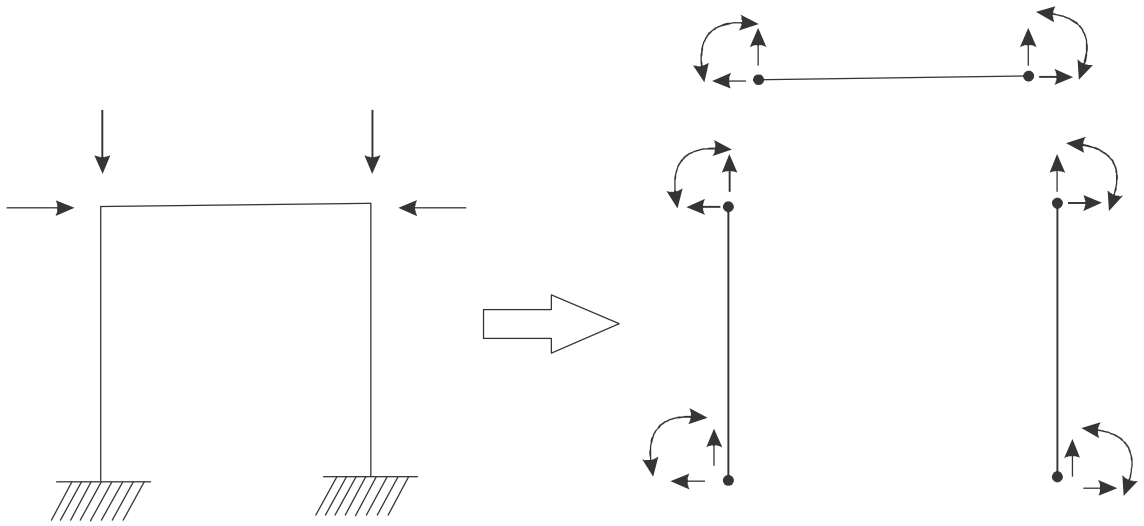


Figura 23: Pórtico da figura (a) representado de forma contínua e o mesmo pórtico discretizado (b) em elementos equivalentes.

Neste trabalho, para facilitar a determinação dos deslocamentos, foram consideradas as seguintes características:

1- Por se tratar de pavimentos de concreto armado com uma rigidez muito grande, foram desconsideradas as forças axiais nos elementos que compõem a grelha. Portanto, surgirão apenas três esforços internos: força cortante, momento fletor e momento torçor.
2- Sendo os elementos de barra do tipo prismático, considera-se o módulo de elasticidade e as características geométricas (Ec, Gc, I, I_t etc.) como constantes ao longo do comprimento dos elementos.
3- Desconsideraram-se também, nesta fase inicial, os recalques (indução de tensão) devido as variações de temperatura e as deformações de montagem da estrutura, pois o estudo desses fatores não fazem parte dos objetivos deste trabalho.

3.1.5 Princípio da Reciprocidade de Efeitos

O teorema de Maxwell-Betti, também conhecido como o teorema da reciprocidade de Betti, demonstra que, em uma estrutura que exibe comportamento elástico linear, se forem considerados dois sistemas de forças, f_{Fi} e f_{Gi}, que provocam dois campos de deslocamento, d_{Fi} e $d_{Gi,}$ então o produto das forças do sistema F com o

deslocamento no ponto de aplicação da força obtido no sistema G é igual ao produto das forças do sistema G com o deslocamento no ponto de aplicação da força obtido no sistema F.

Ou seja:

$$\sum f_{Fi} \,.\, d_{Gi} = \sum f_{Gi} . \, d_{Fi} \tag{7}$$

Considere-se um corpo sólido sujeito a um par de sistemas de forças exteriores, referidos como F_i^P e F_i^Q e que casa sistema de forças provoca um campo de deslocamentos, com os deslocamentos observados no ponto da aplicação das forças exteriores referidos por d_i^P e d_i^Q.

Quando o sistema de forças exteriores F_i^P é aplicado isoladamente ao corpo sólido, o equilíbrio entre o trabalho das forças exteriores e a energia de deformação do corpo é dado por:

$$\frac{1}{2} . \sum_{i=1}^{n} F_i^P \,.\, d_i^P = \frac{1}{2} . \int_{\Omega} \sigma_{ij}^P \,.\, \epsilon_{IJ}^p d\Omega \tag{8}$$

Esta equação representa o equilíbrio entre o trabalho das forças exteriores e a energia de deformação associada à aplicação do sistema de forças F_i^Q isoladamente.

Agora, considerando-se que o sistema de forças F_i^Q é aplicado ao corpo quando o sistema de forças F_i^P já se encontra aplicado. Como o sistema de forças F_i^P já se encontra aplicado e por isso não provocará mais nenhum deslocamento então o equilíbrio do trabalho das forças exteriores e a energia de deformação é descrito através da seguinte expressão:

$$\frac{1}{2} . \sum_{i=1}^{n} F_i^Q \,.\, d_i^Q + \sum_{i=1}^{n} F_i^P \,.\, d_i^P = \frac{1}{2} . \int_{\Omega} \sigma_{ij}^Q \,.\, \epsilon_{IJ}^Q d\Omega + \int_{\Omega} \sigma_{ij}^P \,.\, \epsilon_{IJ}^p d\Omega \tag{9}$$

O mesmo vale para o inverso dos campos de forças.

Se a equação do equilíbrio de energia dos casos em que os sistemas de força são aplicados isoladamente for subtraída da respectiva equação do equilíbrio de energia dos casos em que ambos os sistemas de força são aplicados, então se obtém as seguintes expressões:

$$\sum_{i=1}^{n} F_i^P \,.\, d_i^Q = \int_\Omega \sigma_{ij}^P \,.\, \varepsilon_{ij}^Q .\, d\Omega \tag{10}$$

$$\sum_{i=1}^{n} F_i^Q \,.\, d_i^P = \int_\Omega \sigma_{ij}^Q \,.\, \varepsilon_{ij}^P .\, d\Omega \tag{11}$$

Se o corpo sólido no qual os sistemas de forças exteriores estão sendo aplicados for composto por um material de natureza elástica e linear e os sistemas de forças exteriores provocarem deformações pequenas no corpo, então a equação constitutiva do material, que seguirá a lei de Hooke, poderá ser expressa da seguinte forma:

$$\sigma_{ij} = D_{ijkl} . \varepsilon_{kl} \tag{12}$$

Substituindo esse resultado no conjunto anterior de equações obtém-se o seguinte resultado:

$$\sum_{i=1}^{n} F_i^P \,.\, d_i^Q = \int_\Omega D_{ijkl} . \epsilon_{kl}^P \;\; . \varepsilon_{ij}^Q .\, d\Omega \tag{13}$$

$$\sum_{i=1}^{n} F_i^Q \,.\, d_i^P = \int_\Omega D_{ijkl} . \epsilon_{kl}^Q \;\; . \varepsilon_{ij}^P .\, d\Omega \tag{14}$$

Se ambas as equações forem subtraídas então chega-se finalmente à expressão do teorema de Maxwell-Betti.

$$\sum_{i=1}^{n} F_i^P \,.\, d_i^Q = \sum_{i=1}^{n} F_i^Q \,.\, d_i^P \tag{15}$$

Sejam duas cargas R_i e R_j que atuam em uma determinada estrutura, nos pontos i e j, respectivamente. Conforme o teorema de Betti, o deslocamento no ponto i, na direção de R_i, causado pela ação R_j é igual ao deslocamento no ponto j_n na direção de R_j causado pela ação R_i.

Para ilustrar, tem-se a viga isostática como a da Figura 24, submetida a duas cargas R_i e R_k aplicadas em dois pontos distintos (i e k), acarretando nos deslocamentos d_i e d_k. Os deslocamentos d_i e d_k podem ser obtidos por superposição de efeitos, aplicando-se separadamente as cargas unitárias nos pontos i e k.

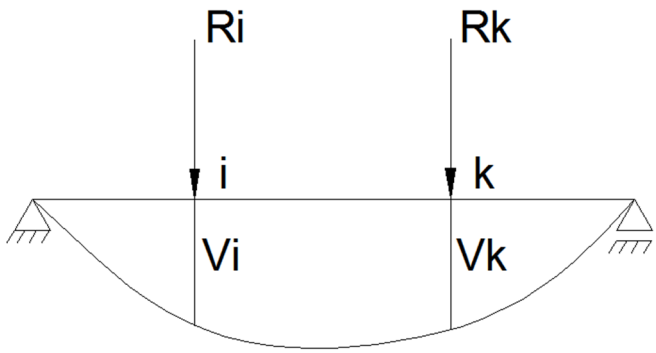


Figura 24: Viga isostática e seus deslocamentos devido a solicitações externas.

Considerando-se apenas uma carga unitária na direção de R_i, têm-se os deslocamentos ∂_{ii} e ∂_{ik}, sendo que ∂_{ii} corresponde ao deslocamento no ponto i na direção de R_i devido a uma carga unitária no ponto i, enquanto que ∂_{ki} corresponde ao deslocamento no ponto k, na direção de R_k, devido à mesma carga unitária no ponto i. Ao se multiplicar essa carga unitária por R_i, têm-se os deslocamentos $R_i\partial_{ii}$ e $R_i\partial_{ki}$, como pode ser visto na Figura 25

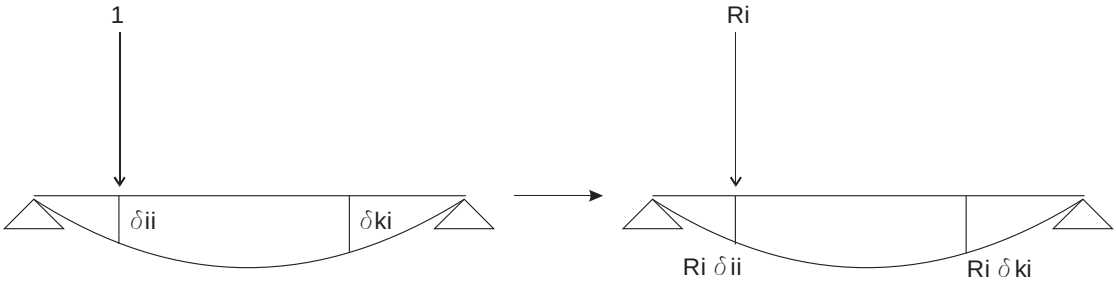


Figura 25: Deslocamentos devidos à carga R_i.

Analogamente considera-se uma carga unitária na direção R_k obtendo-se os deslocamentos unitários ∂_{ik} e ∂_{kk}. Multiplicando-se a carga unitária atuante no ponto k por R_k têm-se os deslocamentos $R_k\partial_{ik}$ e $R_k\partial_{kk}$ conforme a Figura 26.

Vale ressaltar que esse teorema pode ser generalizado para qualquer combinação ou número de cargas que estejam solicitando a estrutura, bem como para qualquer tipo de estrutura dentro das condições de linearidade.

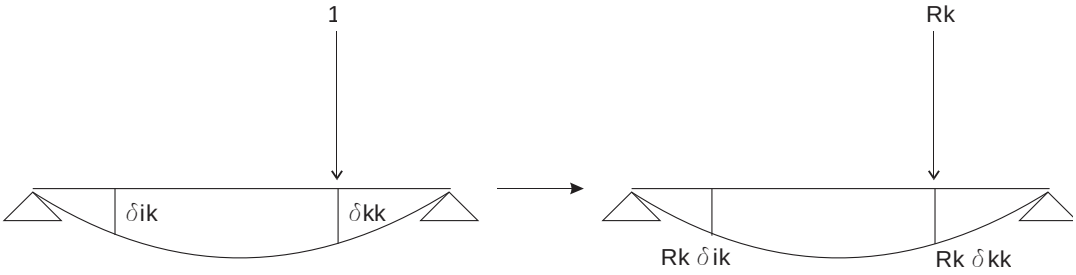


Figura 26: deslocamentos devidos à carga R_K.

3.1.6 Divisão em Elementos

As estruturas analisadas de forma matricial são divididas em elementos de dimensão finita, ligados entre si por pontos nodais (nós) onde, supõe-se, são concentradas todas as forças atuantes nos elementos.

Os nós são pontos fictícios que representam a junção entre duas barras. Os nós podem ser inseridos em qualquer ponto da estrutura, até mesmo entre duas barras, dividindo assim a barra em duas. Como mencionado anteriormente, este é um artifício interessante quando se quer criar um ponto de referência ou de controle na análise estrutural.

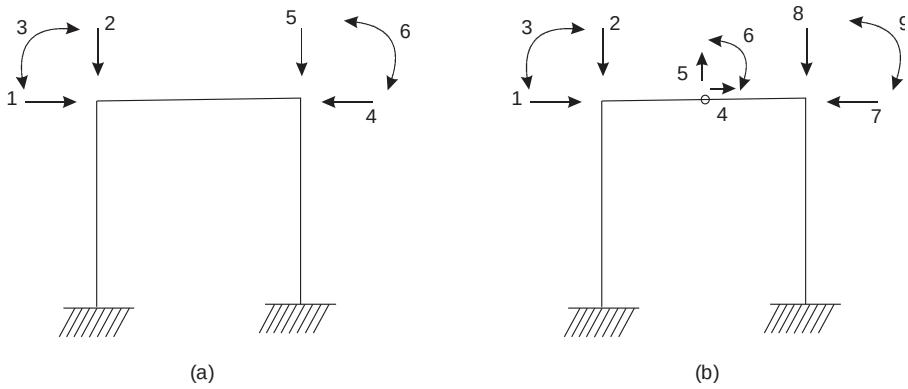


Figura 27: Inserção de nó fictício cortando a barra em dois, gerando um novo ponto de controle.

3.1.7 Sistemas de Coordenadas

Com a finalidade de se criar um sistema de identificação dos nós e barras e ordenar matricialmente as ações solicitantes (forças e momentos) e os deslocamentos

(lineares ou angulares) existentes nos elementos de nós, torna-se imprescindível a criação de um sistema de coordenadas arbitrário.

Na verdade são necessários dois sistemas de coordenadas interdependentes, o primeiro chamado de **Sistema de Coordenadas Globais** e o segundo de **Sistema de Coordenadas Locais**.

O sistema de coordenadas global se refere aos graus de liberdade da estrutura como um todo, já o sistema de coordenadas locais se refere aos graus de liberdade dos elementos discretizados da estrutura.

3.2 FLEXIBILIDADE E RIGIDEZ

3.2.1 Método das Forças

No método das forças, determinam-se diretamente os esforços solicitantes e de forma indireta os deslocamentos. Faz-se uso deste método para a resolução de estruturas hiperestáticas, ou seja, estaticamente indeterminadas.

A estrutura é modificada por meio de cortes arbitrários (liberações) que a torna isostática. O sistema de equações que resolve esse problema é constituído por equações de compatibilidade de deformações cujas incógnitas são os esforços nas liberações ou cortes.

O número de equações (incógnitas) é igual ao grau de hiperestaticidade da estrutura.

3.2.2 Método dos Deslocamentos

Neste método determinam-se inicialmente os deslocamentos (de forma direta) e por meio destes (indiretamente) os esforços solicitantes.

Esse método pode ser usado para qualquer tipo de estrutura hiperestática ou isostática. Na verdade a única estrutura que não pode ser resolvida por este método é a composta por uma única barra biesgastada.

A estrutura é modificada introduzindo-se fixações. O sistema de equações que resolve o problema é constituído por equações de equilíbrio de forças em torno dos nós. As incógnitas são os respectivos deslocamentos, rotações ou translações. O

número de equações é igual ao número de indeterminação da estrutura, ou seja, é igual ao número de graus de liberdade da mesma.
Este método, por não precisar da conversão da estrutura numa estrutura isostática para a análise, torna-se o mais conveniente e é, praticamente, o único utilizado em sistemas computacionais de análise estrutural.

3.2.3 Método da Rigidez

O método da rigidez é um método aplicado em estruturas hiperestáticas compostas por barras que têm um comportamento elástico e linear. O método foi criado para ser adaptado na análise computacional de qualquer estrutura. O método consiste em associar a estrutura de barras à uma matriz de rigidez, que relaciona o deslocamentos nodais da estrutura com as forças externas aplicadas. Essa relação é expressa pela seguinte equação:

$$\begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix} = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n1} & k_{n2} & \dots & k_{nn} \end{bmatrix} . \begin{Bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{Bmatrix} \Rightarrow \{F\} = [k].\{\delta\} \quad (16)$$

Onde:
{F} é o vetor de forças externas na estrutura;
[k] é a matriz de rigidez;
$\{\delta\}$ é o vetor de deslocamentos.

A matriz de rigidez é composta pelos coeficientes k_{nn} de rigidez das barras que compõem a estrutura, que terá uma conformação cujo resultado é oriundo da imposição das condições de contorno na estrutura.

3.2.4 Rigidez e Flexibilidade

A relação entre ação e deslocamento é mostrada na Figura 28 onde a mola k tem um deslocamento u sob o efeito de F: essa relação é definida na equação (17).

$$F = k.u \quad (17)$$

Analogamente pode-se, para um deslocamento unitário, supor a condição da Figura 29. No caso a força F tem uma magnitude igual á rigidez k da mola.

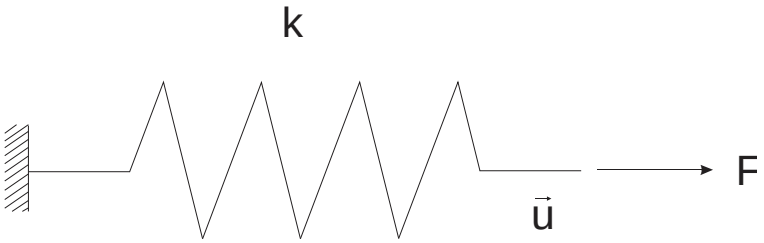


Figura 28: Relação entre ação e deslocamento.

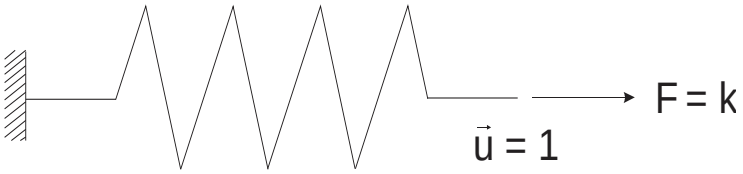


Figura 29: Deslocamento unitário.

Sendo u unitário tem-se que:

$$F = k \tag{18}$$

A relação de ação/deslocamento apresentada na Figura 29 também pode ser formulada em termos de força, como apresentado na Figura 30, onde se tem um deslocamento d para uma força unitária F=1.

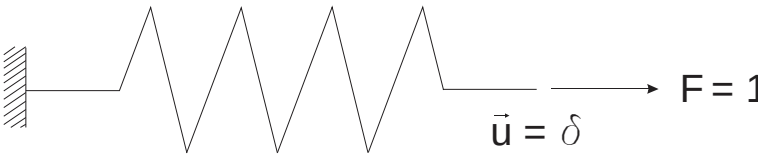


Figura 30: Força unitária.

$$u = \delta . F \tag{19}$$

Onde:

$$\delta = \frac{1}{k} \tag{20}$$

Sendo δ a deformabilidade da mola, geralmente denominada de flexibilidade, onde o deslocamento é determinado por unidade de força.

Se em vez de uma mola for analisada uma barra contínua, como uma viga de um edifício discretizada, de acordo com a resistência dos materiais tem-se:

$$\delta = E.\varepsilon \tag{21}$$

$$\delta = \frac{F}{A} \tag{22}$$

Aplicando a igualdade tem-se:

$$\frac{F}{A} = E.\varepsilon \tag{23}$$

Sabe-se que:

$$\varepsilon = \frac{\Delta \mathrm{l}}{\mathrm{l}_0} = \frac{\mathrm{u}}{\mathrm{L}} \tag{24}$$

Substituindo a equação 24 na equação 23 tem-se que:

$$\frac{F}{A} = E.\frac{\mathrm{u}}{\mathrm{L}} \tag{25}$$

Ou seja

$$F = \frac{E.A}{L}.u \tag{26}$$

Da equação 17 sabe-se então que:

$$F = \frac{E.A}{L}.u \Rightarrow k = \frac{E.A}{L} \tag{27}$$

E analogamente para o coeficiente de Flexibilidade:

$$\delta = \frac{L}{E.A} \tag{28}$$

Por convenção adota-se a letra “S” para o coeficiente de rigidez e a letra ”C” para o coeficiente de flexibilidade.

S_{ij}: Representa a ação na direção i causado por um deslocamento unitário na direção j, enquanto que todos os outros deslocamentos são considerados nulos.

C_{ij}: Representa o deslocamento na direção i, causado por uma ação de valor unitário na direção j enquanto todas as outras solicitações são consideradas nulas.

3.2.5 Obtenção da Matriz de Rigidez de Uma Estrutura

A matriz de rigidez depende da estrutura em análise. Portanto para melhor entender-se o processo deve-se analisar um exemplo específico. Na Figura 31 pode-se considerar uma barra contínua composta por duas hastes e solicitada por esforço normal.

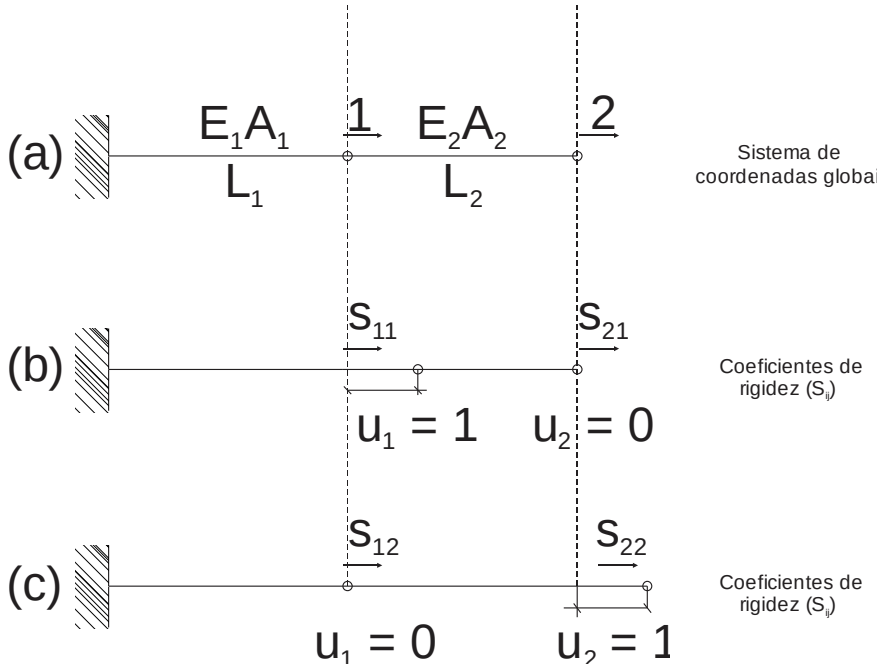


Figura 31:Coeficientes de rigidez em barra composta por duas hastes e solicitada por esforço normal.

No caso acima são conhecidas as ações que atuam nas coordenadas 1 e 2 (A_1 e A_2) e os coeficientes de rigidez (S_{11}, S_{12}, S_{21} e S_{22}) e deseja-se obter os deslocamentos nas coordenadas 1 e 2 (U_1 e U_2).

Para que os nós das coordenadas 1 e 2 estejam em equilíbrio a força externa deve ser igual ao somatório das forças internas resultantes dos deslocamentos ocorridos ao longo da estrutura, ou seja:

$$A_1 = S_{11}.u1 + S_{12}.u2 \quad (29)$$

$$A_2 = S_{21}.u1 + S_{22}.u2 \quad (30)$$

Relacionando-se as equações (29) e (30), matricialmente, pode-se escrever que:

$$\begin{Bmatrix} A_1 \\ A_2 \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} . \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \Rightarrow \{A\} = [S].\{u\} \quad (31)$$

Onde:

{A} é o vetor de ações externas (solicitações);

{u} é o vetor de deslocamentos nos nós 1 e 2;

[S] é a matriz de rigidez da estrutura, de dimensão (2x2), correspondente ao número de coordenadas utilizadas. A matriz de rigidez é uma matriz de transformação linear que transforma o vetor de deslocamentos no vetor de solicitações.

A matriz de rigidez de uma estrutura pode ser obtida pela conceituação de seus coeficientes, e das relações existentes na haste submetida a carregamentos axiais.

S_{11} é a força na coordenada 1 decorrente da imposição de um deslocamento unitário também na coordenada 1, mantendo-se as demais coordenadas restringidas.

$$\begin{matrix} u_1 = 1 \\ u_2 = 0 \end{matrix} \Rightarrow S_{11} = \left(\frac{E_1.A_1}{L_1}\right) + \left(\frac{E_2.A_2}{L_2}\right) \quad (32)$$

S_{21} é a força na coordenada 2, decorrente da imposição de um deslocamento unitário na coordenada 1, mantendo-se as demais coordenadas restringidas.

$$\begin{matrix} u_1 = 1 \\ u_2 = 0 \end{matrix} \Rightarrow S_{21} = -\left(\frac{E_2.A_2}{L_2}\right) \quad (33)$$

S_{12} é a força na coordenada 1 decorrente da imposição de um deslocamento unitário na coordenada 2, mantendo as demais coordenadas restringidas.

$$\begin{matrix} u_1 = 0 \\ u_2 = 1 \end{matrix} \Rightarrow S_{12} = -\left(\frac{E_2.A_2}{L_2}\right) \quad (34)$$

S_{22} é a força na coordenada 2 decorrente da imposição de um deslocamento unitário na coordenada 2, mantendo-se as demais restringidas.

$$\begin{matrix} u_1 = 0 \\ u_2 = 1 \end{matrix} \Rightarrow S_{22} = \left(\frac{E_2.A_2}{L_2}\right) \quad (35)$$

Dessa forma obtém-se a matriz de rigidez da estrutura:

$$[S] = \begin{bmatrix} \left(\frac{E_1.A_1}{L_1} + \frac{E_2.A_2}{L_2}\right) & -\frac{E_2.A_2}{L_2} \\ -\frac{E_2.A_2}{L_2} & \frac{E_2.A_2}{L_2} \end{bmatrix} \quad (36)$$

Na equação 36 vê-se a matriz de rigidez mediante as imposições de condições de contorno da estrutura da Figura 31.

3.3 MATRIZES DE RIGIDEZ LOCAL E GLOBAL

3.3.1 Matriz de Rigidez Local

As operações envolvendo a resolução de estruturas consideram sempre duas situações: a primeira referindo-se às coordenadas locais da estrutura discretizada em elementos independentes, solicitados por esforços {S}, a segunda referindo-se às coordenadas globais onde os elementos estão integrados e submetidos a ações nodais {R}.

A forma mais comum de se relacionarem os dois sistemas de coordenadas é por meio de uma matriz de incidência estática.

$$\{S\} = [B].\{R\} \quad (37)$$

Onde:

{S} é o vetor de esforços nos elementos prismáticos que compõem a estrutura;

[B] á a matriz retangular de incidência estática de dimensão nxm;

{R} é o vetor de ações externas;

Os valores de ‘m’ e ‘n’ são definidos pelo número de coordenadas locais e de coordenadas globais.

A importância da transformação de coordenadas locais em globais reside no fato que as influências de deslocamentos unitários na direção dos esforços (ou deslocamentos) considerados já se encontram tabelados.

3.3.2 Matriz de Rotação

Em casos como o deste trabalho, que se direciona ao estudo de pavimentos por meio de analogia de grelha, é conveniente o uso de matrizes de rotação, que transformam as coordenadas globais em locais. A matriz de rotação permite que as ações atuantes em cada elemento possam ser representadas em um único sistema de eixo global.

Para o caso de uma estrutura plana, como a estrutura de grelha estudada neste projeto, a matriz de rotação será formulada considerando-se que o eixo z permanece na mesma direção e sentido, como mostrado na Figura 32.

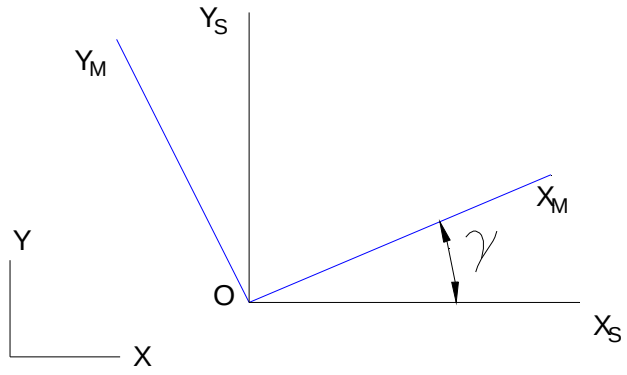


Figura 32: Rotação de eixos para uma estrutura plana.

Neste caso a matriz de rotação apresenta o formato:

$$[R] = \begin{bmatrix} \cos\gamma & -\operatorname{sen}\gamma & 0 \\ \operatorname{sen}\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Esta é a matriz utilizada pelo sistema de cálculo para representar as coordenadas de cada elemento em um sistema de eixos globais.

4 FLUXOGRAMAS E PROCESSO DO PROGRAMA DE GRELHA

4 FLUXOGRAMAS E PROCESSOS DO PROGRAMA DE GRELHA

4.1 INTRODUÇÃO

Em programas similares ao de Grelha, como o GPLAN, a entrada de dados é feita manualmente via editor de texto. As versões anteriores do programa de Grelha também seguiram este princípio (um arquivo texto de entrada "*input*" e outro de saída "*output*" com os resultados). Inicialmente foi proposto para este trabalho, a criação de um ambiente gráfico que permitisse a criação de uma estrutura idealizada sob a forma de elementos gráficos contendo os dados necessários para a geração do arquivo inicial de dados. O processo geral é mostrado na Figura 33 sob a forma de fluxograma.

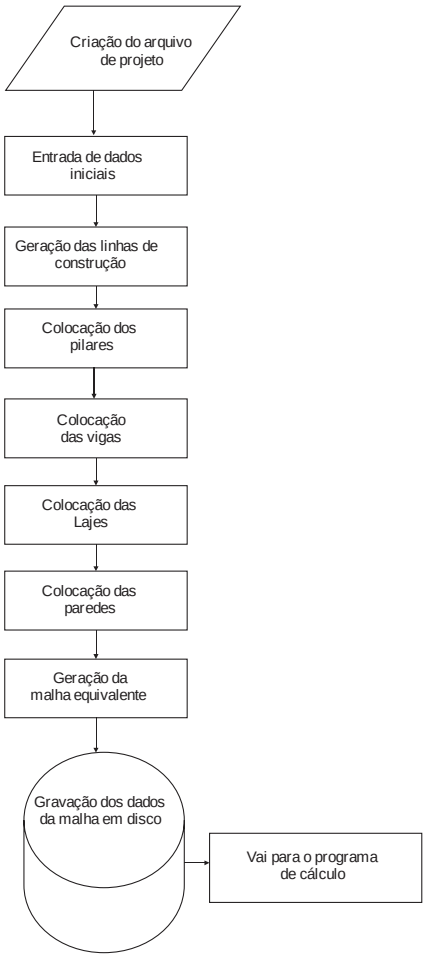


Figura 33: Fluxograma do programa de parametrização da estrutura.

4.2 METODOLOGIA E DEFINIÇÃO DE PROJETO

4.2.1 Criação de Linhas de Construção

Inicialmente cria-se o arquivo de projeto. Este arquivo conterá todas as entidades gráficas do projeto. Essas entidades conterão dados estendidos como as características geométricas, dados de material etc. Entra-se com os dados iniciais de projeto, f_{ck}, Ec, Gc, densidade do concreto e demais dados relevantes, como mostrado na Figura 34.

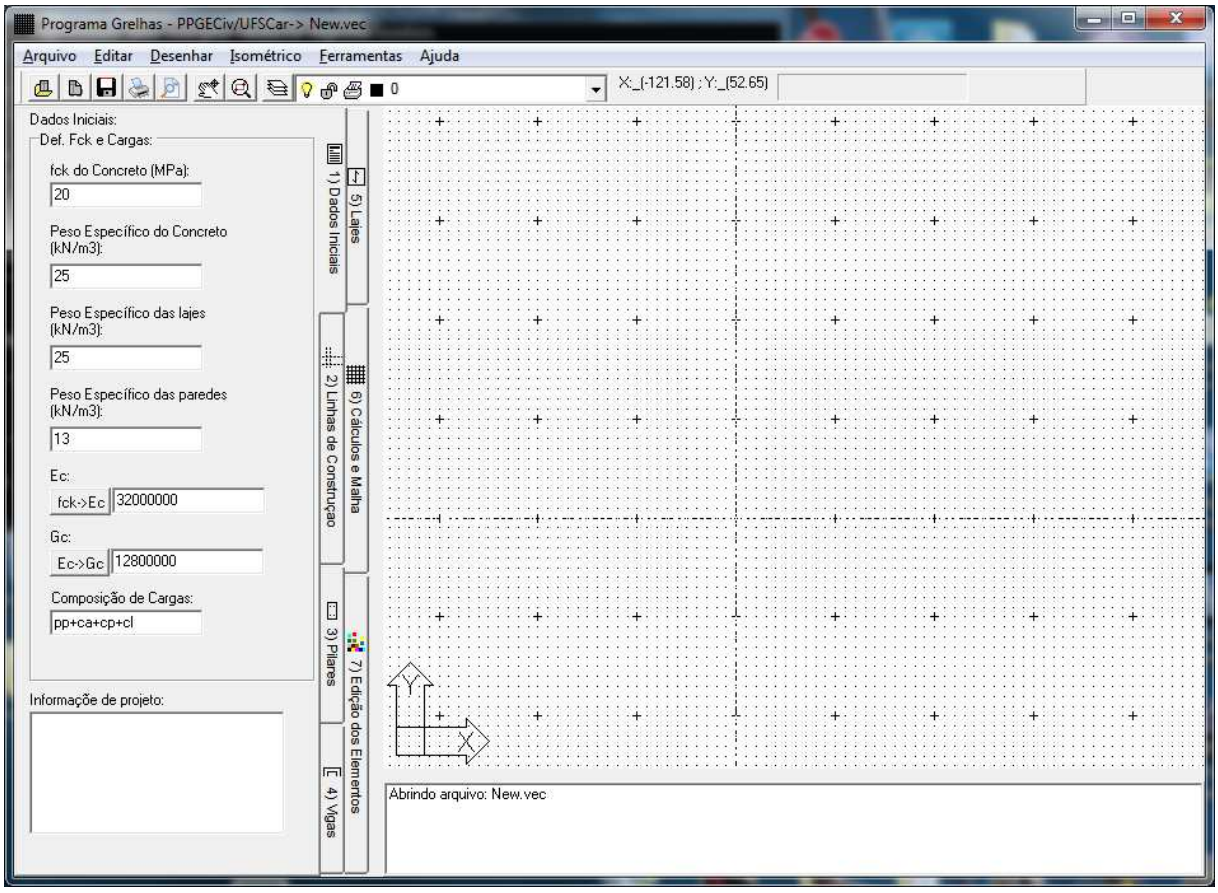


Figura 34: Tela inicial do programa Grelha.

Em seguida entra-se com as linhas de construção, que servirão como eixos de referência para a inserção dos demais elementos estruturais. Os eixos podem ser horizontais ou verticais. No exemplo da Figura 35 vêm-se alguns eixos criados, no exemplo são dois eixos horizontais com 200 cm de distância e três eixos verticais com espaçamento de 200 cm.

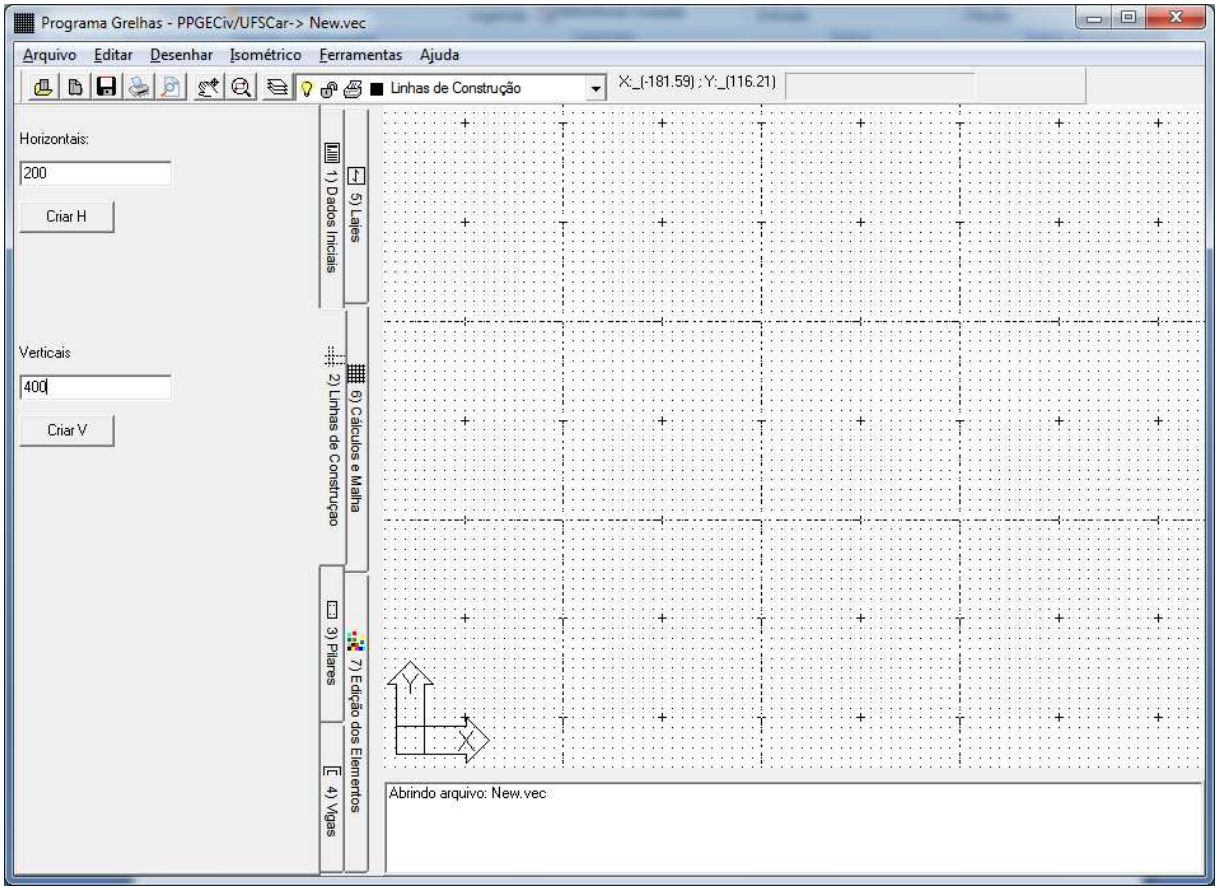


Figura 35: Projeto com alguns eixos criados.

4.2.2 Colocação dos Elementos Estruturais

Após a colocação dos eixos-guia inicia-se a colocação dos elementos estruturais. A ordem sugerida é pilares, vigas e lajes, no entanto esta ordem não é obrigatória.

Definem-se as dimensões do pilar, as condições de apoio na cabeça do pilar e o seu nome. Não é obrigatória a alteração do nome "*Default*" do pilar. Colocam-se os pilares sobre os eixos de referência. O sistema tem a função de aproximação magnética de pontos notáveis, conhecida no universo CAD como "*SNAP*". Na Figura 36 vêm-se alguns pilares inseridos no projeto.

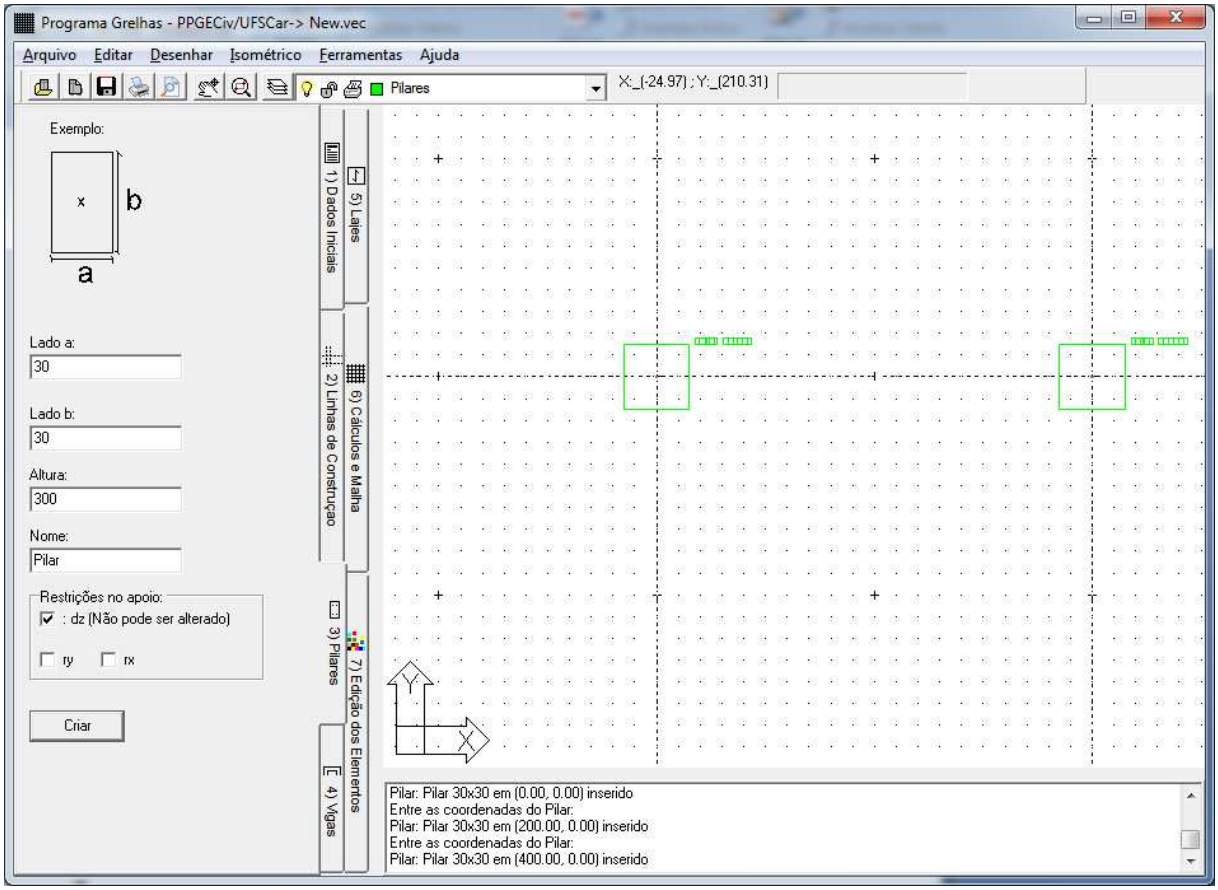


Figura 36: Pilares inseridos.

Em seguida são colocadas as vigas, que devem seguir exatamente os eixos de referência, no caso contrário serão desconsiderados. Antes de se criar uma viga deve-se definir suas dimensões e o seu nome. Como no caso anterior, o nome serve somente para a referência visual, não tento utilidade nenhuma para a análise estrutural em si.

No exemplo da Figura 37 vêm-se quatro vigas inseridas de 30x40cm. As vigas podem cruzar vários pilares de uma vez, não sendo necessária a criação de uma viga para cada um dos vãos. Bastando, portanto, criar-se uma única viga até o último pilar de apoio pertencente ao eixo de referência.

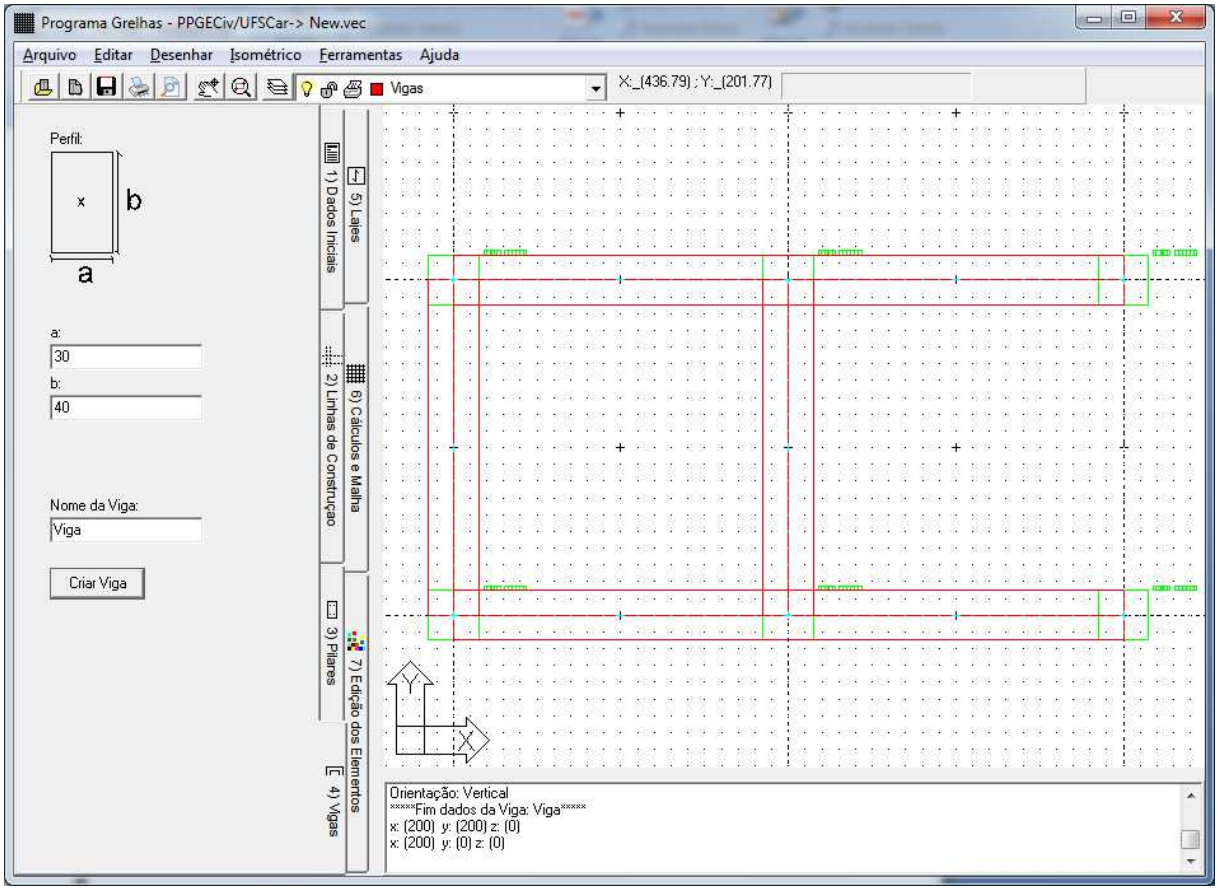


Figura 37: projeto com vigas inseridas.

Finalmente insere(m)-se a(s) laje(s). A laje pode ser definida como uma única laje com espessura “e” ou pode-se inserir várias lajes nos diversos vãos disponíveis da estrutura. Na Figura 38 vêm-se duas lajes, a primeira com 10 cm e a segunda com 18 cm.

Nesta etapa tem-se a estrutura devidamente idealizada, caracterizada e definida geometricamente. Uma vez definida a estrutura passa-se para a próxima etapa, que é a de montagem da malha equivalente.

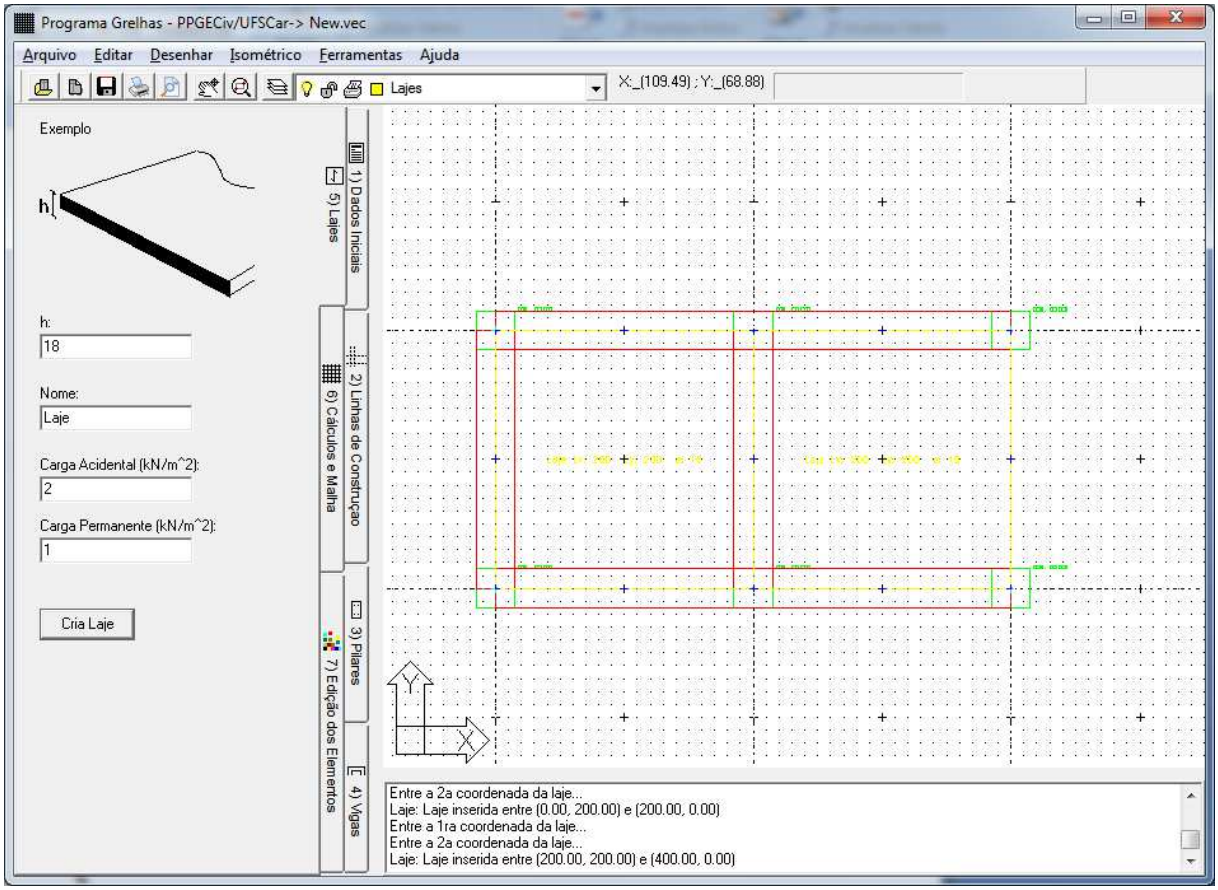


Figura 38: estrutura com duas lajes inseridas.

4.2.3 Criação da Malha Equivalente

A criação da malha deve seguir alguns critérios, que são:

1. A malha será gerada dentro dos limites da(s) laje(s) do projeto, ou seja, toda a área diametral entre a menor e a maior coordenada de lajes será preenchida pela malha[18].
2. A malha deve ser definida de forma que os nós e barras passem pelos eixos relevantes da estrutura, ou seja, passem por pilares e vigas. Para os pilares

[18] Posteriormente planeja-se implementar formas mais eficientes de geração de malhas, que inclusive, levem em conta os vazios das lajes.

há uma exceção, pois qualquer nó dentro dos limites do pilar será considerado indeslocável.

No caso da estrutura da Figura 38 têm-se dimensões de 200cmx400cm o que permite uma malha de 10x20 elementos, que satisfarão às condições acima citadas. Vê-se na Figura 39 a malha criada.

4.2.4 Definição de Cargas Extras

Após a malha definida pode-se aplicar cargas nos nós. Por hora o programa permite a inserção de cargas nodais locais. Para a inserção de cargas distribuídas deve-se proceder de forma manual, dividindo-se a carga distribuída total pelo número de nós menos um, como mostrado na equação (38).

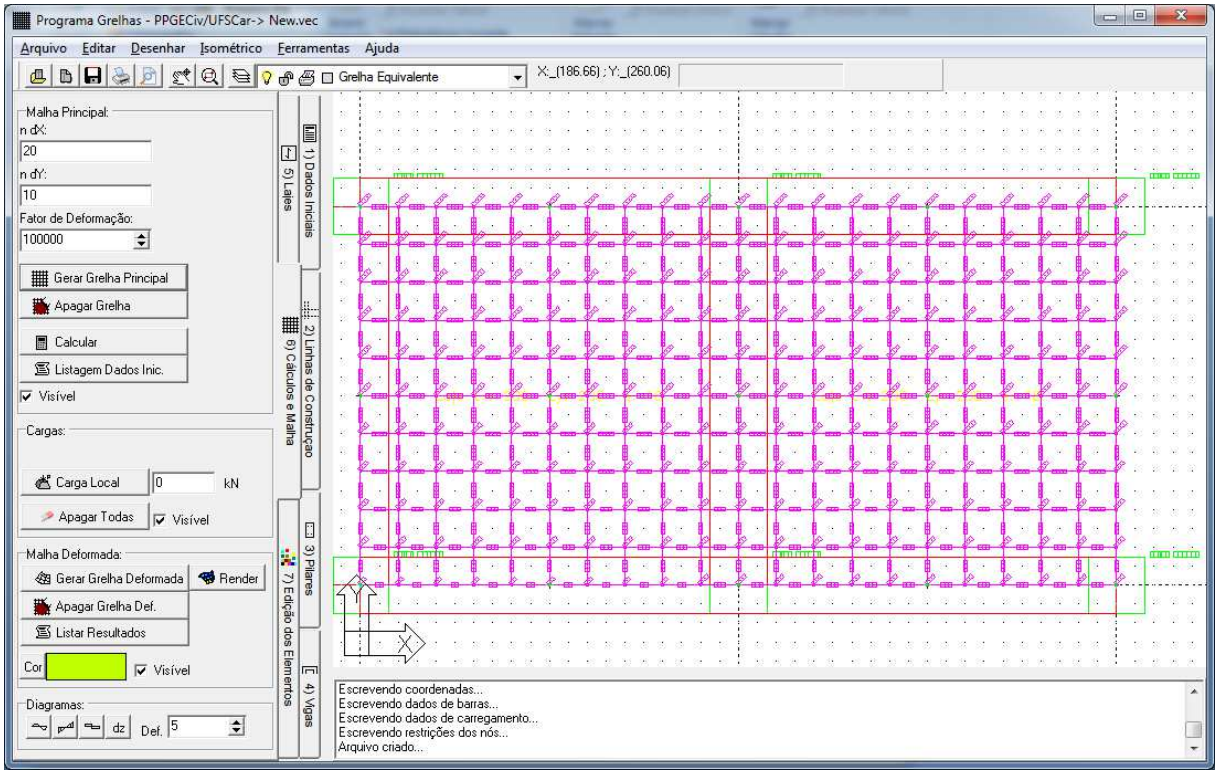


Figura 39: Estrutura com malha de 10x20 elementos criada.

$$C_l = \frac{C_d}{n-1} \tag{38}$$

Onde:

C_l: carga local equivalente;

C_d: Carga distribuída total;

n: número de nós selecionados.

No vão onde é aplicada a carga coloca-se a carga nodal equivalente, enquanto que nas extremidades coloca-se meia carga, ou seja, $Cl/2$, o que satisfaz a condição de zona de influência das cargas. Pretende-se implementar o processo automatizado da inserção de cargas distribuídas em versão posteriores do programa.

Na Figura 40 vêm-se algumas cargas de -2kN inseridas no projeto.

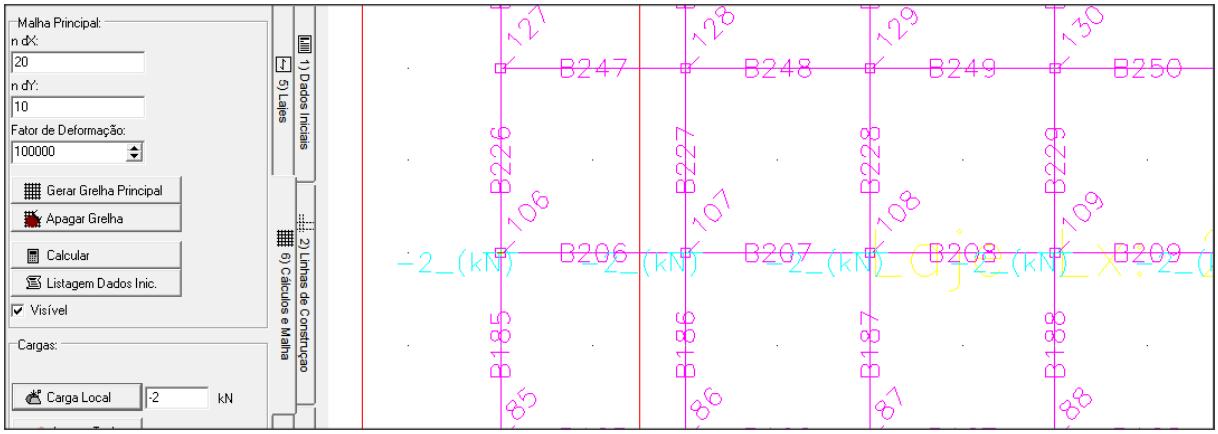


Figura 40: cargas de 2KN inseridas em alguns nós.

As cargas não são obrigatórias para o cálculo da malha equivalente deformada.

4.2.5 Geração da Malha Equivalente Deformada

Antes da geração da malha equivalente deformada deve-se definir o valor do 'fator de deformação' que é o fator de ampliação dos deslocamentos em 'dz' aplicado sobre a malha deformada. Valores muito baixos produzem uma malha deformada com deformações muito pequenas e pouco visíveis.

O passo seguinte é calcular a malha gerar a malha deformada apertando-se o botão 'Calcular', e após o término do cálculo, dever-se apertar o botão 'Gerar grelha deformada'. Na verdade pode-se apertar o botão 'Gerar grelha deformada' logo de início, pois o programa irá verificar se a estrutura já foi calculada pelo menos uma vez ou se algum elemento estrutural foi alterado, fazendo com que a estrutura necessite ser recalculada automaticamente.

Na Figura 41 vê-se a grelha deformada criada, representada na cor verde (pode-se alterar a cor da representação). Neste estágio fica pouco clara a deformação da

malha, pois ela está sendo representada a partir do plano superior da grelha. Para se evidenciar os deslocamentos é necessário produzir uma vista em perspectiva. Para tanto basta ir para o menu ‘Isométrico’ e escolher uma vista isométrica, como por exemplo a ‘Vista NE’. Pode-se ver a malha deformada renderizada ao apertar o botão ‘Render’. Será gerado um gráfico de superfície que representa a malha deformada como, por exemplo, a que se pode ver na Figura 43.

Na Figura 42 vê-se a mesma estrutura representada de forma isométrica. Podem-se observar as deformações bem definidas e podem-se, inclusive, observar os símbolos que representam as cargas nodais aplicadas com mais facilidade, como mostrado no detalhe da Figura 44.

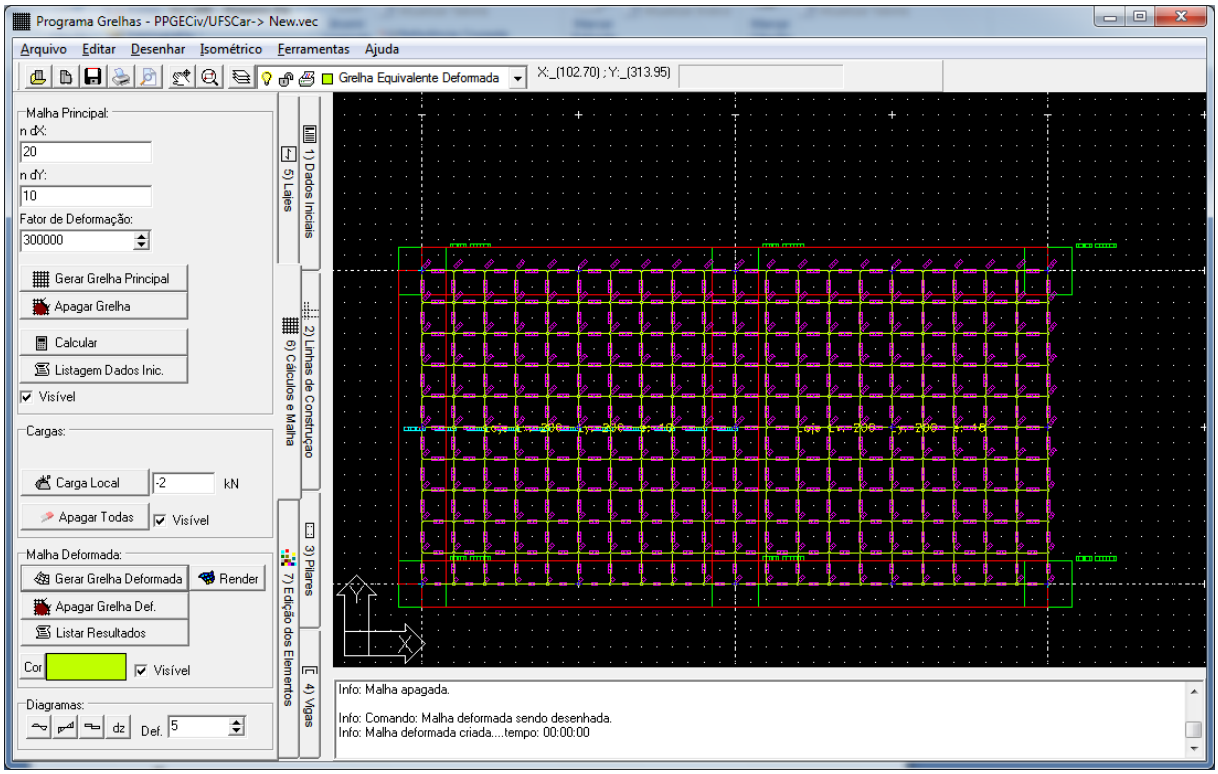


Figura 41: Malha deformada criada, identificada em verde.

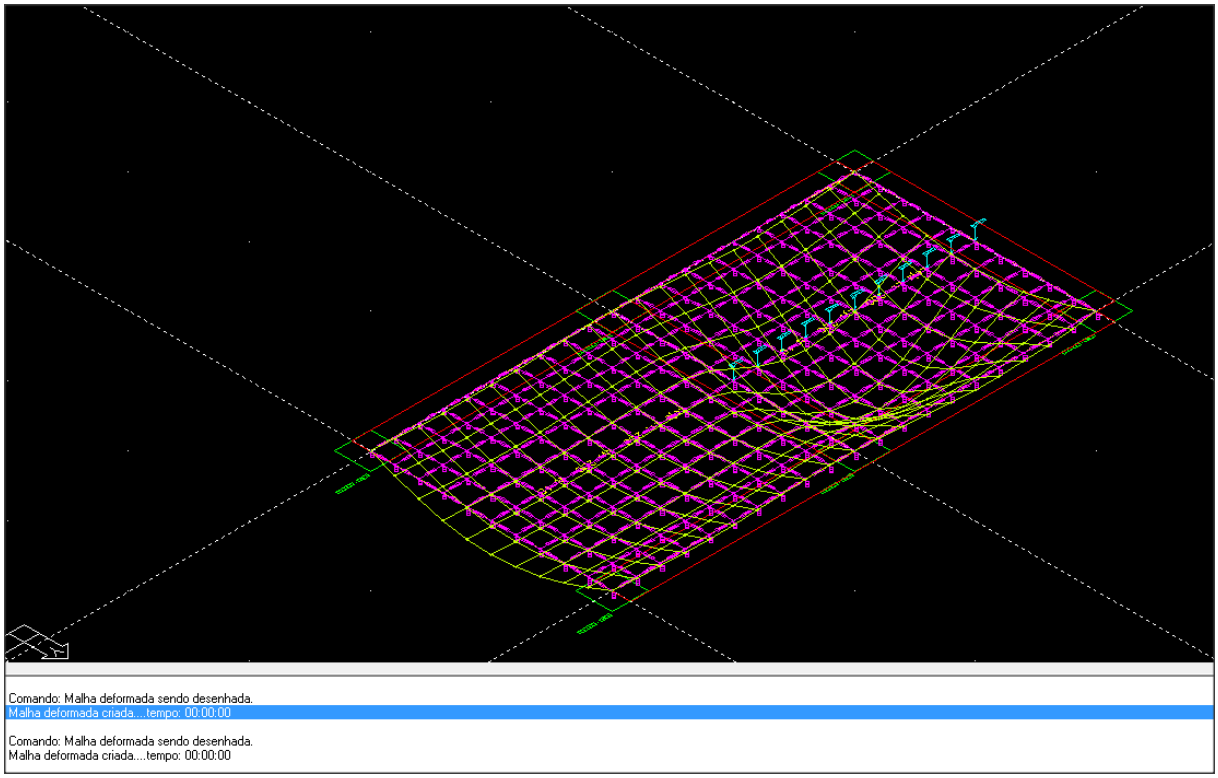


Figura 42: Malha deformada em vista isométrica NE.

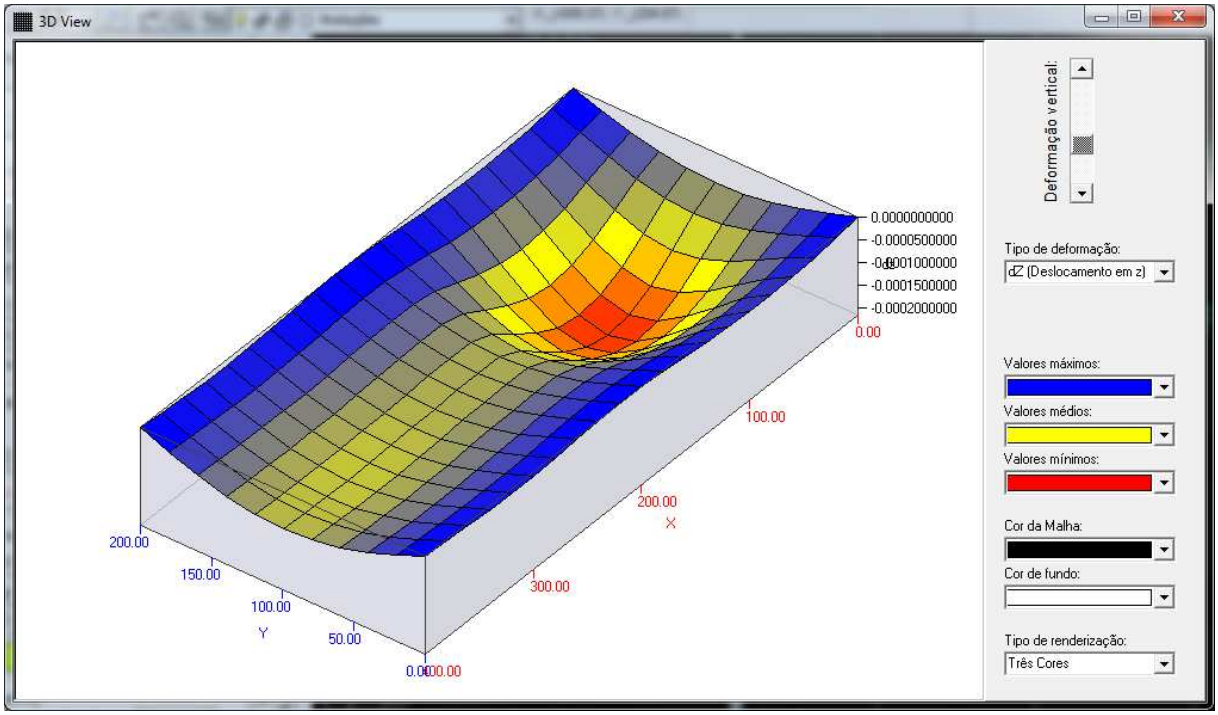


Figura 43: Malha deformada renderizada.

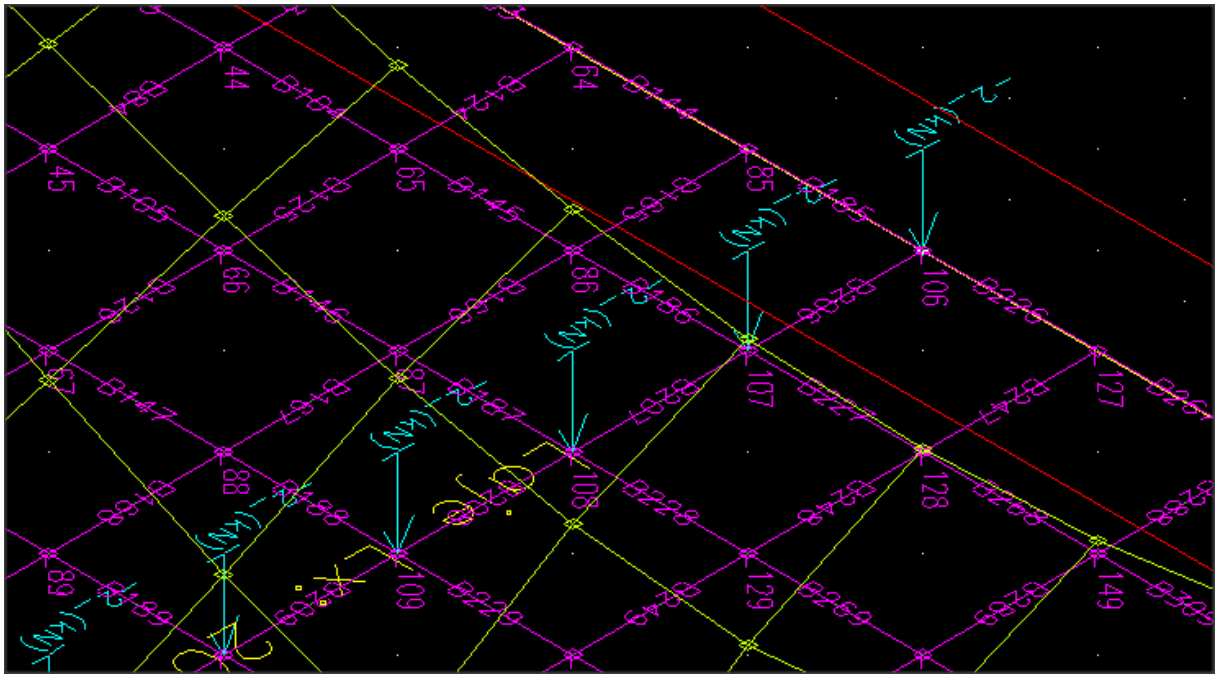


Figura 44: Detalhe da malha deformada com as cargas locais aplicadas.

As cargas nodais podem ser movidas de forma semelhante à que se faz nos programas CAD comuns, selecionando-se a carga e movendo-a para outro nó. Cargas inseridas fora dos nós serão eliminadas no processo de cálculo.

Após a movimentação das cargas basta apertar o botão ‘Gerar malha deformada’. Assim o programa irá automaticamente recalcular a malha deformada para as novas condições. A maioria dos elementos pode ser movida ou pode ter alterada seus dados característicos. Para tanto basta selecionar a aba ‘Edição de elementos’ e em seguida selecionar o elemento do qual se quer alterar as propriedades. Para maiores informações disponibilizar-se-á um manual específico para orientação e aprendizado do uso do sistema.

4.2.6 Geração de Diagramas Diversos

Uma vez terminado o cálculo da malha pode-se gerar alguns tipos de diagramas de esforços para a análise da estrutura. Estes são os diagramas de momento fletor, de momento torçor, de força cortante e de deslocamento.

Na aba ‘Cálculos e Malha’ pode-se ver o grupo ‘Diagramas’, como mostrado na Figura 46. Define-se primeiro o fator de deformação do diagrama e em seguida opta-se pelo diagrama que se quer criar, apertando-se o botão correspondente.

Figura 45: Grupo diagramas – Fletor, Cortante, Torços e Linha Elástica respectivamente.

Uma vez selecionado o diagrama selecionam-se os nós que se quer representar no diagrama (o usual é optar-se por um eixo completo). Em seguida o programa irá solicitar um ponto de inserção para o diagrama.

É importante durante este processo a vista ser recolocada como vista de topo (não é obrigatório, porém fica muito mais fácil a seleção dos nós).

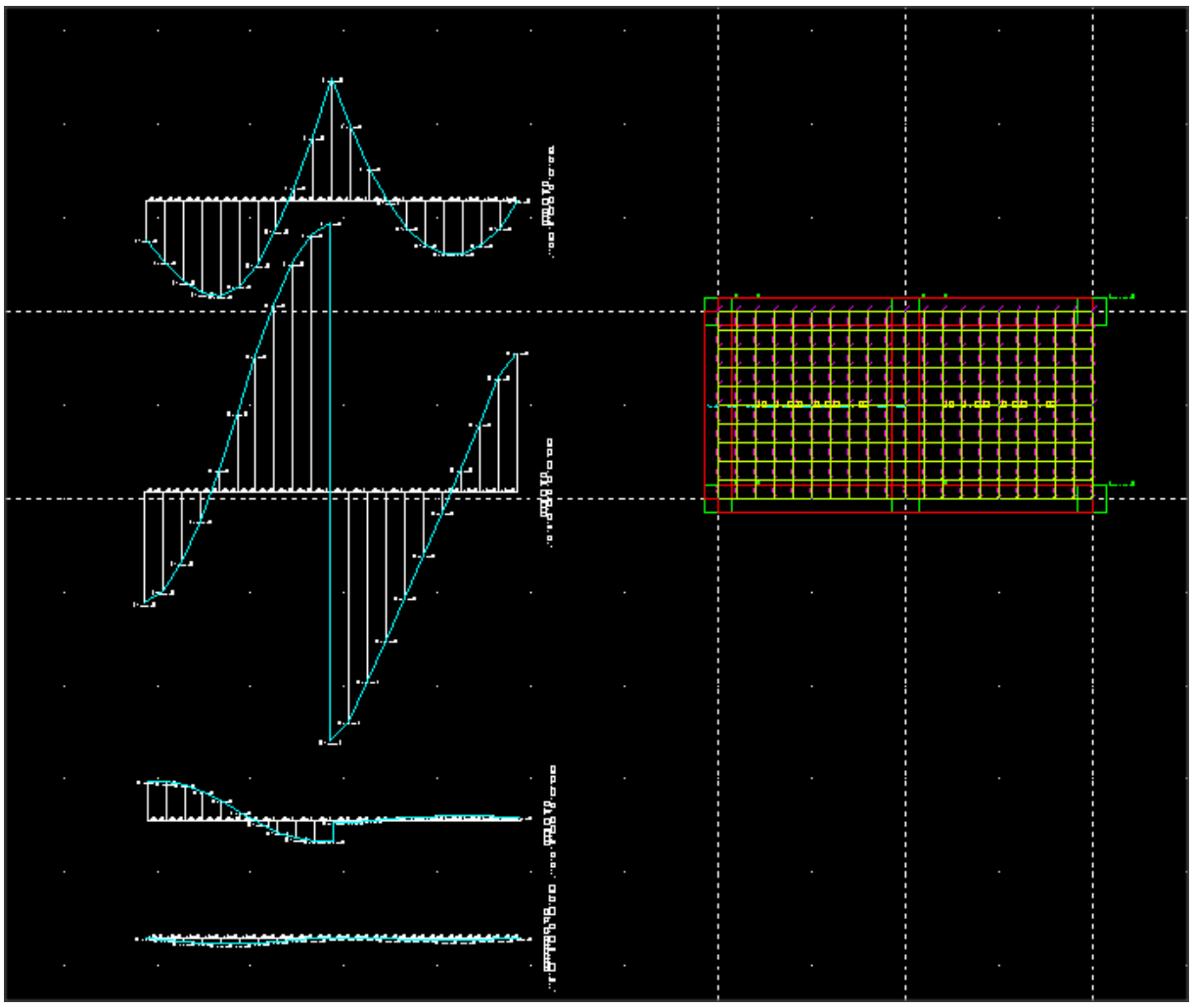


Figura 46: Diagramas de solicitações inseridos.

“O desenho final pode ser gravado ou exportado para formatos comuns como o “.dwg” ou “.dxf”. Um desenho exportado não poderá mais ser calculado, pois os

formatos “.dwg e .dxf” não contém os dados estendidos necessários para o cálculo da estrutura, são somente entidades gráficas. “Portanto vale lembrar que é importante sempre gravar, inicialmente, o projeto no seu formato nativo “.vec” primeiro. Como se verá mais adiante este tipo de arquivo assemelha-se aos arquivos de banco de dados que contêm informações de diversos tipos, que vão bem além das necessárias para o gerador gráfico comum.

4.3 FLUXOGRAMA GERAL DE CÁLCULO DOS ESFORÇOS

Pode-se dizer que basicamente qualquer programa de resolução de estruturas tem o fluxograma parecido com o da Figura 47, cujas etapas são descritas na sequência.

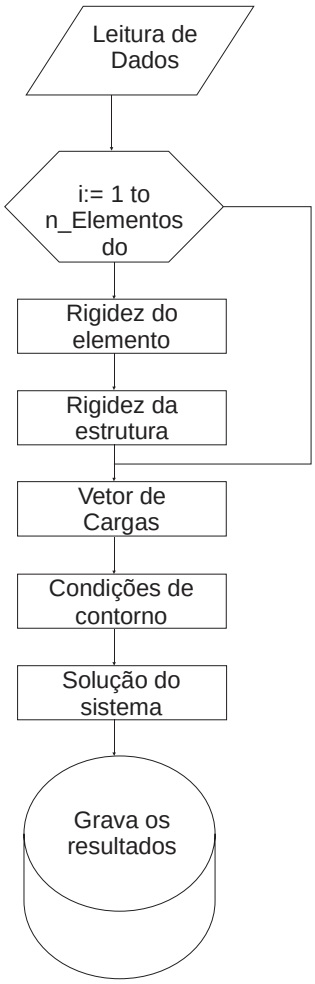


Figura 47: Fluxograma do módulo de cálculo de grelhas.

1. A leitura de dados consiste na caracterização da geometria da estrutura, dos seus elementos constituintes, restrições impostas pelos apoios existentes e da definição dos materiais utilizados. Esses dados são fornecidos pelo sistema gráfico de parametrização da estrutura.
2. Geração e inversão da matriz de rigidez. A matriz de rigidez é uma propriedade intrínseca à estrutura e está relacionada exclusivamente com suas condições de contorno, independendo das ações que a estão solicitando. A matriz de rigidez é obtida introduzindo-se vínculos fictícios à estrutura, impondo-se deslocamentos unitários correspondentes a esses vínculos. Uma vez que o usuário tenha informado os dados relacionados a estrutura, o programa é capaz de construir a matriz de rigidez.
3. Reunião dos dados de carregamento. Nesta fase, são informadas todas as cargas que estão solicitando a estrutura, sejam elas concentradas, distribuídas ou sob a forma de momentos. Uma vez lidos os carregamentos pelo programa, estes devem ser convertidos em ações nodais equivalentes.
4. Resolução dos sistemas de equações e consequente determinação dos deslocamentos.
5. A partir dos deslocamentos são determinados os esforços internos da estrutura (usando-se o método de Gauss-Jordan), bem como as reações de apoio.

4.4 ENTRADA E SAÍDA DE DADOS "*INPUT/OUTPUT*"

4.4.1 Arquivos de Dados do Módulo Gráfico

Como visto nos fluxogramas anteriores, vê-se que o programa de grelhas é dividido em dois módulos básicos: o primeiro que disponibiliza um ambiente gráfico de prototipagem de estruturas, e o segundo que processa os dados gerados pelo primeiro.

Como mencionado anteriormente, os arquivos do sistema gráfico de dados é muito semelhante ao de um banco de dados tradicional. O arquivo de projeto tem o formato binário onde são guardadas as informações de projeto com estrutura descrita na Figura 48.

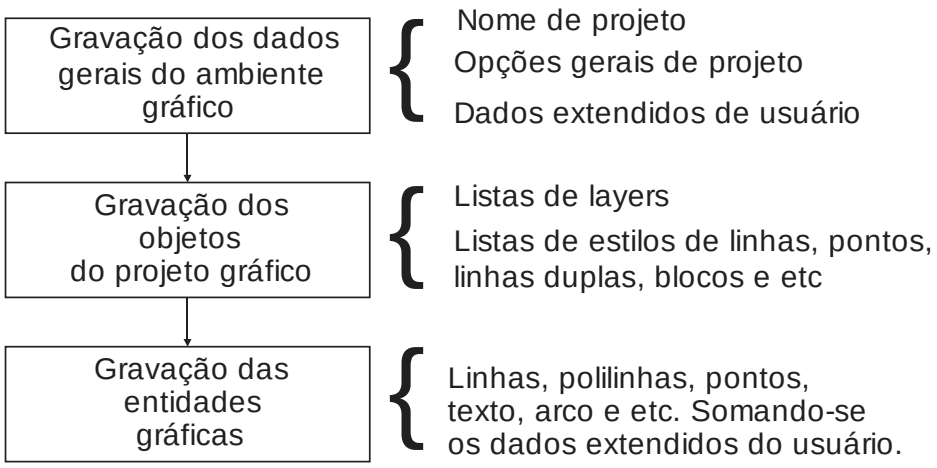


Figura 48: Conformação geral do arquivo de dados.

A estrutura de gravação no seu formato binário não será mostrada aqui, pois o formato em si não é relevante para esta dissertação, mas pode-se esclarecer que o formato adotado foi a estrutura dos arquivos de banco de dados do "*B-Tree*"[19] que é uma estrutura muito eficiente e muito utilizada pelos fabricantes de banco de dados para 'Delphi' e 'Lazarus'. Salienta-se também que existem vários "*softwares*" livres ou de domínio público disponíveis, como o "*Borland b-tree filer libraries*", Turbopower FlashFiler, TDBF, Tiny Database e o Firebird.

4.4.2 Arquivos de Dados do Sistema de Cálculo de Grelhas

O arquivo de entrada do módulo de cálculo é bem mais simples e optou-se pelo formato de texto. Esse formato não é muito eficiente, mas como a quantidade de

[19] O "B-Tree" (Binary Search Tree) é uma estrutura de dados que mantém os dados sequenciados e ordenados. Isso permite a realização de buscas, acesso sequencial, inserções e deleções numa faixa de tempo logarítmica. Esse formato é muito utilizado em sistemas que devem gravar blocos de dados relativamente grandes. Os arquivos completos do B-Tree para Pascal podem ser encontrados no site: http://sourceforge.net/p/tpbtreefiler/news/2005/12/-b-tree-filer-557a-released/

mais informações em: http://en.wikipedia.org/wiki/B-tree

dados é relativamente pequena, a velocidade de processamento não é prejudicada. A grande vantagem desse formato é a possibilidade de permitir que o usuário, eventualmente, possa aportar alterações aos dados de entrada antes do processamento. É um ponto muito interessante principalmente quando se quer utilizar o sistema para o ensino, já que dessa forma podem-se alterar alguns dados e ver o resultado dessas mudanças na tela gráfica.

Os dados são colocados no arquivo na forma de seções de dados, que são descritos a seguir:

1. Número de características de barras (número de tipos de barras diferentes);
2. Lista das características de barras;
3. Número de nós e de barras;
4. Coordenadas X e Y de casa nó;
5. Lista das barras com o número do nó inicial, número do nó final e tipo de característica de barra;
6. Lista das solicitações em cada nó, momento fletor, torçor e força vertical;
7. Lista das restrições nodais à flexão, torção e cortante.

Na Tabela 1 tem-se um exemplo de como o arquivo de entrada é configurado. Obviamente o arquivo não está completo, estando presente somente uma fração de cada seção do arquivo, pois o arquivo original é extenso.

Na sequência, na Figura 49, tem-se o fluxograma do processo.

Tabela 1: Exemplo de arquivo de entrada, unidades em (m e kN).

```
8
 3.20000000000000E+0007 1.00000000000000E-0004 1.28000000000000E+0007 2.25000000000000E-0004
 3.20000000000000E+0007 1.00000000000000E-0004 1.28000000000000E+0007 2.25000000000000E-0004
 3.20000000000000E+0007 1.00000000000000E-0004 1.28000000000000E+0007 2.25000000000000E-0004
 3.20000000000000E+0007 8.00000000000000E-0004 1.28000000000000E+0007 4.50000000000000E-0004
 3.20000000000000E+0007 2.88000000000000E-0005 1.28000000000000E+0007 5.76000000000000E-0005
 3.20000000000000E+0007 2.88000000000000E-0005 1.28000000000000E+0007 5.76000000000000E-0005
 3.20000000000000E+0007 1.44000000000000E-0005 1.28000000000000E+0007 2.88000000000000E-0005
 3.20000000000000E+0007 1.44000000000000E-0005 1.28000000000000E+0007 2.88000000000000E-0005
121 220
0.000 0.000
0.200 0.000
0.400 0.000
0.600 0.000
0.800 0.000
1.000 0.000
1.200 0.000
1.400 0.000
1.600 0.000
1.800 0.000
2.000 0.000
0.000 0.200
0.200 0.200
0.400 0.200.....
```

Número de tipos de materiais

Lista de materiais: Ec, I,Gc,It

Número de nós e barras

Coordenadas dos nós

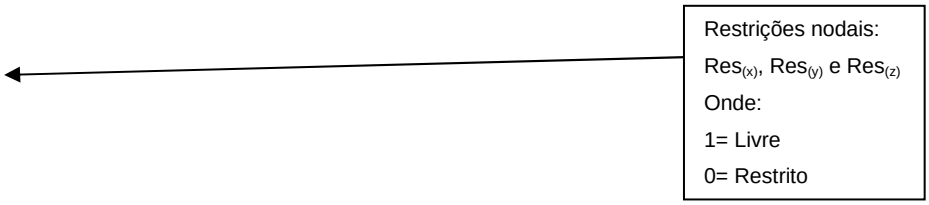
```
1 2 1
2 3 1
3 4 1
4 5 1
5 6 1
6 7 1
7 8 1
8 9 1
9 10 1
10 11 1
1 12 4
2 13 5
3 14 5
4 15 5
5 16 5
6 17 5
7 18 5
8 19 5
```

Dados das barras:
Nó inicial, nó final e
topo de material.

Cargas nodais:
$M_{(x)}$, $M_{(y)}$ e $F_{(z)}$

```
0.000 0.000 -0.285
0.000 0.000 -0.270
0.000 0.000 -0.270
0.000 0.000 -0.270
0.000 0.000 -0.270
0.000 0.000 -0.270
```

0.000 0.000 -0.270 0.000 0.000 -0.270 0.000 0.000 -0.270 0.000 0.000 -0.270 0.000 0.000 -0.210 0.000 0.000 -0.420
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1



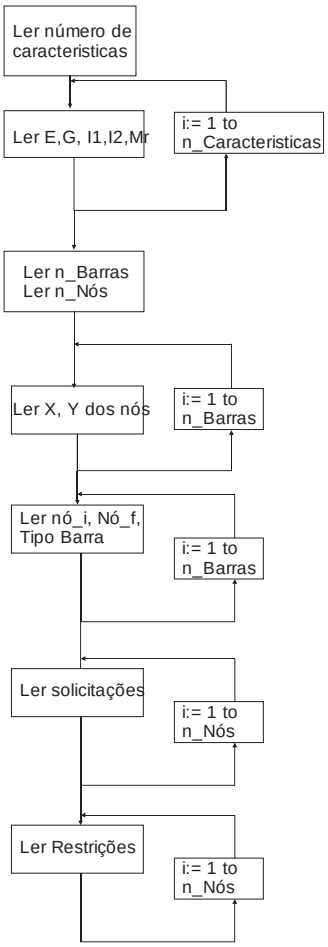


Figura 49: Fluxograma de leitura de dados.

4.4.3 Saída de Dados

Não faz sentido falar-se de entrada e saída de dados do sistema de prototipagem gráfica, pois este não é processado desta forma. O arquivo é mantido em tempo real, refletindo aquilo que se vê na tela, sendo alterado à medida que o projeto se desenvolve. Portanto falar-se-á neste parágrafo somente sobre o arquivo de saída do sistema de cálculo.

A plataforma de cálculo produz um arquivo de saída também no formato de texto, que é dividido em duas seções, como mostrado na Tabela 2, a primeira com os deslocamentos nodais (rotação em X, rotação em Y e deslocamento em Z), e a segunda com as solicitações nas barras (Força cortante, momento fletor e momento torçor). As solicitações são apresentadas aos pares, pois em cada barra existem

dois nós e as ações são impostas em cada nó. Por isso referem-se às solicitações nodais pelas ações à direita ou à esquerda do nó.

Tabela 2: Amostra do arquivo de saída do processador de cálculo (unidades em m, kN e kN.m).

****

Deslocamento nodal

Nó	Rotação em torno do eixo X	Rotação em torno do eixo Y	Deslocamentos na direção do eixo Z
1	0.0000000	0.0000000	0.0000000
2	-0.0000015	0.0000285	-0.0000033
3	0.0000004	0.0000361	-0.0000100
4	0.0000036	0.0000305	-0.0000169
5	0.0000063	0.0000170	-0.0000217

Reações de extremidade

Barra	Nó	Momento torçor (x)	Momento fletor (y)	Esforço cortante (z)
1	1	0.0219	-0.6577	2.0198
	2	-0.0219	0.2537	-2.0198
2	2	-0.0274	-0.2634	1.4133
	3	0.0274	-0.0192	-1.4133
3	3	-0.0465	-0.0016	0.9217
	4	0.0465	-0.1827	-0.9217
4	4	-0.0380	0.1629	0.5155
	5	0.0380	-0.2660	-0.5155
5	5	-0.0111	0.2547	0.1605
	6	0.0111	-0.2868	-0.1605
6	6	0.0230	0.2879	-0.1747

A partir dos dados apresentados na Tabela 2 podem-se associar esses resultados às entidades gráficas da malha equivalente. Desse momento em diante pode-se gerar a malha equivalente deformada e os demais gráficos de solicitações, como apresentado na Figura 46 do parágrafo 4.2.6.

5 SISTEMA GRÁFICO PARA A ANÁLISE DE ESTRUTURAS

5 SISTEMA GRÁFICO PARA A ANÁLISE DE ESTRUTURAS

5.1 INTRODUÇÃO AOS CONCEITOS DE COMPUTAÇÃO GRÁFICA

5.1.1 Generalidades e Aplicativos Gráficos

A computação gráfica pode ser entendida como sendo o conjunto de técnicas e metodologias para o tratamento e a representação gráfica de informações por meio da criação, armazenamento e manipulação de desenhos com o auxílio de computadores e periféricos gráficos. Em termos de aplicações, a CG (computação Gráfica) é dividida em várias áreas, entre elas:

1. CAD, desenho técnico por computador;
2. Apresentações gráficas;
3. Arte por computador, ou digital;
4. Entretenimento;
5. Educação e treinamento;
6. Visualização e tratamento científico de dados.

No caso do escopo desta dissertação, estende-se o tema nos itens um e seis, principalmente.

Um aplicativo gráfico é um programa ou um sistema composto por vários programas que permitem a geração de uma determinada apresentação gráfica, que pode ser composta de recursos 2D ou 3D (bidimensional ou tridimensional). Atualmente, os aplicativos gráficos possuem recursos de interação homem-máquina que permitem a composição e geração interativa de um projeto por meio gráfico.

Na implementação de um sistema gráfico, três parâmetros são importantes de consideração:

1. **Sistema operacional:** Implica-se a priorização do desenvolvimento do aplicativo para um determinado sistema operacional. De fato as GUI's[20] são sempre bem diferentes entre um sistema operacional e outro, o que faz com que dificilmente um aplicativo desses possa ser compilado sem grandes esforços de tradução entre um sistema e outro. Neste caso optou-se pelo Windows por ser o que mais oferece em termos de ferramentas de desenvolvimento.
2. **Biblioteca Gráfica:** A biblioteca gráfica contém as funções que o aplicativo aciona para a geração de primitivas gráficas. Um importante fator a ser planejado nas bibliotecas gráficas é a portabilidade do aplicativo em termos de ambiente computacional e de dispositivos ("*Device Independent Computing*"), ou seja, a biblioteca gráfica utilizada deve ser passível de execução na maior gama possível de computadores. Dessa forma garante-se a sua independência em relação aos dispositivos gráficos. Isso é possível através de "*Drivers*", que fazem o interfaceamento da biblioteca (GDI) com os dispositivos gráficos. Além da GDI existem várias interfaces gráficas muito poderosas, mas que exigem acesso via linguagem de máquina. Tem-se, por exemplo, as bibliotecas: Core, DCore, OPenGl, DirectX etc. Essas bibliotecas geralmente fazem parte da própria placa gráfica do computador ou do 'SO', o que permite o processamento de imagem em velocidades bem superiores ao normal.
3. **Sistema de janelas:** Os sistemas "*windowed*", a interface com o usuário por meio de janelas gráficas API "*Application Program Interface*" atualmente é um padrão mínimo como o Windows, Linux's, OS2 e Apple.

[20] "*Graphic User Interface*" ou Interface Gráfica ao Usuário é o sistema de janelas, caixas de texto e seleção para diálogos que fazem parte do sistema visual do Sistema Operacional (SO). GUI's fazem com que o sistema operacional fique bem mais amigável e acessível ao usuário comum.

5.1.2 Conceitos de Desenho Vetorial e Geometria Afins

A palavra CAD refere-se a "*Computer Aided Design*", que é o processo de se utilizar um computador para o auxílio ao projeto de forma gráfica, com rapidez, precisão, eficiência e qualidade. Eventualmente sistemas CAD podem estar associados a sistemas CAM - "*Computer Aided Manufactoring*".

Esses tipos de sistemas são baseados em transformações Afins[21], ou seja, em transformações de coordenadas da forma:

$$\overrightarrow{v'} = A.\vec{v} + \vec{b} \tag{39}$$

Ou:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{40}$$

No caso da equação (40) as coordenadas (x',y',z') do vetor ($\overrightarrow{v'}$), que definem um ponto no espaço, são uma função linear de (x,y,z) e a_{ij} é a matriz de transformação, enquanto que b_i são constantes determinadas pelo tipo de transformação. Basicamente as transformações afins têm como função de modificar a posição dos pontos. Rotação, translação, escalonamento, espelhamento e cisalhamento nada mais são do que deslocamentos de pontos (Bataiolla, 2006).

5.2 TRANSFORMAÇÕES

5.2.1 Translação

A translação é a alteração da posição de um ponto por meio da soma de constantes de deslocamento à coordenada inicial. A translação é sempre aplicada em todos os

[21] A geometria afim é a geometria que não está envolvida em qualquer noção de origem, extensão ou ângulo, mas sim em noções de subtração, adição, multiplicação e divisão de pontos, gerando vetores. É um espaço pensando, informalmente, como espaço onde se esqueceu do ponto de origem.

pontos que constituem a figura (a menos, claro, que se queira modificá-la), de modo a proporcionar a movimentação da figura no espaço.

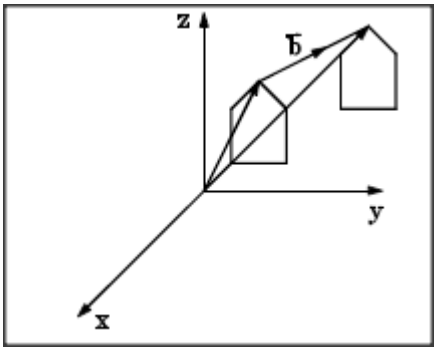


Figura 50: translação de uma figura (Bataiolla, 2006)

A translação é dada pela matriz de transformação da equação (41), onde as coordenadas transformadas (x',y',z') são o resultado da soma ($x+t_x$,$y+t_y$,$z+t_z$).
Note que a translação pode ser usada por uma função "*move*" que move realmente o objeto gráfico de posição no projeto, ou pode ser usada pela função de controle de tela "Pan", que move o objeto gráfico somente no "*Viewport*", ou seja, somente no monitor e não no projeto.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (41)$$

5.2.2 Escala

Conhecido como "*Scaling*", é a mudança de escala de uma figura por meio da multiplicação das coordenadas da figura por um escalar. Geralmente multiplicam-se todos os pontos pelo mesmo escalar, a menos que se queira mudar as proporções da figura. Isso acontece quando se quer representar uma figura numa perspectiva que requer escalas diferentes no plano x e no plano y. Isso acontece, por exemplo, com a perspectiva cavaleira. O escalonamento é utilizado tanto para a mudança de proporção de um objeto gráfico como também para os controles visuais do programa por meio das funções "*Zoom*", "*Zoom In*" e "*Zoom out*", que não são transformações no projeto, mas somente na tela do computador.

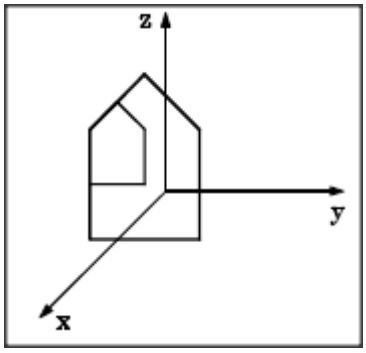


Figura 51: Escalonamento de uma figura.

A escala é representada pela equação (42).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (42)$$

5.2.3 Rotação

A rotação não é utilizada pelas funções de tela, mas somente para as funções de alteração de objetos no projeto. A rotação é definida através de um giro de um determinado ângulo em torno de outro ponto de referência, sem nenhuma alteração da distância entre eles. A rotação é aplicada em todos os pontos do objeto gráfico, proporcionando uma rotação homogênea. Para o cálculo da matriz de rotação, são consideradas inicialmente apenas duas coordenadas. Assim como mostrado na Figura 52, o ponto P de coordenadas (x,y,z) e ângulo inicial de β^0 é rotacionado em um ângulo de α entorno de P.

Matricialmente tem-se o que é mostrado a equação 43. Primeiro combinam-se as matrizes de transformação, e finalmente multiplica-se o vetor de coordenadas pela matriz de transformação e têm-se as coordenadas transformadas.

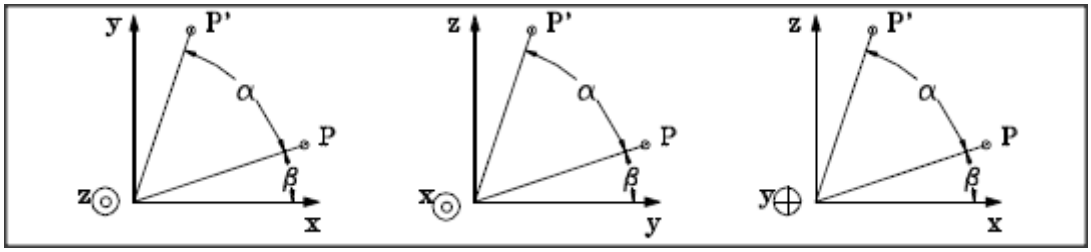


Figura 52: Rotação.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{RotX}} \underbrace{\begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{RotY}} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{RotZ}} \quad (43)$$

5.2.4 Espelhamento

O espelhamento ou reflexão, aplicado a um objeto produz um objeto com as coordenadas espelhadas. O espelhamento pode ser em torno dos eixos x, y ou z.
A equação de transformação é dada pela equação 44, que no caso, representa o espelhamento em torno do eixo x.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (44)$$

O espelhamento, de forma simplificada, é a inversão do sinal do componente das coordenadas pertencentes ao eixo que se quer espelhar.

5.3 TECNOLOGIA EMPREGADA

5.3.1 Introdução

Como visto nas justificativas desta dissertação, adotou-se o princípio de que a interface gráfica para a parametrização de dados se insere naturalmente na evolução dos sistemas de cálculo estrutural e oferecem, dessa forma, uma plataforma ou "*Front-End*"[22] muito mais amigável e precisa no processo de

[22] Em ciências da computação, "**front-end"** e "**back-end"** são termos generalizados que se referem ao processo inicial e final de um sistema. O "front-end" é responsável pela entrada de dados

prototipagem estrutural. Podem-se citar vários exemplos bem sucedidos de plataformas comerciais desenvolvidas para esse fim e que, atualmente, são consagradas no meio profissional da engenharia. Estas são: o Eberick, Cypecad, TQS, STRAP, SAP e várias outras.

O AutoCad, já mencionado, que foi inicialmente desenvolvido somente como uma ferramenta de desenho, historicamente acabou sendo muito utilizado como ferramenta de apoio para o processamento gráfico de dados, principalmente com aplicativos do tipo SIG (Sistemas de Informações Geográficas). No entanto sabe-se que o desenvolvimento de programas para AutoCad não é uma tarefa fácil já que não existe uma IDE[23] de qualidade para o desenvolvimento de aplicativos em LISP[24]. O LISP tem vários agravantes, pois é essencialmente uma linguagem interpretada (linguagens compiladas são muito mais rápidas), manipula somente dados de tipos simples e quase não tem ferramentas de acesso à leitura e interação de dados.

Fica claro que, para o caso deste trabalho, linguagens desse tipo têm pouca ou nenhuma utilidade, já que para o cálculo estrutural tratam-se matrizes, vetores, processos e funções com tipos complexos de dados. Além desses tipos complexos tratam-se listas com dados característicos dos elementos estruturais de tamanhos não suportados por linguagens comuns de programação (acesso direto à "*Heap*"). O acesso a essas informações requisitam uma linguagem capaz de lidar com arquivos binários complexos, organizados na forma de banco de dados indexados que podem ter grande extensão.

---

sob várias formas pelo usuário e processá-la para adequá-la a uma especificação em que o "back-end" possa utilizá-la. O "front-end" é uma espécie de interface entre o usuário e o "back-end".

[23] IDE, do inglês "*Integrated Development Environment*" -Ambiente Integrado de Desenvolvimento-, é um programa de computador que reúne ferramentas de apoio ao desenvolvimento de software que tem como objetivo o de agilizar o processo de desenvolvimento.

[24] Lisp, é uma família de linguagens de programação interpretadas, concebida por John McVarthy em 1958. O princípio da linguagem baseia-se na interpretação de dados por meio de listas.

Na mesma linha do AutoCad surge uma série de outros programas alternativos, como por exemplo os pertencentes ao grupo da ODA[25] como a BriksCad, ZWCad, NanoCad e outros que surgem por iniciativas individuais como o FreeCad e TCad. Esses programas seguem a trilha lógica básica do AutoCad, que é a de se desenvolver uma ferramenta de desenho e, portanto, não oferecem um ambiente efetivo de programação que permita a manipulação de grandes quantidades de dados de tipos variados. Vale salientar que a Autodesk vem investindo pesadamente em um sistema derivado do AutoCad, chamado de plataforma 'REVIT', que além de fornecer um ambiente gráfico poderoso, oferece a capacidade de se criar complexos bancos de dados de construção civil interconectáveis, compatíveis com as estruturas BIM[26] e programáveis via linguagem C#[27].

Diante desta conjuntura é que se encaixa este capítulo, pois se analisa a melhor forma de se criar uma ferramenta gráfica via API que seja capaz de se conectar a outros programas clientes. Faz-se isso seguindo alguns dos conceitos básicos de engenharia de software (Pressman, 2001) que normalizam a produção de plataformas computacionais de maneira que estas possam ser desenvolvidas de forma colaborativa, sem repetições de procedimentos. Espera-se assim que com maior organização poder-se-á obter uma base de bibliotecas muito mais sólidas e

[25] ODA "*Open Design Aliance*" é grupo formado por várias empresas no intuito de se criar uma biblioteca para a base de programas CAD semelhante a do Autodesk. As empresas participantes podem, por meio de uma taxa anual, ter acesso a essas bibliotecas para o desenvolvimento de programas. Além das taxas anuais é requerida uma taxa proporcional (Royalty) na venda de cada cópia de software que inclui essas bibliotecas.

[26] BIM ou "*Building Information Manager*", em português Modelador de Informação para a Construção, é um conjunto de informações geradas e mantidas durante todo o ciclo de vida de uma edificação. O BIM abrange geometria, relações espaciais, informações geográficas, quantitativos e propriedades construtivas de um modelo de edificação.

[27] C♯, também escrito como C# ou C Sharp, é uma linguagem de programação orientada a objetos, fortemente tipada, desenvolvida pela Microsoft como parte da plataforma .NET. A sua sintaxe orientada a objetos foi baseada no C++, mas inclui muitas influências de outras linguagens de programação, como Object Pascal e Java.

amplas que permitirão a rápida extensão dessa mesma base para a resolução de uma margem mais extensa de problemas típicos de engenharia.

5.3.2 Ambientes de Desenvolvimento

A escolha do Pascal como linguagem de desenvolvimento deste projeto, deve-se a um fator muito simples: è uma linguagem muito difundida e popular nos meios acadêmicos, principalmente nas áreas de engenharia. A linguagem foi portada para os ambientes de desenvolvimento “Lazarus free Pascal” e “Delphi”, o primeiro gratuito e o segundo comercial, que são IDEs muito eficientes e que permitem, para aqueles que estão iniciando a aprendizagem da linguagem, a obtenção de resultados muito rapidamente devido a curva de aprendizado típica muito acentuada. Além do mais é uma linguagem que sempre se beneficia de atualizações, proporcionadas por uma grande comunidade internacional, que transformou o Pascal, junto ao C, em uma linguagem poderosa e eficiente.

Atualmente estas plataformas oferecem a possibilidade de compilações multiplataforma, ou seja, programas que podem ser compilados para sistemas operacionais específicos como o Windows, Linux, Apple, Android e IOS.

O Lazarus permite, além da programação estruturada, orientada a objetos, classes e eventos, permite ainda que sejam desenvolvidos componentes reutilizáveis nos moldes propostos, inicialmente pelo Delphi da Borland, que uma vez instalados na IDE podem ser facilmente acessados e reutilizados no desenvolvimento de outros programas.

Como será visto mais adiante a POO (programação Orientada a Objetos) é fundamental para este projeto, pois permite que sejam criadas as classes de objetos que seguem os princípios de herança e polimorfismo. Propriedades que permitem

trabalhar de uma forma mais estruturada e organizada os objetos visuais, como as primitivas gráficas[28].

5.3.3 Definições e Paradigmas de POO

Um objeto em programação é uma entidade que tem um propósito bem definido, que tem a capacidade de receber e enviar informações. O objeto é estruturado por métodos e propriedades[29]. As propriedades são os valores das variáveis internas ao objeto (dentro do seu domínio, chamado de escopo), enquanto que os métodos são os procedimentos e funções internas a esse objeto. O programa cliente invoca uma função ou procedimento de um objeto servidor e este devolve ao cliente uma resposta ou serviço específico. Um objeto tem encapsulado em seu núcleo todos os métodos e propriedades necessários para a realização de uma tarefa (ou classe de tarefas) bem definida. O sistema cliente nunca terá que acessar ao administrar diretamente o conteúdo do objeto, somente os procedimentos e variáveis (métodos e propriedades) que fazem parte do seu escopo público[30]. Observa-se que uma mensagem pode endereçar diferentes procedimentos, dependendo da classe receptora. Por exemplo, a operação '+' (mais) numa linguagem orientada a objetos, pode se referir a:

1. Soma de dois números;
2. Concatenação entre duas cadeias de caracteres;
3. Soma de duas matrizes;

[28] Primitivas Gráficas são os elementos primordiais que formam uma figura. Por exemplo: ponto, segmento de reta, círculo, retângulo etc. Primitivas elementares podem dar origem, por meio de herança, a objetos gráficos mais complexos como a polilinha, blocos, grupos etc.

[29] Os métodos são semelhantes aos procedimentos e funções, enquanto que as propriedades são variáveis dentro do escopo do objeto. O Objeto fica com uma estrutura semelhante à de um programa comum.

[30] Na POO foram criados os conceitos de escopos Privado e Público. Os métodos e propriedades pertencentes ao escopo público são diretamente acessíveis pelo sistema cliente, enquanto que os privados são de acesso exclusivo do objeto.

4. Soma de uma matriz e um vetor;
5. Adição de elementos de duas listas.

Esta característica é chamada de polimorfismo ou sobrecarga de métodos, onde um determinado método é iniciado em função dos argumentos oferecidos. Além dessas características tem-se o conceito de herança que permite a especialização da informação ao longo de uma linhagem hierárquica de classes, onde aquelas situadas em níveis mais baixos herdam características (propriedades e métodos) das situadas mais acima. Assim é possível estender, ou reutilizar, classes em várias aplicações diferentes, bastando programar as variáveis e métodos necessários para a complementação das novas especializações desejadas.

Pode-se oferecer como exemplo dessa estrutura a formação das primitivas gráficas utilizadas no programa de grelhas, fruto deste trabalho. A estrutura é mostrada na Figura 53, onde se tem uma classe inicial ‘TCustom_Graphic’, que tem as propriedades de coordenadas X, Y, um atributo de cor, um atributo de tipo de linha e um método que usa esses dados para desenhar uma primitiva.

O método ‘Desenha’ de ‘TCustom_Graphic’ ainda é virtual, ou seja, é criado o método mas que ainda não é definido, pois neste nível de herança ainda não se desenha nenhuma primitiva. A definição do método será instanciada nos descendentes deste objeto.

De ‘TCustom_Graphic’ descendem três novos objetos: Ponto, Linha, e Círculo. Estes três novos objetos são descendentes de ‘TCustom_Graphic’, portanto herdam as propriedades X, Y, atributo de cor e atributo de linha e finalmente podem ter implementados seus métodos ‘Desenha’, que são particulares a cada descendente. Finalmente, descendendo de linha, tem-se um objeto chamado polilinha, que não somente herda todos os métodos e propriedades de ‘TCustom_Graphic’, mas também os de Linha, ou seja, o objeto pode usar a capacidade do objeto linha de produzir uma linha para gerar um conjunto de linhas na forma de uma polilinha.

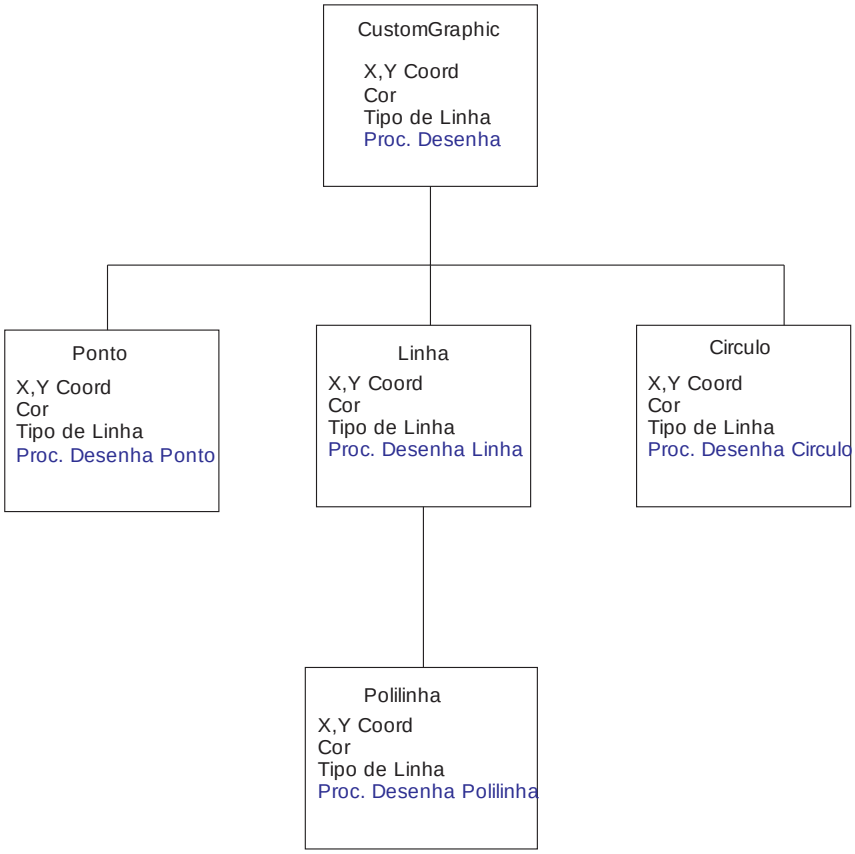


Figura 53: Exemplo de herança e polimorfismo.

Em termos práticos, o que se tem é um objeto com características bem definidas, cujas propriedades são transferidas para as classes descendentes sem haver a necessidade de programação de um novo código para as propriedades e métodos que são compartilhados por toda a linhagem desses objetos. Tais estruturas ajudam a manter o código bem estruturado e organizado.

Este paradigma é de fundamental importância quando se pretende desenvolver sistemas de forma colaborativa. A indústria do software adotou esta filosofia em grande escala, oferecendo no mercado, inúmeros componentes e objetos pré-compilados para serem vendidos como módulos facilmente incorporáveis aos sistemas clientes. No meio acadêmico pode-se encontrar vários exemplos de adaptação bem sucedida da POO na programação de sistemas de cálculo estrutural por elementos finitos (Devloo, et al., 1991) (Drummond, 2012).

Para exemplificar pode-se conceituar um projeto CAD que tenha a necessidade de um componente para a exportação de arquivos gráficos nativos para o formato DXF. Um componente desse tipo pode ser adquirido ou desenvolvido separadamente do

sistema cliente principal. Na verdade, basicamente, o componente só precisa que lhe seja fornecido o nome do arquivo a ser criado e a lista de entidades gráficas para que seja gerado o arquivo de exportação no formato DXF. O objeto, por sua vez, ao terminar sua tarefa, sinaliza ao programa cliente se o processo foi bem sucedido ou não, como visto no mapa conceitual da Figura 54. Na forma computacional, tem-se o fluxograma da Figura 55.

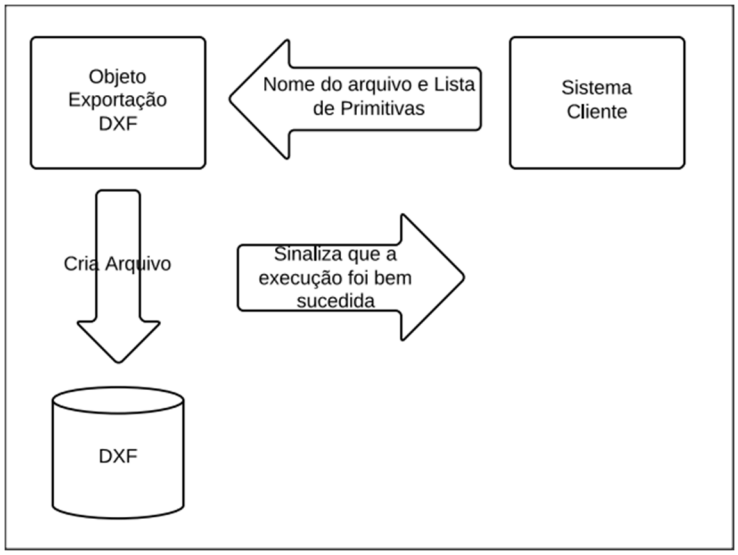


Figura 54: Diagrama de interação entre processos.

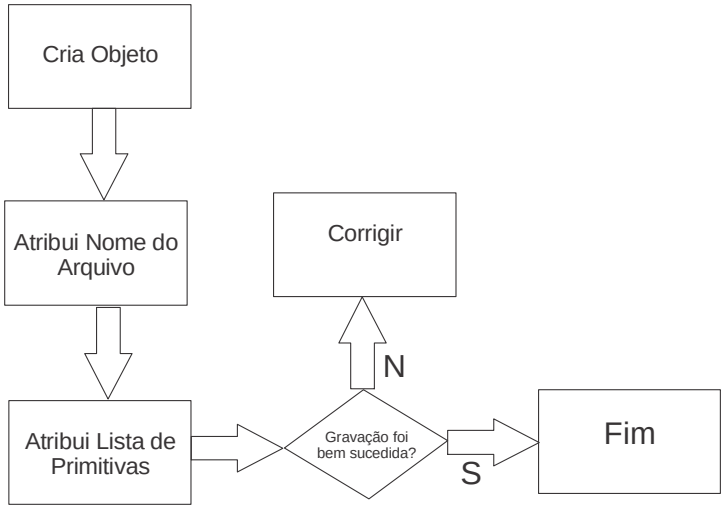


Figura 55: Fluxograma de interação.

Percebe-se que, a forma como o arquivo é gravado ou criado é irrelevante para o sistema principal. No exemplo da Figura 55, o objeto retorna o booleano '*True'* no caso do processo ter terminado corretamente ou '*False'* no caso de um insucesso.

No programa deste trabalho o objeto retorna inclusive o código de erro ao sistema principal para que este possa tomar as medidas corretivas necessárias.

5.3.4 Conceitos de POO

Os conceitos básicos de POO são listados abaixo:

Classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos. Exemplo de classe: Os seres humanos.

Subclasse é uma nova classe que herda características de sua(s) classe(s) ancestral(is).

Objeto / instância de uma classe é um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Exemplo de objetos da classe Humanos: João, José, Maria.

Atributo são características de um objeto. Basicamente a estrutura de dados que vai representar a classe. Exemplos: Funcionário: nome, endereço, telefone, CPF; etc. Carro: nome, marca, ano, cor; Livro: autor, editora, ano. Por sua vez, os atributos possuem valores. Por exemplo, o atributo cor pode conter o valor azul. O conjunto de valores dos atributos de um determinado objeto é chamado de estado.

Métodos definem as habilidades dos objetos. ‘Rex’ é uma instância da classe Cachorro, portanto tem habilidade para latir, programada através do método (procedimento) latido. Um método em uma classe é apenas uma definição. A ação só ocorre quando o método é invocado através do objeto, no caso Rex->Latido. Dentro do programa, a utilização de um método deve afetar apenas o estado do objeto em particular. Todos os cachorros podem latir, mas você quer que apenas Rex dê o latido. Normalmente, uma classe possui diversos métodos, que no caso da classe Cachorro poderiam ser sentar, comer e morder.

Mensagem é uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento definido por sua classe.

Herança é o mecanismo pelo qual uma classe (subclasse) pode estender suas habilidades/propriedades para outra classe (superclasse), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos). Um exemplo de herança: Mamífero é superclasse de Humano. Ou seja, um Humano é um mamífero. Há herança múltipla quando uma subclasse possui mais de uma superclasse. Um exemplo de herança múltipla: Mamíferos e Terrestres ou Mamíferos e Aquáticos. Heranças múltiplas não são possíveis em Pascal.

Associação é o mecanismo pelo qual um objeto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de". Por exemplo: Um humano usa um telefone. A tecla "1" é parte de um telefone, que é um exemplo de acoplamento de objetos.

Encapsulamento consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente apenas os métodos que alteram estes estados se necessário for. Exemplo: não é necessário conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo uma interface mais amigável (os botões, o fone e os sinais de tom).

Abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.

Polimorfismo consiste em propriedades que a linguagem pode ter para tratar vários tipos de dados diferentes por meio de um único método (note que nem toda linguagem orientada a objeto tem implementados todos os tipos de polimorfismo).

Sobrecarga: duas funções/métodos com o mesmo nome, mas assinaturas diferentes.

Coerção numa linguagem existe quando esta faz conversões implicitamente (como por exemplo, atribuir um int a um float em C++, isto é aceito mesmo sendo tipos diferentes, pois a conversão é feita implicitamente). O processo de coerção é raro no Pascal padrão, pois é uma linguagem que justamente pretende manter um controle rígido de tipos, no entanto no sistema Lazarus-Pascal vários tipos de coerção estão presentes.

Interface: é uma assinatura entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface. Qualquer diferença entre a interface e a classe pode provocar um erro de sistema. O Pascal não permite diferenças entre a interface e a implementação de classe.

5.4 INTERFACE GRÁFICA

5.4.1 Considerações Sobre *'Framework'* e '*API*' do Programa de Grelhas

No caso do programa de Grelhas, tem-se uma estrutura operacional bem mais complexa do que se discutiu até agora. A API gráfica do programa incorporou uma grande quantidade de objetos e classes diferentes, que têm uma relação bem definida entre si.

Para que esses objetos fossem individualizados corretamente foi, primeiramente, necessário definir a amplitude mínima do sistema, que deveria ter um "*Framework*"[31] composto por:

1. Capacidade de edição e/ou criação de um modelo gráfico que permita a geração de novas primitivas gráficas como as linhas, círculos, elipses, retângulos etc.;
2. Capacidade de organizar essas primitivas em camadas "*Layers*" e blocos[32];
3. Capacidade de edição das primitivas por meio de métodos de cópia, colagem, translação e deleção;

[31] Um "*Framework*" (ou arcabouço), em programação, é uma abstração que une códigos comuns entre vários projetos de "*software*", provendo uma funcionalidade genérica. Um "*Framework*" pode atingir uma funcionalidade específica, por configuração ou por projeto de classes e objetos. O "*Framework*" é que dita o fluxo de controle da aplicação por meio de um conjunto de classes que colaboram para realizar uma determinada tarefa para um domínio ou um subsistema da aplicação.

[32] Blocos são entidades que reúnem um determinado número de primitivas num grupo. Este grupo é sempre referenciado de forma única, como se fosse uma única entidade.

4. Capacidade de visualização interativa na tela com a programação dos métodos se "*Zoom*" e "*Pan*"[33];
5. Capacidade de visualização tridimensional dos elementos gráficos, com a programação de rotinas de gerenciamento de perspectivas isométricas;
6. Capacidade de incorporar ferramentas de auxílio ao desenho como "*Snap*", o gerenciamento e configuração de grades de apoio e modos ortométricos;
7. Capacidade de gravação e leitura de arquivos de dados num formato nativo ao sistema principal que preserve todas as propriedades das entidades gráficas parametrizadas no projeto, preservando assim os dados estendidos acoplados às entidades gráficas, como, por exemplo, as propriedades geométricas das barras, as propriedades do material que as compõem ou qualquer outro dado que for relevante ao projeto;
8. Capacidade de exportação do projeto em formatos compatíveis com outros sistemas CAD, como por exemplo, o DXF, SVG ou DWG.

No entanto, como mencionado anteriormente neste trabalho, para que a ferramenta aqui apresentada seja mais do que uma simples ferramenta de desenho vetorial, deve-se expandir a capacidade das primitivas, ou objetos gráficos, de forma que estas possam interpretar dados mais complexos externos à própria entidade gráfica. Explicando melhor, deve-se reservar um espaço de memória em cada objeto para que este possa carregar dados extras de projeto[34]. Como exemplo pode-se idealizar um programa de cálculo estrutural qualquer que deva permitir que as entidades gráficas (as barras e nós) devam carregar dados extras (além dos dados de representação gráfica), como as propriedades geométricas da seção das barras (CG, I, Ii, Ec, Gc, Nó Inicial, nó final etc). Dessa forma o projeto gráfico constitui-se numa estrutura semelhante a de um banco de dados cujas informações têm uma

[33] "*Zoom*" e "*Pan*" são funções de visualização interativa para facilitar a visualização global ou parcial de uma imagem. Esses métodos não afetam as entidades gráficas em si.

[34] Outra forma de acoplamento de dados ao objeto é reservar uma variável do tipo pointer no objeto que apontará para o endereço dos dados estendidos na memória principal. Na verdade este foi o método utilizado no programa desta dissertação.

representação visual (gráfica) e virtual de dados discretos acoplados às entidades gráficas.

Segundo o mapa conceitual da Figura 56, primeiro foi necessário criar-se um ambiente gráfico que foi complementado com objetos de edição de entidades gráficas, visualização de tela, leitura e gravação de dados.

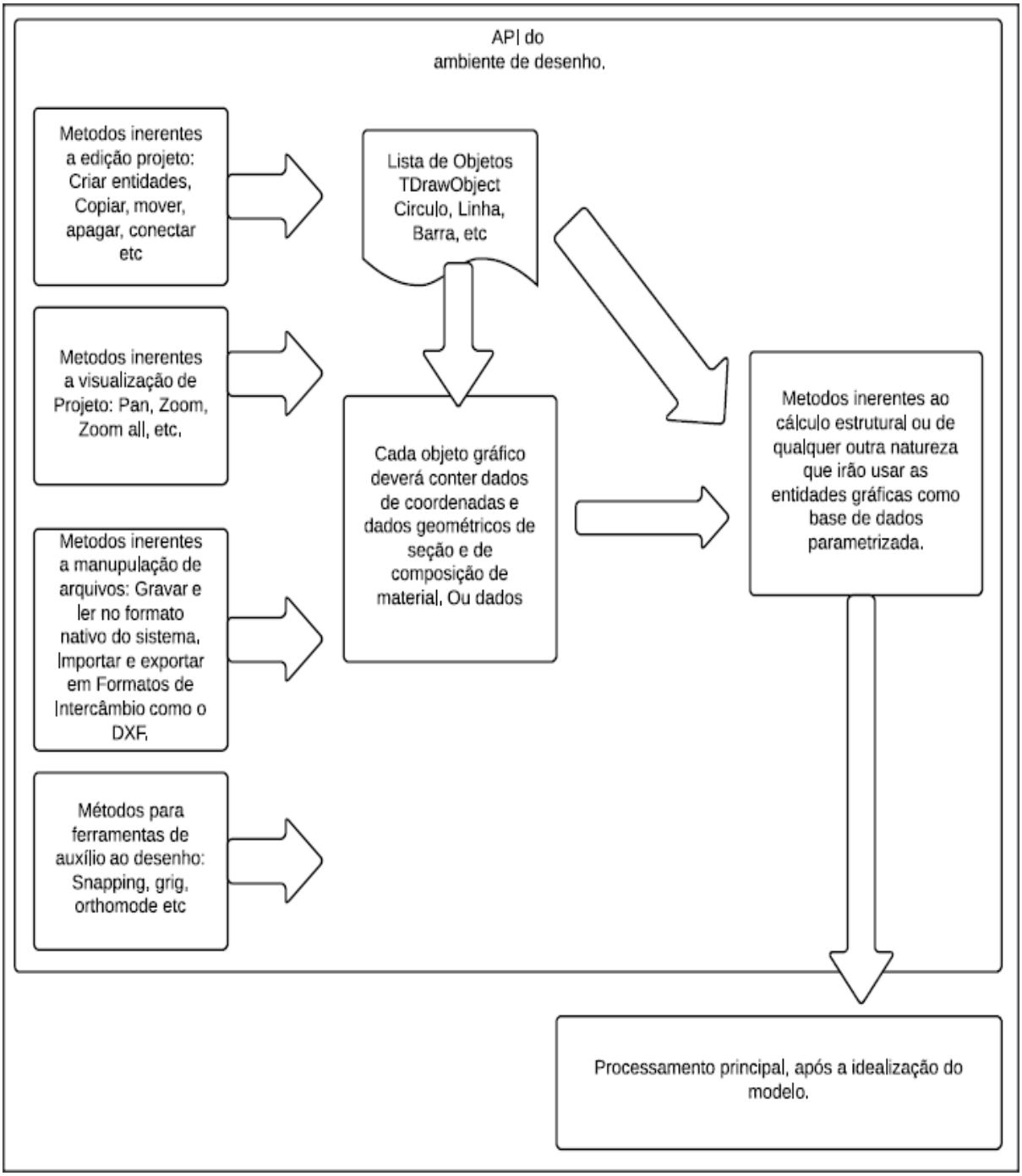


Figura 56: Mapa conceitual do "*Framework*" da "*API*" do programa Grelhas.

O ambiente, por comodidade, foi derivado de uma classe pré-existente no 'Windows', que é a classe 'Canvas'. A classe canvas aninha uma grande quantidade de métodos de manipulação de objetos gráficos, tanto para impressoras como para os monitores[35]. Nas IDE's do Lazarus e Delphi encontra-se a classe PaintBox, que na verdade é uma subclasse de ImageBox, que é uma subclasse de Canvas. No programa desta dissertação usou-se a classe PaintBox, que fornece os métodos básicos para a geração de primitivas gráficas e diferentes ferramentas para o tratamento gráfico de imagens.

5.4.2 Interação dos Elementos Gráficos com a Tela

5.4.2.1 Domínios de Tela e de Projeto

O desenho de entidades gráficas sobre uma tela é só o início do desenvolvimento de um sistema gráfico vetorizado. Primeiramente o ambiente deve ser coerente (homogêneo). O que está sendo representado na tela nada mais é que uma parte do que, na verdade, são as primitivas gráficas. De fato a telas têm um limite de representação e esse limite, de forma alguma, pode alterar a qualidade das informações contidas nas entidades representadas.

Deve-se entender que o ambiente deve preservar as coordenadas originais universais dos objetos gráficos, ou seja, mesmo quando há mudanças de visualização de tela (como ocorre quando se aplica um "*Zoom*" ou um "*Pan*"), as coordenas reais das entidades devem permanecer inalteradas. Mesmo que o objeto

[35] Impressoras e monitores, no Windows, são tratados como objetos semelhantes, ou seja, com as mesmas habilidades. A diferença primordial reside na quantidade de DPI ("*Dot Per Inch*" - pontos por polegada) entre uma classe e outra. De fato as impressoras trabalham com um número de DPI's (na casa dos 1700 dpi) bem superior aos dos monitores (por volta de 75 dpi).

tenha sofrido alterações na "*Viewport*"[36] ou tenha sido recortado por um "*Clipping*"[37]. Estes conceitos implicam que devam existir dois tipos de planos gráficos (ou sistemas de coordenadas), o de visualização e o de modelagem gráfica, sendo o primeiro útil para a interação do sistema como usuário, o segundo, necessário para a interação dos elementos gráficos do sistema.

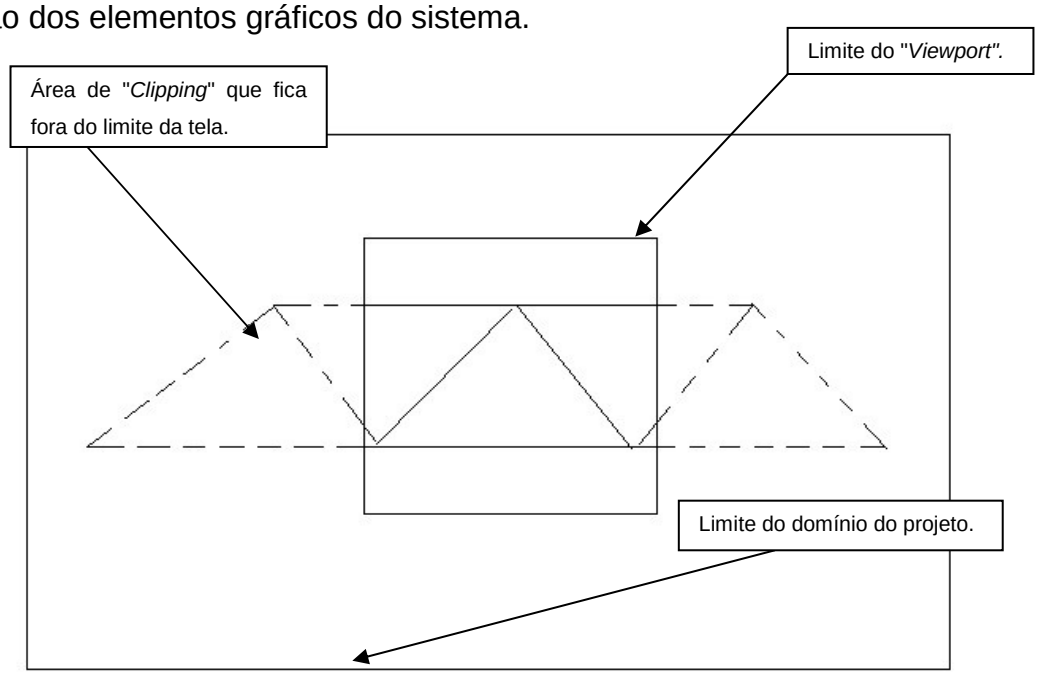


Figura 57: Representação de um "*Viewport*" e "*Clipping*".

Pode-se facilmente perceber, na Figura 57, a área de corte é a diferença entre o domínio de projeto e o "*Viewport*".

Este é um conceito básico e essencial para sistemas deste tipo. Assim os projetos desenvolvidos no programa de Grelha têm elementos estruturais com coordenadas dentro de um sistema origem fixo pertencentes a um domínio **IR** que é sempre coerente e imutável.

---

[36] "*Viewport*" é uma região virtual retangular em 2D na qual é representada totalmente ou parcialmente uma imagem. "*Viewports*" geralmente são definidos por: Coordenadas (iniciais, finais e de origem), área útil e por escala de representação.

[37] "*Clipping*", que significa corte, numa "*Viewport*", é a parte de uma imagem que, por não caber na área útil da tela, é cortada e fica parcialmente invisível.

Entende-se melhor este conceito quando colocado de forma de um exemplo numérico. Quando se aplica um "*PAN*" numa imagem soma-se à base do desenho o vetor de deslocamento desejado (geralmente indicado pelo "*Mouse*"). A base (origem) do desenho é deslocada de (x_1,y_1,z_1) para $(x_2,y_2,z_2) = (x_1,y_1,z_1)+(d_x,d_y,d_z)$. Esta translação opera-se no domínio do "*Viewport*" e não no domínio do projeto, o que em termos práticos é uma translação somente da vista sistema/usuário, e não sistema/elementos gráficos.

5.4.2.2 Precisão Gráfica

A manipulação de elementos vetoriais na tela implica em fatorações e divisões sucessivas que podem produzir erros de precisão nas coordenadas, principalmente ao se trabalhar com escalas muito grandes ou muito pequenas (próximas de zero). Logo nas versões iniciais do programa deste trabalho deparou-se com os problemas típicos dessa ordem como o reproduzido na Figura 58, onde a linha e o círculo deveriam ter um ponto em comum, que na figura fica dentro do retângulo pontilhado de "*zoom*".

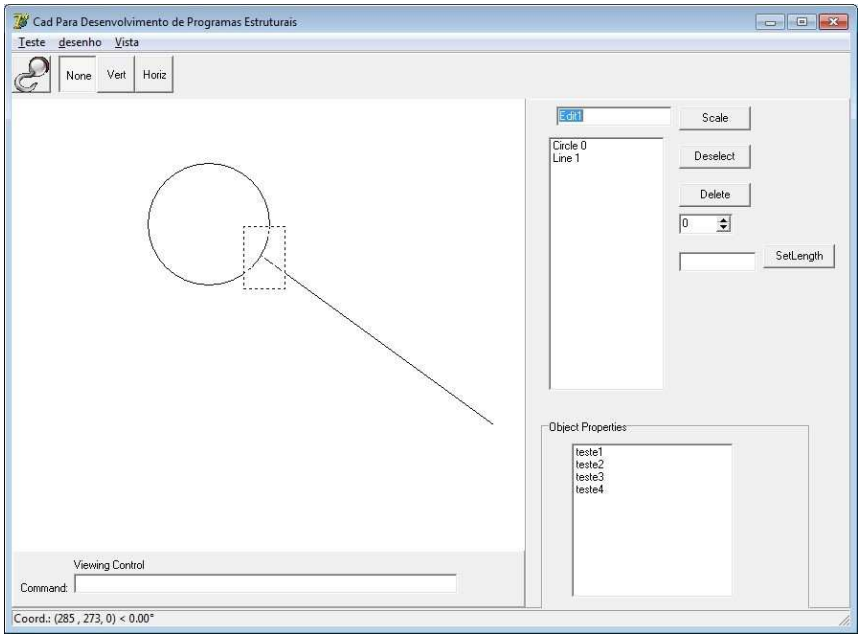


Figura 58: Círculo e linha com um ponto colinear.

Após a aplicação do "*Zoom*" (Figura 59), observa-se que a coordenada na extremidade da linha não confere com a linha do círculo. O erro ocorre devido à imprecisão na conversão de coordenadas de projeto para as coordenadas de tela.

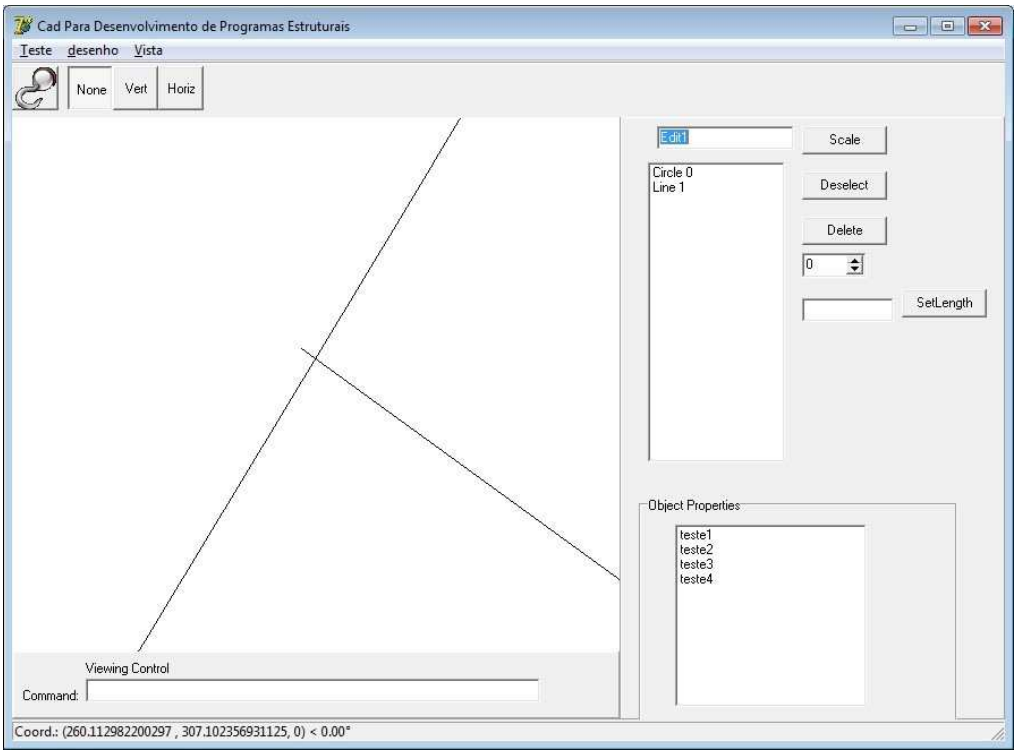


Figura 59: Vista ampliada do ponto de interseção entre a linha e o círculo.

Os erros gráficos mais comuns são os:

1. De proliferação, onde operações geométricas com um número de 'k' dígitos de precisão que podem gerar resultados que necessitem de precisão ainda maior;
2. De irracionalidade, onde algumas operações resultam em números sem precisão finita, como por exemplo $\frac{\pi}{3}$ ou $\sqrt{3}$.

Segundo Hoffmann (Hoffman, 2001), os "*softwares*" de cálculo geométrico ou que dependem de uma interface gráfica, ocasionalmente, tendem a ser frágeis e falhos. Este problema de robustez deve-se essencialmente à dificuldade em se tomar decisões inequívocas sobre a incidência ou não incidência, fundamentalmente

prejudicando a confiabilidade do sistema geométrico, o que é antagônico ao próprio princípio almejado pelo ambiente gráfico que gera um grande número de situações.

No intuito de inibir este tipo de erro, adotam-se escalas de desenho que sejam adequadas ao tipo de projeto gráfico do sistema cliente. Por exemplo, para programas de cálculo estrutural onde a maior parte das medidas é em metros, pode-se eleger como escala básica o centímetro que dará a precisão necessária para os cálculos sem gerar números grandes demais ou números fracionários muito pequenos. No fator de escala de 'um' adotou-se para cada (pixel) um (mm), ou seja, a cada cem píxeis tem-se um metro no projeto.

As coordenadas reais das entidades gráficas são mantidas em registradores específicos e nunca são alterados, claro, a menos que sejam aplicadas mudanças reais nesses objetos por meio de edição direta do projetista. As entidades são geradas na tela usando-se essas coordenadas reais como base, que são convertidas em coordenadas de tela[38]. As coordenadas de tela, por terem um erro intrínseco, são descartadas e recalculadas toda vez que as entidades são redesenhadas na tela.

Em escalas muito baixas, que produzem uma relação de píxeis por unidade de medida menor que um ($\frac{Pix}{d} < 1$), ou quando uma determinada coordenada encontra-se entre dois píxeis, não é mais possível, ao clicar na tela, obter-se uma coordenada precisa de inserção. Nessas situações, deve-se recorrer a métodos de eliminação de imprecisões (Wilczynska, 2010). De fato a eliminação de incertezas pode ser aplicada por meio de métodos baseados em lógica nebulosa[39], que permitem a determinação de um ponto provável na tela.

---

[38] É interessante notar aqui que as coordenadas reais podem ser fracionárias enquanto que as coordenadas de tela, em píxeis, são coordenadas inteiras. Fica portanto intrínseco que a conversão de um sistema para outro produz um erro de arredondamento.

[39] A lógica nebulosa é um método matemático complexo que permite solucionar problemas com soluções pouco óbvias ou com pouca precisão, principalmente por terem um grande número de entradas e saídas de informações. Essa lógica oferece um resultado baseado numa recomendação

A precisão de coordenadas é um tópico de primeira importância no caso da parametrização de dados gráficos para o cálculo estrutural. Erros de conexões nodais no programa podem facilmente gerar um sistema de equações indeterminado ou sem solução (ou deslocamentos infinitos).
Por isso, ferramentas como as linhas de construção, de aproximação magnética ("*Snapping*") ou a grade são essenciais no momento de conectar as entidades entre si.

5.4.3 Definição das Primitivas Gráficas

As primitivas gráficas são objetos que seguem os paradigmas da POO, que neste caso descendem da classe 'Object', que é uma classe básica tanto no Delphi como no Lazarus. No diagrama da Figura 60 tenta-se ilustrar simplificadamente essa classe.

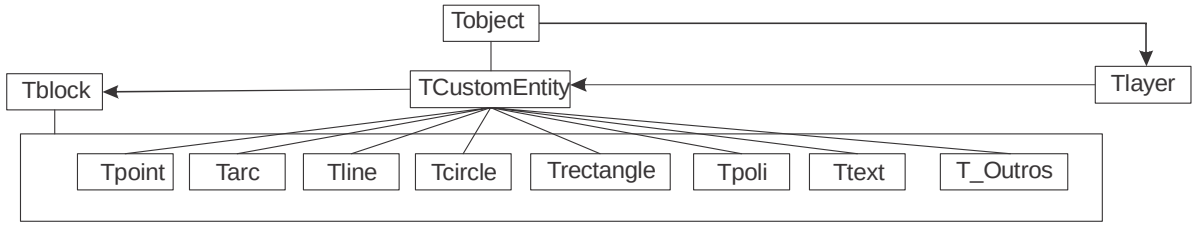


Figura 60: Relação de descendência entre os objetos gráficos.

Algumas das propriedades e métodos do objeto 'CustomEntity' são relacionados na Tabela 3. Note que na verdade o objeto tem um número bem maior se propriedades e métodos.

Tabela 3: Algumas das propriedades e métodos de uma entidade gráfica.

Métodos ou Atributos	Comentários
Function GetShape	Retorna o tipo de entidade.
Procedure SetShape	Define o tipo de entidade que ira representar o objeto.
Function GetSelected	Retorna a lista de entidades selecionadas

para um determinado intervalo (centro de massa de informações), portanto é essencialmente diferente das lógicas tradicionais como a booleana.

Procedure SetSelected	Define uma entidade como selecionada
Procedure Move	Translada as entidades selecionadas até uma determinada coordenada
Procedure Rotate	Gira as entidades selecionadas e 'x' graus.
Procedure Deleted	Marca a entidade como estando deletada.
Coords	Lista de coordenadas da entidade
fPen	Tipos de pena que usada para desenhar a entidade
fVisible	Define se a entidade é visível ou não (usado nos layers)
fLineType	Define o tipo de linha da entidade
fExtendedData	Dados estendidos do usuário
fExtendedDataSize	Tamanho em bytes dos dados do usuário
fLayerId	Define o layer pai da entidade
fScale	Define a escala de representação da entidade
fKey	Define uma chave de identificação da entidade
fHandle	É o número único de identificação da entidade
...	...

As primitivas gráficas são armazenadas em "*layers*", ou seja, camadas, que são listas (objeto TStringList). Cada uma dessas listas contém os objetos que devem ser administrados por elas. Por exemplo, para que todas as entidades de um projeto sejam desenhadas, deve-se proceder como mostrado no código da Tabela 4.

Tabela 4: Código simplificado do processo de desenho de todas as entidades.

```
DrawBoard.Clear;
For i:= 0 to Layers.Count-1 do begin
 For j:= 0 to Layers[i].EntityCount-1 do begin
    TDrawObject(Layers[i].Entities[j]).Draw;
 End;
End;
```

5.5 REPRESENTAÇÃO GRÁFICA DOS ELEMENTOS ESTRUTURAIS DO PROGRAMA GRELHA

5.5.1 Linhas de Construção

As linhas de construção são entidades discretas (não têm espessura ou escala), que são definidas por horizontalidade ou verticalidade e mais uma coordenada de inserção. Essas linhas não têm início nem fim. Elas são desenhadas horizontalmente ou verticalmente através todo o "*Viewport*" atravessando o ponto de inserção. No

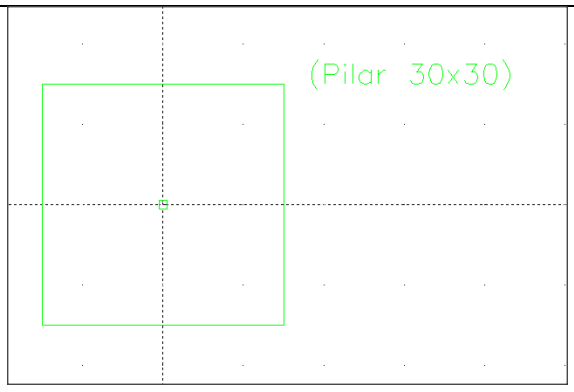
caso do programa Grelhas, a linhas verticais são definidas por um deslocamento horizontal, enquanto que as horizontais por deslocamentos verticais. Essas linhas são necessárias para que sejam evidenciadas as coordenadas de inserção dos elementos estruturais. Esta foi a forma mais eficaz encontrada para evitarem-se erros de conexão entre esses elementos.

A função "*Snap*" tem a habilidade de encontrar os pontos notáveis entre essas linhas, como por exemplo, os cruzamentos.

5.5.2 Pilares

Os pilares Tabela 5 são entidades gráficas compostas, na forma de blocos, que contêm um quadrilátero, uma caixa de texto e um ponto (com estilo pré-definido). O ponto de inserção do bloco é definido pela coordenada do ponto no bloco. Durante a colocação dos pilares é importante procurar colocá-los em coordenadas notáveis definidas pelas linhas de construção. Os nós da malha equivalente que estiverem dentro dos limites da área dos pilares serão considerados como pontos indeslocáveis em 'z', e tendo as restrições definidas no momento da criação do pilar.

Tabela 5: Definições da entidade gráfica pilar.

Tipo de dado estendido associado	Representação gráfica
PPilarData = ^TPilarData; TPilarData = record Lado_a: Double; Lado_b: Double; h: Double; xC,yC,zC,x1,y1,z1,x2,y2,z2: Double; bz,bx,by: Boolean; Nome: string50; end;	

5.5.3 Vigas

As vigas Tabela 6 são entidades definidas por uma linha tripla, fechada nas extremidades. A distância entre as linhas laterais é a mesma definida para a largura da viga. As vigas devem ser inseridas no projeto respeitando os eixos definidos

pelas linhas de construção. A malha equivalente deverá, necessariamente, ter barras e nós que passem pelos eixos das vigas.

Tabela 6: Definições da entidade gráfica viga.

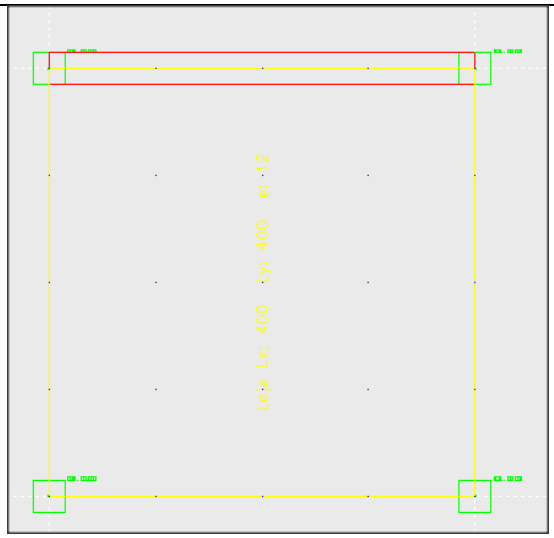
Tipo de dado estendido associado	Representação gráfica

PVigadados = ^TVigaDados;
TVigaDados = record
Lado_a, Lado_b: Double; //Lados da viga
Compr: Double; //Comprimento
xC,yC,zC,x1,y1,z1,x2,y2,z2: Double;
I, It, CargaAdic,Peso: Double;
Nome: string[50
Orient: TOrientation;
hId: THandle;
end;
	

5.5.4 Lajes

A laje Tabela 7 é uma entidade composta, na forma de um bloco, constituída por um quadrilátero e uma caixa de texto. As lajes definem um espaço com propriedades predefinidas. As malhas são geradas de forma a englobarem todas as lajes.

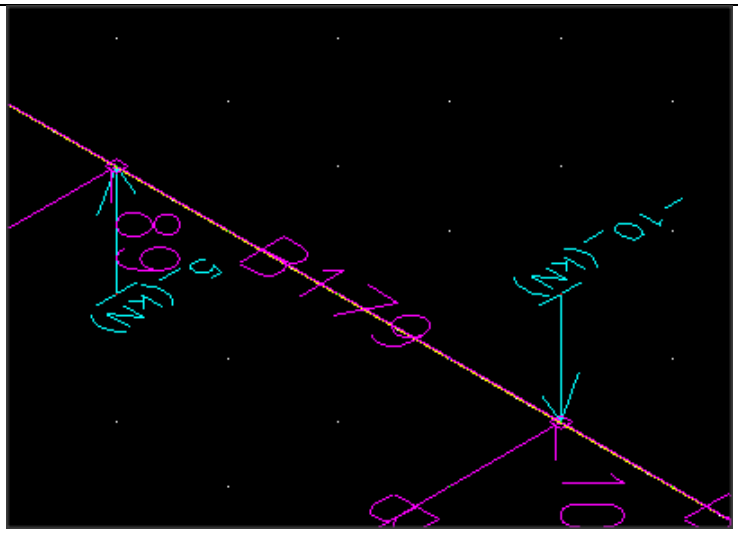
Tabela 7: Definições da entidade gráfica laje.

Tipo de dado estendido associado	Representação gráfica
PLajedados = ^TLajeDados; TLajeDados = record Lado_a, Lado_b: Double; nAcid, nPerm: Real e: Double; I1, I2, It1, It2: Double; LPeso, LPesoAcid, LPesoPerm: Real; xC,yC,zC,x1,y1,z1,x2,y2,z2, p: Double; Nome: string[50]; LajeId: THandle;	

5.5.5 Carga Nodal

A carga Tabela 8 nodal é uma entidade gráfica composta, na forma de um bloco, composta por linhas, uma caixa de texto e um ponto. O sentido (rotação) é determinado pelo sentido da carga. As cargas devem sem inseridas obrigatoriamente sobre um nó. Cargas deslocadas fora dos nós são automaticamente removidas no processo de cálculo e do ambiente gráfico.

Tabela 8: Entidade gráfica da carga local.

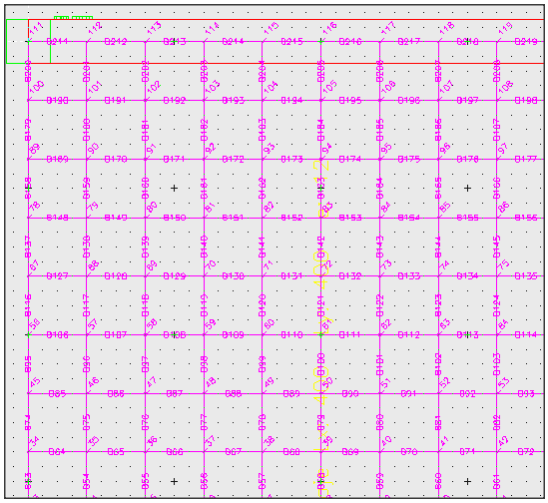
Tipo de dado estendido associado	Representação gráfica
PNodalLoad = ^TNodalLoad; TNodalLoad = record LoadValue: Real; x,y,z: Double; Nome: string[50]; end;	

5.5.6 Malhas Equivalente e Equivalente Deformada

A malha equivalente Tabela 9 é um conjunto de entidades constituídas por barras e nós. A malha equivalente é criada de forma a ocupar todo o espaço das lajes. Se houver mais de uma laje, a malha irá cobrir a área das lajes combinadas. O procedimento para a criação da malha é mostrado na Tabela 28 – Anexo 1.

Tabela 9: Entidades de barras e nós que geram a malha equivalente

Tipo de dado estendido associado	Representação gráfica

PBarData = ^TBardata;
TBarData = Record
P1,P2: TRCoord;
iN1, iN2, iBar : Word;
I, Load: Extended;
hN1, hN2, hB: THandle;
MT1, MT2, MF1, MF2, V1, V2: Double;
MatType: Word;
Orientation: TOrientation;
BarType: TBarType;
end;
 	
PNodeData = ^TNodeData;
TNodeData = record // Node data
//P1: TRCoord;
X,Y: Double;
N,iN1: Word;
mx,my,nz: Extended;
rx,ry,rz: Boolean;
Rotx, Roty, Deslz: Double;
pLaje, pAcid, pPerm, pViga, pParede, pLocal: Real;
hN1: THandle; //Node Identity Handle
Pos: TNodePosType;
end;
	

6 DESENVOLVIMENTO: ALGORITMOS E NUMÉRICO

6 DESENVOLVIMENTO: ALGORITMOS E NUMÉRICO

6.1 CONSIDERAÇÕES INICIAIS

O objetivo deste trabalho, essencialmente, é o 'lançamento' da estrutura, onde são representados de forma gráfica os diversos elementos estruturais que compõem o projeto. As lajes, vigas, pilares e elementos auxiliares são representados com suas características geométricas e condições de carregamento.

Considera-se que a estrutura tem múltiplos pavimentos e que mesmo especificando-se plantas e cortes, ela está ligada de um andar ao outro por meio de pilares. Uma possível análise dessa estrutura se faz associando-se sua fôrma a um modelo físico composto, normalmente, de uma malha de nós e barras. A geração desses elementos é condição primordial para o sucesso e a precisão da análise numérica que será feita posteriormente. Assim o grupo, ao qual se insere o autor deste trabalho, tem procurado desenvolver uma série de programas livres que suprem essa necessidade. Atualmente vários programas encontram-se disponíveis para o uso ou estão em desenvolvimento[40]. Com a defesa da dissertação de Cotta (Cotta, 2006)[41] pode-se dispor de uma ferramenta extremamente poderosa que permitirá resolver estruturas de concreto moldadas "*In Loco*" ou pré-moldadas. Outros programas ainda estão sendo desenvolvidos e poderão ser complementados pelos contidos neste trabalho.

Como mencionado anteriormente neste trabalho, algoritmos para o cálculo estrutural vêm sendo desenvolvidos há mais de trinta anos em algumas universidades, porém, em geral, de forma caótica, sem concatenação, de forma individual e não pública. Essa conjuntura dificulta qualquer projeto de continuidade e aprimoramento desses sistemas, ficando a necessidade de se programar novamente as mesmas rotinas toda vez que se inicia um projeto deste tipo.

---

[40] Para o CALCO: http://www.deciv.ufscar.br/calco/

[41] A dissertação é acessível pelo link:
http://www2.ufscar.br/interface_frames/index.php?link=http://www.bco.ufscar.br

O desenvolvimento de um programa que resolva os esforços solicitantes em estruturas como o de Cotta, por si só não são suficientes para ter-se um sistema completo. De fato o sistema deve ser integrado a um processador gráfico que permita a entrada de dados de forma rápida, precisa, segura e editável. Essa associação de sistemas permite o desenvolvimento de uma plataforma capaz de gerar dados e manipulá-los de forma mais didática. Observa-se também que essas plataformas devem ser desenvolvidas e adaptadas aos paradigmas modernos de programação, que permitam a manutenção dos códigos gerados de forma colaborativa e incremental, permitindo-se assim que com o tempo a plataforma adquira novas capacidades e estendam-se ainda mais seus limites.

6.2 ROTINAS

6.2.1 Rotinas do Sistema de Cálculo

6.2.1.1 Considerações Iniciais

O sistema de cálculo do programa Grelhas, como estava inicialmente, fora programado de forma bastante convencional. Essa forma de programação com variáveis globais estáticas produz um programa que enfrenta problemas de estouro de pilha "*Stack Overflow*". No Pascal as variáveis globais e locais são armazenadas na pilha de dados do programa, e como esta pilha tem um tamanho fixo alocado pelo compilador programas que usam matrizes, têm grandes chances de provocar um estouro de pilha. Programas de cálculo estrutural que trabalham naturalmente com matrizes de grandes dimensões (ou esparsas) requerem uma metodologia diferente: o uso de matrizes dinâmicas e listas de endereçamento por ponteiros. Dessa forma as variáveis ficam alocadas na área dinâmica da memória (a "*Heap*") e tem-se acesso a toda a memória instalada no computador. Este artifício permitiu que se calculassem grelhas de praticamente qualquer tamanho. Nos testes feitos pelo autor deste trabalho, grelhas com 4000 barras foram calculadas normalmente, ficando somente o inconveniente do tempo de processamento, que para uma grelha dessa dimensão ficou em aproximadamente cinquenta minutos.

Portanto, ao cronograma inicial deste projeto, incluiu-se naturalmente a reformulação da alocação de memória do programa de cálculo original, que estava limitado a grelhas de aproximadamente 300 barras e 1000 nós.

Alguns testes com sistemas geradores de matrizes dinâmicas (JEDI, 1997) de terceiros[42] e gratuitas foram aplicados com sucesso, obtendo-se matrizes de números de ponto flutuante maiores que 20.000 x 20.000 sem erros de memória. No entanto essas bibliotecas não foram utilizadas nesta fase do trabalho, já que se estenderia por demais o cronograma limite de finalização desta dissertação.

6.2.1.2 Modulação

A modulação geral ficou bem simples, ficando limitada a três etapas básicas. São elas:

1. **Leitura de dados:** leitura das listas das características dos materiais dos elementos, das características geométricas, das restrições de vínculo, carregamentos nodais (ver Tabela 1);
2. **Processamento:** geração das matrizes de rigidez local e global. Determinação dos deslocamentos e demais solicitações (M_f, M_t e V_z);
3. **Gravação de resultados:** geração da lista de solicitações nas barras e lista de solicitações nodais, como mostrado no exemplo da Tabela 2.

6.2.1.3 Tipos de Variáveis e Constantes Utilizados

Segundo Gere (Gere, et al., 1987), as variáveis utilizadas em um programa de cálculo estrutural podem ser classificados em:

---

[42] As bibliotecas JEDI "*Joint Endeavour of Delphi Inovators*" são oferecidas gratuitamente na internet. Essa bibliotecas estendem em muito as possibilidades de soluções dos programadores. Destaca-se a biblioteca JEDI Math que oferece um grande número de soluções para o cálculo numérico, estatístico, probabilístico, matricial e vetorial. Pode-se baixar essa biblioteca pelo link:

http://jedimath.sourceforge.net/

1. Dados de controle;
2. Dados de Estrutura;
3. Dados de carregamento.

Dessa forma as variáveis utilizadas foram agrupadas da forma como mostrado na Tabela 10.

Tabela 10: Classificação das variáveis utilizadas.

Conjunto de dados		Quantidade de conjuntos.	Variáveis relacionadas
Dados de controle		1	estrutura, arqmatriz,teste, carac_dif
Dados da estrutura	a. Parâmetros dos elementos estruturais	1	NUMENO, NUMEBAR, NOINICIAL, NOFINAL
	b. Coordenadas dos nós	NUMENO	X, Y
	c. Designação dos membros e propriedades.	NUMEBAR	Rig1, R, Rtrans, SMG, SMGdes,
	d. Lista da restrição dos nós.	3*NUMENO	XX
Dados do carregamento	a. Ações aplicadas nos nós	3*NUMENO	F, Q
	b. Ações nas extremidades de membros.	6*NUMEBAR	extrem, extrem1, extrem_nlinear
	c. Reações de vínculo.	3*NUMENO	reagir, reac
	d. Deslocamentos gerados.	3*NUMENO	Des, Destotal, Destotalnlin
	e. Análise não linear.	1	et

6.2.1.4 Leitura de Dados

Nesta etapa são lidos os dados pertinentes à estrutura. Para tanto se programou um procedimento **Ler_Do_Arquivo** (Tabela 21-Anexo 1) cujo propósito é o de ler os dados da estrutura a partir de um arquivo de texto, como definido na Tabela 1. Ressalta-se que os valores são separados por espaços.

6.2.1.5 Geração da Matriz de Rigidez

Concluída a leitura de dados, a partir do arquivo de integração o programa gera-se a matriz de rigidez de cada um dos elementos prismáticos que compõem a estrutura e, por fim, gera-se a matriz de rigidez global da estrutura (Tabela 22-Anexo 1). As

questões teóricas envolvidas já foram discutidas nos capítulos anteriores desta dissertação.

Programou-se para tanto um laço de forma a considerar a contribuição da rigidez de todos os elementos constituintes da estrutura. Neste laço estão as seguintes etapas:

1. Cálculo da matriz de rotação R do elemento (Tabela 24 - Anexo 1);
2. Cálculo da matriz de rotação transposta Rtrans (Tabela 24 - Anexo 1);
3. Cálculo da matriz de rigidez do elemento.

Multiplicação da matriz de rotação transposta pela matriz do elemento, resultando ma matriz Rig1.

Multiplicação da matriz Rig1 pela matriz de rotação do elemento, resultando na matriz de rigidez definitiva Rig.

6.2.1.6 Rigidez do Elemento

Esse procedimento (Tabela 23-Anexo 1) tem a função de gerar a matriz de rigidez de cada elemento prismático que compõe a estrutura. Para cada matriz de rigidez, os coeficientes encontrados são inseridos na matriz de rigidez global e em seguida é utilizada para a geração da matriz de rigidez do próximo elemento.

6.2.1.7 Solução das equações

Uma vez terminada a montagem da matriz de rigidez global, inicia-se o cálculo dos deslocamentos nodais, reações de vínculos e reações nas extremidades das barras. Para esses cálculos é necessários o uso de métodos numéricos para a resolução de sistemas de equações lineares. Qual o melhor método para tanto não faz parte do escopo desta dissertação, no entanto admitiu-se que o melhor método é o de solução direta, como o de eliminação de Gauss (Tabela 27 - Anexo 1). O fluxograma é mostrado na Figura 61. Este método transforma uma matriz numa matriz triangular superior que, em seguida, é resolvida facilmente. Fazendo a retrosubstituição obtêm-se a solução das demais variáveis.

No Anexo 2 apresenta-se o algoritmo, fluxograma e rotina padrão da eliminação de Gauss.

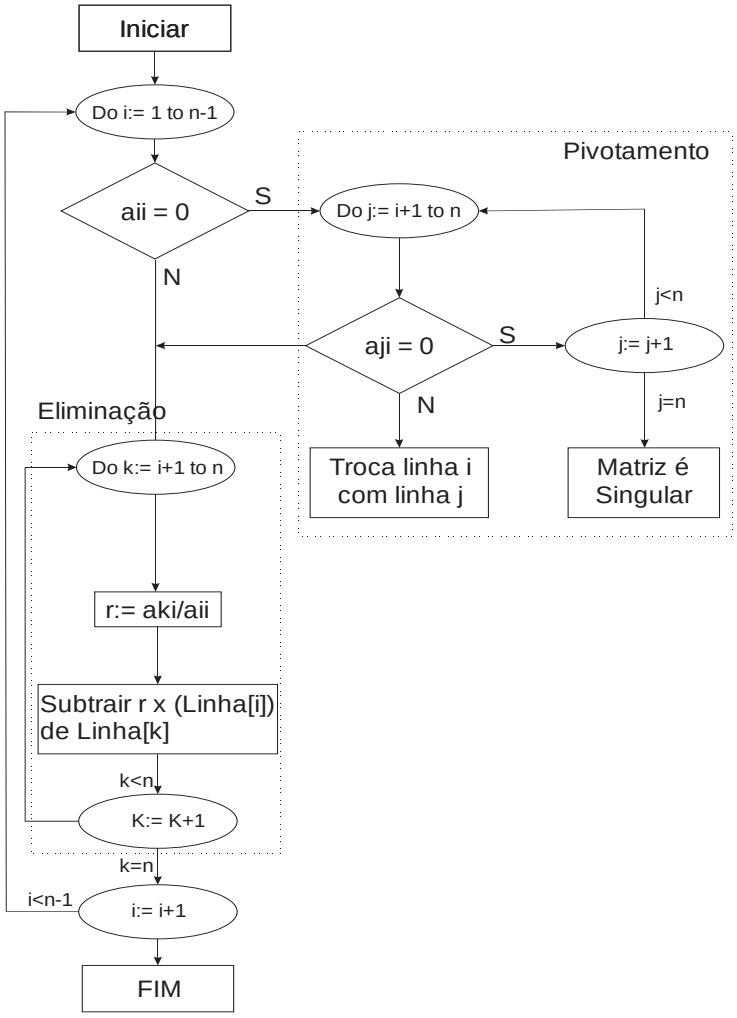


Figura 61: Fluxograma do método de eliminação de Gauss.

6.2.1.8 Procedimento Para o Cálculo dos Deslocamentos

Após a determinação dos deslocamentos gera-se o vetor de deslocamentos nodais totais, considerando-se as reações de apoio. Para tal finalidade executa-se um laço que percorre todos os nós da estrutura. Havendo uma restrição no nó, atribui-se o valor zero para o vetor de deslocamentos totais na posição correspondente. Não havendo restrição, atribui-se o valor contido no vetor de deslocamentos. Após o fim deste laço, o programa finalmente gera as saídas de dados no formato de arquivo de texto, como mostrado na Tabela 2.

6.2.2 Rotinas do Sistema Gráfico

6.2.2.1 Considerações Iniciais

Após a finalização do processo de cálculo tem-se um arquivo 'GOutput.txt' que é criado. Os dados desse arquivo são lidos para a apresentação dos resultados. Os resultados são:

1. Grelha equivalente deformada;
2. Gráficos de solicitações diversos;
3. Possibilidade de consultar os valores das solicitações em pontos específicos da estrutura.

Ressalta-se que a programação da unidade gráfica segue os preceitos da 'POO', com amplo uso da programação orientada a objetos e eventos. Portanto, alguns algoritmos podem parecer ambíguos se comparados com as estruturas de programação procedural[43] comum.

[43] O termo **'Programação procedural'** (ou programação procedimental) é um paradigma de programação baseado no conceito de chamadas a procedimento ''*procedure call''*. Os Procedimentos, também conhecidos como rotinas, subrotinas, métodos, ou funções (que não devem ser confundidas com funções matemáticas, mas são similares àquelas usadas na programação funcional) simplesmente contêm um conjunto de passos computacionais a serem executados. Um dado procedimento pode ser chamado a qualquer hora durante a execução de um programa, inclusive por outros procedimentos ou por si mesmo.
A programação procedural é geralmente uma escolha melhor que a programação sequencial e não estruturada em muitas situações que envolvem uma complexidade média e requerem facilidade de manutenção. Possíveis benefícios são:

- A habilidade de reutilizar o mesmo código em diferentes lugares no programa sem copiá-lo;
- Uma forma mais fácil de organizar o fluxo do programa que uma coleção de comandos "goto (programação)" ou "jump;
- A habilidade de ser fortemente modular e estruturado.

6.2.2.2 Arquivo de Modelagem da Estrutura

A estratégia utilizada no sistema gráfico foi a de manter todos os dados relevantes à estrutura diretamente associados aos elementos gráficos do projeto. Evita-se desta forma a necessidade da criação de vários arquivos auxiliares durante a modelação do projeto. Este formato inclusive permite que, no arquivo de projeto, sejam incluídos todos os dados necessários para o cálculo da grelha equivalente, simplificando assim o manuseio de arquivos de dados pelo usuário.
Como comentado anteriormente, o sistema de arquivos adotado foi o semelhante ao de um banco de dados, que inclui todas as informações de configuração do projeto, tabelas de estilos, entidades gráficas e os dados estendidos associados a elas.
Esses dados são gravados e indexados sequencialmente como indicado na

Tabela 11. Assim, toda vez que um arquivo gráfico é lido, todas as informações do projeto como um todo são carregados na memória, sendo desnecessário qualquer tipo de arquivo auxiliar.

Tabela 11: Sequência de gravação de dados no arquivo gráfico.

Tipo de objeto	**Tamanho**
Dados gerais de projeto	n . bytes
Tabelas de estilos: Tipos de Linhas Tipos de texto Tipos de pontos Tipos de dimensões	 n . estilos m . tipos
Lista de blocos	n . blocos
Lista de Layers	n . layers
Entidades gráficas	n . entidades

Vale lembrar que cada entidade tem associada a ela um bloco de dados na forma de uma estrutura "*Record*" complexa de tamanho variável (dependendo do tipo de entidade) que é lida toda vez que necessário. No programa 'Grelhas' usou-se este recurso para individualizar cada elemento por meio de seus dados específicos, e não pelas suas propriedades gráficas. Na Tabela 12 tem-se dois exemplos de estruturas de dados.

Tabela 12: Exemplo de estruturas de dados usadas no programa.

PBarData = ^TBardata;
TBarData = record // Bar data
P1,P2: TRCoord; // Bar coordinates
iN1, iN2, iBar : Word; // Node 1, 2 and bar identity
I, Load: Extended; // Inertia and load values
hN1, hN2, hB: THandle; // Handle to node 1, 2 and Bar
MT1, MT2, MF1, MF2, V1, V2: Double;
MatType: Word; //Tipo de material
Orientation: TOrientation;
BarType: TBarType;
end;
	Tipo BarData, que é um tipo complexo de dados que inclui vários tipos pré-definidos de dados. Uma estrutura semelhante é associada a cada barra da grelha equivalente.

PNodeData = ^TNodeData;
TNodeData = record // Node data
//P1: TRCoord;
X,Y: Double; //X coord, Y coord
N,iN1: Word; // NodeNumber, Node identity
mx,my,nz: Extended; // node solicitations
rx,ry,rz: Boolean; // node restrictions
Rotx, Roty, Deslz: Double; //Ref aos resultados Rotação x, y e deslocamento em z
pLaje, pAcid, pPerm, pViga, pParede, pLocal: Real; // cargas laje, acidental, permanente, viga, parede e pontual
hN1: THandle; //Node Identity Handle
Pos: TNodePosType;
end;
	Tipo NodeData que é o tipo associado a cada elemento nodal da grelha equivalente.

6.2.2.3 Tipos dos Objetos Gráficos

Entidades são objetos gráficos que têm uma representação gráfica: linha, círculo, retângulo, texto, elipse, polilinha etc. As entidades podem ter dados típicos, específicos e estendidos.

Os dados típicos são os dados que todas as entidades possuem, como as coordenadas de inserção, tipo de linha, camada a que pertencem, cor etc.

Os dados específicos são aqueles necessários para a geração de uma dada entidade, como por exemplo, o 'Raio', que é um dado específico da entidade círculo.

Os dados estendidos são aqueles associados a cada entidade pelo usuário, ou as necessidades da natureza do sistema. No caso específico do programa de grelhas, as entidades possuem dados estendidos do tipo: Altura de pilar, Lados de pilar e viga, Inércia a torção e a flexão, peso específico do material etc.

Um exemplo desse conjunto de dados é mostrada na Tabela 13.

Tabela 13: Conjunto de dados de entidades.

ID	Identificador único que serve para localizar a entidade.
Left, Right, Bottom e Top	Valores que definem a extensão da entidade em projeção.
LayerId	Identificador do layer a que pertence a entidade.
LineTypeId	Identificador do estilo de linha a ser usada para desenhar a entidade. Pode ser ByLayer
Scale	Fator de escala de representação da entidade.
Color	Identidade da cor na paleta de cores corrente. Pode ser ByLayer.
LineWeight	Espessura da caneta de desenho. Pode ser ByLayer.
UserData	Número específico, pode servir como definição extra da entidade.
ExtendedDataSize	Informa o tamanho em bytes dos dados estendidos.
ExtendedData	Pointer para o local onde estão gravados os dados estendidos.
Visible	Booleano que define se a entidade é visível ou não.
Locked	Booleano que define se uma entidade pode ser editada ou não.
...	...

6.2.2.4 Tipos de Objetos de Controle

O sistema gráfico é composto por objetos gráficos e não gráficos (invisíveis). Esses últimos são os objetos de controle. Eles são necessários para a administração do desenho e do projeto, ou seja, especificam como os objetos gráficos devem se comportar na tela. Alguns desses objetos são exemplificados na Tabela 14.

Tabela 14: Alguns objetos invisíveis do sistema.

DrawSpace	Gerencia a área de desenho
Layers	Gerencias as diferentes camadas do desenho. É responsável pelas variáveis ByLayer.
Lines	Gerencia a lista de tipos de linhas.
Blocks	Gerencia os diferentes blocos do sistema.
Points	Gerencia os diferentes tipos de estilos de pontos.
TextStyle	Gerencia os diferentes tipos de estilos de Texto.
DimStyle	Gerencia os diferentes tipos de estilos de dimensões.
Options	Gerencia as opções do sistema
ColorMap	Gerencia as diferentes paletas de cores do sistema.

6.2.2.5 Eventos de Controle

O programa não flui linearmente, mas sim respondendo a eventos. Criar um pilar é um evento. Esses eventos são definidos no sistema de forma a atenderem a maior

parte dos eventos possíveis que possam ocorrer durante o processo de prototipagem.

Alguns dos eventos básicos são listados na Tabela 15.

Tabela 15: Alguns eventos do sistema.

On_Mouse_Move	Inicia quando o usuário moveu o mouse. Utilizado por exemplo para se determinar as coordenadas atuais do cursor.
On_Mouse_Down	Iniciado quando o usuário clica algum botão do mouse.
On_Mouse_up	Inicia quando um botão foi solto.
On_Double_Click	Inicia ao se detecter um duplo clique sobre o mouse.
On_Mouse_Wheel	Inicia quando a roda central do mouse foi girada. Usado pelas funções de Zoom.
On_Key_Down	Inicia quando o usuário apertou uma tecla do teclado.
On_Key_Up	Inicia quando a tecla foi solta.
On_Regen	Inicia quando o desenho irá ser regenerado
On_File_load e On_File_Save	Iniciados ao carregar um arquivo ou ao gravar um arquivos
On_Window Resize	Inicia quando a janela é redimensionada. Usado para regenerar o desenho após a mudança de tamanho da tela.
On_Ent_Create	Inicia quando uma nova entidade foi criada. Útil para iniciar os dados estendidos das entidades em particular.
On_Ent_Copy e On_Ent_Erase	Iniciados a se copiar uma entidade ou ao se apagar uma.
...	...

6.2.2.6 Parâmetros e Métodos Globais

Os parâmetros globais são aqueles que o sistema usa como padrões de acesso. Esses parâmetros podem ser gravados em disco em arquivos diferentes do de projeto. Esses arquivos podem ter várias configurações acessadas individualmente, permitindo certo grau de particularização do sistema (é conhecido como opções do usuário ou de perfil).

Esses métodos são mostrados na Tabela 16.

Tabela 16: Métodos globais.

Profile_Load	Carrega um arquivo de perfil
Profile_Save	Grava um arquivo de perfil
Profile_Get_Name	Pega o nome do arquivo corrente de perfil
DefaultFont	Define o fonte default do sistema
Sis_Cursor	Booleano que define se o cursor do sistema deve ou não estar visível
BackColor	Cor de fundo do DrawSpace
FileNew	Cria um novo DrawSpoace
Save_AS	Salca o DrawSpace para um arquivo com nome a ser definido

Purge	Elimina as entidades apagadas do DrawingSpace. Após PURGE não é possível usar a função UNDO.
Regen	Apaga todas as entidades e as redesenha novamente.
FileName	Nome do DrawSpace atual
PathFileName	Trilha associada ao DrawSpace
ReadOnly	Booleano que define se o DrawSpace é editável ou não.
Dirty	Informa se o DrawSpace foi alterado ou não.
ExtendedData	Dados estendidos associados ao DrawSpace
Grid	Retorna os dados do Grid

6.2.2.7 Organograma das Bibliotecas

As bibliotecas são chamadas no Pascal de "*Units*". São bibliotecas que, normalmente, são organizadas em arquivos separados para cada "*Unit*". O programa principal pode fazer uso de uma biblioteca, referenciando-se a ela através da clausula "*Uses*". Quando um programa faz referência a uma biblioteca, todas as funções e procedimentos dessa biblioteca (que são indexadas na "*interface*") ficam disponíveis para serem usadas pelo programa ou módulo que fez a referência. No programa de Grelhas, as bibliotecas foram criadas de modo a encapsularem as estruturas de programa por afinidade, por exemplo, todos os métodos referentes às malhas estarão na unidade "*Unit_Mesh*", e assim por diante. O organograma geral pode ser visto na Figura 62, e as diferentes unidades são descritas mais adiante.

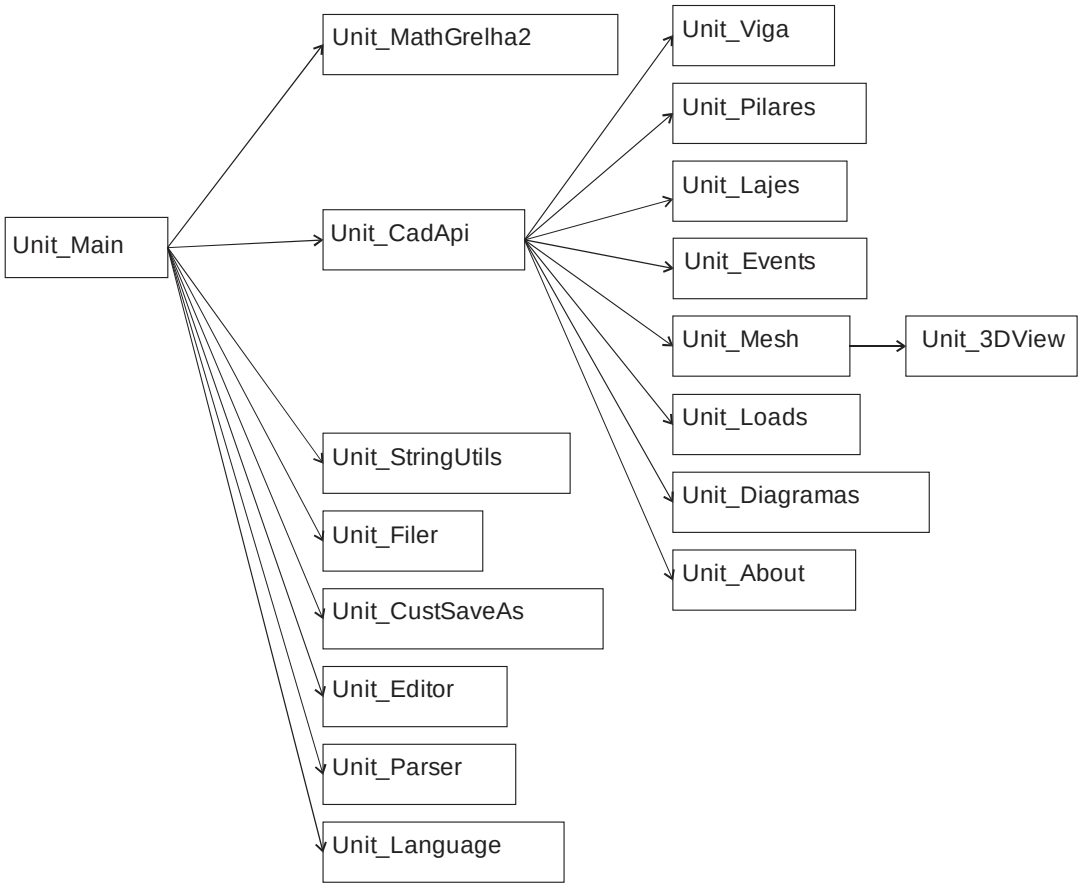


Figura 62: Organograma das unidades do sistema Grelhas.

- **Unit_Main**: Nesta unidade ficam as estruturas principais do programa. A janela principal, os objetos dessas janelas etc. Pode-se dizer que todo o "*Front-End*" do sistema está nesta unidade.
- **Unit_MathGrelha**: Nesta unidade estão as estruturas necessárias para o cálculo estrutural, como, por exemplo, a resolução de sistemas lineares por Gauss, multiplicação de matrizes, rotação etc. Está inclusa nesta unidade dialogo tradicional para abertura de arquivos de grelha para o cálculo. Diálogo útil quando se deseja fazer cálculos a parte do sistema principal.
- **Unit_CADAPI**: nesta unidade fica o núcleo do sistema gráfico, com todas as chamadas de interface para acessar suas estruturas.
- **Unit_StringUtils**: Esta unidade contém utilitários para se trabalhar "*Strings*", que são cadeias de caracteres. Esta unidade é muito utilizada na leitura de arquivos de dados e de resultados, pois é ela que faz o serviço de separar as

linhas de dados (no formato de texto) e transformá-las em informações coerentes ao sistema.

- **Unit_Filer**: Esta unidade contém as estruturas necessárias para a leitura e gravação de dados em arquivos.
- **UnitCustSaveAS**: Esta unidade é a implementação de um diálogo específico para a função ‘Salvar Como’, que necessitou ser particularizada para o sistema Grelhas.
- **Unit_Parser**: É uma biblioteca que é facilmente encontrada na internet que permite a leitura e interpretação de uma cadeia de caracteres que representam uma fórmula.
- **Unit_Language**: Nesta unidade são implementadas as cadeias de caracteres para a gestão de línguas. A Unidade foi idealizada no intuito de se ter o sistema traduzido em várias línguas.
- **Unit_Viga**: Nesta unidade estão as estruturas necessárias para a criação e edição e visualização do elemento viga. Estão presentes também ferramentas para a gestão de dados de tal elemento.
- **Unit_Pilares**: Esta unidade contém as ferramentas de acesso e edição do elemento pilar.
- **Unit_Lajes**: Semelhante aos elementos anteriores, mas para lajes.
- **Unit_Events**: Esta unidade contém todas as estruturas de eventos referentes ao sistema gráfico. Estes eventos são acionados na criação ou deleção de primitivas, na alteração de coordenadas etc.
- **Unit_Mesh**: Esta unidade contém todas as rotinas para a criação da malha equivalente e a deformada. Contém também várias ferramentas para a edição e alteração de nós e barras.
- **Unit_Loads**: Esta unidade contém as estruturas responsáveis por criar os elementos de carga nodal.
- **Unit_Diagramas**: Esta unidade administra a criação de diagramas de momento fletor, torçor, cortante e elástica de um conjunto de nós.
- **Unit_About**: Unidade com a descrição do sistema e faz referência aos seus autores.

- **Unit_3DView**: Esta unidade é responsável por gerar a vista renderizada da grelha equivalente deformada. O gráfico de superfície faz parte do sistema SDL de ferramentas científicas “SDL Component Suite 10.3”[44] (Lohninger, 2004). No sistema Grelhas o gerador gráfico de superfícies foi integrado na sua versão gratuita cujo limite é o número de nós.

[44] O sistema SDL é desenvolvido pela “Epina Software Labs”, que desenvolveram uma série de componentes de cunho científico para Delphi e Lazarus. Estes são ferramentas gráficas, estatísticas, matriciais, de medição etc.

7 RESULTADOS

7 RESULTADOS

7.1 COMPARATIVOS ENTRE OS PROGRAMAS GPLAN E GRELHAS

Neste capítulo estudam-se dois tipos de pavimentos, o primeiro sendo uma laje isolada sobre pilares, e o segundo sendo uma laje apoiada sobre vigas e pilares. Comparam-se os resultados obtidos com os resultados obtidos pelo programa GPLAN (com as mesmas estruturas) para a validação dos dados.
Para cada estrutura escolheram-se dois nós, o primeiro com o maior deslocamento e o segundo como sendo próximo a um elemento estrutural.
Na sequência se analisará o potencial do programa com alguns exemplos onde se verão algumas das diferentes possibilidades de combinações de elementos estruturais, de cargas, de efeitos e diferentes tipos de malhas.

7.1.1 Considerações Iniciais

O GPLAN é uma ferramenta pertencente ao projeto ANSER (Análise de Sistemas Estruturais Reticulados) que foi desenvolvido na USP pelos Engenheiros Márcio R.S. Correa, Marcio A. Ramalho e Luiz H. Ceotto. O programa foi compilado na linguagem FORTRAN 16 bits e, atualmente, só pode ser executado num ambiente com emulador DOS ou numa máquina virtual com o sistema DOS instalado. O GPLAN é um programa que foi muito usado, inclusive como ferramenta de ensino, para a análise elástico-linear de grelhas planas durante os anos 80. É um programa consagrado que foi utilizado neste trabalho para o cálculo de duas estruturas, a primeira na configuração de laje sobre pilares (laje isolada), e a segunda de laje sobre vigas e pilares. Os resultados são comparados com os oriundos do programa Grelhas para efeito de validação dos dados.

7.1.2 Modelo Laje Sobre Pilares Sem Vigas (Laje Isolada)

Neste modelo, mostrado na planta de forma da Figura 63, tem-se um quadrilátero de nove pilares por face, com uma laje de dez centímetros diretamente sobre eles. A grelha equivalente montada pelo programa Grelha é apresentada na Figura 64. Já a grelha equivalente montada pelo sistema GPLAN é apresentada na Figura 65.

Os pontos escolhidos para a comparação são: o nó 41 (nó com o maior deslocamento) que, coincidentemente, tem o mesmo numeral em ambas as grelhas, e o nó 17 no programa Grelhas com o nó 71 do programa GPLAN, que ficam próximos ao pilar inferior direito.

Os resultados são apresentados na Tabela 17, onde são apresentados o erro absoluto e o relativo.

A malha deformada gerada pelo programa Grelhas é ilustrada na Figura 66 e na Figura 67.

Dados da estrutura:

Concreto: Resistência característica do concreto à compressão: F_{ck} = 30,0 MPa
Módulo de deformação longitudinal do concreto:

$$E_c = 0{,}85.5600.\sqrt{Fck} = 0{,}85.5600.\sqrt{30} = 26072\ \text{MPa} = 2{,}6072.10^7\ \text{kN/m}^2$$

Coeficiente de Poisson: ν = 0.2

Módulo de deformação transversal do concreto:

$$G_c = 0{,}40.E_c = 0{,}4.26072 = 10429\ \text{MPa} = 1{,}0429.10^7\ \text{kN/m}^2$$

Dados do carregamento da placa:

a) Peso próprio: g_1 = 0.10*25 = 2.5 kN/m^2
b) Sobrecarga permanente: g_2 = 1.0 kN/m^2
c) Sobrecarga de utilização: q = 2.0 kN/m^2
d) Carga total: p = 5.5 kN/m^2

Grelha Equivalente: Adotou-se uma grelha equivalente composta de 81 nós e 144 barras, com espaçamento de100 cm entre as barras nas duas direções.

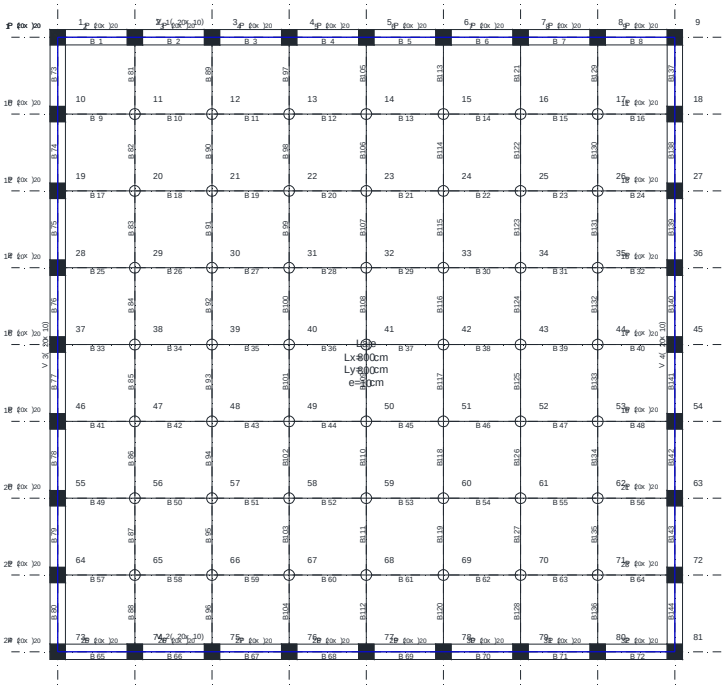


Figura 63: Forma da estrutura.

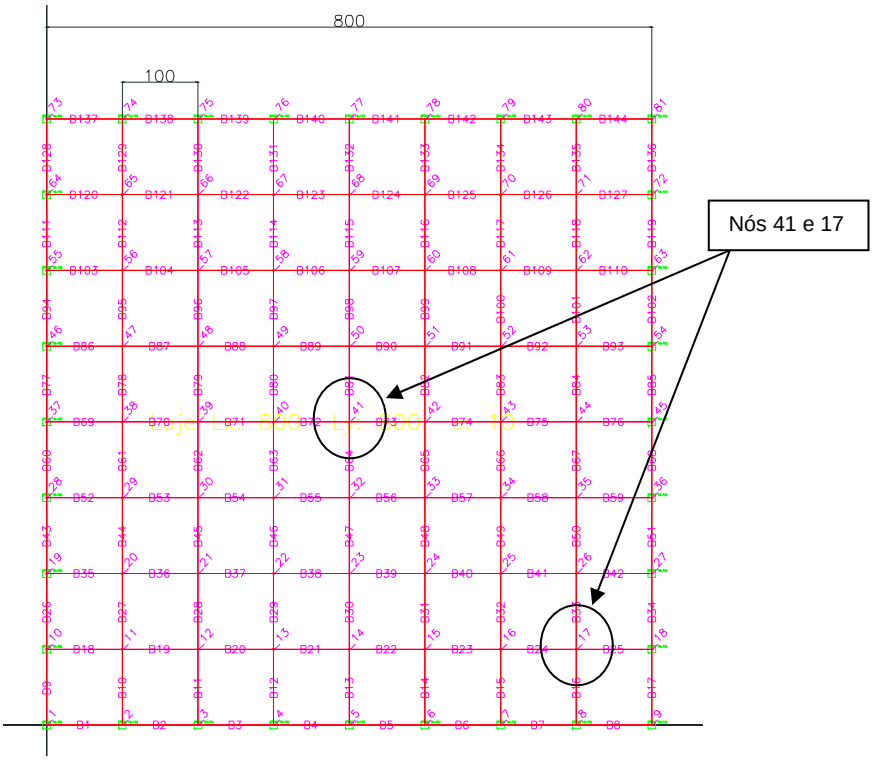


Figura 64: Grelha gerada pelo programa Grelhas.

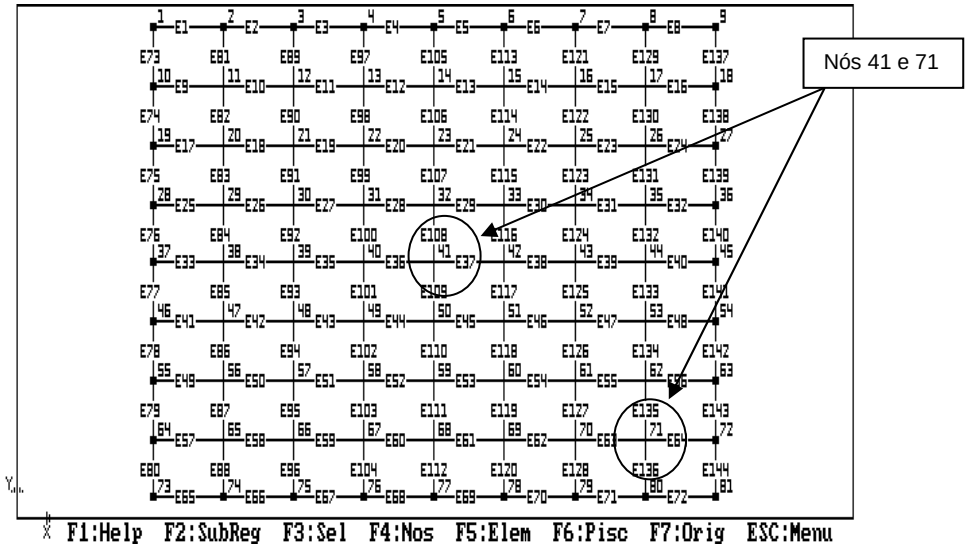


Figura 65: Grelha gerada pelo programa GPLAN.

Tabela 17: Comparativo GPLAN e Grelhas, erros entre os nós (41-41) e (71-17).

Sistema	Nó	Dz	DRx	DRy	V	Mf	Mt
GPLAN	41	-0.0474073	-0.0000001	0	1.375	15.061	0
GRELHA	41	-0.0474094	0	0	1.375	15.0638	0
Erro Abs.		0.0000021	0	0	0	0.0028	0
Erro Rel.		0.0000443	0	0	0	0.000186	0
GPLAN	71	-0.0076143	0.0070088	0.0070087	2.11	-3.264	9.730
GRELHA	17	-0.0076145	0.007009	0.007009	2.1105	-3.2641	9.7272
Erro Abs.		0.0000002	0.0000002	0.0000003	0.0005	0.0001	0.0028
Erro Rel.		0.00002627	0.0000285	0.0000428	0.000237	0.0000306	0.000288

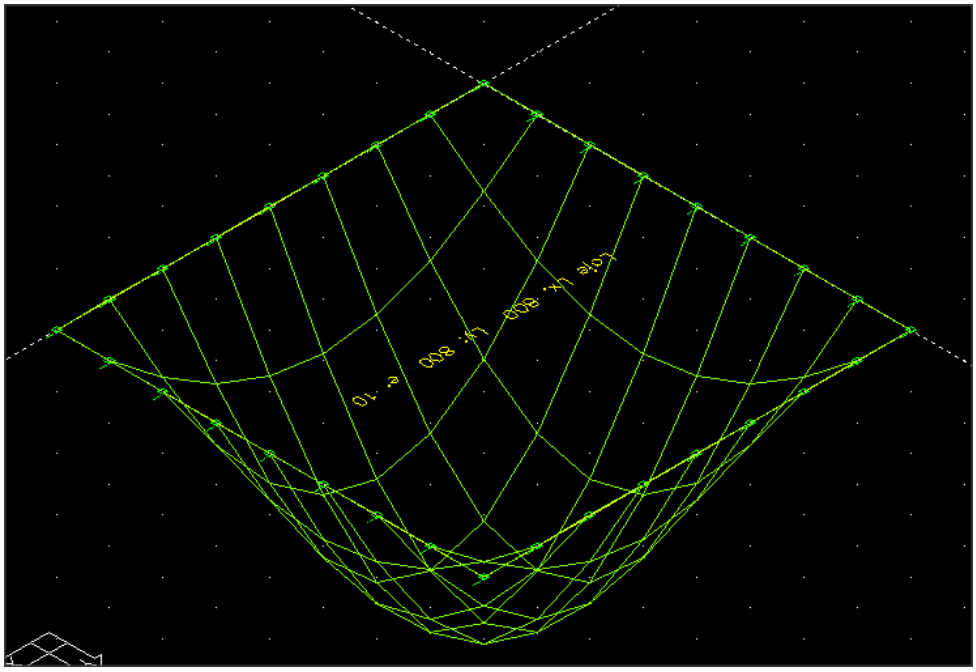


Figura 66: Malha equivalente deformada pelo programa Grelhas.

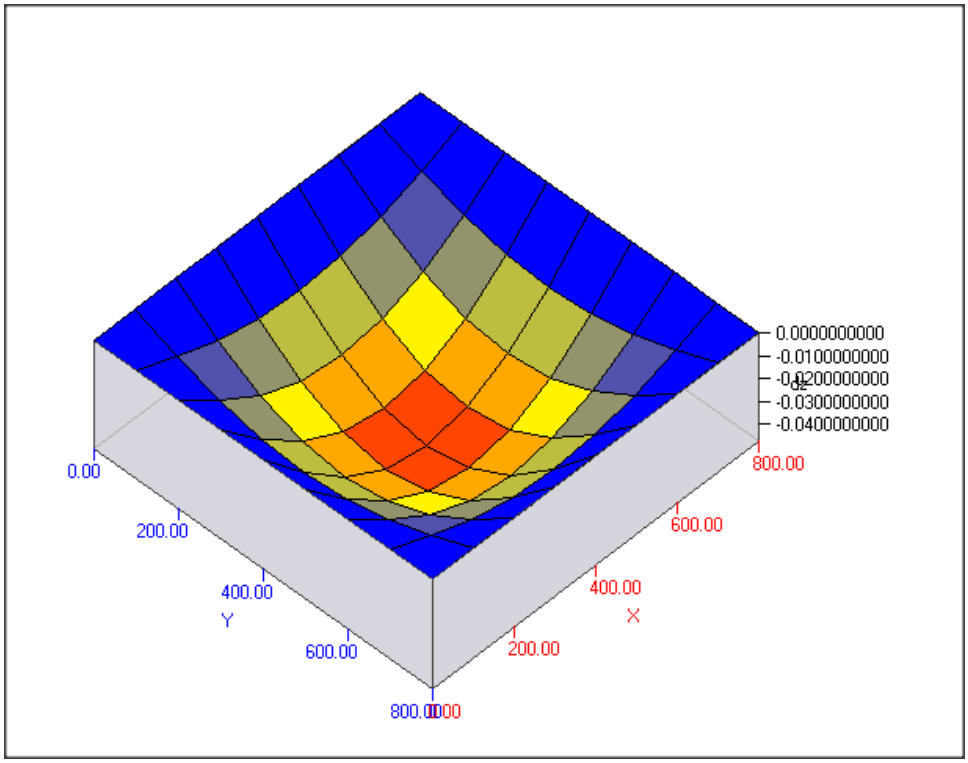


Figura 67: Gráfico de superfície da grelha equivalente pelo programa Grelhas.

7.1.3 Modelo Laje com Vigas e Pilares

Neste modelo tem-se uma estrutura com cinco vigas e duas lajes com dez centímetros cada e seis pilares. A planta de formas é mostrada na Figura 68, a

grelha equivalente pelo programa Grelhas na Figura 69, e finalmente a grelha equivalente pelo programa GPLAN na Figura 70.

Dados da estrutura:

Concreto: Resistência característica do concreto à compressão: F_{ck} = 30,0 MPa

Módulo de deformação longitudinal do concreto:

$$E_c = 0{,}85.5600.\sqrt{Fck} = 0{,}85.5600.\sqrt{30} = 26072\ \text{MPa} = 2{,}6072.10^7\ \text{kN/m}^2$$

Coeficiente de Poisson: ν = 0,2

Módulo de deformação transversal do concreto:

$$G_c = 0{,}40.E_c = 0{,}4.26072 = 10429\ \text{MPa} = 1{,}0429.10^7\ \text{kN/m}^2$$

Dados do carregamento da placa:

Peso próprio: g_1 = 0.10*25 = 2.5 kN/m^2

Sobrecarga permanente: g_2 = 1.0 kN/m^2

Sobrecarga de utilização: q = 2.0 kN/m^2

Carga total: p = 5,5 kN/m^2

Grelha equivalente: Adotou-se uma grelha equivalente composta de 153 nós e 280 barras, com espaçamento de 100 cm entre as barras nas duas direções.

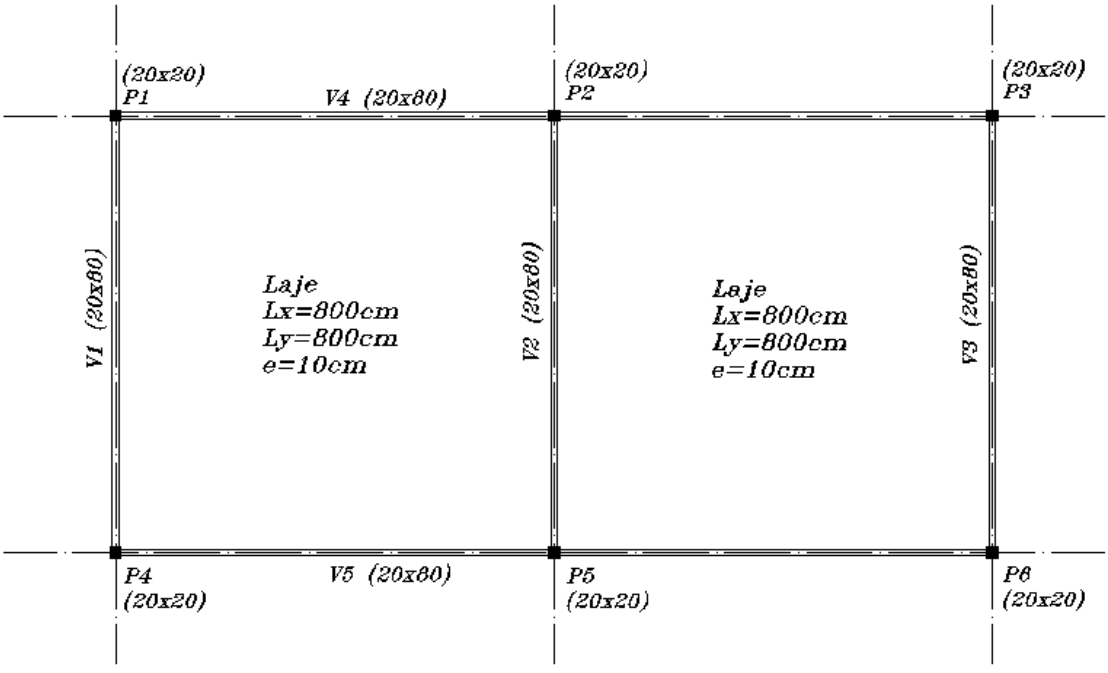


Figura 68: Forma da estrutura.

.

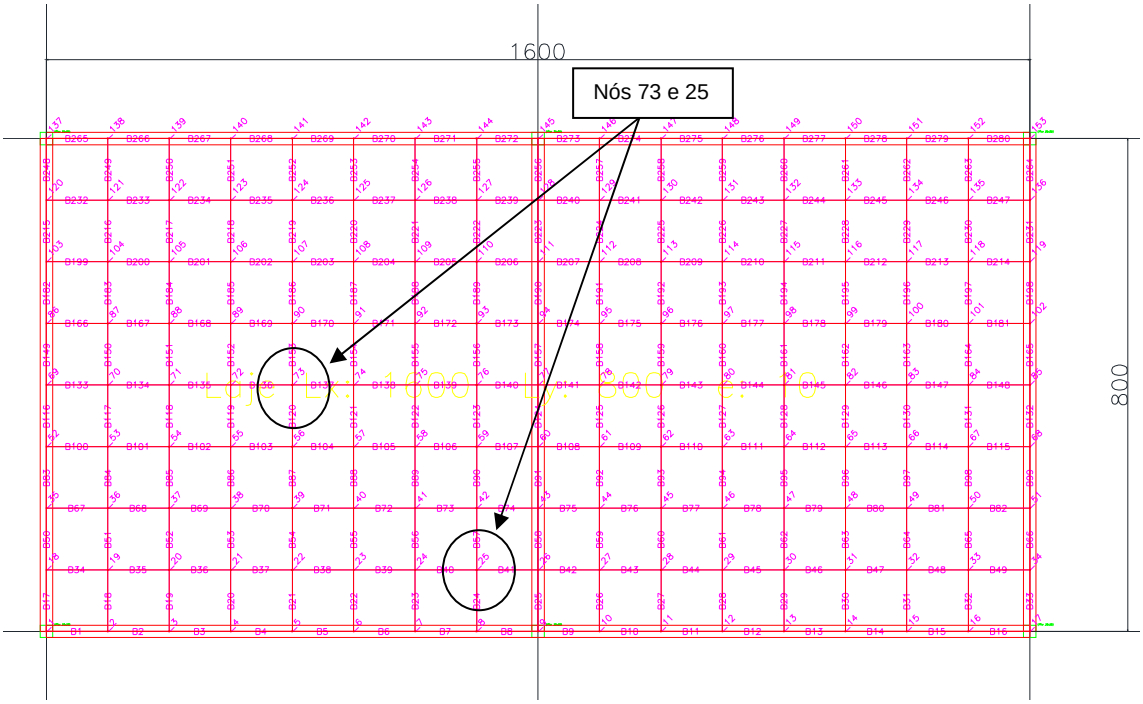


Figura 69: Grelha equivalente gerada pelo programa Grelhas.

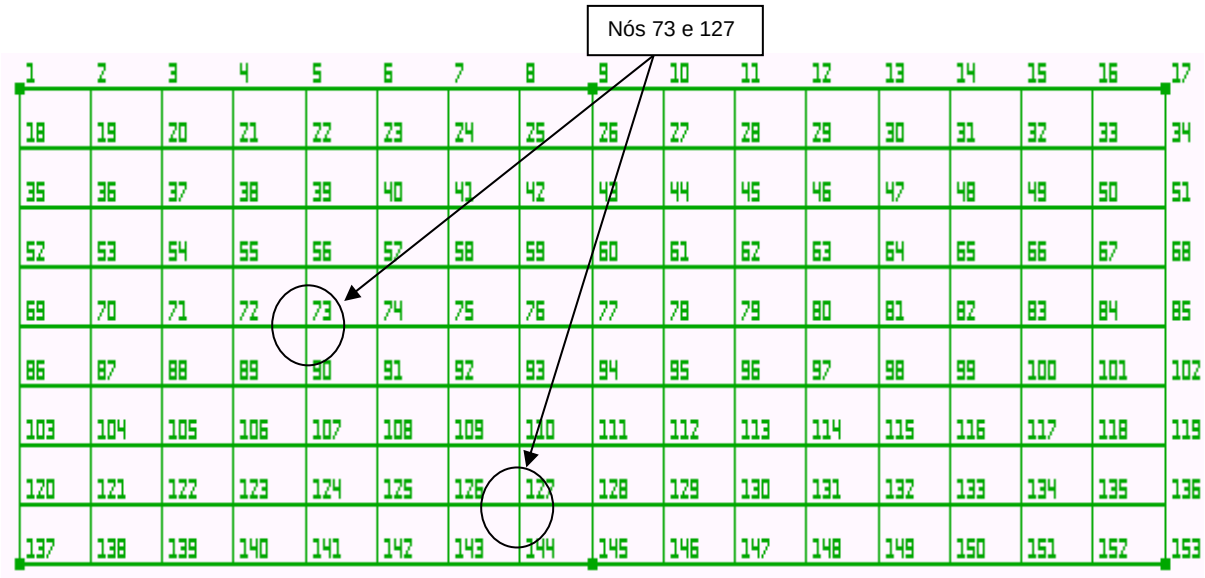


Figura 70: Grelha equivalente gerada pelo GPLAN.

Tabela 18: Comparativo GPLAN e Grelhas, erros entre os nós 73-73 e 127-25.

Sistema	Nó	Dz	DRx	DRy	V	Mf	Mt
GPLAN	73	-0.0269392	-0.0000001	-0.0002567	-0.908	-8.696	0
GRELHA	73	-0.02694	0	-0.0002567	-0.9077	-8.6977	0
Erro Abs.		0.0000008	0	0	0.0003	0.0017	0
Erro rel.		0.0000297	0	0	0.00031	0.000195	0
GPLAN	127	-0.0052451	-0.0048554	-0.0021824	2.163	-1.283	3.482
GRELHA	25	-0.0052453	-0.0048556	-0.0021824	2.1635	-1.2831	3.4809
Erro Abs.		0.0000002	0.0000002	0	0.0005	0.0001	0.0011
Erro rel.		0.0000381	0.0000412	0	0.00023	0.000078	0.00032

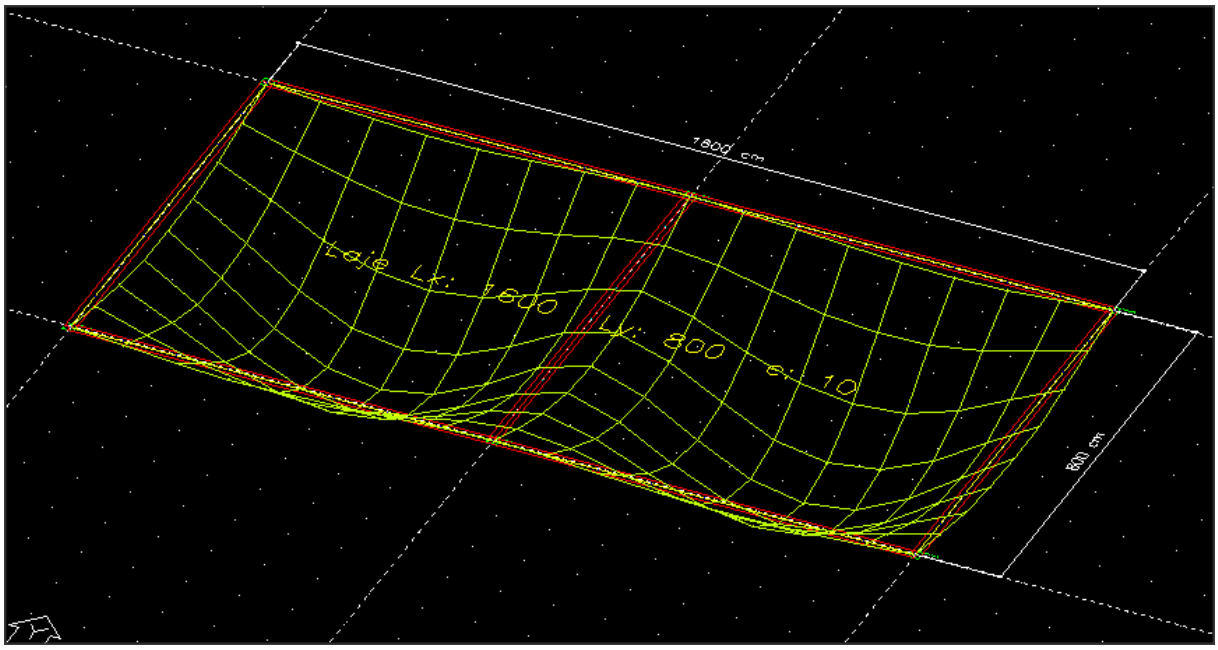


Figura 71 : Grelha deformada pelo programa Grelhas.

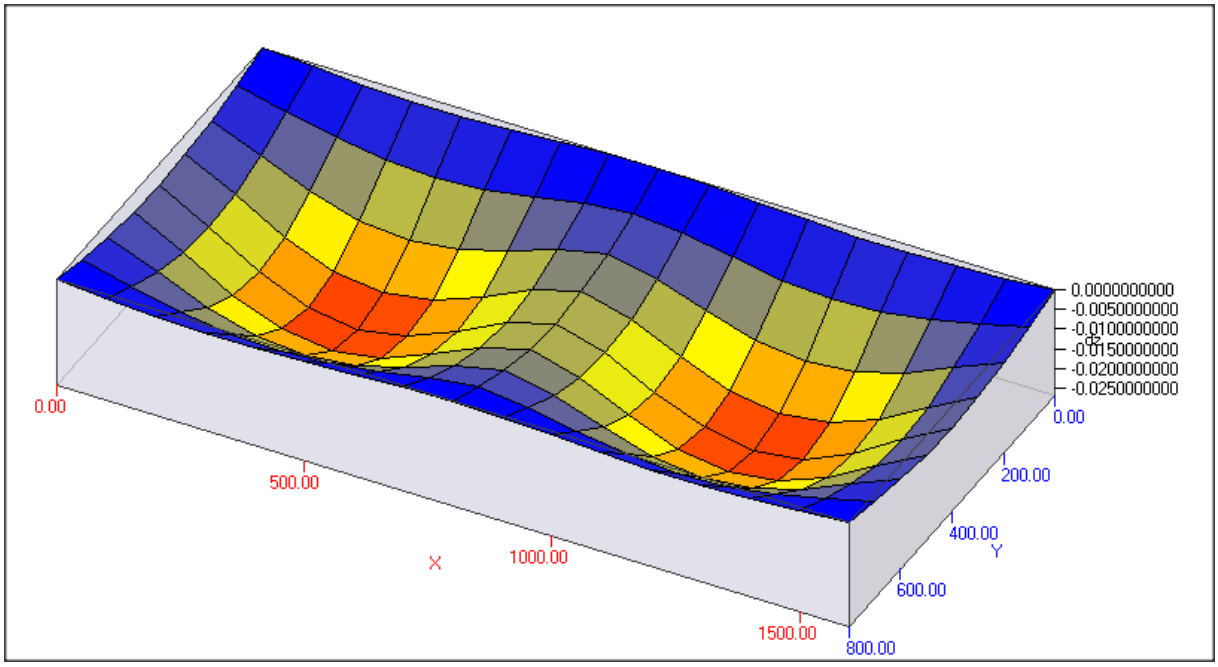


Figura 72: Gráfico de superfície da grelha deformada pelo programa Grelhas.

7.1.4 Conclusões Sobre os Resultados Obtidos nas Duas Estruturas

Os resultados apresentados na Tabela 17 e Tabela 18 mostram claramente que os nós calculados estão corretos, pois o erro relativo ficou abaixo da quarta casa decimal. Isso implica que o sistema gráfico está gerando os dados estruturais para o cálculo de forma correta, além do fato que o sistema de cálculo também está calculando corretamente os deslocamentos e esforços nas barras.

7.2 POTENCIAL DO PROGRAMA GRELHAS

A vantagem mais marcante do programa Grelha é a relativa rapidez que se faz a prototipagem da estrutura, na ordem de alguns minutos. Os elementos estruturais são editáveis, o que facilita em muito no caso da necessidade de correções. A remoção ou colocação de novos elementos é muito simples, e as alterações são automaticamente inseridas no contexto do projeto. Grelhas são alteradas e recalculadas em segundos.

Para ilustrar essas habilidades, pode-se usar a grelha da Figura 41 que contém vários elementos estruturais e cargas distribuídas em nós.

A estrutura da Figura 73 é a mesma estrutura citada anteriormente, com um trecho a menos de viga. A grelha equivalente deformada é desenhada sobre a anterior, e podem-se observar as diferenças entre elas. Note que o tempo total para o recalculo e desenho da grelha ficou em menos de um segundo (como mostrado na área de informações do programa)

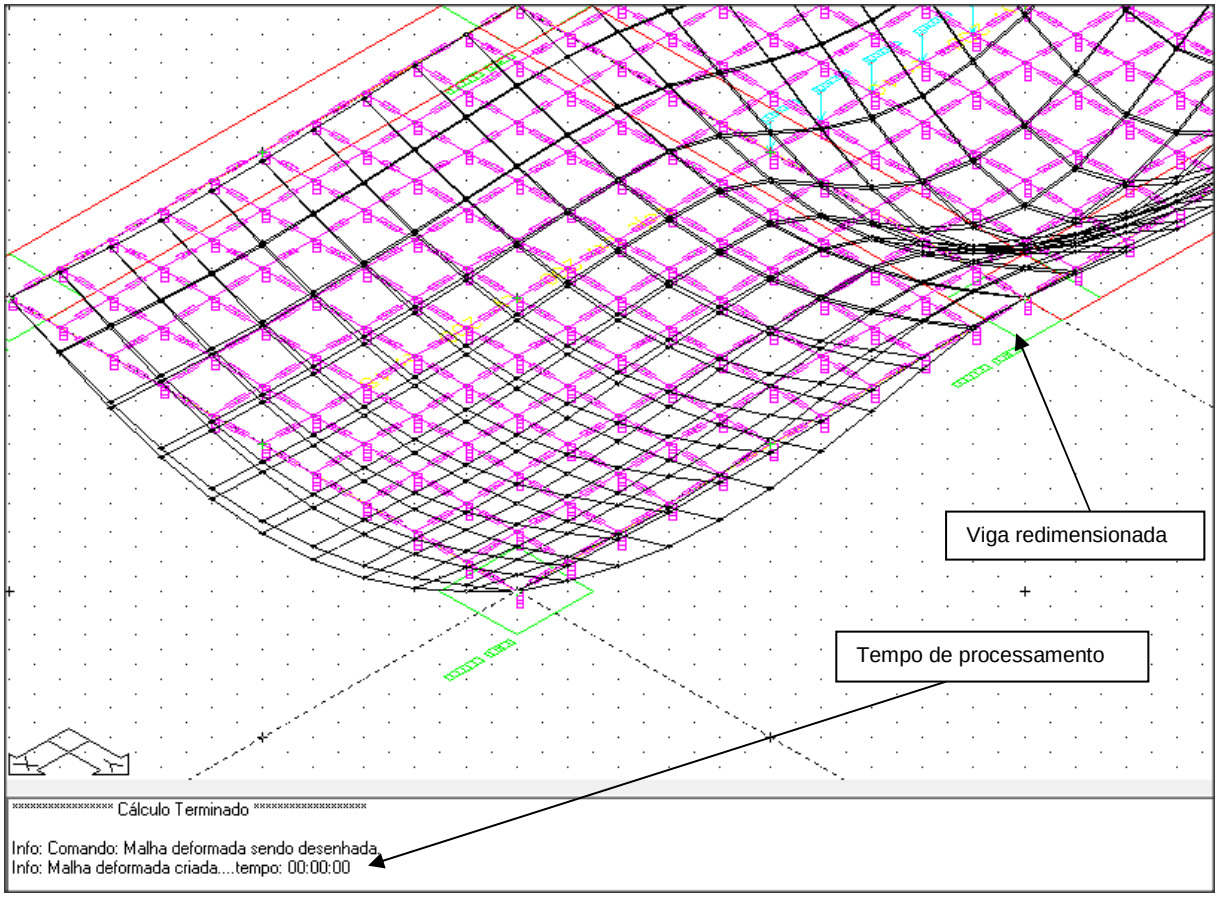


Figura 73: Estrutura editada e recalculada. As duas grelhas resultantes ficam visíveis e podem ser comparadas para efeito didático.

Na Figura 74 recoloca-se a viga na posição original, mas elimina-se o pilar de canto. A estrutura é recalculada, novamente, em menos de um segundo.

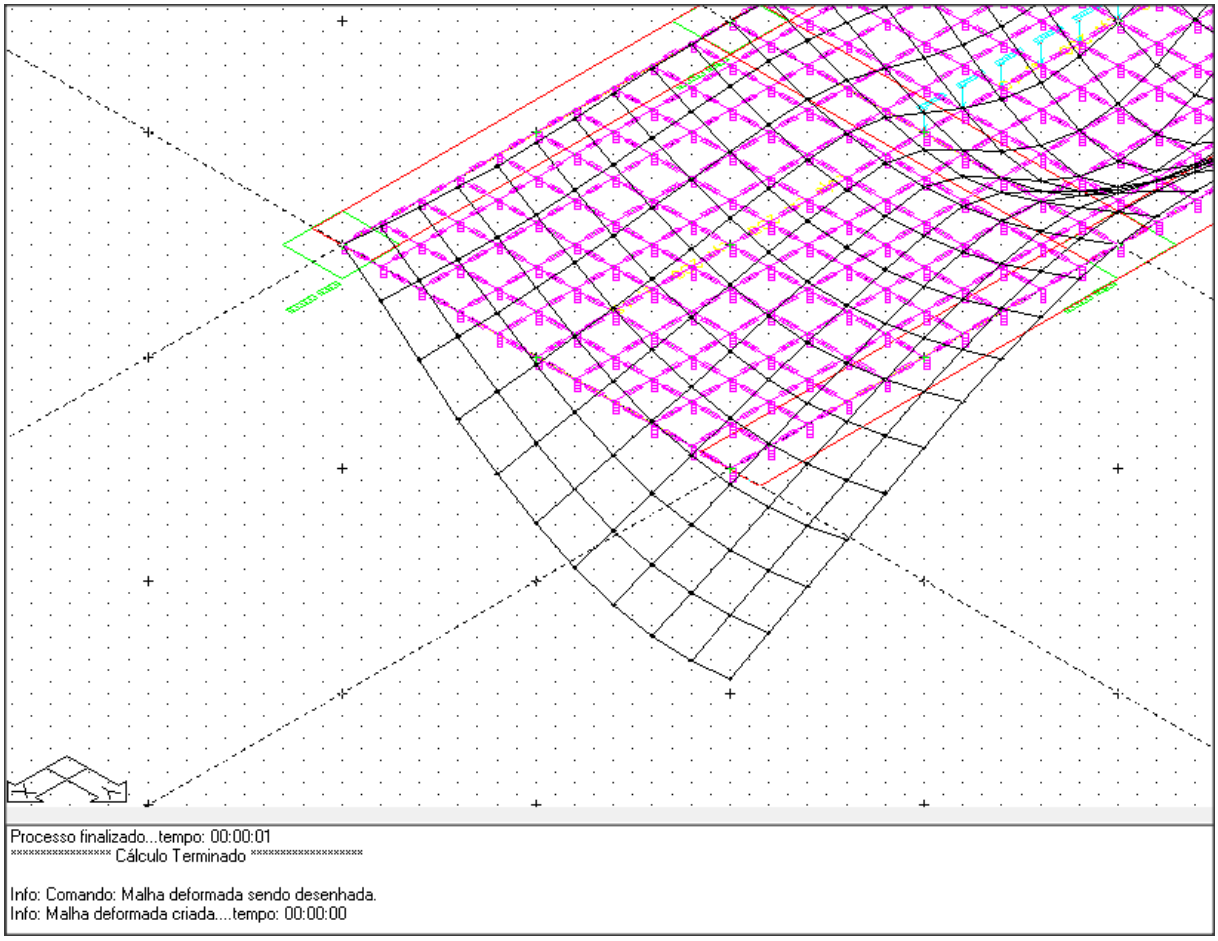


Figura 74: Estrutura alterada e recalculada.

O fator agilidade fica óbvio pela maneira simples que os elementos podem ser alterados ou trocados de posição e pela rápida apresentação dos resultados recalculados.

7.3 MALHAS DENSAS

Outro fator importante é o número praticamente ilimitado de nós e barras que o sistema pode gerir, permitindo-se assim que se possam calcular grelhas de, praticamente, qualquer tamanho (ver Figura 77). O limite reside na memória do computador e na velocidade de processamento. De fato, grelhas com malhas muito densas podem demorar entorno de uma hora para serem processadas. Na Figura 75 vê-se a estrutura anterior com a malha refinada com 60 divisões em ‘X’ e 30 divisões em ‘Y’, gerando uma malha de 3690 barras e 1891 nós.

Nas imagens da Figura 77 e Figura 78 vê-se malhas exemplo extremamente densas já calculadas.

A questão é se realmente malhas de grande densidade melhoram significativamente os resultados obtidos e se, analogamente, o custo computacional compensa.

Na Tabela 19 observa-se na coluna “erro relativo” (coluna 5) cujos valores são flutuantes significando que entre uma densidade de malha e outra se tem valores que podem, inclusive, aumentar com o refinamento da malha. No entanto ao se aceitar-se a premissa de que uma malha mais refinada é, obrigatoriamente, a mais precisa pode-se aplicar a comparação de erro relativo à malha mais densa, que é a de 51X51 na Tabela 19. Percebe-se então que os valores de erro relativo (coluna 6) decaem geometricamente.

Tabela 19: Quadro comparativo entre malhas com diferentes densidades de nós para a mesma estrutura.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Malhas	Tipo de Malha	Número do nó	Densidade (nó/pav)	Flecha (m)	Erro Relativo (n)/(n+1)	Err. Rel. $(n)/(n_{max})$	Tempo de Exec.
(1)	9X9	5	81	-6.23E-5	1.15	11.46	0m:0s
(2)	17x17	9	289	-3.2E-5	0.95	5.4	0m:3s
(3)	21X21	11	441	-2.55E-5	0.55	4.1	0m:9s
(4)	31X31	16	961	-1.65E-5	2.3	2.3	1m:31s
(5)	41X41	21	1681	-5E-6	0	0	7m:36s
(6)	51X51	26	2601	-5E-6	0	0	26m:34s

De fato a malha 31X31 com uma densidade de 64 $\frac{nós}{Pav}$ tem um erro relativo com pouco mais de 2% (ver Gráfico 1) necessitando um tempo de processamento aceitável de 1m:31s. Malhas superiores à malha de 41X41 fornecerão dados com precisão pouco superior à de 31X31, com um custo de tempo de processamento exponencialmente maior, podendo chegar a mais de uma hora (o aumento no tempo de processamento entre as malhas 31X31 e 51X51 foi de 1751,64%) para um ganho de precisão de 2.3%. A progressão do custo computacional é mostrada no Gráfico 2.

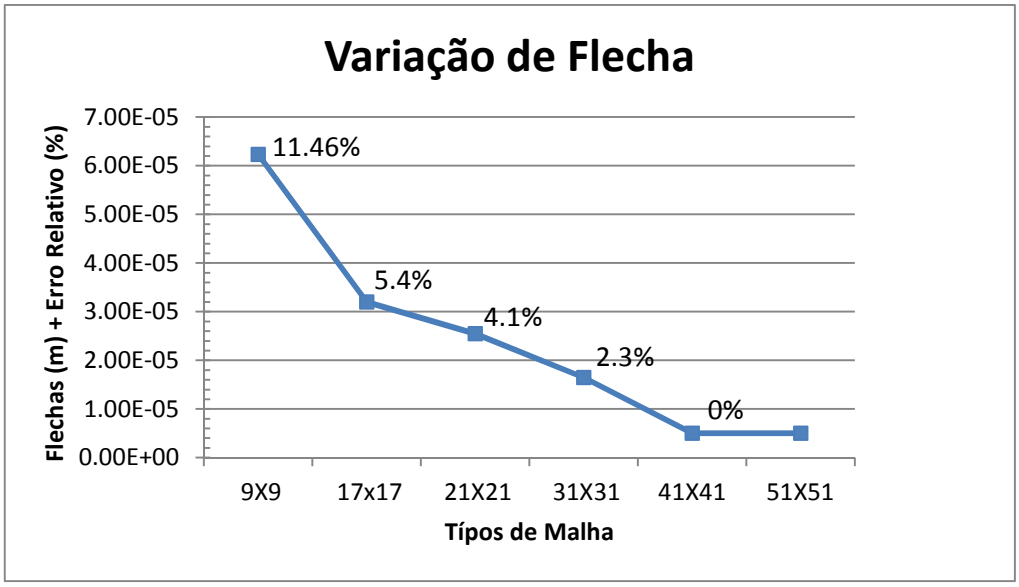


Gráfico 1: Progressão do erro relativo percentual e flechas nas malhas.

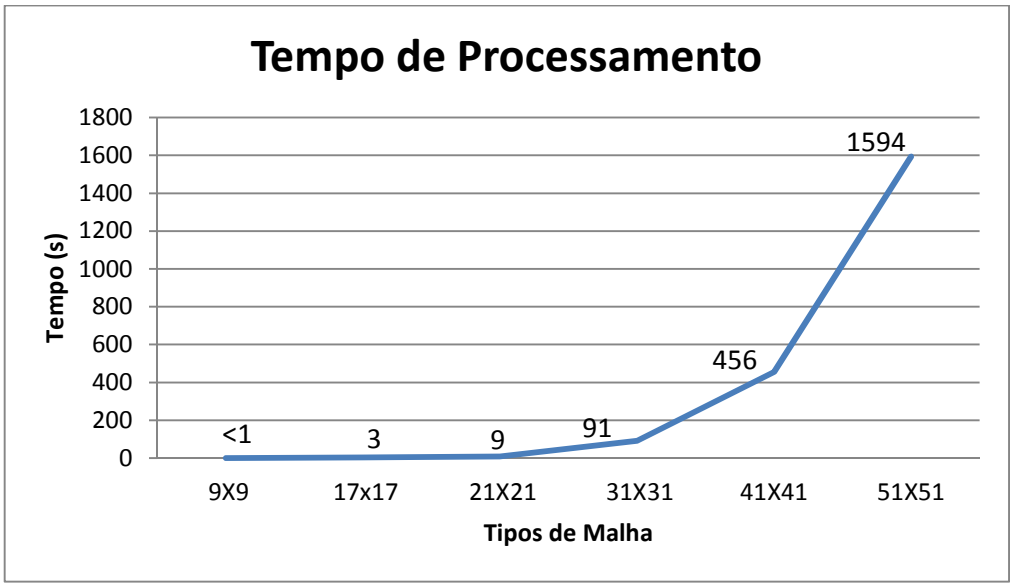


Gráfico 2: Tempo de processamento para cada malha.

Pode-se dizer que malhas com densidade superior a 256 $\frac{nós}{pav}$ não são ideais, pois trarão poucas melhorias nos resultados com um custo de tempo de processamento muito maior. Pode-se observar que a grelha de 51X51 teve um tempo de processamento muito maior que a de 41X41, sem melhorar em nada a precisão do modelo. Claro, ficam a critério do projetista os fatores a serem utilizados nas malhas.

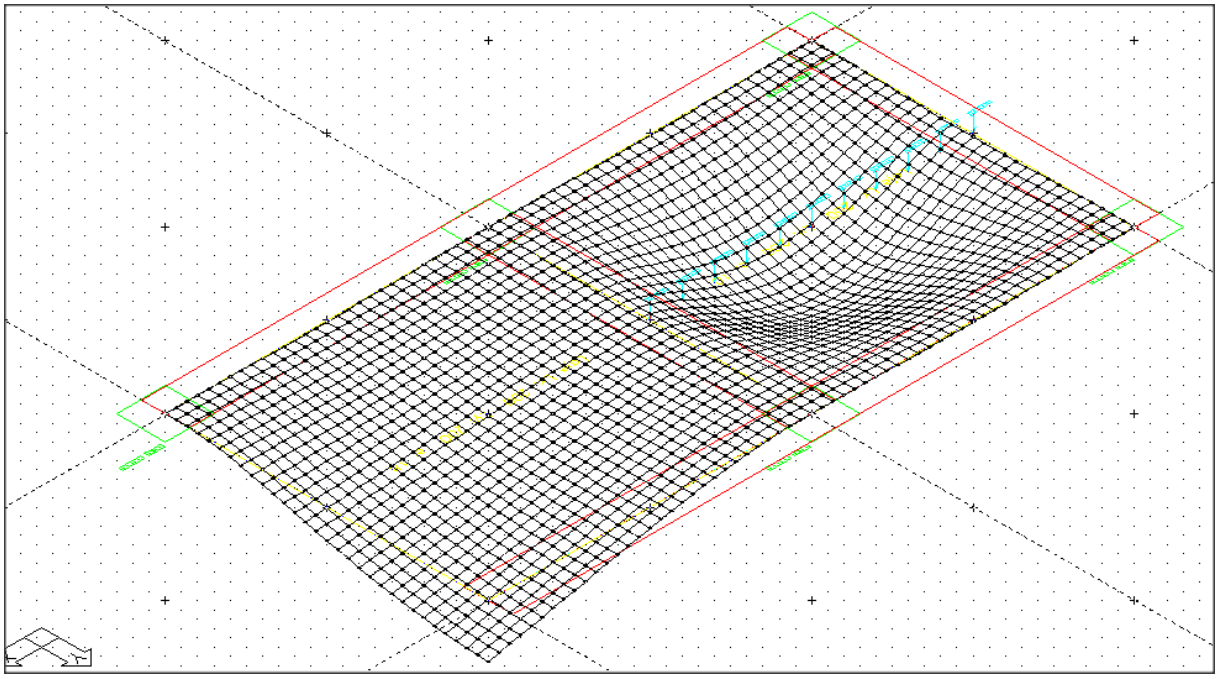


Figura 75: Estrutura com malha refinada, com 3690 barras e 1891 nós. Tempo de processamento: 00h:09m:26s

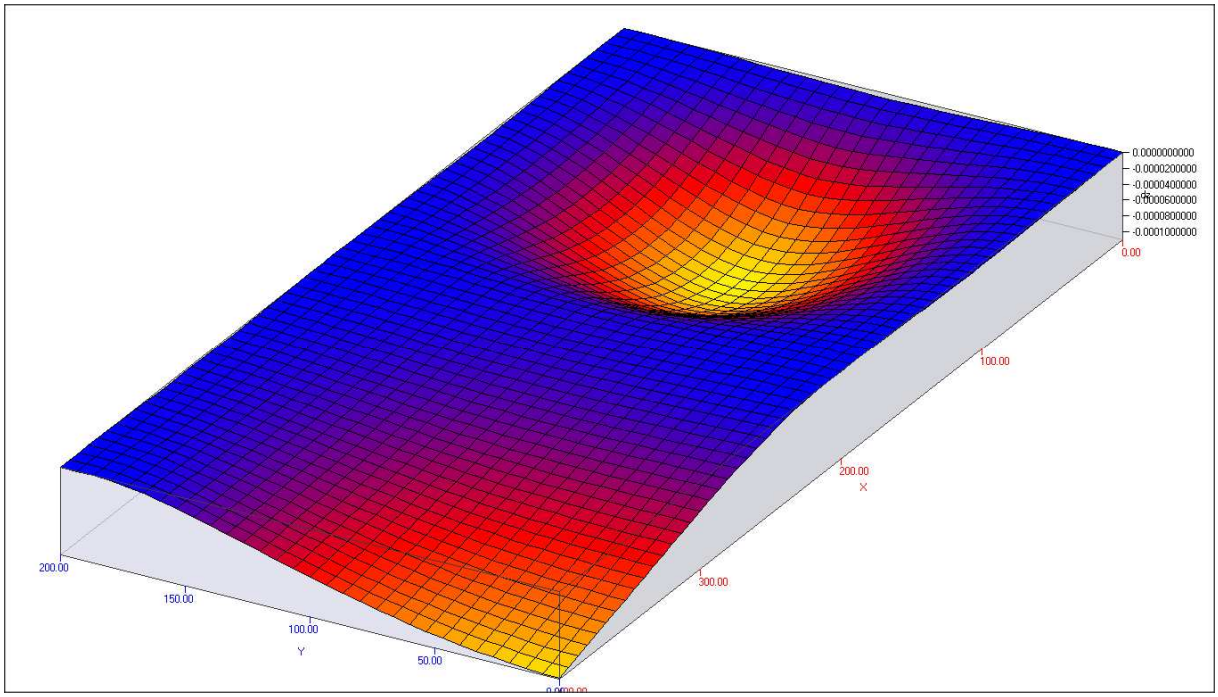


Figura 76: Estrutura com malha refinada renderizada do programa Grelha.

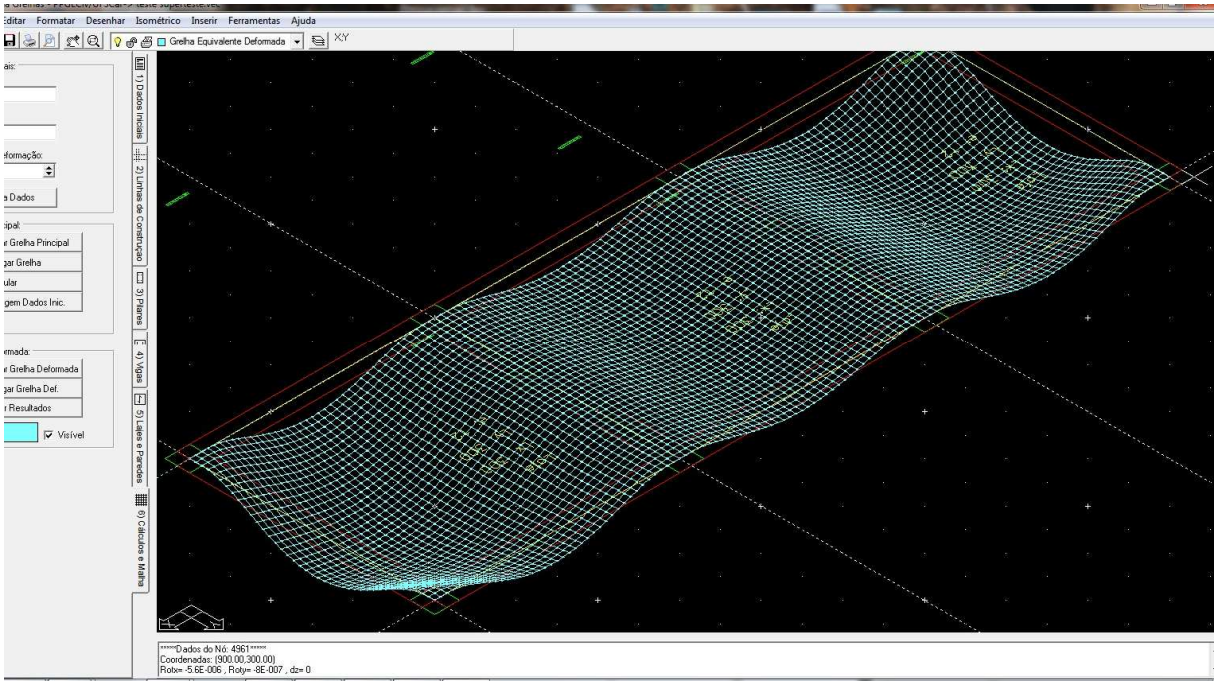


Figura 77: Grelha extremamente densa: observam-se vários nós que ficam na região dos pilares, todos esses nós ficam indeslocáveis.

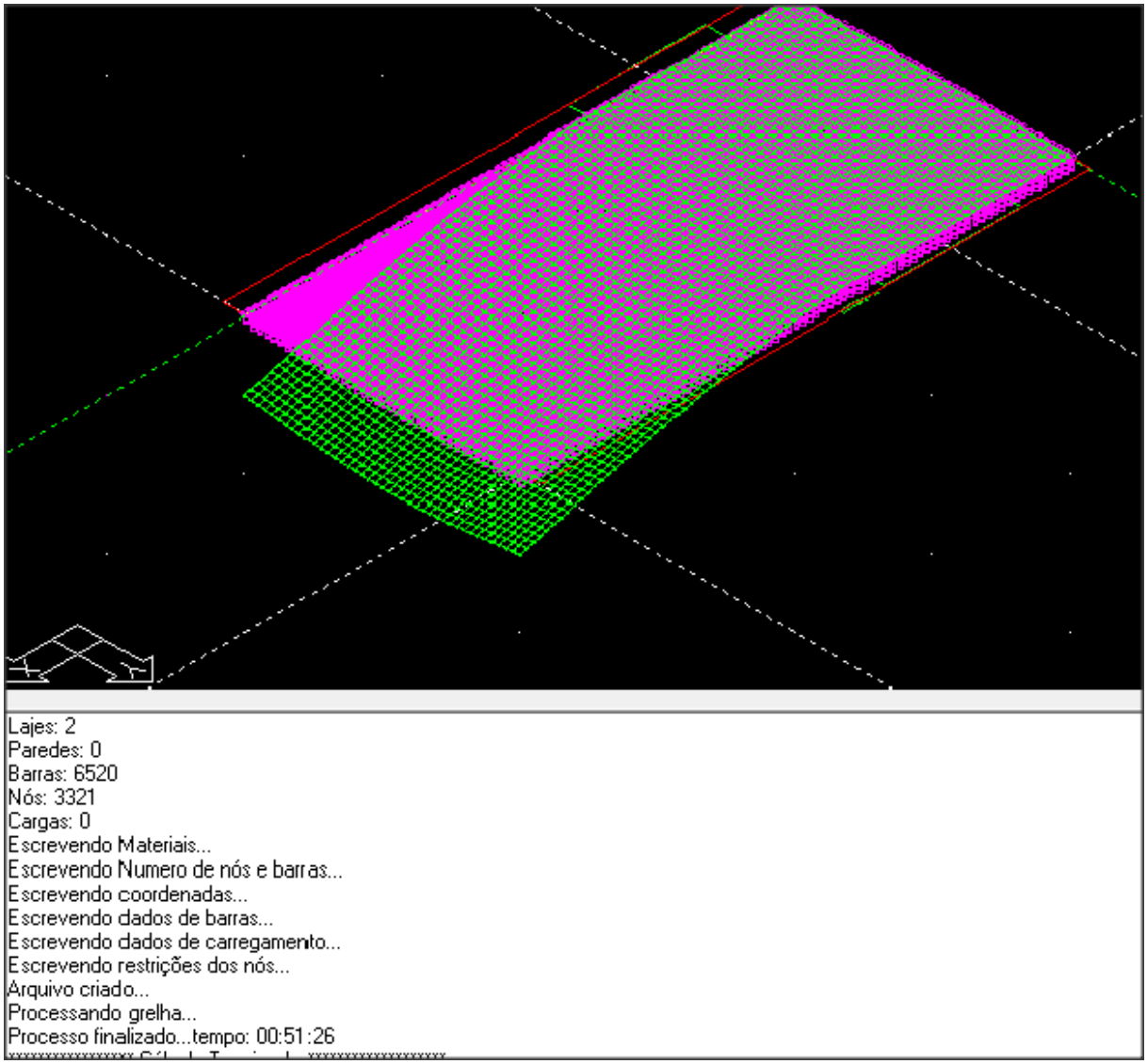


Figura 78: Grelha extremamente densa com 6520 barras e 3321 nós. Tempo de processamento: 00h:51m:26s.

7.4 COMPARAÇÃO COM O EBERICK

Deve-se ressaltar que os sistemas comerciais nem sempre deixam claro quais os critérios utilizados com relação aos algoritmos e componentes utilizados. Os sistemas como o Eberick são essencialmente ferramentas de cálculo e não, especificamente, de análise estrutural: introduz-se o desenho de uma estrutura e o sistema criará um projeto estrutural, seguindo a norma corrente.

Os métodos de cálculo nem sempre são os mesmos entre um sistema e outro. Por exemplo, o Cypecad utiliza o sistema de diferenças finitas para o cálculo de lajes, já o Eberick usa o sistema de grelha equivalente. Este último gera uma grelha nos

moldes e definições que são tidos como ideais pela empresa produtora do software, não oferecendo meios simples para o refino ou alteração da constituição das malhas geradas. Portanto a comparação do programa de Grelhas com este tipo de sistema irá, naturalmente, evidenciar diferenças nos resultados.

Como vê-se anteriormente, o sistema Grelhas oferece resultados precisos e confiáveis (se comparado com o GPLAN3). Portanto, as variações entre o sistema GRELHAS e os demais comerciais deverão ser entendidas como diferenças causadas pelos diferentes métodos utilizados e condições de contorno atribuídas a esses métodos.

A título de curiosidade apresenta-se neste capítulo a mesma estrutura que foi calculada no Eberick e no sistema Grelhas. A forma é apresentada na Figura 79.

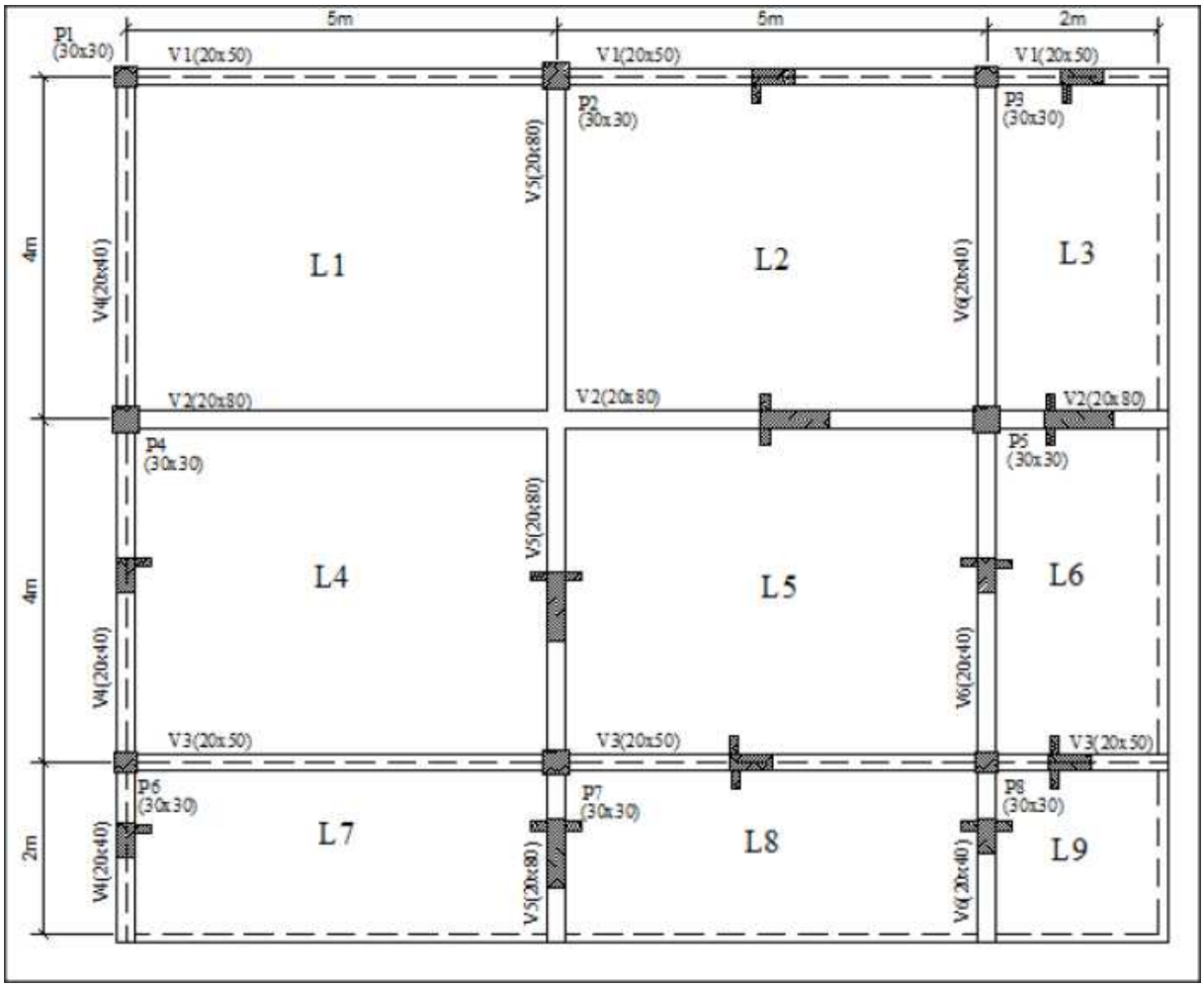


Figura 79: Forma da estrutura.

O Eberick gerou uma grelha equivalente cujos elementos nem sempre têm tamanho constante, provavelmente para harmonizar as grelhas de cada laje do pavimento. A grelha deformada é apresentada na Figura 80. Apesar da grelha ter conformação diferente, pode-se perceber a semelhança entre os eixos da grelha. A diferença entre as deformações ficou em menos de 10% nos nós 1, 2 e 3 mostrados na Figura 80. Isso, claro, mesmo tomando-se em conta os fatoramentos de carga e demais condições de contorno do Eberick. Os resultados são apresentados na Tabela 20.

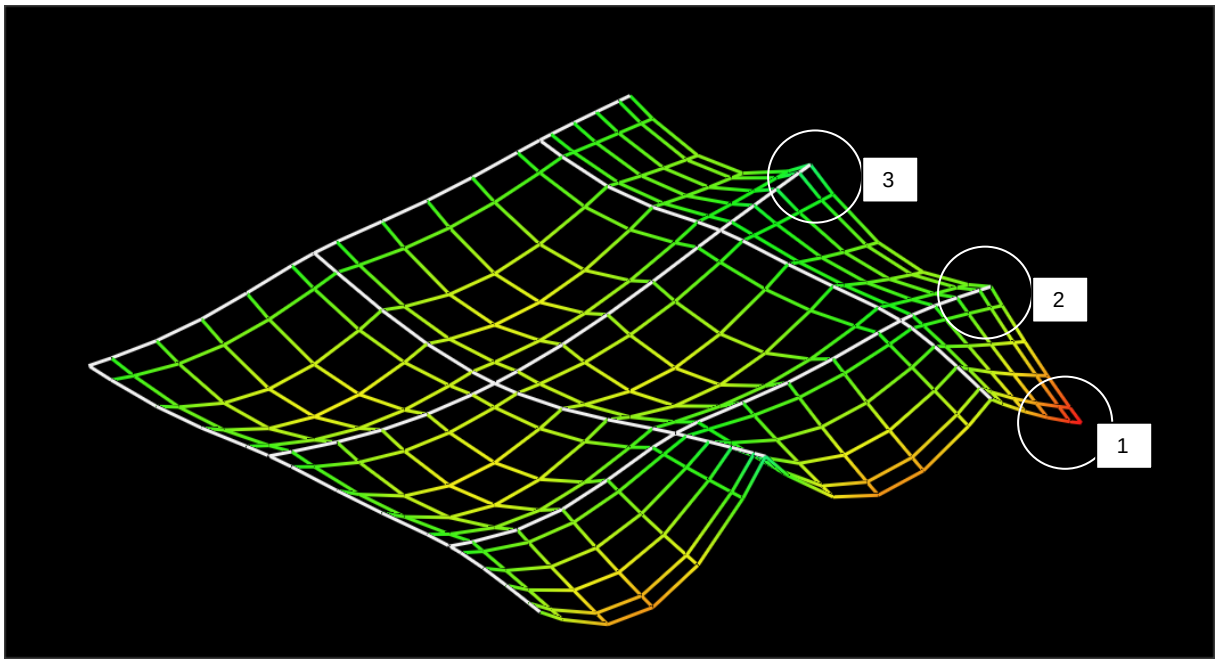


Figura 80: Grelha deformada gerada pelo Eberick.

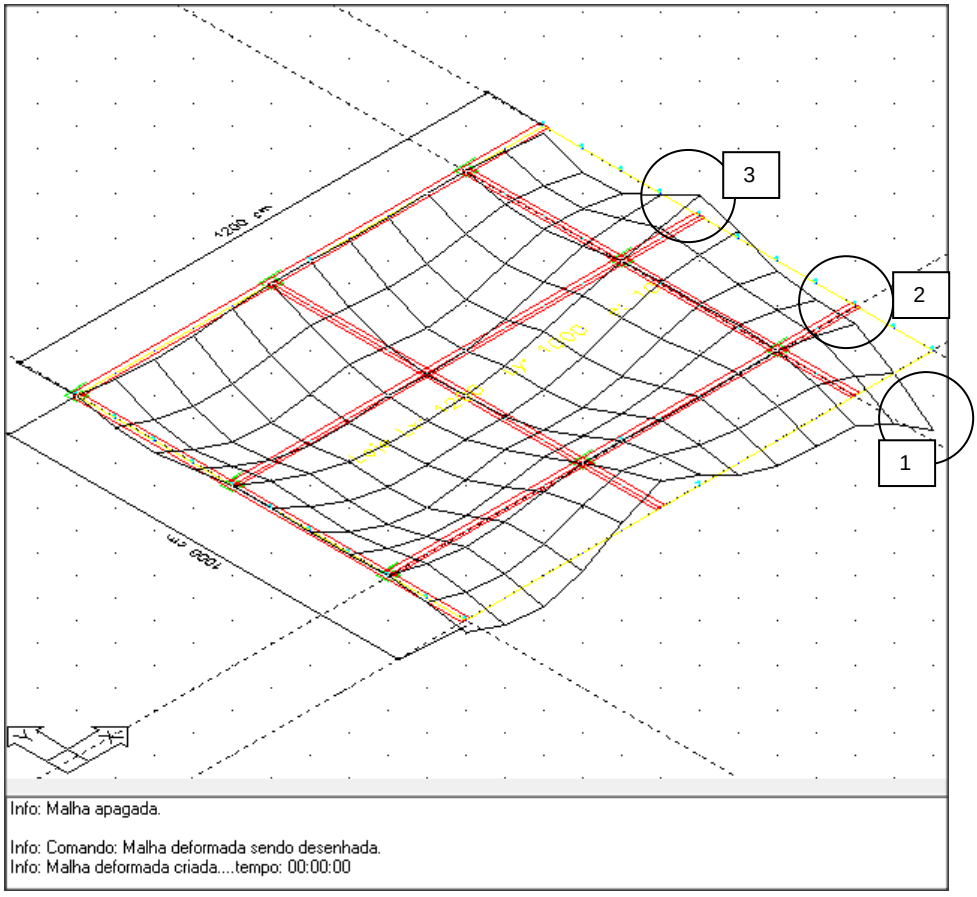


Figura 81: Grelha deformada gerada pelo programa Grelhas.

Tabela 20: Comparação entre o Eberick e o programa Grelhas nos pontos 1, 2 e 3.

Sistema	Nó	Dz (cm)
Eberick	1	<1.34
GRELHA	1	-1.33711
Erro Abs.	0.00216	
Erro rel.	0.00289	
Eberick	2	-0.3
GRELHA	2	-0.21242
Erro Abs.	0.08758	
Erro rel.	0.4123	
Eberick	3	0.31
GRELHA	3	0.3237
Erro Abs.	0.0137	
Erro Rel.	0.0423	

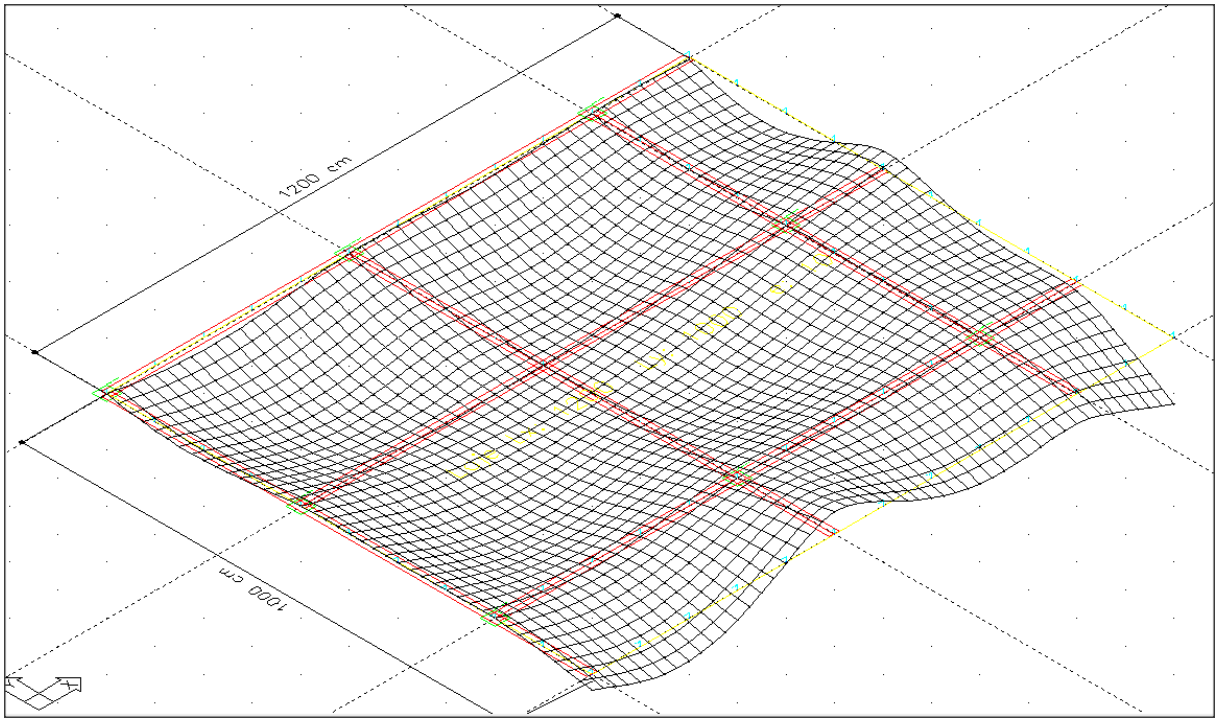


Figura 82: Estrutura com malha refinada.

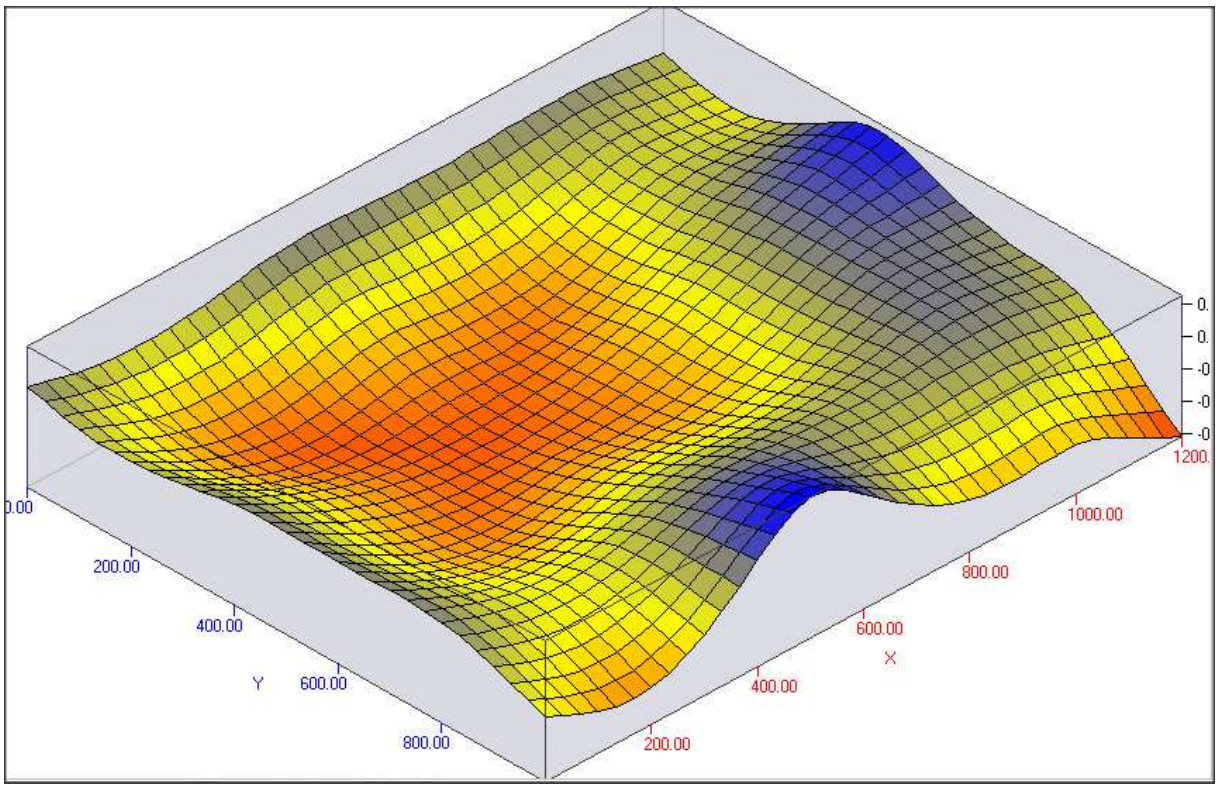


Figura 83: Malha refinada renderizada.

8 CONCLUSÕES E CONSIDERAÇÕES FINAIS

8 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Primeiramente pode-se afirmar que os objetivos apresentados inicialmente neste trabalho foram amplamente alcançados, ou seja, foi desenvolvida uma plataforma gráfica de prototipagem de grelhas que é acoplada com o módulo de cálculo inicialmente desenvolvido por (Cotta, 2006). Esta plataforma já está sendo utilizada, em fase de teste, pelos alunos de graduação de engenharia civil da UFSCar e brevemente será disponibilizada pela internet.

A plataforma gera arquivos completos com os elementos necessários para o cálculo de grelhas equivalentes, fornecendo um rol completo de informações a respeito dos componentes estruturais, permitindo a navegação visual panorâmica da estrutura e a exportação dos dados gráficos em formatos usuais como o ‘DWG’ e o ‘DXF’.

Observa-se também que, praticamente, os objetivos complementares foram alcançados, como:

1. Por meio da remodelagem da plataforma de cálculo pode-se processar um número quase ilimitado de elementos estruturais;
2. Foi desenvolvido um pós-processador de resultados (esforços e deslocamentos) que facilita o entendimento do comportamento das grelhas sob o efeito de um conjunto pré-estabelecido de ações, mostrando, inclusive, o pavimento deformado renderizado;
3. O módulo de cálculo foi melhorado sendo aumentada a velocidade de processamento, o limite do tamanho da grelha e a capacidade de resolução de grelhas sob o efeito de diversas combinações de carregamentos simultâneos;
4. Pode-se considerar o monolitismo dos pavimentos de concreto armado;
5. O programa já está sendo utilizado como ferramenta auxiliar de ensino de estruturas e em cursos de especialização na UFSCar;
6. Finalmente, entende-se que, após a disponibilização na internet, esta ferramenta será de grande utilidade para os profissionais que ainda necessitem de uma ferramenta gratuita.

Como visto nos resultados apresentados no capítulo anterior, o programa apresentado para esta dissertação mostrou-se preciso e confiável em todos os aspectos e em todos os modelos gerados até o momento. Mostrou-se também de

fácil manuseio e bem didático, proporcionando uma forma rápida e eficiente de se analisar pavimentos de concreto. No entanto para que o sistema seja considerado completo, ainda se pode sugerir uma série de melhorias. Apresentam-se na sequência algumas delas.

8.1 MELHORIAS APLICÁVEIS AO SISTEMA

8.1.1 Resolução da Estrutura para Diversos Tipos de Ações de Forma Simultânea

Simplificando fez-se a composição das cargas determinada por uma fórmula que pode ser alterada conforme a vontade do projetista. A fórmula contém as variáveis cp, ca, pp e cl, onde:
cp: é a carga permanente;
ca: é a carga acidental;
pp: é o peso próprio;
cl: é a carga nocal.
Na inicialização de qualquer projeto, a composição de cargas é definida pelo programa, como: ‘pp+ca+cp+cl’. Ao se gerar a carga nodal, o programa resolve a formula e, assim, determina o valor dela antes da resolução do sistema.
A forma correta de se gerarem os deslocamentos seria calculando-se, para cada tipo de carregamento, os deslocamentos parciais relativos a cada ação e, em seguida, aplicar-se a fórmula combinatória para se obter o deslocamento total por uperposição de efeitos. Esta forma de cálculo não onera o sistema, pois a matriz de rigidez é a mesma para o cálculo de todos os deslocamentos parciais.

8.1.2 Consideração de Ações Devidas à Variação de Temperatura

A variação de temperatura induz ações em qualquer estrutura. Esta variação pode ter dois tipos básicos de conformação:

1. A variação constante de temperatura onde as fibras superiores e inferiores de um elemento estrutural têm a mesma variação de temperatura. Neste caso há um alongamento do elemento estrutural que produz uma força axial simétrica.

2. A variação diferencial de temperatura onde as fibras inferiores e superiores de um determinado elemento estrutural têm variações de temperatura diferentes, provocando, além do alongamento dos eixos, a rotação das secções transversais, que podem ser dadas pela expressão da equação 45.

$$d\theta = -\frac{\alpha.(T_1 - T_2).dx}{d} \quad (45)$$

Onde:

α: é a constante de dilatação térmica do material.

T_1 e T_2: são as temperaturas entre as fibras inferior e superior do elemento.

d: é a altura da seção da viga.

8.1.3 Apoios Semi Rígidos

Existem casos onde a placa não está apoiada sobre apoios rígidos, ou seja, que os apoios sofrem uma deformação proporcional aos esforços atuantes. É o caso de fundações do tipo Radier. Neste caso o interessante seria permitir que o programa induza o efeito de mola nos deslocamentos verticais, simulando, assim, vigas sobre bases elásticas. Mostra-se um exemplo de grelha com os apoios sobre base elástica na Figura 84.

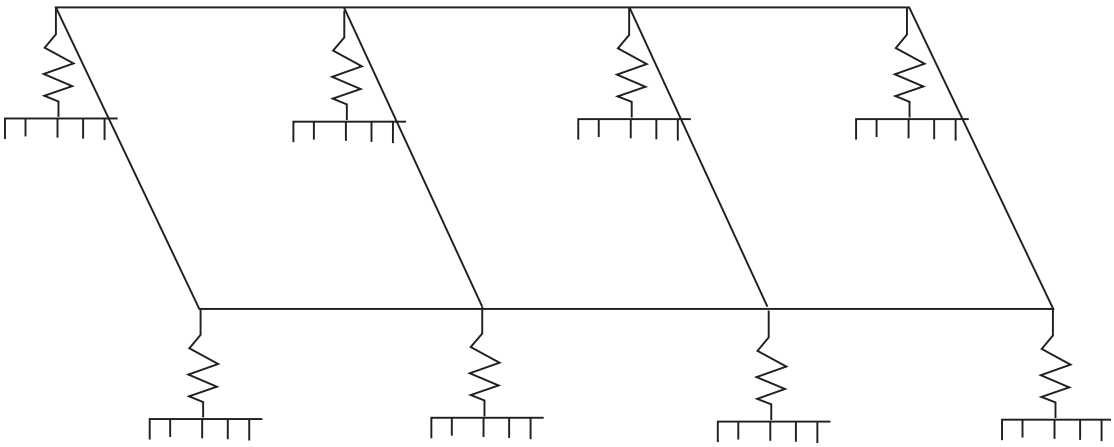


Figura 84: Grelha sobre molas, com deformação linearmente proporcional a uma constante K.

8.1.4 Rótulas em Nós

A introdução de rótulas é necessária para se ter um cálculo mais próximo da realidade. No caso da grelha é interessante a liberação à flexão, pois desta forma pode-se analisar as descontinuidades nos pavimentos. Este cenário é frequente nos casos de pavimentos pré-moldados, onde não se estabelece a continuidade típica do concreto moldado "*in-loco*".

8.1.5 Carregamento Incremental, Consideração do Efeito da Fissuração e Fluência do Concreto

O carregamento incremental é interessante quando se quer analisar a estrutura em um estádio intermediário de carregamento e eventualmente tomar-se em conta a não linearidade das características físicas ou geométricas do concreto.

A não linearidade é uma questão complexa que requer especial atenção, pois se inserem nos cálculos fatores que são alterados de forma recursiva onde, em cada iteração, fazem-se verificações específicas.

Como pode ser visto em (Carvalho, 1994) e (Cotta, 2006), os efeitos da fissuração do concreto é investigado de forma semelhante. Os elementos fissurados têm sua inércia reduzida. A consideração da não linearidade pode ser feita por meio de carregamentos incrementais onde, em cada etapa, faz-se uma análise linear e altera-se a rigidez dos elementos fissurados (quando o momento atuante é maior que o momento de fissuração), substituindo-se a inércia deles pela inércia de Branson. A fluência é outro efeito que pode ser resolvido da mesma forma, mas desta vez considerando-se incrementos de tempo (ou o efeito do tempo na rigidez) no cálculo dos deslocamentos. Mostra-se no fluxograma na Figura 85 o incremento de cargas. Um exemplo do procedimento de substituição do momento de inércia pelo momento de Branson é mostrado na Tabela 31- Anexo 2. O fluxograma proposto por (Anjoletto Filho, 2012) é mostrado na Figura 88 – Anexo 2.

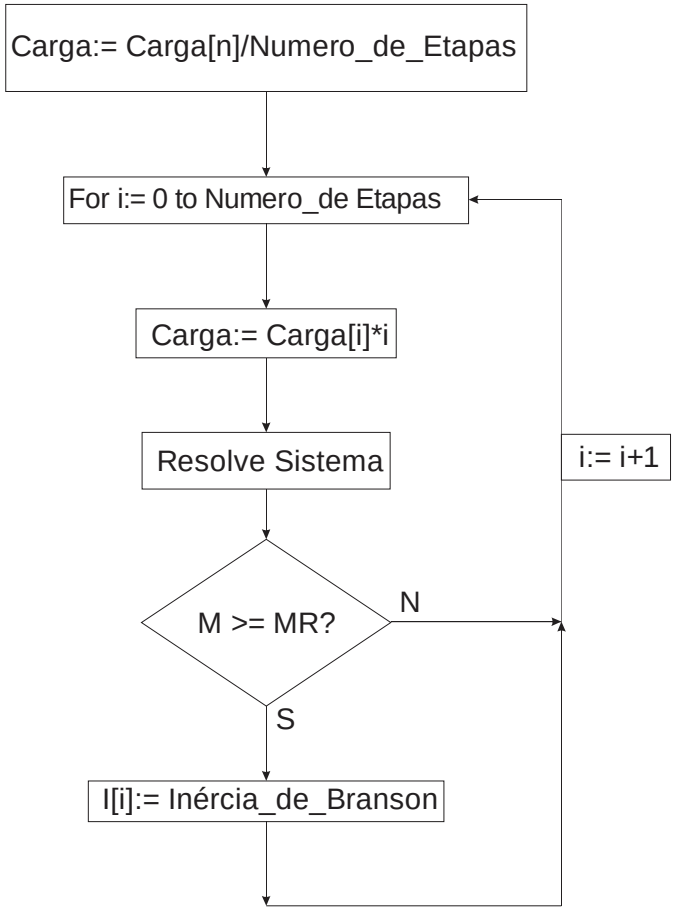


Figura 85: Fluxograma carregamento incremental e verificação do momento atuante em relação ao momento de fissuração.

8.1.6 Efeitos de Protensão e Ações nos Elementos

Para se poder analisar pavimentos com lajes protendidas, deve-se permitir a inserção de ações uniformes devidas aos cabos de protensão.

Como exemplo, pode-se analisar uma laje maciça com dois tramos como a do traçado da Figura 86, que tem uma atuação como a representada na Figura 87.

Para tanto se poderia criar uma rotina na qual o usuário forneceria os dados da curva do cabo e o programa induziria as ações equivalentes, permitindo, assim, que se determine o efeito da protensão na laje.

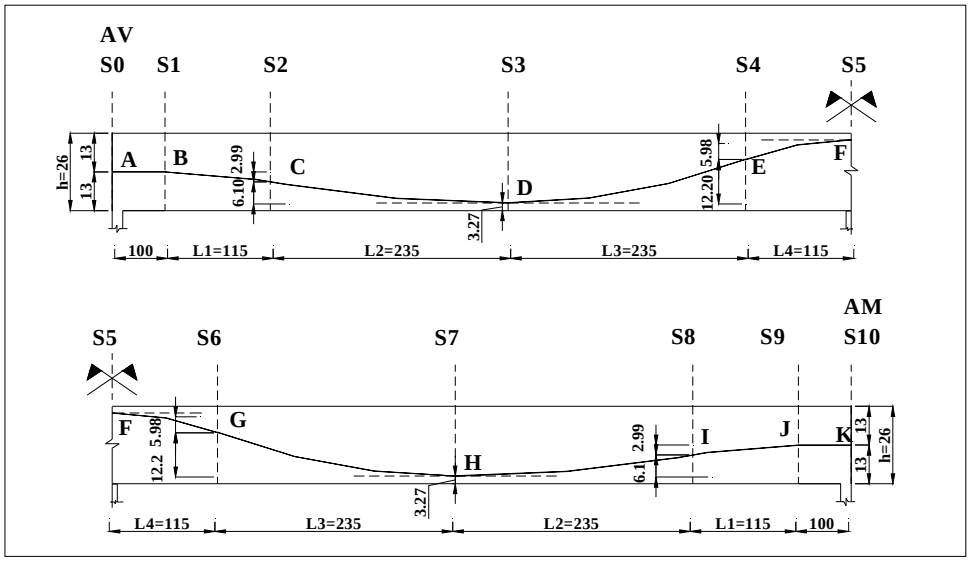


Figura 86: Perfil do traçado de um cabo em uma laje maciça.

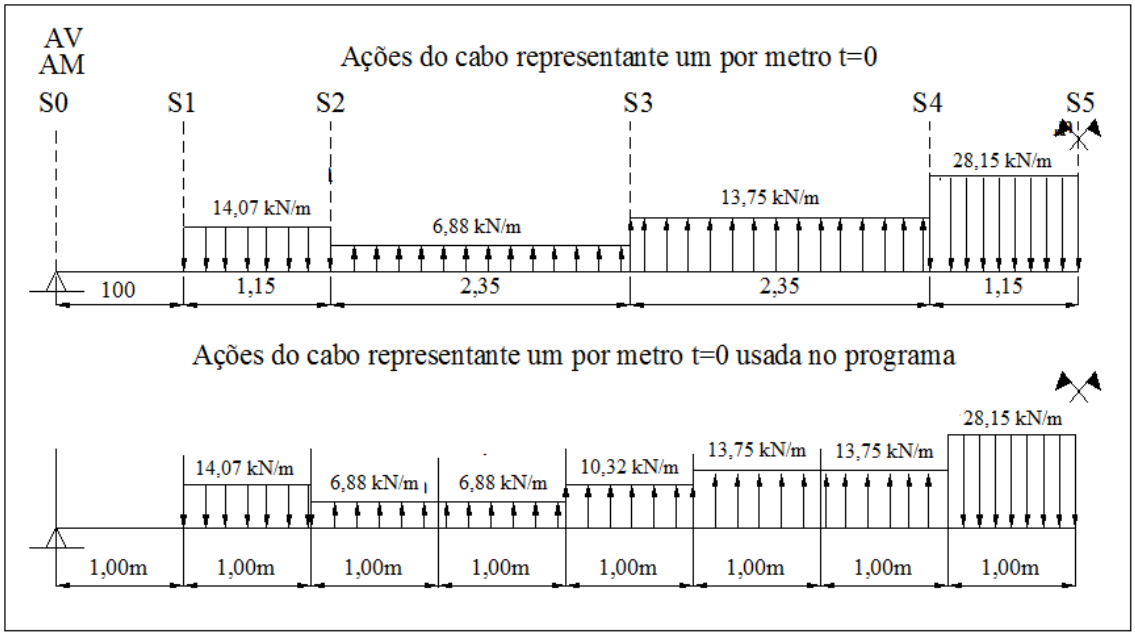


Figura 87: Carregamento equivalente em elementos de grelha devido a protensão.

8.1.7 Eficiência do Gerador de Malhas

Atualmente o gerador de malhas é bastante limitado. De fato, basicamente, este gera a malha equivalente por cima da projeção das lajes, tomando-se em conta a extensão de área total delas. Eventuais vazios nas lajes são ignorados.

Além de se tomar em conta as áreas efetivas das lajes, o sistema deveria permitir a inserção, remoção e edição direta de nós e barras. Dessa forma se poderia aumentar a precisão dos resultados em pontos específicos da laje.

Outro ponto importante é a possibilidade de se editarem as propriedades geométricas das barras, o que permitiria a simulação de diferentes materiais, imperfeições ou casos particulares em pontos específicos da malha.

9 BIBLIOGRAFIA

9 BIBLIOGRAFIA

Anjoletto Filho, Milton César. 2012. *Otimização de um programa de grelha equivalente do sistema CALCO para resolução de pavimentos de Concreto Armado.* DECiv, UFSCar. São Carlos : s.n., 2012. Relatório de iniciação científica.

Arenales, Selma Helena de Vasconcelos e Salvador, José Antônio. 2010. *Cálculo Numérico.* São Carlos : EDUFSCar, 2010. 978-85-7600-187-4.

Bank, R. E. 2012. *Mesh User's Guide.* La Jolla, California : National Science Foundation, 2012.

Bataiolla, André Luiz. 2006. Apostila do Curso de Computação Gráfica. *Apostila do Curso de Computação Gráfica.* São Carlos : EDUFSCar, 2006. Vol. 1.

Beer, Ferdinand P. e Johnston Jr., Russel E. 1980. *Mecânica Vetorial para Engenheiros.* São Paulo : McGraw-Hill - Brasil, 1980. Vol. 1. 80-1010.

Branson, D. E. 1968. Deflections of Reinforced Concrete Flexural Members. *Journal of American Concrete Institute.* 1968.

Carvalho, R. C. 1994. *Análise Não Linear de Pavimentos de Edifícios de Concreto Através da Analogia de Grelha.* 1994.

Carvalho, Roberto Chust e Fiqueiredo Filho, Jasson Rodrigues de. 2013. *Cálculo e Detalhamento de Estruturas Usuais de Concreto Armado.* 5a. São Carlos : EdUFSCar, 2013. Vol. I. 978-85-7600-086-0.

Carvalho, Roberto Chust e Pinheiro, Libânio Miranda. 2013. *Cálculo e Detalhamento de Estruturas Usuais de Concreto Armado.* São Paulo : PINI, 2013. Vol. 2. 978-85-7266-276-5.

Chuang, Ming-Chieh. 2006. Development Of An Object-Oriented Graphical User Interface For The Structural Analisys Program. *4th International Conference on Earthquake Engiineering.* 12-13 de 10 de 2006, p. 8.

Clèment, Jasmin, et al. 2013. Framework For Delaunay Mesh Generation. *INRIA - Informatics Mathematics.* 1 Março 2013, 2013, p. 31.

Colaborative Commons. 2014. *Code With C.* [Online] Code With C Team, 2014. http://www.codewithc.com/c-program-for-gauss-elimination-method/.

—. 2009. Regular Grid. *Wikipédia.* [Online] Wikipedia, 12 de 2009. http://en.wikipedia.org/wiki/Regular_grid.

Corrêa. 1987. *Sistema laser de análise estrutral.* São Paulo : s.n., 1987.

Cotta, I. S. 2006. Desenvolvimento de programa livre automático para a determinação de esforços solicitantes, deslocamentos e armadura de pavimentos em concreto armado usando a analogia de grelha não linear. São Carlos : UFSCar, 2006.

Darkov, A., et al. 1983. *Structural Mechanics.* 4th. Moscou : MIR, 1983.

Devloo, Philippe Remy Bernard e Alves Filho, José Sérgio Rodrigues. 1991. Object Oriented programming In Scientific Computations: The Beginning Of a New Era. *Engineering Computations.* 1991.

Drummond, Fabiana Paula. 2012. Desenvolvimento de Software de Análise Estrutural de Sistemas Reticulados Espaciais Usando o Método dos Deslocamentos. *Exacta.* 2012.

Ferreira, Fernando Luís e Santos, João. 2002. *Programação em AutoCAD.* Lisboa : FCA, 2002. 972-722-208-0.

Flòrio, M. C., et al. 2003. Flecha em lajes com vigotas pré-moldadas considerando a fissuração e uso da Expressão de Branson. *45 Congresso Brasileiro do Concreto.* Vitória : IBRACON, 2003.

Gere, James M. e Weaver, William Jr. 1987. *Análise de Estruturas Reticuladas.* Rio de Janeiro : Editora Guanabara S.A., 1987.

Hambly, E. C. 1976. *Bridge Deck Behaviour.* London : Chapman and Hall, 1976.

Hoffman, C. M. 2001. *Robustness In Geometric Computations.* 2001.

JEDI. 1997. Project Jedi Portal. *Project Jedi.* [Online] Joint Endeavour of Delphi Inovators, 1997. [Citado em: 2015 de 03 de 10.] http://www.delphi-jedi.org/.

Keith, Weiskamp e Loren, Heiny. 1992. *Programação Gráfica em Turbo Pascal 6.* Rio de Janeiro : Editora Ciência Moderna, 1992. 0-471-54736-0.

Lischke, Mike. 2006. GLScene. *GLScene OpenGl Solution for PASCAL.* [Online] 2006. http://glscene.sourceforge.net/wikka/HomePage.

Lohninger. 2004. Epina Software Labs. *SDL Component Suite.* [Online] Epina, 2004. [Citado em: 10 de 03 de 2015.] http://www.lohninger.com/index.html.

Martha, Luiz Fernando. 2010. *Análise de Estruturas - Conceiros Básicos.* 12. Rio de Janeiro : Elsevier, 2010. p. 524. 9788535234558.

Mukhin, N. V., Pershin, A. N. e Shishman, B. A. 1983. *Statics of Structures.* Moscou : MIR, 1983.

Müller, Marina F. 2011. A Interoperabilidade Entre Sistemas CAD de Projeto de Estruturas de Concreto Armado Baseada em Arquivos IFC. Curitiba : UFP, 2011.

Pissarenko, G. S., Lakovlev, A. P. e Matveiev, V. V. 1985. *Prontuário de Resistência dos Materiais.* Moscou : MIR, 1985.

Pizzolato, Ednaldo Brigante. 2010. *Introdução à Programação Orientada a Objetos com C++ e Java .* São Carlos : EDUFSCar, 2010. 978-85-7600-204-8.

Pressman, Roger S. 2001. *Software Engineering a Practitioner's Approach.* New York : Mc Graw Hill, 2001. 0073655783.

Raymundo, Henrique. 2008. *Programa para representação da forma de pavimentos de concreto, geração de dados correspondentes para programa de cálculo de estruturas prismáticas e representação da estrutura deformada e esforços solicitantes.* São Carlos : s.n., 2008. Iniciação Científica. Processo Fapesp nº 06/54378-0.

Thompson, Joe F. 1998. *Grid Generation.* [ed.] Weatherill editors. New York : CRC Press, 1998. p. 1096.

Timoshenko, Stephen P. e Gere, James E. 1982. *Mecânica dos Sólidos.* Rio de Janeiro : Livros Técnicos e Científicos S.A., 1982. Vol. 1 e 2. 85-216-0246-4.

Vaz, Luiz Eloy. 2014. *Método Dos Elementos Finitos Em Análise De Estruturas.* s.l. : Campus, 2014. 978-85-352-3929-4.

Wilczynska, D. 2010. *Direction of Research Into Methods of Defuzzification.* 2010.

10 ANEXOS

10 ANEXOS

10.1 ANEXO 1 - PROCEDIMENTOS RELEVANTES DOS MÓDULOS GRÁFICOS E DE CÁLCULO

Procedimentos das unidade de cálculo e gráfica:

procedimento **Ler_Do_Arquivo:**

Tabela 21: Procedimento de leitura de dados.

```
Procedure ler_do_arquivo;
Var i,j:integer;
   titulo: textfile;
Begin
   Assignfile(titulo, inFileName);
   Reset(titulo);
   Readln(titulo,carac_dif);
   For i:=1 to carac_dif do
   Begin
    Read(titulo,geometria_material[i].E1);
    Read(titulo,geometria_material[i].I1);
    Read(titulo,geometria_material[i].G1);
    Read(titulo,geometria_material[i].J1);
    Readln(titulo);
   End;
   Read(titulo,numeno,numebar);
   Readln(titulo);
   For i:=1 to numeno do
   Begin
    Read(titulo,X[i]);
    Read(titulo,Y[i]);
    Readln(titulo);
   End;
   For i:=1 to numebar do
   Begin
    Read(titulo,noinicial[i]);     // Leitura do nó inicial
    Read(titulo,nofinal[i]);       // Leitura do nó final
    Read(titulo,elemento[i].Class_tipo); // Leitura da classe do elemento

    elemento[i].L:= Comprimento(NOINICIAL[i],NOFINAL[i],X,Y);
    // Cálculo do comprimento do elemento

    elemento[i].cos:= (X[NOFINAL[i]]-X[NOINICIAL[i]])/elemento[i].L;
    elemento[i].sen:= (Y[NOFINAL[i]]-Y[NOINICIAL[i]])/elemento[i].L;
    Readln(titulo);
   End;

   For i:=1 to numeno do
   Begin                  // Leitura dos carregamentos nodais
```

```
    Read(titulo,F[3*i-2]);
    Read(titulo,F[3*i-1]);
    Read(titulo,F[3*i]);
    Readln(titulo);
  End;

  For i:=1 to numeno do
  Begin      // Leitura da vinculação
    Read(titulo,XX[3*i-2]);
    Read(titulo,XX[3*i-1]);
    Read(titulo,XX[3*i]);
    Readln(titulo);
  End;
  close(titulo);

  //Writeln('Final da leitura dos dados. ');
  //Readln;

End;
```

Procedimento **Rigidez_Global**:

Tabela 22: Geração da matriz de rigidez.

```
Procedure Rigidez_global(NUMENO, NUMEBAR:integer;R, Rtrans, Rig, Rig1: matriz2;
                  NOINICIAL,NOFINAL:Pont2;var SMG:matriz;
                  elemento:registro1; geometria_material: registro2);

Var n,i,j:integer;

Begin
  //For i:=1 to 3*numeno do
  //For j:=1 to 3*numeno do
  //SMG[i,j]:=0;
  FillChar(SMG, SizeOf(SMG),0);
 { SetLength(SMG, VectorSize, VectorSize);}
  For n:=1 to numebar do
  {Laço para percorrer todas as barras da estrutura}

  Begin
    matriz_rotacao(n,elemento,R);
    transposta(R,Rtrans);

    Rigidez_do_elemento(n,geometria_material,elemento,Rig);

    multiplica_matriz_matriz(Rtrans,Rig,Rig1,6,6);
    multiplica_matriz_matriz(Rig1,R,Rig,6,6);

    For i:= 1 to 6 do
    For j:= 1 to 6 do
```

```
      Begin
       if ((i<3)or (i=3)) and ((j<3)or(j=3)) then
       SMG[3*NOINICIAL[n]-3+i,3*NOINICIAL[n]-3+j]:=Rig[i,j]+SMG[3*NOINICIAL[n]-3+i,3*NOINICIAL[n]-3+j];
       if (i>3) and (j>3) then
       SMG[3*NOFINAL[n]-6+i,3*NOFINAL[n]-6+j]:=SMG[3*NOFINAL[n]-6+i,3*NOFINAL[n]-6+j]+Rig[i,j];
       if (i>3) and ((j<3)or(j=3))  then
       SMG[3*NOFINAL[n]-6+i,3*NOINICIAL[n]-3+j]:=SMG[3*NOFINAL[n]-6+i,3*NOINICIAL[n]-3+j]+Rig[i,j];
       if ((i<3)or(i=3)) and (j>3) then
       SMG[3*NOINICIAL[n]-3+i,3*NOFINAL[n]-6+j]:=SMG[3*NOINICIAL[n]-3+i,3*NOFINAL[n]-6+j]+Rig[i,j];
      End;
    End;
End;
```

procedimento **Rigidez_do_Elemento**:

Tabela 23: Rigidez do elemento de barra.

```
Procedure Rigidez_do_elemento(k:integer;geom_mat:registro2;elemento:registro1;
        var SML:matriz2);

{Este procedimento irá fornecer a matriz de rigidez do Elemento}
{E = módulo de elasticidade longitudinal, I = momento de inércia à flexào,
 G = módulo de elasticidade transveral, J= momento de inércia à torsão}

Var r1,r2,r3,r4,r5:real;
   i,j:integer; {contadores}

Begin
  r1:= geometria_material[elemento[k].Class_tipo].G1*geometria_material[elemento[k].Class_tipo].J1/elemento[k].L;
  r2:= 4*geometria_material[elemento[k].Class_tipo].E1*geometria_material[elemento[k].Class_tipo].I1/elemento[k].L;
  r3:=
6*geometria_material[elemento[k].Class_tipo].E1*geometria_material[elemento[k].Class_tipo].I1/(elemento[k].L*elem
ento[k].L);
  r4:= 2*geometria_material[elemento[k].Class_tipo].E1*geometria_material[elemento[k].Class_tipo].I1/elemento[k].L;
  r5:=
12*geometria_material[elemento[k].Class_tipo].E1*geometria_material[elemento[k].Class_tipo].I1/(elemento[k].L*ele
mento[k].L*elemento[k].L);

  SML[1,1]:= r1;
  SML[1,4]:=-r1;
  SML[2,2]:= r2;
  SML[2,3]:=-r3;
  SML[2,5]:= r4;
  SML[2,6]:= r3;
  SML[3,3]:= r5;
  SML[3,5]:=-r3;
  SML[3,6]:=-r5;
  SML[4,4]:= r1;
  SML[5,5]:= r2;
  SML[5,6]:= r3;
```

```
  SML[6,6]:= r5;

  SML[1,2]:=0;SML[1,3]:=0;SML[1,5]:=0;SML[1,6]:=0;
  SML[2,4]:=0;SML[3,4]:=0;SML[4,5]:=0;SML[4,6]:=0;

  For i:=1 to 6 do
    For j:=i to 6 do
    SML[j,i]:= SML[i,j];

End;
```

procedimento **Transposta**:

Tabela 24: Transposta da matriz de rigidez local.

```
Procedure transposta(R:matriz2;var Rtrans:matriz2);

{Procedimento para transposição de matriz}

Var i,j:integer;
Begin
  For i:=1 to 6 do
    For j:= 1 to 6 do
    Rtrans[j,i]:=R[i,j];
End;
```

Tabela 25: Matriz de rotação.

```
Procedure matriz_rotacao(k:integer; elemento: registro1; var R:matriz2);

{Procedimento para calcular a matriz de rotação do elemento}

Var cx,cy: real;
    i,j:integer;
Begin
    cx:=elemento[k].cos;
    cy:=elemento[k].sen;

    For i:=1 to 6 do
    For j:=1 to 6 do
    R[i,j]:=0;
    R[1,1]:= cx;
    R[1,2]:= cy;
    R[2,1]:=-cy;
    R[2,2]:= cx;
    R[3,3]:= 1;
    For i:=4 to 6 do
      For j:= 4 to 6 do
      R[i,j]:= R[i-3,j-3];
End
```

procedimentos **Multiplica_Matriz_Matriz e Matriz_Vetor**

Tabela 26: Procedimentos multiplica matriz x matriz e matriz x vetor.

```
Procedure multiplica_matriz_matriz(A,B:matriz2; Var C:matriz2;n,m:integer);

{Procedimento para multiplicação de matriz}

Var   k:integer;
      i,j:integer;
Begin
  For i:=1 to n do
    For j:=1 to m do
    Begin
      C[i,j]:=0;
      For k:=1 to m do
      C[i,j]:=C[i,j]+A[i,k]*B[k,j];
    End;
End;

{------------------------------------------------------------------------}
Procedure multiplica_matriz_vetor(A:matriz2;B:pont1; Var C:pont1;n,m:integer);

Var   i,k:integer;          {Procedimento para multiplicação de matriz por vetor}
Begin
  For i:=1 to n do
    Begin
      C[i]:=0;
      For k:=1 to m do
      C[i]:=C[i]+A[i,k]*B[k];
    End;
End;
```

procedimento **Gauss**:

Tabela 27: Resolução do sistema de equações pelo método de eliminação de Gauss.

```
procedure Gauss({A:matriz;} var Deslocamento:pont1; B:pont1; XX:pont1; numeno:integer);

{Cálculo dos deslocamentos}
var i,k,j,l:integer;
    p,prov:real;
    X_prov:pont1;

Begin
  //1a etapa do calculo do deslocamento
 { SetLength(Deslocamento, 3*VectorSize);
  SetLength(B, 3*VectorSize);
  SetLength(XX, 3*VectorSize);
  SetLength(X_prov, 3*VectorSize);}
  For k:=1 to 3*numeno-1 do
```

```
  Begin
    For i:=(k+1) to 3*numeno do
    Begin
      p:={A}SMG[i,k]/{A}SMG[k,k];
      {A}SMG[i,k]:=0;
      For j:=(k+1) to 3*numeno do
      {A}SMG[i,j]:={A}SMG[i,j]-p*{A}SMG[k,j];
      B[i]:=B[i]-p*B[k];
    End;
  End;

  //2a etapa do calculo do deslocamento

  X_prov[3*numeno]:=B[3*numeno]/{A}SMG[3*numeno,3*numeno];
  For l:=(3*numeno-1) downto 1 do
  Begin
    prov:=0;
    For j:=(l+1) to 3*numeno do
    prov:=prov+{A}SMG[l,j]*X_prov[j]/{A}SMG[l,l];
    X_prov[l]:=B[l]/{A}SMG[l,l]-prov;
  end;
  For i:=1 to 3*numeno do
  Begin
  Deslocamento[i]:=X_prov[i]+Deslocamento[i];
  End;
End;
```

Procedimento **CreateMesh**

Tabela 28: Procedimento para criar a malha equivalente.

```
procedure CreateMesh(x1,y1,x2,y2: Extended);
var
  hEnt, Cols, Rows, i, j: Integer;
  dx, dy, px1, py1, px2, py2: Extended;
  iNo, iBar: Word;
  sNo, sBar: string;
  fBar: TBarData;
  fNode: TNodeData;
  hPoints_Id: array of THandle;
begin
  dx:= ((X2-X1)/ProjExtData.G_dX);
  dy:= ((y2-y1)/ProjExtData.G_dY);
  ProjExtData.dX:= dx; ProjExtData.dY:= dy;
  iBar:=0; iNo:= 0; px1:=0; px2:=0; py1:=0; py2:=0;
  nBarras:= 0; nNos:= 0; //Variáveis globais
  For j:= 0 to ProjExtData.G_dY do begin //Points generator and sets hPoints_Id[i] vector of entity id's

    For i := 0 to ProjExtData.G_dX do begin // Make Nodes
      FillChar(fNode, SizeOf(fNode), 0);
      // Point marks generator and numerator
      px1:= x1+(dX)*i; py1:= y1+(dy)*j; px2:= x1+(dX)*(i+1); py2:= y1+(dY)*(j+1);
      hEnt:= CadAddPoint(hDwg, px1, py1, 0);
      CadEntityPutUserData(hEnt, MESH_USR_NODE);          //Put user data at line entity with MESH_USR_ID value
      CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA);
      Inc(iNo);
      sNo:= '_'+IntToStr(iNo);
```

```
        SetLength(hPoints_Id, iNo); //Redim Array of point entity handle
        hPoints_Id[iNo-1]:= CadEntityGetID(hEnt); //Add handle of entity to array
        fNode.hN1:= hPoints_Id[iNo-1]; //save handle to node extended data
        //fNode.P1.X:= px1; fNode.P1.Y:= py1;
        fNode.X:= px1; fNode.Y:= py1;
        fNode.iN1:= iNo;
        fNode.mx:= 0; fNode.my:= 0; fNode.nz:= 0;
        fNode.rx:= True; fNode.ry:= True; fNode.rz:= True;
        fNode.N:= iNo;
        fNode.Pos:= ntInterno;
        if (j = 0) or (i = 0) or (i = ProjExtData.G_dX) or (j = ProjExtData.G_dY) then begin
          fNode.Pos:= ntlateral;
          if ((j = 0)and(i =0)) or ((j = 0)and(i = ProjExtData.G_dX)) or ((j = ProjExtData.G_dY)and(i=0)) or
            ((j = ProjExtData.G_dY)and(i = ProjExtData.G_dX))then begin
              fNode.Pos:= ntCanto;
            end;
        end;
        CadEntityPutExData(hEnt, @fNode, SizeOf(TNodeData)); //Put Extended data into Point entity
        hEnt:= CadAddText(hDwg, PCStr(sNo), px1, py1, 0);
        CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA);
        CadTextPutAngle(hEnt, 45);
        CadTextPutHeight(hEnt, ProjExtData.dY/10);
      end;
    end;

    For j:= 0 to ProjExtData.G_dY do begin

      For i := 0 to ProjExtData.G_dX-1 do begin // Make horizontal bars
        FillChar(fBar, SizeOf(fBar), 0);
        px1:= x1+(dX)*i; py1:= y1+(dy)*j; px2:= x1+(dX)*(i+1); py2:= y1+(dY)*(j+1); // calculates point 1 and point 2 of
each bar
        hEnt:= CadAddLine(hDwg, px1, py1, 0, px2, py1, 0); // creates a line between those points
        CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA);       //Set line to magenta color
        CadEntityPutUserData(hEnt, MESH_USR_BAR);         //Put user data at line entuity with MESH_USR_ID value
        iBar:= iBar+1; sBar:= 'B'+IntToStr(iBar);       // Counts the number of bars and creates the bar label;

        //Formats the extended data for each line
        fBar.iBar:= iBar;
        fBar.P1.X:= px1; fBar.P1.Y:= py1; fBar.P2.X:= px2; fBar.P2.Y:= py1; // coordinates
        fBar.iN1:= (i)+(j*(ProjExtData.G_dX+1))+1; //Node 1
        fBar.iN2:= (fBar.iN1+1); //Node 2
        fBar.hN1:= hPoints_Id[fBar.iN1-1]; //saves bar Point 1 id handle to bar data
        fBar.hN2:= hPoints_Id[fBar.iN2-1]; // saves bar point 2 id handle to bar data
        fBar.hB:= CadEntityGetID(hEnt);
        fBar.Orientation:= GetOrientation(fBar); //pega orientação da barra
        fBar.BarType:= GetBarType(fBar); // pega tippo de barra

        CadEntityPutExData(hEnt, @fBar, SizeOf(TBarData)); //Put Extended data into line entity

        hEnt:= CadAddText(hDwg, PCStr(sBar), px1+(px2-px1)/2, py1, 0); //Adds label to the bar
        CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA); // label color to magenta
        CadTextPutAlign(hEnt, CAD_TA_MIDCENTER);    //Define labels aligment
        CadTextPutHeight(hEnt, ProjExtData.dX/10);
      end;

      if j<ProjExtData.G_dY then
        For i := 0 to ProjExtData.G_dX do begin //Make vertical bars
          FillChar(fBar, SizeOf(fBar), 0);
          px1:= x1+(dX)*i; py1:= y1+(dy)*j; px2:= x1+(dX)*(i+1); py2:= y1+(dY)*(j+1); // Calculates coords
          hEnt:= CadAddLine(hDwg, px1, py1, 0, px1, py2, 0); // add line entity
          CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA);    // define color
          CadEntityPutUserData(hEnt, MESH_USR_BAR);    // put user data to line entity
          iBar:= iBar+1; sBar:= 'B'+IntToStr(iBar); // Count bars and create bar label

          //Formats the extended data for each line
          fBar.iBar:= iBar;
          fBar.P1.X:= px1; fBar.P1.Y:= py1; fBar.P2.X:= px1; fBar.P2.Y:= py2; // coordinates
          fBar.iN1:= (i)+(j*(ProjExtData.G_dX+1))+1; //Node 1
          fBar.iN2:= fBar.iN1+(ProjExtData.G_dX+1); //Node 2
          fBar.hN1:= hPoints_Id[fBar.iN1-1];
          fBar.hN2:= hPoints_Id[fBar.iN2-1];
          fBar.hB:= CadEntityGetID(hEnt);
          fBar.Orientation:= GetOrientation(fBar); // pega orientação da barra
```

```
        fBar.BarType:= GetBarType(fBar);  //pega tipo de barra

        CadEntityPutExData(hEnt, @fBar, SizeOf(TBarData)); //Put Extended data into line

        hEnt:= CadAddText(hDwg, PCStr(sBar), px1, py1+(py2-py1)/2, 0); //add label to bar
        CadEntityPutColor(hEnt, CAD_COLOR_MAGENTA);   // define label color
        CadTextPutAlign(hEnt, CAD_TA_MIDCENTER);     // define label aligment
        CadTextPutAngle(hEnt, 90);              //define label angle rotation
        CadTextPutHeight(hEnt, ProjExtData.dY/10);
      end;
      Application.ProcessMessages;
  end;
   nBarras:= iBar; //Stores Total bar number to global var
   nNos:= iNo;     //Strores total nodes number to global var
   HistAdd('Malha gerada com ' + IntToStr(nBarras) +' Barras e '+IntToStr(nNos)+' nós');
   SetLength(hPoints_Id, 0); //Close hPoints_Id data and frees memory;
   CadUpdate( hDwg );
   CadWndRedraw( CadGetWindow( hDwg ) );
 end;
```

Procedimento **CreateDefMesh** (criação da malha Equivalente Deformada)

Tabela 29: Procedimento para a criação da malha deformada.

```
 procedure CreateDefMesh;
 var
   sInFile: TFileName;
   Rotx, Roty, Deslz, Factor, MT, MF, V : Double;
   i, Node: Integer;
   sLine: string;
   N1,N2: TNodeData;
   B: TBarData;
   Dados: TStringList;

   procedure ReadBlancs(Lines: Byte);
   var
     i: Integer;
   begin
    for i:= 1 to Lines do
     ReadLn(fInFile);
   end;

 begin
  DecimalSeparator:= '.'; //Antes da leitura do arquivo dizer ao sistema que o separador decimal é o ponto (.)
  Factor:= ProjExtData.DefFactor;
  sInFile:= ExtractFilePath(Application.ExeName)+OutFilename;
  if FileExists(sInFile) then begin
    Dados:= TStringList.Create; // cria lista de dados
    AssignFile(fInFile, sInFile);
    Reset(fInFile);
    ReadBlancs(6); // descarta as 6 primeiras linhas do arq.
    CadSetCurLayerByName(hDwg,MeshStrDef);
    For i:= 1 to nNos do begin // Aqui lêem-se os dados
      ReadLn(fInFile,sLine);
      TextLineParser(sLine, Dados); // separa os argumento da linha de arquivo numa lista
      Rotx:= StrToFloat(Dados[1]);
      Roty:= StrToFloat(Dados[2]);
      Deslz:= StrToFloat(Dados[3]);
      N1:= TNodedata(NodeList.Items[i-1]^);//Pega o dado do nó atual na memória
      N1.Rotx:= Rotx; // insere os valores lidos no nó atual
      N1.Roty:= Roty;
      N1.Deslz:= Deslz;
      TNodedata(NodeList.Items[i-1]^):= N1;// coloca as alterações devolta no nó direto na memória
      hEnt := CadGetEntityByID(hDwg,N1.hN1); //Pega endereço do nó
      CadEntityPutExData(hEnt, @N1, SizeOf(N1)); //Coloca os dados no nó
      hEnt:= CadAddPoint(hDwg, N1.X, N1.Y,N1.Deslz*Factor);// fatora a deformação
      //CadEntityPutColor(hEnt,10);
      CadEntityPutUserData(hEnt, MESH_USR_DEFNODE);
      CadEntityPutExData(hEnt, @N1, SizeOf(N1)); //Coloca os dados no nó deformado
    end;
    ReadBlancs(6); //descarta 6 linhas
    For i:= 0 to BarList.Count-1 do begin  //here we plot the bars that must match the nodes
       B:= TBarData(BarList[i]^);
       CadEntityGetExData(CadGetEntityByID(hDwg, B.hB), @B); // pega os dados originais da barra
```

```
        ReadLn(fInFile,sLine);
        TextLineParser(sLine, Dados); // separa os argumento da linha de arquivo numa lista
        MT:= StrToFloat(Dados[2]);//StrToFloat(Trim(Copy(sLine,18,18)));
        MF:= StrToFloat(Dados[3]);//StrToFloat(Trim(Copy(sLine,36,18)));
        V:= StrToFloat(Dados[4]);//StrToFloat(Trim(Copy(sLine,54,18)));
        B.MT1:= MT;
        B.MF1:= MF;
        B.V1:= V;
        ReadLn(fInFile,sLine);
        TextLineParser(sLine, Dados); // separa os argumento da linha de arquivo numa lista
        MT:= StrToFloat(Dados[1]);//StrToFloat(Trim(Copy(sLine,18,18)));
        MF:= StrToFloat(Dados[2]);//StrToFloat(Trim(Copy(sLine,36,18)));
        V:= StrToFloat(Dados[3]);//StrToFloat(Trim(Copy(sLine,54,18)));
        B.MT2:= MT;
        B.MF2:= MF;
        B.V2:= V;
        TBarData(BarList[i]^):= B;
        CadEntityGetExData(CadGetEntityByID(hDwg, B.hN1), @N1);
        CadEntityGetExData(CadGetEntityByID(hDwg, B.hN2),@N2);
        hEnt:= CadAddLine(hDwg,N1.X,N1.Y,N1.Deslz*Factor,N2.X,N2.Y,N2.Deslz*Factor);
        CadEntityPutExData(hEnt, @B, SizeOf(B)); //Coloca os dados na barra deformada
       // CadEntityPutColor(hEnt,10 ); //Color 10 reserved for deformed grid
        CadEntityPutUserData(hEnt, MESH_USR_DEFBAR);
        hEnt:= CadGetEntityByID(hDwg, B.hB); // pega a barra original não def.
        CadEntityPutExData(hEnt, @B, SizeOf(B)); //Coloca os dados na barra original
      end;
     CloseFile(fInFile);
     CadUpdate( hDwg );
     CadWndRedraw( CadGetWindow( hDwg ) );
     if Dados <> nil then Dados.Free; //apaga lista de dados
   end else begin
     HistAdd(STR_ERRO_NAOHARESULTADOS);
   end;
 end;
```

.

ANEXO 2

10.2 ANEXO 2 – REFERÊNCIAS DIVERSAS

Tabela 30: Rotina: Método da eliminação de Gauss (Colaborative Commons, 2014).

```
program gauss_elimination;
uses crt;

const MAXX=4;
const MAXEQNS=MAXX; {MAXEQNS should be = MAXX}
const SIGPLACES=4;
const DECPLACES=4;

var a:array[1..MAXEQNS,1..MAXX+1] of real;
    x:array[1..MAXX] of real;
    i,j,k:integer;
    temp:real;

begin
clrscr;
{write the equations in general form}
for i:=1 to MAXEQNS do
 begin
  for j:=1 to MAXX do
   begin
    write('a',i,j,'.x',j);
    if j < MAXX then write(' + ');
   end;
   writeln(' = b',i);
 end;

{get the values of a[i,j]}
for i:=1 to MAXEQNS do
begin
  for j:=1 to MAXX do
   begin
    write('enter a',i,j,' ');
    readln(a[i,j]);
   end;
   write('enter b',i,'  ');
   readln(a[i,MAXX+1]);
end;

readln;
clrscr;

{write the given eqns}
for i:=1 to MAXEQNS do
 begin
  for j:=1 to MAXX do
   begin
    write(a[i,j]:SIGPLACES:DECPLACES,' x',j);
    if j < MAXX then write(' + ');
   end;
   writeln(' = ',a[i,MAXX+1]:SIGPLACES:DECPLACES);
 end;

{calculate the output matrix}
for i:=1 to MAXEQNS-1 do
 for j:=i+1 to MAXEQNS do
 begin
  temp:=a[j,i];
  for k:=1 to MAXX+1 do
   a[j,k]:=a[j,k] - temp*a[i,k]/a[i,i];
 end;

{calc the values of x1 x2 etc}
x[MAXEQNS] := a[MAXEQNS,MAXX+1] / a[MAXEQNS,MAXX];
for i:=MAXEQNS downto 1 do
begin
 x[i]:=a[i,MAXX+1];
 for j:=i+1 to MAXX do
  begin
```

```
    x[i]:=x[i] - a[i,j]*x[j];
   end;
  x[i]:=x[i]/a[i,i];
end;

{answers}
writeln;
for i:=1 to MAXX do
 write('x',i,' = ',x[i]:SIGPLACES:DECPLACES,'  ');

readln;
end.
```

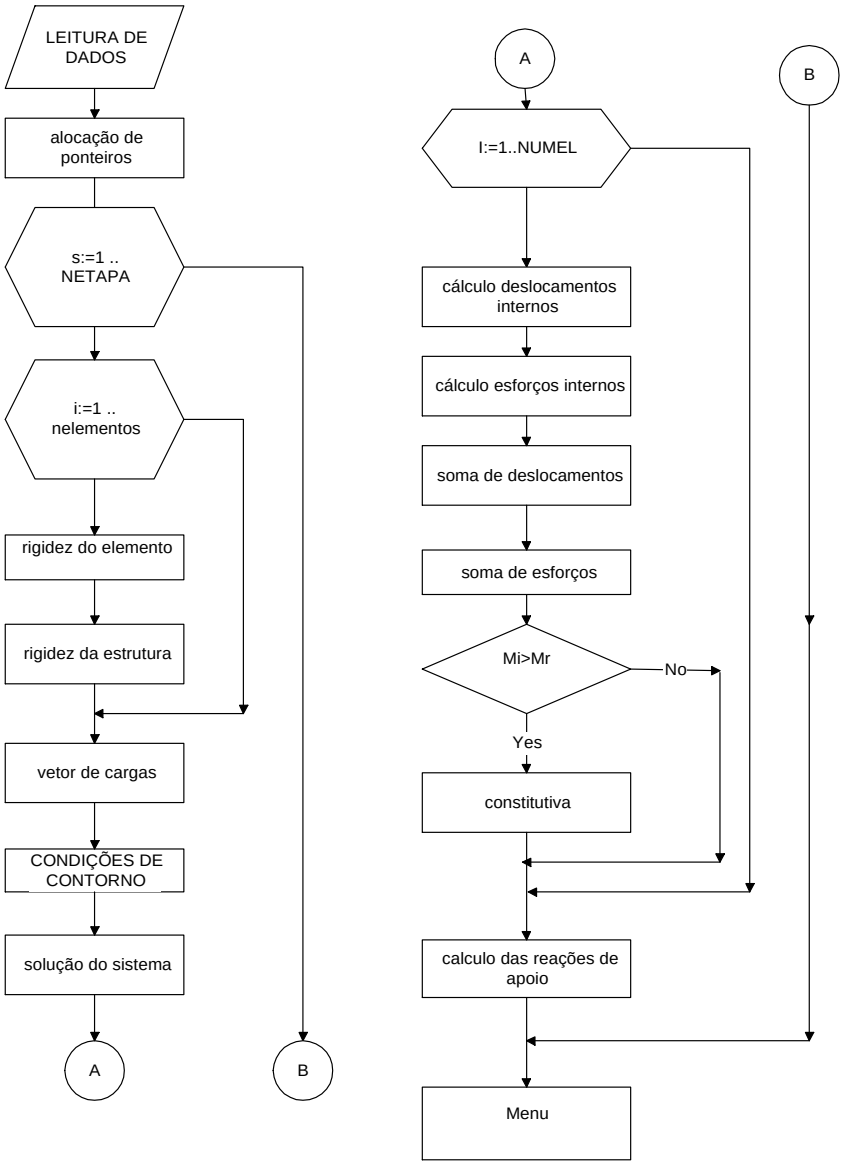


Figura 88: Fluxograma do programa Grelhas alterado para considerar as cargas de forma incremental.

Tabela 31: Exemplo de substituição da inércia padrão pela inércia de Branson (exemplo aplicado para vigas).

```
function BRANSON(K: Integer): Extended

var
  Inercia: Extended
begin
    AUX4:=MR[L]/AUX3
    Inercia:= AUX4*AUX4*AUX4*IFL[L]+(1-AUX4*AUX4*AUX4)*IFLII[L]
    IFLM[K]:= Inercia
//WRITELN('ETAPA ',KIL, ' BARRA ',K,' M=',AUX3,' L=',L,' IFLM=',IFLM[K])
    MostraStr('Branson ETAPA: '+MyFloat(KIL)+' BARRA '+MyFloat(K)+' M='+MyFloat(AUX3)+' L='+MyFloat(L)+'
IFLM='+MyFloat(Inercia))

    writeln(num7, 'Branson ETAPA: '+MyFloat(KIL)+' BARRA '+MyFloat(K)+' M='+MyFloat(AUX3)+' L='+MyFloat(L)+'
IFLM='+MyFloat(Inercia))
//PAUSA
    Result:= Inercia
end

function COMPARA(k:Integer): Extended
var
   Inercia: Extended
begin
  L:= TIPOB[K]
  CJ:=NNOI[K]
  CJR:=NNOF[K]
  AUX1:=ESFINI[2*CJ]
  AUX2:=ESFFIN[2*CJR]
  //IF AUX1<0 THEN AUX1:=0.0-AUX1
  //IF AUX2<0 THEN AUX2:=0.0-AUX2
  Aux1:= Abs(Aux1)
  Aux2:= Abs(Aux2)
  if Aux1>Aux2 then Aux3:= Aux1 else
                Aux3:= Aux2
  //IF AUX1<AUX2 THEN AUX3:=AUX2
  //IF AUX2<AUX1 THEN AUX3:=AUX1

  if (AUX3>=MR[L]) and bBranson then begin
    Inercia:= BRANSON(K)
  end else
    Inercia:=IFL[L]
  Result:= Inercia
end
```