MANUAL DO PROGRAMA:
OblqCalco – Flexão Oblíqua Composta

1 CARACTERÍSTICAS GERAIS E ENTRADA DE DADOS

Este programa tem como foco resolver problemas envolvendo esforços normais, portanto aqui se desprezam as solicitações tangenciais. As solicitações avaliadas são forças axiais e momento fletor em qualquer ângulo.

O programa divide-se em 4 abas, isto é, 4 iterações com diferentes objetivos:

- ELU: Gera um gráfico que mostra as solicitações que a seção suporta com segurança, apresentando a altura da linha neutra de ruptura também;
- Momento-Curvatura: Gera um gráfico que relaciona a grandeza momento com a grandeza curvatura, sob uma força axial resultante fixa na seção;
- Calcular Solicitações: Nessa aba, o programa calcula a partir dos dados fornecidos pelo usuário: ângulo, deformação na fibra superior e deformação na fibra inferior, calcula as solicitações correspondentes;
- Verificar Solicitações / ELS: A partir de solicitações escolhidas pelo usuário, calcula o estado de deformações correspondente (ângulo e deformações nas bordas inferior e superior) e verifica os 3 estados-limite de serviço: abertura de fissuras (ELS-W), formação de fissuras (ELS-F, parcialmente alterado, sugere-se consulta) e descompressão (ELS-D).

A seguir, se apresenta a imagem inicial do programa, que é o principal ambiente de trabalho. Os retângulos vermelhos apresentam cada área discriminada como a seguir:

1 – Nesta área se define a seção de concreto, de forma poligonal, descritas como pontos num plano cartesiano *xy*, em centímetros. Não é necessário que a figura seja fechada (último ponto = primeiro ponto), pois o programa faz o processo automaticamente. Além disso, a posição do polígono em relação à origem (0,0) não importa, isto é, o centro de gravidade da seção pode estar posicionado em qualquer ponto.

2 – Nesta área o usuário preenche se a seção tem ou não armadura passiva. Caso haja armadura passiva, deve-se indicar a área de cada barra, e sua posição no mesmo plano cartesiano usado para se posicionar o polígono de seção de concreto;

3 – A área 3 é referente à presença e posição da armadura ativa na seção. Caso haja armadura ativa, o usuário deve fornecer sua posição no plano cartesiano xy, sua área, sua deformação de pré-alongamento (apesar de ser de tração, o usuário deve usar sinal positivo). Caso o usuário deseje fazer a análise de ELS-W, é necessário fornecer o diâmetro da armadura também.

4 – O gráfico exposto na região 4 apresenta os resultados da análise de estado-limite último, isto é, o diagrama de momentos resistentes, e o gráfico da relação momento-curvatura. As outras 2 análises (Calcular Solicitações e Estados-limite de serviço) não utilizam de tal recurso para apresentar os seus resultados;

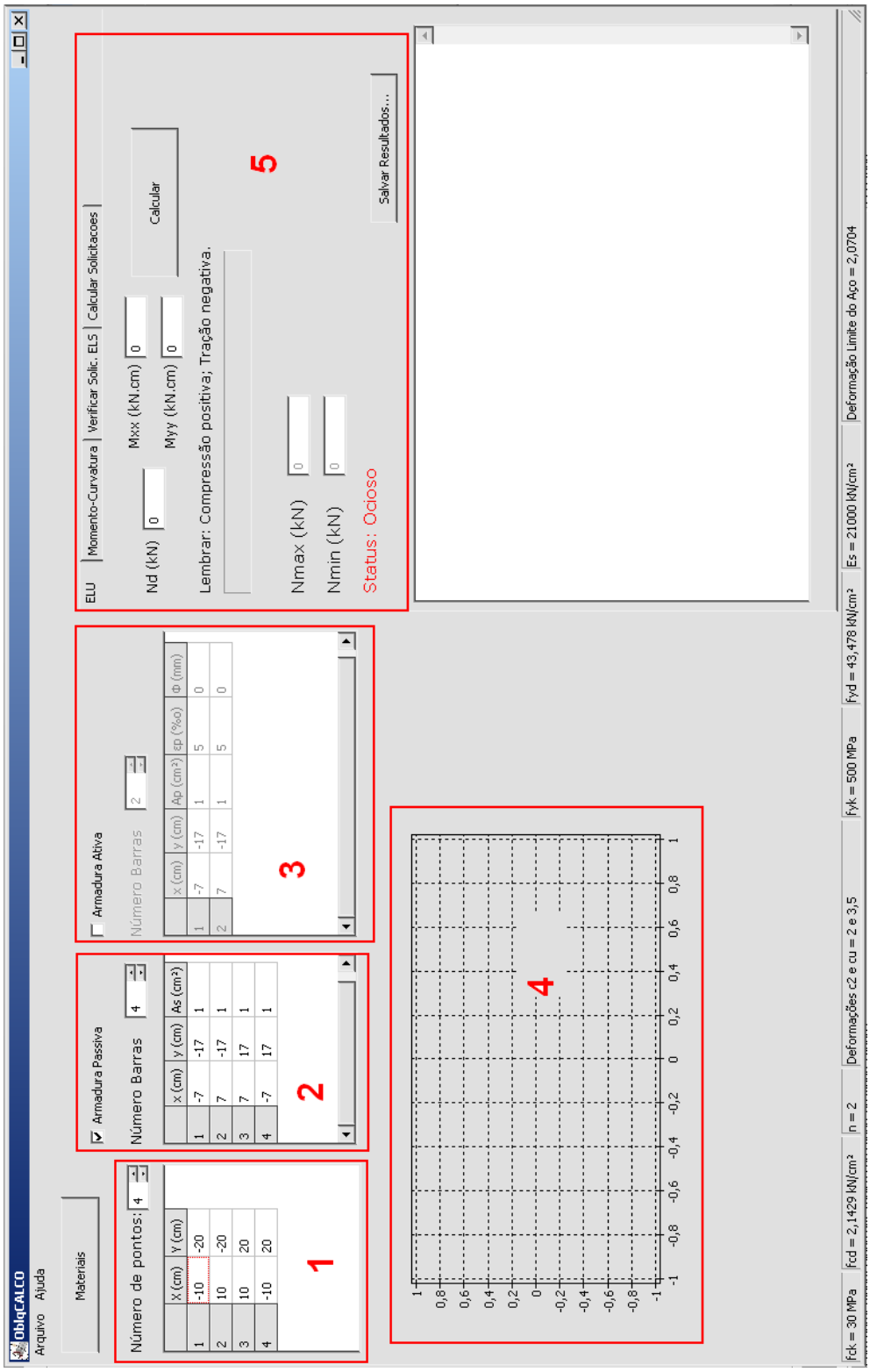
5 – A área 5 dispõe dados específicos às análises: por exemplo, na análise de estado-limite último, o usuário deve indicar qual a força normal atuante na seção; Cada área possui dados diferentes utilizados:

- Estado-limite último: Há um campo para se indicar a força normal na seção, e campos para apresentar momento atuante na seção; além disso, há o botão que inicia o processo de cálculo “Calcular”;

- Momento-Curvatura: Há o campo para indicar qual a força normal na seção, e o botão gatilho de cálculo “Calcular”;

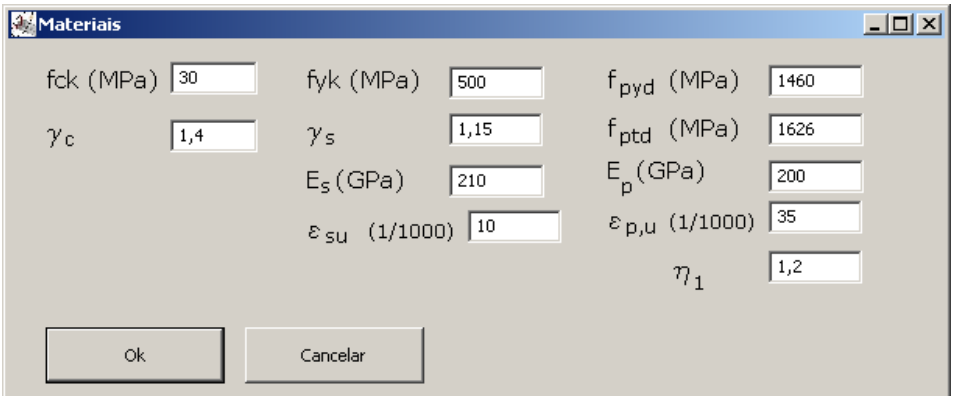
- Verificar Solicitação de serviço: Neste ambiente, há três campos:força axial, momento em relação ao eixo *x* e momento em relação ao eixo *y*, além do botão que inicia o processo de cálculo “Calcular”;

- Calcular Solicitações: Nesta região há três campos: um para a deformação absoluta na fibra superior da seção ε_1, um para a deformação absoluta na fibra inferior da seção ε_2, e outro para o ângulo da linha-neutra α em graus (sendo positivo no sentido horário).



Além disso, há 2 janelas de controle: uma para materiais e uma para os cálculos:

Na janela de materiais, apresentada pela figura a seguir, o usuário escolhe as características do concreto, aço de armadura passiva e aço da armadura ativa. O significado de cada é explicado na sequência.



- f_{ck} : resistência característica à compressão do concreto;
- γ_c : coeficiente que relaciona f_{ck} com f_{cd} (resistência de cálculo à compressão do concreto), de padrão 1,4;

f_{yk} : tensão de escoamento característica do aço de armadura passiva: na norma, sob nomenclatura E_{cs}, ver figura 8.4 da norma ABNT NBR 6118:2014;

- f_{yd} : tensão de escoamento de cálculo do aço de armadura passiva: na norma, sob nomenclatura E_{cs}, ver figura 8.4 da norma ABNT NBR 6118:2014;
- E_s : módulo de elasticidade do aço de armadura passiva: na norma, sob nomenclatura E_{cs}, ver figura 8.4 da norma ABNT NBR 6118:2014 ;
- ε_{su}: deformação última do aço (cujo valor padrão é 10 ‰) figura 8.4 da norma ABNT NBR 6118:2014 ;
- f_{pyd}: resistência de cálculo ao escoamento de escoamento da armadura de protensão, ver figura 8.5 da norma ABNT NBR 6118:2014;
- f_{ptd}: resistência de cálculo à tração da armadura de protensão, ver figura 8.5 da norma ABNT NBR 6118:2014 ;
- E_p: módulo de elasticidade do aço da armadura ativa (cujo valor padrão é 200 GPa) figura 8.5 da norma ABNT NBR 6118:2014;
- ε_{pu}: deformação última do aço de protensão (cujo valor padrão é 35 ‰) figura 8.5 da norma ABNT NBR 6118:2014;

Na janela de configurações, o usuário escolhe os critérios de parada para os processos iterativos: $(\varepsilon_i - \varepsilon_{i-1})$ quer dizer a diferença entre uma iteração e a próxima, que são os critérios de parada usados no cálculo numérico.

Também é relevante conhecer os outros dois campos que não fazem parte do critério de parada do processo iterativo:

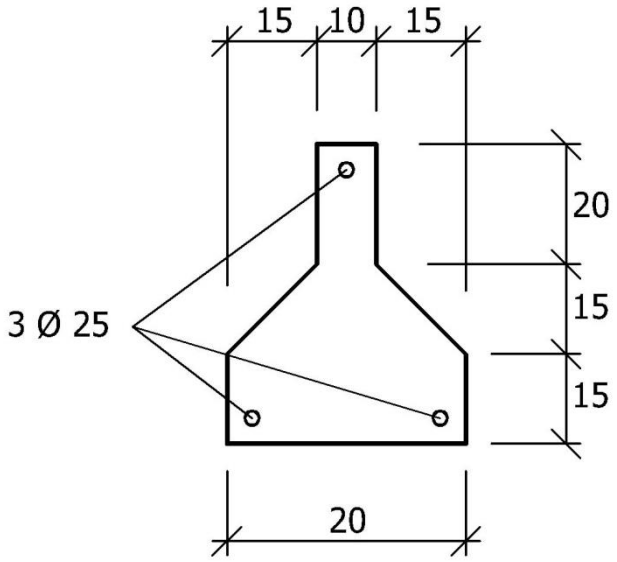
- "*Critérios ELU – α"* : um diagrama de Flexão Oblíqua Composta é uma curva: mas como ele é aproximado, neste trabalho essa curva é trocada por vários pontos: um ponto para cada ângulo que a linha-neutra girar em relação à seção a cada iteração concluída: α é exatamente a distância entre esses ângulos.

- "*Critérios ELS-W, ELS-F e ELS-D – ε_{ELS-F}"* : o usuário pode escolher a deformação limite para o estado-limite de formação de fissuras. Sugere-se que se consultar a seção que trata deste assunto na dissertação completa e o tópico da norma ABNT NBR 6118:2014. Além disso, o programa não suporta considerar tensões de tração no concreto.

1.1 ENTRADA DE DADOS

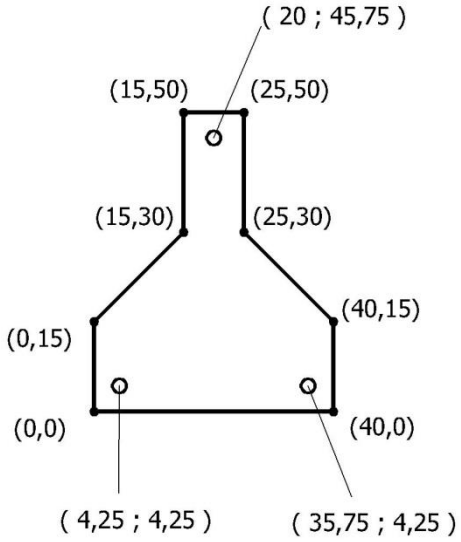
A entrada de dados é realizada pela entrada das coordenadas cartesianas que definem a seção.

Apresenta-se como tal entrada deve ser realizada mostrando uma seção exemplo. Essa mesma seção será a seção avaliada em todos os ambientes do programa. É composta de concreto C30 e armadura passiva CA 50. Não é utilizada a armadura ativa.



Para isso, aplica-se no programa os dados:

- Polígono que define a seção de concreto: resistência característica à compressão do concreto: foram aplicados os pontos: (0 , 0) ; (40 , 0) ; (40 , 15) ; (25 , 30) ; (25 , 50) ; (15 , 50) ; (15 , 30) ; (0 , 15) .
- Número de barras da armadura e as coordenadas de seus centros: (4,25 ; 4,25) ; (35,75 ; 4,25) ; (20 ; 45,75) com seus respectivas áreas que são iguais nesse caso: 4,9 cm².



1.2 EXPORTAÇÃO DE DADOS

Outra utilidade que o programa pode ter é permitir que os dados sejam exportados de alguma maneira. Isso permite que outros usuários possam comparar resultados: pois com o acesso a um gráfico, não se conhece exatamente os valores apresentados. Porém, com acesso aos valores, pode-se gerar gráficos que comparem diferentes análises.

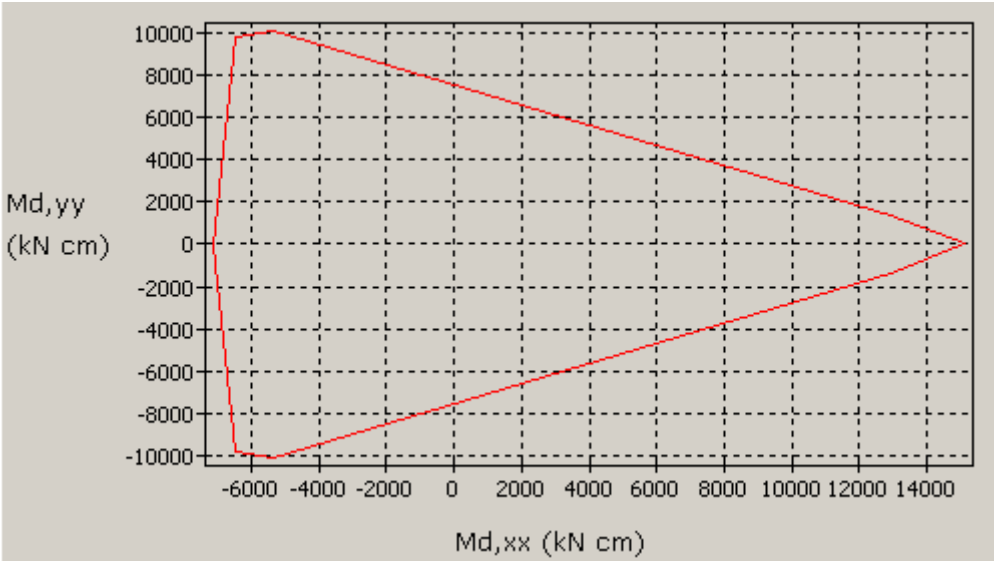
O programa não gera arquivos, ou tabelas prontas, mas exporta os dados num campo de texto. Neste campo, cada análise está numa linha, para que o usuário possa facilmente "*Colar*" os dados numa planilha. Alguns caracteres não são aceitos por certos programas. A seguir é apresentada uma linha do campo de texto: "*Eps1=0; Eps2=0; força=0; Curv.=0 x 10/m; Mxx = 0; Myy = 0*". Neste exemplo, o usuário pode, por exemplo, passar por um gerenciador de texto, como o *Bloco de Notas*, por exemplo, e aplicar a ferramenta "Substituir" para trocar os "x" por *nada*, eliminando assim caracteres indesejados. Os programas de planilha também transformam texto em tabela, para facilitar essa utilidade. Na seção referente ao ELU a ultima imagem foi gerada com uso dessa utilidade.

2 ABA "ELU"

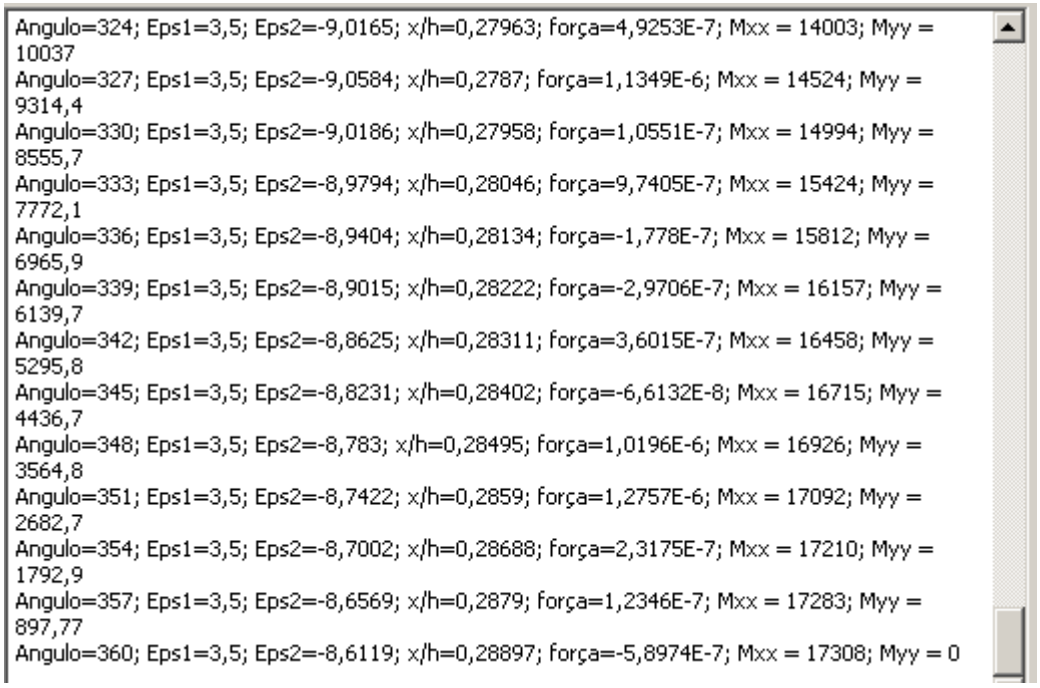
Pode-se obter diagramas de estado-limite último de uma seção especifica, apresentando os momentos resistentes M_{xx} e M_{yy} segundo uma força axial desejada, por exemplo, deseja-se verificar a seguinte seção, com materiais concreto C30 e aço CA-50.

Como resultado, o programa apresenta o gráfico e os dados num painel.

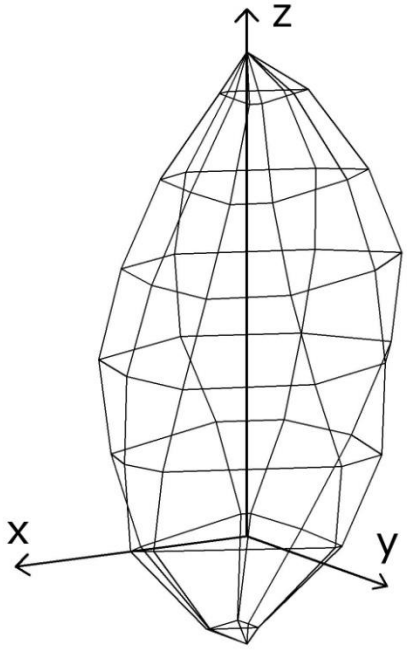
O gráfico é apresentado como mostra a imagem:



O programa também extrai as duplas de pontos (M_{xx} e M_{yy}) sob um campo de texto, como apresenta a imagem a seguir.



Esse texto foi elaborado assim com intenção de poder ser extraído, assim o usuário pode facilmente desenhar o gráfico em quaisquer outros lugares. Foi elaborado um diagrama com dados de várias iterações para a seção padrão. Para elaborar o diagrama, adotou-se um ângulo de 45º entre os resultados, e as forças normais de cada patamar é de -500, 0, +500, +1000, +1500, +2000, +2500 e os pontos extremos de forças axiais. Foi usada uma escala de 20 na força axial para tornar o entendimento mais intuitivo.



Inclusive, sugere-se para trabalhos futuros o desenvolvimento de uma forma automática de desenhar os diagramas mostrados, usando programação LISP em conjunto com o programa mostrado, por exemplo. Para mais informações sugere-se a leitura da dissertação relativa ao programa.

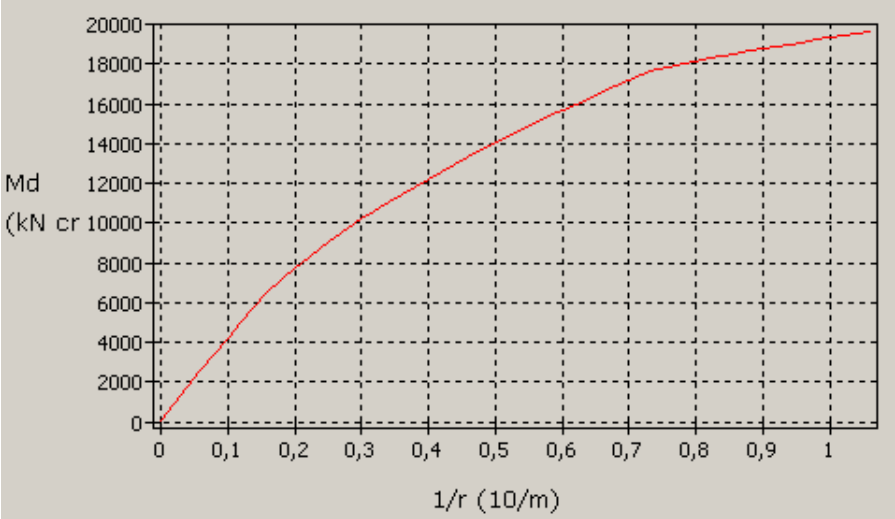
3 ABA "MOMENTO-CURVATURA"

Este ambiente foi elaborado com a intenção de descrever a relação sugerida pelo título: a relação momento-curvatura, segundo uma força axial constante. O programa primeiro gira a seção segundo o ângulo desejado, e depois calcula a posição da linha neutra que tem a força axial e curvatura correspondente e registra o valor. O programa então calcula os momentos M_{xx} e M_{yy} e os apresenta no campo de texto. Além disso, o programa gera um gráfico de M_{xx} *x 1/r*.

Por que o programa não apresenta um gráfico "*M* x *1/r"*? Pois um gráfico assim levaria o usuário à falsa impressão que o ângulo de inclinação da linha-neutra na seção é igual ao ângulo entre o vetor *M* e o eixo *y*. Ou ainda a falsa impressão de M_{xx} e M_{yy} manterem relação linear entre si.

Por esse motivo o programa avisa quando houver momento nas duas direções: para prevenir o usuário de usar dados de momento incorretamente ou sem devido aviso: evitando que a resistência à momentos numa direção seja superestimada.

Finalmente, apresenta-se um gráfico obtido pelo programa para a relação momento-curvatura. Esse gráfico é apresentado para a seção exemplo deste texto apresentada no título 1.1. Além disso, a força axial é de 500 kN.



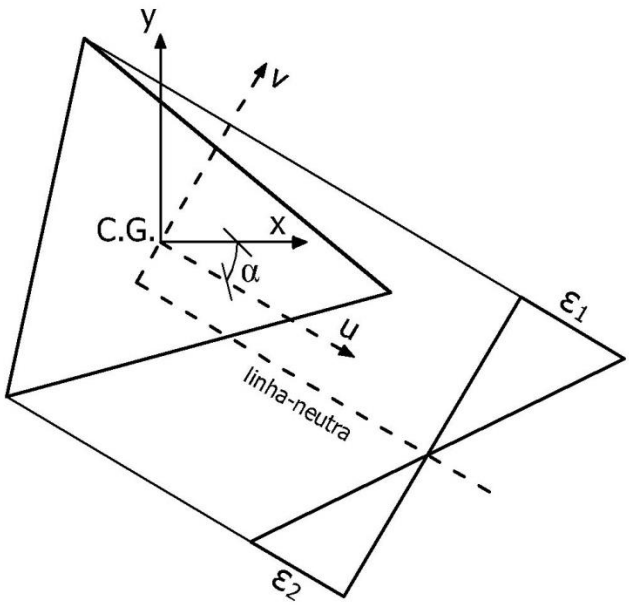
4 ABA "CALCULAR SOLICITAÇÕES"

A aba calcular solicitações tem por objetivo calcular diretamente os esforços análogos a certa conformação de deformações. O usuário fornece as deformações da seção usando as variáveis α, ε_1 e ε_2, como indica a figura logo após a descrição de cada variável:

- α: é o ângulo de inclinação da linha-neutra;

- ε_1: é a deformação no ponto da seção com a maior deformação (ou ponto com extremo máximo em relação ao eixo *uv*).

- ε_2: é a deformação no ponto da seção com a menor deformação (ou ponto com extremo mínimo em relação ao eixo *uv*.



O programa apenas resolve situações em que a borda do extremo máximo (maior compressão na seção) está mais comprimida que a borda do extremo mínimo (maior tração na seção). Caso for necessário resolver uma situação inversa (tração nas bordas superiores e compressão nas bordas inferiores – sendo superior e inferior em relação ao eixo *uv*), então inverte-se a seção por 180º. O programa faz isso automaticamente e avisa o usuário.

Então, o programa retorna como resultado a força axial, momento em relação ao eixo *x* e momento em relação ao eixo *y*. E também, o ângulo do vetor momento fletor: sendo que 0º significa um momento em relação ao eixo *x* convencional para vigas sob carregamento apenas de origem gravitacional.

A principal utilidade dessa aba é possibilitar ao usuário estudar o comportamento das seções ao alcançar certos valores de deformações. Além disso, a aba pode ser usada para conferir o funcionamento do programa, principalmente para comparação de resultados de diferentes rotinas, pois muitos programas de análise de Flexão Oblíqua Composta consideram forças e momentos como função das deformações, e não o contrário.

5 ABA “VERIFICAR SOLICITAÇÕES - ELS”

A quarta e última aba do programa faz com que o programa procure a conformação de deformações que apresentam certa solicitação. Isto é, o usuário escolhe as solicitações (força axial, momento fletor em relação ao eixo *x*, momento fletor em relação ao eixo *y*) e o programa procura pela conformação de deformações, retornando o resultado nas variáveis α, ε_1 e ε_2.

Além disso, o programa auxilia o usuário para verificações dos estados-limites de serviço de abertura de fissuras (ELS-W), formação de fissuras (ELS-F) e descompressão (ELS-D).

Salienta-se que a resistência à tração do concreto não é considerada em qualquer análise, mesmo aquelas em que o concreto não fissura (ELS-F e ELS-D). Além disso, o critério usado para analisar o ELS-F não é levado em conta de maneira idêntica à norma, mas usando uma aproximação: na norma, sugere-se realizar um cálculo que leva em conta a força da seção, sua inércia, e assim obter o momento de fissuração, como incida a seção 17.3 da norma ABNT NBR 6118:2014.

Neste trabalho levou-se em conta a relação tensão-deformação do concreto apenas. Na seção 8.2 a mesma norma caracteriza o concreto estrutural, e apresenta um gráfico que mostra o comportamento à tração do concreto, que indica a ruptura ao atingir deformação de 0,15‰ à tração.

• ELS-W: O programa calcula a tensão de cada armadura, e calcula a equação da norma da abertura de fissuras. A equação da norma é o menor valor entre 2 expressões. No presente trabalho, apenas se calcula a primeira (que não considera o concreto no envolvimento da armadura). Mesmo assim, se fornece a tensão, caso o usuário deseje calculá-la.

A equação que o programa calcula é:

$$w_{k,1} = \frac{\phi_i}{12{,}5\eta_1}\,\frac{\sigma_{si}}{E_{si}}\,\frac{3\sigma_{si}}{f_{ctm}}$$

• ELS-F: Para essa verificação, o programa compara a deformação de maior tração na seção (isto é, menor deformação, visto que deformação negativa = alargamento; deformação positiva = encurtamento) com o a máxima deformação permitida (limite de ruptura à tração do concreto, cujo padrão é 0,15‰). Caso nenhuma deformação na seção seja menor que 0,15‰ o ELS-F é atendido.

ELS-D: Funciona da mesma do ELS-F, porém com limites diferentes: no limite de descompressão, nenhum ponto da seção apresenta tração. Logo, a menor deformação permitida é *zero*. Logo, compara-se a menor deformação na seção com *zero*. Caso nenhuma deformação na seção seja menor que *zero*, o ELS-D é atendido.

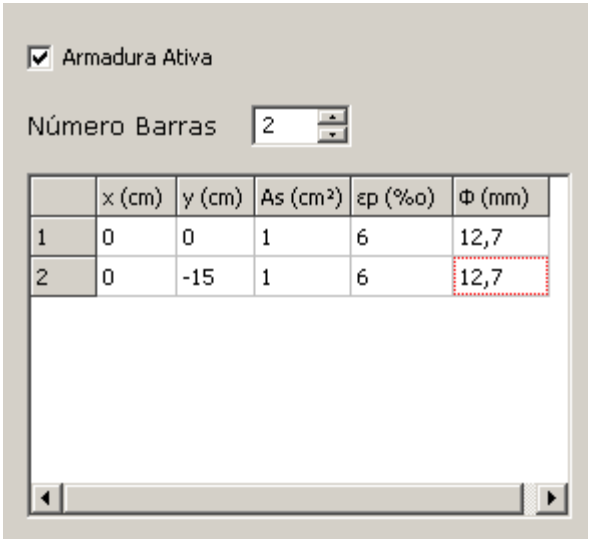
6 USO DE PROTENSÃO

O programa tem capacidade de avaliar as seções com presença de armadura ativa. Para isso, o usuário precisa incluir dois dados mais que a armadura passiva (além de caracterizar os materiais):

- ε_p : a deformação de pré alongamento ε_p em 1/1000. Para calculá-la, o usuário deve calcular a força de protensão com a consideração das perdas, e calcular qual a deformação correspondente. Essa deformação é de alargamento nos cabos, mesmo assim, seu valor no programa deve ser positivo.

- Φ(mm): diâmetro da seção da armadura. É o diâmetro da armadura, APENAS para cálculo do ELS-W. Isso é necessário para possibilitar várias considerações, por exemplo calcular o diâmetro a partir da área (usando a fórmula de áreas de circunferências) ou usar o diâmetro total da cordoalha. Caso a análise em ELS-W não seja importante, o usuário pode preencher com "zeros". NÃO devendo deixar a tabela com vazios.

A força de protensão ocasiona a seção à forças de compressão. O usuário não precisa considera-las separadamente, o programa faz isso automaticamente. A seguir, um exemplo de inserção da armadura Φ12,7, com ε_p = *6‰ (tração).*



UNIDADE PRINCIPAL

```
procedure TFrmMainFrm.ButGoELU(Sender: TObject);
var
iContador:integer;
```

```
fpdEpsilC2,fpEpsilonNow,fpEpsilonPrevious,fpNNow,fpNPrevious,fpNAuxiliar,fpEpsilonA
uxiliar
  ,fpNretaA,fpNdominio23,fpNdominio45,fpNretaB:real;

begin

  if fVarBin=True then
     begin
       LabelStatusELU.Caption:='Status: Calculando... Aguarde';
       LabelStatusELU.Refresh;

              // Inicio da preparação para iteração
       falfatemp:=0;
       Serie1.Clear;
       SeriePonto.Clear;
       MemoELU.Clear;
       ProgressBarELU.Position:=0;
       LabelChartX.Caption:='Md,xx (kN cm)';
       LabelChartYSup.Caption:='Md,yy';
       LabelChartYInf.Caption:='(kN.cm)';

  repeat
    RotacaoDePontos(falfatemp,CoordSecT,CoordSec);
    if fbArmPas then RotacaoDePontos(falfatemp,CoordArmPasT,CoordArmPas);
    if fbArmAtiv then RotacaoDePontos(falfatemp,CoordArmAtivT,CoordArmAtiv);
    PontosMaximoMinimo(CoordSec,fvmax,fvmin);
    fhalfa:=fvmax-fvmin;
    if fbArmPas then PontosMaximoMinimo(CoordArmPas,fAvmax,fAvmin);
    if fbArmPas then fdalfa:=fvmax-fAvmin;

    // PRIMEIRO, CALCULA-SE EM QUAL REGIAO ESTA A FORCA NORMAL
DESEJADA
    // N RETA A - LIMITE DOMINIO 1 - Nem se calcula a força no concreto.
    fEpsilon1:=fEpsilonsu;
    fEpsilon2:=fEpsilonsu;
    fpNretaA:=0;

    if fbArmPas then begin
      for fTemp:=0 to Length(CoordArmPasT)-1 do begin // Inicio força do aço
         ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
         aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
         fpNretaA:=fpNretaA+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
      end; //fim força do aço
    end;

    if fbArmAtiv then begin
      for fTemp:=0 to Length(CoordArmAtivT)-1 do begin // Inicio força do aço
         ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
         aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
         varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];
```

```
fpNretaA:=fpNretaA+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varT
emp);
     end; //fim força do aço
   end;
   // N DOMINIO 23
   fEpsilon1:=fEpsilonCU;
   fEpsilon2:=fEpsilonsu; // ESSE VALOR DE EPSILON2 É USADO CASO A RUPTURA
SEJA NO CONCRETO
   if fModoDeRuptura=1 then fEpsilon2:=fEpsilon1-(fEpsilon1-fEpsilonsu)*fhalfa/fdalfa; //
SE FOR NA ARMADURA PASSIVA, ESSE VALOR
   if fEpsilon1=fEpsilon2 then begin
     fxEpsilonC2:=10e6;
     fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
   end;
   if fEpsilon1<>fEpsilon2 then begin
     fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);  //Explicacoes na secao 4.1.2;
     fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);  //Explicacoes na secao 4.1.2;
   end;

   fpNdominio23:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

   if fbArmPas then begin
     for fTemp:=0 to Length(CoordArmPasT)-1 do begin // Inicio força do aço
        ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
        aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;

fpNdominio23:=fpNdominio23+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
     end; //fim força do aço
   end;
   if fbArmAtiv then begin
     for fTemp:=0 to Length(CoordArmAtivT)-1 do begin // Inicio força do aço
        ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
        aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
        varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNdominio23:=fpNdominio23+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,
ffptd,varTemp);
     end; //fim força do aço
   end;
   // N DOMINIO 45
   fEpsilon1:=fEpsilonCU;
   fEpsilon2:=0;

   if fEpsilon1=fEpsilon2 then begin
     fxEpsilonC2:=10e6;
     fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
   end;
   if fEpsilon1<>fEpsilon2 then begin
     fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
```

```
    fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);  //Explicacoes na secao 4.1.2;
  end;

  fpNdominio45:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

  if fbArmPas then
    begin
      for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
        begin
          ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
          aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;

fpNdominio45:=fpNdominio45+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
        end;
    end; //fim força do aço
  if fbArmAtiv then
    begin
      for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
        begin
          ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
          aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
          varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNdominio45:=fpNdominio45+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,
ffptd,varTemp);
        end; //fim força do aço
     end;


  // N REBA B - LIMITE DOMINIO 5
  fEpsilon1:=fEpsilonC2;
  fEpsilon2:=fEpsilonC2;

  if fEpsilon1=fEpsilon2 then begin
    fxEpsilonC2:=10e6;
    fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
  end;
  if fEpsilon1<>fEpsilon2 then begin
    fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
    fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);  //Explicacoes na secao 4.1.2;
  end;

  fpNretaB:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

  if fbArmPas then begin
    for fTemp:=0 to Length(CoordArmPasT)-1 do begin // Inicio força do aço
      ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
      aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
      fpNretaB:=fpNretaB+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
    end; //fim força do aço
```

```
   end;
   if fbArmAtiv then begin
    for fTemp:=0 to Length(CoordArmAtivT)-1 do begin // Inicio força do aço
       ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
       aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
       varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNretaB:=fpNretaB+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varT
emp);
     end; //fim força do aço
   end;

   if fNdFixado<=fpNretaB then begin
    fVarredura:=3; //Regiao I - Epsilon 2 usado como referencia - pois se Concreto for C90,
Epsilon1 é fixo
     fpNPrevious:=fpNdominio45;
     fpEpsilonPrevious:=0;
     fpNNow:=fpNretaB;
     fpEpsilonNow:=fEpsilonC2;
     fpdEpsilC2:=(fEpsilonCU-fEpsilonC2)/fEpsilonCU*fhalfa;
   end;
   if fNdFixado<=fpNdominio45 then begin
     fVarredura:=2; //Regiao II - Epsilon 2 usado como referencia - Epsilon1 assume valor fixo
     fpNPrevious:=fpNdominio23;
     if fModoDeRuptura=0 then fpEpsilonPrevious:=fEpsilonsu;
     if fModoDeRuptura=1 then fpEpsilonPrevious:=fEpsilonCU-(fEpsilonCU-
fEpsilonsu)*fhalfa/fdalfa;
     fpNNow:=fpNdominio45;
     fpEpsilonNow:=0;
     fEpsilon1:=fEpsilonCU;
     fEpsilon2:=0;
   end;
   if fNdFixado<=fpNdominio23 then begin
     fVarredura:=1; //Regiao III - Epsilon 1 usado como referencia - Epsilon 2 fixo
     fpNPrevious:=fpNretaA;
     fpEpsilonPrevious:=-fEpsilonYD;
     fpNNow:=fpNdominio23;
     fpEpsilonNow:=fEpsilonCU;
     if fModoDeRuptura=0 then fEpsilon2:=fEpsilonsu;
     if fModoDeRuptura=1 then fEpsilon2:=fEpsilonCU-(fEpsilonCU-
fEpsilonsu)*fhalfa/fdalfa;
   end;
   if fNdFixado<=fpNretaA then begin
     fVarredura:=0; //Alem do limite
   end;
   while fVarredura<>0 do begin
      if fVarredura=3 then begin // Regiao I - Eps2 referencia - Eps1 funcao de Eps2
        fpEpsilonNow:=fpEpsilonNow-(fpEpsilonNow-fpEpsilonPrevious)/(fpNNow-
fpNPrevious)*(fpNNow-fNdFixado);
```

```
        if fpEpsilonNow<0 then fpEpsilonNow:=0-(0-fpEpsilonPrevious)/(fpNdominio45-
fpNPrevious)*(fpNdominio45-fNdFixado);
        if fpEpsilonNow>fEpsilonC2 then fpEpsilonNow:=fEpsilonC2-(fEpsilonC2-
fpEpsilonPrevious)/(fpNretaB-fpNPrevious)*(fpNretaB-fNdFixado);
        fpEpsilonPrevious:=fEpsilon2;
        fpNPrevious:=fpNNow;
        fEpsilon2:=fpEpsilonNow;
        fEpsilon1:=(fEpsilonC2*fEpsilonCU-(fEpsilonCU-
fEpsilonC2)*fEpsilon2)/fEpsilonC2;
      end;
      if fVarredura=2 then begin // Regiao II
        fpEpsilonNow:=fpEpsilonNow-(fpEpsilonNow-fpEpsilonPrevious)/(fpNNow-
fpNPrevious)*(fpNNow-fNdFixado);
        if fModoDeRuptura=0 then begin
          if fpEpsilonNow<fEpsilonsu then begin
            fpEpsilonNow:=fEpsilonsu;
            fpNNow:=fpNdominio23;
          end;
        end;
        if fModoDeRuptura=1 then begin
          if fpEpsilonNow<(fEpsilonCU-(fEpsilonCU-fEpsilonsu)*fhalfa/fdalfa) then
fpEpsilonNow:=fEpsilonsu-(fEpsilonsu-fpEpsilonPrevious)/(fpNdominio23-
fpNPrevious)*(fpNdominio23-fNdFixado);
        end;
        if fpEpsilonNow>0 then fpEpsilonNow:=0-(0-fEpsilon2)/(fpNdominio45-
fpNNow)*(fpNdominio45-fNdFixado); // Isto é, se sair do intervalo para um valor maior,
compara-se o epsilon com o valor maximo
        fpEpsilonPrevious:=fEpsilon2;
        fpNPrevious:=fpNNow;
        fEpsilon2:=fpEpsilonNow;
      end;
      if fVarredura=1 then begin // Regiao III
        if fpNNow<=fpNretaA then begin
          fpNPrevious:=fpNdominio23;
          fpEpsilonPrevious:=fEpsilonCU;
        end;
        if abs(fpEpsilonNow-fpEpsilonPrevious)<=0.0000000001 then begin
          fpNPrevious:=fpNdominio23;
          fpEpsilonPrevious:=fEpsilonCU;
        end;
        fpEpsilonNow:=fpEpsilonNow-(fpEpsilonNow-fpEpsilonPrevious)/(fpNNow-
fpNPrevious)*(fpNNow-fNdFixado);
        if fpEpsilonNow<fEpsilonsu then fpEpsilonNow:=fEpsilonsu;
        if fpEpsilonNow>fEpsilonCU then fpEpsilonNow:=fEpsilonCU;
        fpEpsilonPrevious:=fEpsilon1;
        fpNPrevious:=fpNNow;
        fEpsilon1:=fpEpsilonNow;
        if fModoDeRuptura=1 then fEpsilon2:=fEpsilon1-(fEpsilon1-fEpsilonsu)*fhalfa/fdalfa;
        // Caso o modo de ruptura seja no concreto, o valor é constante fEpsilonsu
      end;
```

```
      if abs(fpEpsilonNow-fpEpsilonPrevious)<=fEpsilonErroELU then begin
        fVarredura:=0;
      end;


        // Arredondar os epsilon, pois eles estão adquirindo erros (nas casas decimais
avançadas
        fStrtemp:=FloattostrF(fEpsilon1,ffGeneral,8,6);
        fEpsilon1:=StrToFloat(fStrtemp);
        fStrtemp:=FloattostrF(fEpsilon2,ffGeneral,8,6);
        fEpsilon2:=StrToFloat(fStrtemp);
        if abs(fEpsilon1)<0.000000001 then fEpsilon1:=0;
        if abs(fEpsilon2)<0.000000001 then fEpsilon2:=0;

        if fEpsilon1=fEpsilon2 then begin
          fxEpsilonC2:=10e6;
          fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
        end;
        if fEpsilon1<>fEpsilon2 then begin
          fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
          fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);     //Explicacoes na secao
4.1.2;
        end;


        fpNNow:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

        if fbArmPas then
          begin
             for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
               begin
                  ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                  aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;

fpNNow:=fpNNow+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
               end; //fim força do aço
          end;

        if fbArmAtiv then
          begin
            for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força da armadura passiva
              begin
                ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNNow:=fpNNow+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTe
mp);
              end; //fim força da armadura passiva
```

```
        end;


      if fVarredura=0 then // Inicio do calculo dos momentos - se a carga axial N foi encontrada
        begin
          fMtempx:=0;
          fMtempy:=0;

fMtempx:=MomentoXXConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
          //Primeiro, calculando Mxx

fMtempy:=MomentoYYConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
          //Entao, calculando Myy
          if fbArmPas then
            begin
              for fTemp:=0 to Length(CoordArmPas)-1 do // Inicio momento do aço
                begin
                  ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                  aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                  fvarMtemp:=aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
// Calcula-se a força na barra e ela é salva numa variavel temporaria
                  fMtempx:=fMtempx+fvarMtemp*CoordArmPas[fTemp].yi;
                  fMtempy:=fMtempy+fvarMtemp*CoordArmPas[fTemp].xi;
                end; //fim do momento do aço
            end;
          if fbArmAtiv then
            begin
              for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força da armadura
ativa
                begin
                  ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                  aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                  varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fvarMtemp:=aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);  //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                  fMtempx:=fMtempx+fvarMtemp*CoordArmAtiv[fTemp].yi;
                  fMtempy:=fMtempy+fvarMtemp*CoordArmAtiv[fTemp].xi;
                end; //fim força da armadura ativa
            end;

          if abs(fMtempx)<0.000000001 then fMtempx:=0;
          if abs(fMtempy)<0.000000001 then fMtempy:=0;

          fMx:=fMtempx*cos(-falfatemp*3.14159265359/180)+fMtempy*sin(-
falfatemp*3.14159265359/180); //Girando de U para X
```

```
            fMy:=-fMtempx*sin(-falfatemp*3.14159265359/180)+fMtempy*cos(-
falfatemp*3.14159265359/180); //Girando de V para Y

            if                    fModoDeRuptura=0                    then
MemoELU.Append('Angulo='+floattostr(falfatemp)+';
Eps1='+floattostrf(fEpsilon1,ffGeneral,5,4)+';   Eps2='+floattostrf(fEpsilon2,ffGeneral,5,4)+';
x/h='+floattostrf(fxalfa/fhalfa,ffGeneral,5,4)+';  força='+FloattostrF(fpNNow,ffGeneral,5,4)+';
Mxx = '+FloattostrF(fMx,ffGeneral,5,4)+'; Myy = '+FloattostrF(fMy,ffGeneral,5,4));
            if                    fModoDeRuptura=1                    then
MemoELU.Append('Angulo='+floattostr(falfatemp)+';
Eps1='+floattostrf(fEpsilon1,ffGeneral,5,4)+';   Eps2='+floattostrf(fEpsilon2,ffGeneral,5,4)+';
x/d='+floattostrf(fxalfa/fdalfa,ffGeneral,5,4)+';  força='+FloattostrF(fpNNow,ffGeneral,5,4)+';
Mxx = '+FloattostrF(fMx,ffGeneral,5,4)+'; Myy = '+FloattostrF(fMy,ffGeneral,5,4));

            Serie1.AddXY(fMx,fMy);

          end; // Fim do calculo dos momentos - se a carga axial N foi encontrada

      end;


   falfatemp:=falfatemp+fELUalfa;
   ProgressBarELU.Position:=trunc(falfatemp*100/360);
 until falfatemp>360;
 SeriePonto.AddXY(StrToFloat(EdtELUMxx.Text),StrToFloat(EdtELUMyy.Text));
 LabelStatusELU.Caption:='Status: Ocioso';
 LabelStatusELU.Refresh;

  end;
end;
```

ROTINA MOMENTO CURVATURA

```
procedure TFrmMainFrm.ButGoMomCurvClick(Sender: TObject);
var
iContador:integer;
fpEpsilonNow,fpEpsilonPrevious,fpNNow,fpNPrevious,fpNAuxiliar,fpEpsilonAuxiliar
  ,fpdEpsilC2,fpNmax,fpNmin:real;

begin

  if fVarBin=True then
     begin
       LabelStatusELS.Caption:='Status: Calculando... Aguarde';
       LabelStatusELS.Refresh;
       RotacaoDePontos(falfatemp,CoordSecT,CoordSec);
       if fbArmPas then RotacaoDePontos(falfatemp,CoordArmPasT,CoordArmPas);
       if fbArmAtiv then RotacaoDePontos(falfatemp,CoordArmAtivT,CoordArmAtiv);
       PontosMaximoMinimo(CoordSec,fvmax,fvmin);
       fhalfa:=fvmax-fvmin;
```

```
      if fbArmPas then PontosMaximoMinimo(CoordArmPas,fAvmax,fAvmin);
      if fbArmPas then fdalfa:=fAvmax-fAvmin;

      // Inicio da preparação para iteração
      Serie1.Clear;
      MemoELS.Clear;
      LabelChartX.Caption:='1/r (10/m)';
      LabelChartYSup.Caption:='Md';
      LabelChartYinf.Caption:='(kN.cm)';
      ProgressBarELS.Position:=0;
      fVarredura:=1;
      fDeltaEps:=0;

      repeat
      fCurvatura:=fDeltaEps/fhalfa/10; // = 1/r - unidade = 1/m - cuidado ao manusear
curvaturas
      fVarredura:=1;
      // N MINIMO
      fEpsilon2:=fEpsilonsu;
      fEpsilon1:=fEpsilon2+fDeltaEps;

       if fEpsilon1=fEpsilon2 then
         begin
           fxEpsilonC2:=10e6;
           fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
         end;
       if fEpsilon1<>fEpsilon2 then
         begin
            fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
            fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);    //Explicacoes na secao
4.1.2
         end;

       fpNmin:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

       if fbArmPas then
         begin
          for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
             begin
                ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                fpNmin:=fpNmin+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
             end; //fim força do aço
         end;
       if fbArmAtiv then
         begin
          for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
             begin
                ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
```

```
                varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNmin:=fpNmin+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTem
p);
              end; //fim força do aço
           end;
        fpEpsilonPrevious:=fEpsilon2;
        fpNPrevious:=fpNmin;

        // N MAXIMO
        if fDeltaEps>=fEpsilonCU then
          begin
             fEpsilon1:=fEpsilonCU;
             fEpsilon2:=fEpsilon1-fDeltaEps;
          end;
        if fDeltaEps<fEpsilonCU then
          begin
             fpdEpsilC2:=(fEpsilonCU-fEpsilonC2)/fEpsilonCU*fhalfa; //Pode-se excluir essa
linha substituindo seu conteudo na linha seguinte
             fEpsilon1:=fEpsilonC2+fpdEpsilC2*fDeltaEps/fhalfa;
             fEpsilon2:=fEpsilon1-fDeltaEps;
          end;

        if fEpsilon1=fEpsilon2 then
          begin
             fxEpsilonC2:=10e6;
             fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
          end;
        if fEpsilon1<>fEpsilon2 then
          begin
             fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
             fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);    //Explicacoes na secao
4.1.2
          end;

        fpNmax:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
        if fbArmPas then
          begin
            for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
               begin
                  ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                  aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                  fpNmax:=fpNmax+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
               end; //fim força do aço
          end;
        if fbArmAtiv then
          begin
            for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
               begin
                  ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
```

```
                aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNmax:=fpNmax+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTem
p);
             end; //fim força do aço
          end;
   fpEpsilonNow:=fEpsilon2;
   fpNNow:=fpNmax;
   fpEpsilonAuxiliar:=fpEpsilonNow;
   fpNAuxiliar:=fpNNow;

   if fNdFixado>=fpNmax then
     begin
        fVarredura:=3;
     end;

   if fNdFixado<=fpNmin then
     begin
        fVarredura:=3;
     end;

   while fVarredura=1 do
       begin
          fpEpsilonNow:=fpEpsilonNow-(fpEpsilonNow-fpEpsilonPrevious)/(fpNNow-
fpNPrevious)*(fpNNow-fNdFixado);
          if fpEpsilonNow>fEpsilonC2 then fpEpsilonNow:=fEpsilonC2;
          fpEpsilonPrevious:=fEpsilon2;
          fpNPrevious:=fpNNow;
          fEpsilon2:=fpEpsilonNow;
          fEpsilon1:=fEpsilon2+fDeltaEps;

          if abs(fpEpsilonNow-fpEpsilonPrevious)<=fEpsilonErroMomCurv then //Erro
            begin
               fVarredura:=2;
            end;

          if fEpsilon1=fEpsilon2 then
            begin
               fxEpsilonC2:=10e6;
               fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
            end;
          if fEpsilon1<>fEpsilon2 then
            begin
               fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
               fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);  //Explicacoes na secao
4.1.2
            end;
```

```
        // Arredondar os epsilon, pois eles estÃ£o adquirindo erros (nas casas decimais
avanÃ§adas
        fStrtemp:=FloattostrF(fEpsilon1,ffGeneral,8,6);
        fEpsilon1:=StrToFloat(fStrtemp);
        fStrtemp:=FloattostrF(fEpsilon2,ffGeneral,8,6);
        fEpsilon2:=StrToFloat(fStrtemp);
        if abs(fEpsilon1)<0.000000001 then fEpsilon1:=0;
        if abs(fEpsilon2)<0.000000001 then fEpsilon2:=0;
        if abs(fDeltaEps)<0.000000001 then fDeltaEps:=0;

        fpNNow:=0;
        fpNNow:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
        if fbArmPas then
          begin
              for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
                 begin
                    ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                    aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;

fpNNow:=fpNNow+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
                 end; //fim força do aço
          end;
        if fbArmAtiv then
          begin
            for fTemp:=0 to Length(CoordArmAtivT)-1 do
              begin
                 ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                 aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                 varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fpNNow:=fpNNow+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTe
mp);
              end;
           end;
        if fpNPrevious<=fpNmin then // Este bloco evita que a forca fique constante (no
dominio 1, a regiao em que todas as armaduras estao trancionadas, ha forca constante
          begin
             if fpNNow<=fpNmin then
               begin
                  fpNPrevious:=fpNAuxiliar;
                  fpEpsilonPrevious:=fpEpsilonAuxiliar;
               end;
          end;

        if fVarredura=2 then // Inicio do calculo dos momentos - se a carga axial N foi
encontrada
          begin
             fMtempx:=0;
             fMtempy:=0;
```

```
            //Primeiro, calculando Mxx

fMtempx:=MomentoXXConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

            //Entao, calculando Myy

fMtempy:=MomentoYYConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

            if fDeltaEps=0 then   // Essa condicao foi necessaria para evitar um bug dentro do
programa
              begin              // Para entender o processo, basta retirar essas 5 linhas e testar o
programa
                fMtempx:=0;        // Em algumas situacoes de curvatura zero, ainda havera
momento resistido pelo concreto
                fMtempy:=0;        // E isso nao pode ocorrer, pois no centro geométrico o
momento de inércia de primeira ordem vale zero.
              end;


            if fbArmPas then
              begin
                for fTemp:=0 to Length(CoordArmPas)-1 do // Inicio momento do aÃ§o
                  begin
                    ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                    aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                    fvarMtemp:=aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
// Calcula-se a forÃ§a na barra e ela Ã© salva numa variavel temporaria
                    fMtempx:=fMtempx+fvarMtemp*CoordArmPas[fTemp].yi;
                    fMtempy:=fMtempy+fvarMtemp*CoordArmPas[fTemp].xi;
                  end; //fim do momento do aÃ§o
              end;
             if fbArmAtiv then
               begin
                 for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força da armadura
ativa
                   begin
                     ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                     aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                     varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fvarMtemp:=aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);  //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                     fMtempx:=fMtempx+fvarMtemp*CoordArmAtiv[fTemp].yi;
                     fMtempy:=fMtempy+fvarMtemp*CoordArmAtiv[fTemp].xi;
                   end; //fim força da armadura ativa
               end;
```

```
            fMx:=fMtempx*cos(-falfatemp*3.14159265359/180)+fMtempy*sin(-
falfatemp*3.14159265359/180); //Girando de U para X
            fMy:=-fMtempx*sin(-falfatemp*3.14159265359/180)+fMtempy*cos(-
falfatemp*3.14159265359/180); //Girando de V para Y

            if abs(fMx)<0.000000001 then fMtempx:=0;
            if abs(fMy)<0.000000001 then fMtempy:=0;
            if abs(fMy)>0.000000001 then MemoELS.Append('Existe momento residual na
direção perpendicular!');


            MemoELS.Append('Eps1='+FloatToStrF(fEpsilon1,ffGeneral,6,5)+';
Eps2='+FloatToStrF(fEpsilon2,ffGeneral,6,5)+';
força='+FloattostrF(fpNNow,ffGeneral,6,5)+';
Curv.='+FloattostrF(fCurvatura,ffGeneral,6,5)+';                Mxx                =
'+FloattostrF(fMtempx,ffGeneral,6,5)+'; Myy = '+FloattostrF(fMtempy,ffGeneral,6,5));
            Serie1.AddXY(fCurvatura,fMx);

        end; // Fim do calculo dos momentos - se a carga axial N foi encontrada


    end;
  fDeltaEps:=fDeltaEps+0.1;

  until fVarredura=3;

  LabelStatusELS.Caption:='Status: Ocioso';
  LabelStatusELS.Refresh;

 end;
end;
```

Função Procurar

```
procedure TFrmMainFrm.ButProcurarClick(Sender: TObject);
var
iContador:integer;

begin
 fVarBin:=True; // Inicio das medidas defensivas

 if fVarBin=True then
   begin
     SetLength(CoordSec,PlanPolyg.RowCount); // Inicio da aquisição dos dados - Dos Edit
para as variaveis
     VetorReceberPontos(PlanPolyg,CoordSec);
     ObterCaractGeomet(CoordSec,fAc,fxcg,fycg); //Calculo das caracteristicas geometricas
     SetLength(CoordSecT,Length(CoordSec));   //Inicio da translação dos pontos
     TranslacaoDePontos(CoordSec,fxcg,fycg,CoordSecT);
```

```
      if fbArmPas then
        begin
          SetLength(CoordArmPas,PlanArmPas.RowCount-1);
          SetLength(aAs,PlanArmPas.RowCount-1);
          SetLength(ahsi,PlanArmPas.RowCount-1);
          SetLength(aEpsilonsi,PlanArmPas.RowCount-1);
          VetorReceberPontosArmaduraPassiva(PlanArmPas,aAs,CoordArmPas); // Fim da
aquisição dos dados - Dos Edit para as variaveis
          SetLength(CoordArmPasT,Length(CoordArmPas));
          TranslacaoDePontos(CoordArmPas,fxcg,fycg,CoordArmPasT);  //Fim da translação
dos pontos
        end;

      if fbArmAtiv then
        begin
          SetLength(CoordArmAtiv,PlanArmAtiv.RowCount-1);
          SetLength(aAp,PlanArmAtiv.RowCount-1);
          SetLength(ahpi,PlanArmAtiv.RowCount-1);
          SetLength(aEpsilonprealong,PlanArmAtiv.RowCount-1);
          SetLength(aEpsilonpsi,PlanArmAtiv.RowCount-1);
          SetLength(aDiametroAp,PlanArmAtiv.RowCount-1);

VetorReceberPontosArmaduraAtiva(PlanArmAtiv,aAp,aEpsilonprealong,aDiametroAp,Coor
dArmAtiv); // Fim da aquisição dos dados - Dos Edit para as variaveis
          SetLength(CoordArmAtivT,Length(CoordArmAtiv));
          TranslacaoDePontos(CoordArmAtiv,fxcg,fycg,CoordArmAtivT);  //Fim da translação
dos pontos
        end;

       fEpsilon1:=StrToFloat(EdtProcEps1.Text); // Este bloco inverte Epsilon1 e Epsilon2
automaticamente caso Eps2 seja maior que Eps1
       fEpsilon2:=StrToFloat(EdtProcEps2.Text);
       if fEpsilon2>fEpsilon1 then begin
         falfatemp:=StrToFloat(EdtProcAlfa.Text);
         falfatemp:=falfatemp+180;
         EdtProcAlfa.Text:=Floattostr(falfatemp);
         EdtProcEps1.Text:=FloatToStr(fEpsilon2);
         EdtProcEps2.Text:=FloatToStr(fEpsilon1);
       end;


      if fVarBin=True then begin
        // Inicio da preparação para iteração
        falfatemp:=StrToFloat(EdtProcAlfa.Text);
        fEpsilon1:=StrToFloat(EdtProcEps1.Text);
        fEpsilon2:=StrToFloat(EdtProcEps2.Text);
        RotacaoDePontos(falfatemp,CoordSecT,CoordSec);
        if fbArmPas then RotacaoDePontos(falfatemp,CoordArmPasT,CoordArmPas);
        if fbArmAtiv then RotacaoDePontos(falfatemp,CoordArmAtivT,CoordArmAtiv);
        PontosMaximoMinimo(CoordSec,fvmax,fvmin);
```

```
      fhalfa:=fvmax-fvmin;
      if fbArmPas then PontosMaximoMinimo(CoordArmPas,fAvmax,fAvmin);
      if fbArmPas then fdalfa:=fAvmax-fAvmin;
      fNdtemp:=0;
          // Arredondar os epsilon, pois eles estão adquirindo erros (nas casas decimais
avançadas
          fStrtemp:=FloattostrF(fEpsilon1,ffGeneral,8,6);
          fEpsilon1:=StrToFloat(fStrtemp);
          fStrtemp:=FloattostrF(fEpsilon2,ffGeneral,8,6);
          fEpsilon2:=StrToFloat(fStrtemp);
          if abs(fEpsilon1)<0.000000001 then fEpsilon1:=0;
          if abs(fEpsilon2)<0.000000001 then fEpsilon2:=0;

          if fEpsilon1=fEpsilon2 then begin
            fxEpsilonC2:=10e6;
            fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
          end;
          if fEpsilon1<>fEpsilon2 then begin
            fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
            fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);   //Explicacoes na secao
4.1.2
          end;

          fNdtemp:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

          if fbArmPas then
            begin
              for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
                 begin
                    ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                    aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;

fNdtemp:=fNdtemp+aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);
                 end; //fim força do aço
            end;
          if fbArmAtiv then
            begin
              for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
                 begin
                    ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                    aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                    varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fNdtemp:=fNdtemp+aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTe
mp);
                 end; //fim força do aço
             end;

          fMtempx:=0;
          fMtempy:=0;
```

```
fMtempx:=MomentoXXConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
          //Primeiro, calculando Mxx

fMtempy:=MomentoYYConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);
          //Entao, calculando Myy
          if fbArmPas then
            begin
             for fTemp:=0 to Length(CoordArmPas)-1 do // Inicio momento do aço
                begin
                   ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                   aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                   fvarMtemp:=aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd);    //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                   fMtempx:=fMtempx+fvarMtemp*CoordArmPas[fTemp].yi;
                   fMtempy:=fMtempy+fvarMtemp*CoordArmPas[fTemp].xi;
                end; //fim do momento do aço
            end;

          if fbArmAtiv then
            begin
             for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força da armadura ativa
                begin
                   ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                   aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                   varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fvarMtemp:=aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);    //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                   fMtempx:=fMtempx+fvarMtemp*CoordArmAtiv[fTemp].yi;
                   fMtempy:=fMtempy+fvarMtemp*CoordArmAtiv[fTemp].xi;
                end; //fim força da armadura ativa
             end;

                   if abs(fMtempx)<0.000001 then fMtempx:=0;
                   if abs(fMtempy)<0.000001 then fMtempy:=0;

                   fMx:=fMtempx*cos(-falfatemp*3.14159265359/180)+fMtempy*sin(-
falfatemp*3.14159265359/180); //Girando de U para X
                   fMy:=-fMtempx*sin(-falfatemp*3.14159265359/180)+fMtempy*cos(-
falfatemp*3.14159265359/180); //Girando de V para Y

                   if abs(fMx)<0.000001 then fMx:=0;
                   if abs(fMy)<0.000001 then fMy:=0;

                   //Procurando angulo de Mxx e Myy
                   if fMx<>0 then
                     begin
                        falfatemp:=arctan(fMy/fMx)*180/3.14159265359;
                     end;
```

```
                    if fMx<0 then falfatemp:=falfatemp+180;
                    if fMx>0 then
                      begin
                          if fMy<0 then
                            begin
                                falfatemp:=falfatemp+360;
                            end;
                      end;

                    EdtProcResult.Text:='Força='+FloattostrF(fNdtemp,ffGeneral,6,5)+';  Mxx
= '+FloattostrF(fMx,ffGeneral,6,5)+'; Myy = '+FloattostrF(fMy,ffGeneral,6,5)+'; Angulo do
Vetor Momento ='+FloatToStrF(falfatemp,ffGeneral,6,5);
      end;
    end;
end;

procedure TFrmMainFrm.ButCalc2Click(Sender: TObject);
var
iContador:integer;
fpMxx,fpMyy,fpN,fpNNow,fpMxNow,fpMyNow,fphx,fphy,fpEpsilonC,fpDeltaEpsX,fpDelta
EpsY,fpEpsilonCPrevious,fpDeltaEpsXPrevious,fpDeltaEpsYPrevious,fpDeltaEpsAlfa,fpCur
vatura,fpCurvaturaX,fpCurvaturaY,fpVarTemp1,fpVarTemp2:real;
aJacobn,aJacobInverso:array[1..3,1..3] of real;

begin
    if EdtELSNd.Text='' then
       begin
         LabelStatusProcurar2.Caption:='Dados incompletos';
         fVarBin:=False;
       end;
   if EdtELSMxx.Text='' then
      begin
        LabelStatusProcurar2.Caption:='Dados incompletos';
        fVarBin:=False;
      end;
   if EdtELSMyy.Text='' then
      begin
        LabelStatusProcurar2.Caption:='Dados incompletos';
        fVarBin:=False;
      end;
    fVarBin:=True; // Inicio das medidas defensivas
  for iContador:=1 to SpnPolyg.Value do         // Medida defensiva: Tabela de poligonos
incompleta
     begin
        if PlanPolyg.Cells[1,iContador]='' then
          begin
            LabelStatusProcurar2.Caption:='Tabela de pontos do poligonos incompleta!';
            fVarBin:=False;
          end;
        if PlanPolyg.Cells[2,iContador]='' then
```

```
          begin
            LabelStatusProcurar2.Caption:='Tabela de pontos do poligonos incompleta!';
            fVarBin:=False;
          end;
      end;
  if fbArmPas then
    begin
      for iContador:=1 to SpnEdtArmPas.Value do   // Medida defensiva: Tabela de armadura
passiva incompleta
         begin
            if PlanArmPas.Cells[1,iContador]='' then
               begin
                LabelStatusProcurar2.Caption:='Tabela das armaduras passivas incompleta!';
                fVarBin:=False;
               end;
           if PlanArmPas.Cells[2,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras passivas incompleta!';
              fVarBin:=False;
             end;
           if PlanArmPas.Cells[3,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras passivas incompleta!';
              fVarBin:=False;
             end;
         end;
    end;
  if fbArmAtiv then
    begin
      for iContador:=1 to SpnEdtArmAtiv.Value do   // Medida defensiva: Tabela de armadura
ativa incompleta
        begin
           if PlanArmAtiv.Cells[1,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras ativa incompleta!';
              fVarBin:=False;
             end;
           if PlanArmAtiv.Cells[2,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras ativa incompleta!';
              fVarBin:=False;
             end;
           if PlanArmAtiv.Cells[3,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras ativa incompleta!';
              fVarBin:=False;
             end;
           if PlanArmAtiv.Cells[4,iContador]='' then
             begin
              LabelStatusProcurar2.Caption:='Tabela das armaduras ativa incompleta!';
```

```
              fVarBin:=False;
            end;
        end;
    end;
  if fVarBin=True then
    begin
      fpN:=strtofloat(EdtELSNd.Text);
      fpMxx:=strtofloat(EdtELSMxx.Text);
      fpMyy:=strtofloat(EdtELSMyy.Text);
      SetLength(CoordSec,PlanPolyg.RowCount); // Inicio da aquisição dos dados - Dos Edit
para as variaveis
      VetorReceberPontos(PlanPolyg,CoordSec);
      ObterCaractGeomet(CoordSec,fAc,fxcg,fycg); //Calculo das caracteristicas geometricas
      if fbArmPas then
        begin
          SetLength(CoordArmPas,PlanArmPas.RowCount-1);
          SetLength(aAs,PlanArmPas.RowCount-1);
          SetLength(ahsi,PlanArmPas.RowCount-1);
          SetLength(aEpsilonsi,PlanArmPas.RowCount-1);
          VetorReceberPontosArmaduraPassiva(PlanArmPas,aAs,CoordArmPas); // Fim da
aquisição dos dados - Dos Edit para as variaveis
          SetLength(CoordArmPasT,Length(CoordArmPas));
          TranslacaoDePontos(CoordArmPas,fxcg,fycg,CoordArmPasT);  //Fim da translação
dos pontos
        end;

      if fbArmAtiv then
        begin
          SetLength(CoordArmAtiv,PlanArmAtiv.RowCount-1);
          SetLength(aAp,PlanArmAtiv.RowCount-1);
          SetLength(ahpi,PlanArmAtiv.RowCount-1);
          SetLength(aEpsilonprealong,PlanArmAtiv.RowCount-1);
          SetLength(aEpsilonpsi,PlanArmAtiv.RowCount-1);
          SetLength(aDiametroAp,PlanArmAtiv.RowCount-1);

VetorReceberPontosArmaduraAtiva(PlanArmAtiv,aAp,aEpsilonprealong,aDiametroAp,Coor
dArmAtiv); // Fim da aquisição dos dados - Dos Edit para as variaveis
          SetLength(CoordArmAtivT,Length(CoordArmAtiv));
          TranslacaoDePontos(CoordArmAtiv,fxcg,fycg,CoordArmAtivT);  //Fim da translação
dos pontos
        end;
      SetLength(CoordSecT,Length(CoordSec));   //Inicio da translação dos pontos
      TranslacaoDePontos(CoordSec,fxcg,fycg,CoordSecT);
      fAs:=0;         //Inicio do calculo dos valores normais
      if fbArmPas then
        begin
          for fTemp:=1 to PlanArmPas.RowCount-1 do
            begin
              fAs:=fAs+Strtofloat(PlanArmPas.Cells[3,fTemp]);
            end;
```

```
        end;
      fSigmacd:=ffcd*0.85;
      fNdmax:=(fAc*fSigmacd+fAs*TensaoArmPas(fEpsilonC2,fEs,ffyd));
      fNdmin:=(-ffyd*fAs);

      if fbArmAtiv then
        begin
          for fTemp:=1 to PlanArmAtiv.RowCount-1 do
            begin
              varTemp:=(aEpsilonprealong[fTemp-1])+fEpsilonC2;
              fNdmax:=fNdmax+aAp[fTemp-
1]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);
              varTemp:=(aEpsilonprealong[fTemp-1])-10
              fNdmin:=fNdmin+aAp[fTemp-
1]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);
            end;
        end;
//      EdtNmax.Text:=FloatToStrF(fNdmax,ffGeneral,8,6);
//        EdtNmin.Text:=FloattostrF(fNdmin,ffGeneral,8,6);       //Fim do calculo dos valores
normais
      if fpN>=fNdmax then
        begin
          LabelStatusProcurar2.Caption:='Ruptura por compressão atingida. Força de
compressão máxima='+FloatToStrF(fNdmax,ffGeneral,8,6);
          fVarBin:=False;
        end;
      if fpN<=fNdmin then
        begin
          LabelStatusProcurar2.Caption:='Ruptura por tração atingida. Força de tração
máxima='+FloatToStrF(fNdmin,ffGeneral,8,6);
          fVarBin:=False;
        end;
    end;


        if fVarBin=True then
          begin
            MemoProcurar2.Clear;
            RotacaoDePontos(0,CoordSecT,CoordSec);
            if fbArmPas then RotacaoDePontos(0,CoordArmPasT,CoordArmPas);
            if fbArmAtiv then RotacaoDePontos(0,CoordArmAtivT,CoordArmAtiv);
            PontosMaximoMinimo(CoordSec,fpVarTemp1,fpVarTemp2);
            fphy:=fpVarTemp1-fpVarTemp2;

            RotacaoDePontos(90,CoordSecT,CoordSec);
            if fbArmPas then RotacaoDePontos(90,CoordArmPasT,CoordArmPas);
            if fbArmAtiv then RotacaoDePontos(90,CoordArmAtivT,CoordArmAtiv);
            PontosMaximoMinimo(CoordSec,fpVarTemp1,fpVarTemp2);
            fphx:=fpVarTemp1-fpVarTemp2;
```

```
            //Encontrou-se fphy e fphx, agora, calcula-se o jacobiano.
            for iContador:=1 to 4 do
               begin
                  //Para isso, calculam-se 4 estados de deformacao:
                  // 1 - falfatemp=0; Epsilon1 = 0.1; Epsilon2 = 0.1
                  // 2 - falfatemp=0; Epsilon1 = 0.2; Epsilon2 = 0.2 (isto e, varia-se o
fpEpsilonC em 0.1)
                  // 3 - falfatemp=90; Epsilon1 = 0.2; Epsilon2 = 0.0 (isto e, varia-se o
fpCurvaturaY em 0.2)
                  // 4 - falfatemp=0; Epsilon1 = 0.2; Epsilon2 = 0.0 (isto e, varia-se o
fpCurvaturaX em 0.2)

                  if iContador=1 then
                    begin
                      falfatemp:=0;
                      fEpsilon1:=0.1;
                      fEpsilon2:=0.1;
                    end;
                  if iContador=2 then
                    begin
                      falfatemp:=0;
                      fEpsilon1:=0.2;
                      fEpsilon2:=0.2;
                    end;
                  if iContador=3 then
                    begin
                      falfatemp:=90;
                      fEpsilon1:=0.2;
                      fEpsilon2:=0.0;
                    end;
                  if iContador=4 then
                    begin
                      falfatemp:=0;
                      fEpsilon1:=0.2;
                      fEpsilon2:=0.0;
                    end;

                  RotacaoDePontos(falfatemp,CoordSecT,CoordSec); //inicio, obtencao de N,
Mxx e Myy a partir de falfatemp, fEpsilon1 e fEpsilon2;
                  if                                fbArmPas                                then
RotacaoDePontos(falfatemp,CoordArmPasT,CoordArmPas);
                  if                                fbArmAtiv                               then
RotacaoDePontos(falfatemp,CoordArmAtivT,CoordArmAtiv);
                  PontosMaximoMinimo(CoordSec,fvmax,fvmin);
                  fhalfa:=fvmax-fvmin;
                  if abs(fEpsilon1)<0.000000001 then fEpsilon1:=0;
                  if abs(fEpsilon2)<0.000000001 then fEpsilon2:=0;
                  if fEpsilon1=fEpsilon2 then
                    begin
                       fxEpsilonC2:=10e6;
```

```
                    fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
                  end;
                if fEpsilon1<>fEpsilon2 then
                  begin
                    fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
                    fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);     //Explicacoes
na secao 4.1.2
                  end;

fNdtemp:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

fMtempx:=MomentoXXConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

fMtempy:=MomentoYYConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

                if fbArmPas then
                  begin
                    for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
                      begin
                        ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                        aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;

fvarMtemp:=aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd); // Calcula-se a força
na barra e ela é salva numa variavel temporaria
                        fNdtemp:=fNdtemp+fvarMtemp;
                        fMtempx:=fMtempx+fvarMtemp*CoordArmPas[fTemp].yi;
                        fMtempy:=fMtempy+fvarMtemp*CoordArmPas[fTemp].xi;
                      end; //fim força do aço
                  end;

                if fbArmAtiv then
                   begin
                     for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
                       begin
                         ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                         aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                         varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fvarMtemp:=aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);   //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                         fNdtemp:=fNdtemp+fvarMtemp;
                         fMtempx:=fMtempx+fvarMtemp*CoordArmAtiv[fTemp].yi;
                         fMtempy:=fMtempy+fvarMtemp*CoordArmAtiv[fTemp].xi;
                       end; //fim força do aço
                   end;

                if abs(fMtempx)<0.000001 then fMtempx:=0;
                if abs(fMtempy)<0.000001 then fMtempy:=0;
```

```
                fMx:=fMtempx*cos(-falfatemp*3.14159265359/180)+fMtempy*sin(-
falfatemp*3.14159265359/180); //Girando de U para X
                fMy:=-fMtempx*sin(-falfatemp*3.14159265359/180)+fMtempy*cos(-
falfatemp*3.14159265359/180); //Girando de V para Y

                if abs(fMx)<0.000001 then fMx:=0;
                if abs(fMy)<0.000001 then fMy:=0;

                if iContador=1 then
                   begin
                     aJacobn[1,1]:=-fNdtemp;
                     aJacobn[1,2]:=-fMx;
                     aJacobn[1,3]:=-fMy;
                     aJacobn[2,1]:=-fNdtemp;
                     aJacobn[2,2]:=-fMx;
                     aJacobn[2,3]:=-fMy;
                     aJacobn[3,1]:=-fNdtemp;
                     aJacobn[3,2]:=-fMx;
                     aJacobn[3,3]:=-fMy;
                      fpNNow:=fNdtemp;   // Guardando esses 3 valores, para serem utilizados
novamente, ver iContador=4
                       fpMxNow:=fMx;
                       fpMyNow:=fMy;
                   end;
                if iContador=2 then
                   begin
                     aJacobn[1,1]:=(fNdtemp+aJacobn[1,1])/0.1;
                     aJacobn[1,2]:=(fMx+aJacobn[1,2])/0.1;;
                     aJacobn[1,3]:=(fMy+aJacobn[1,3])/0.1;;
                   end;
                if iContador=3 then
                   begin
                     aJacobn[2,1]:=(fNdtemp+aJacobn[2,1])/0.2;
                     aJacobn[2,2]:=(fMx+aJacobn[2,2])/0.2;
                     aJacobn[2,3]:=(fMy+aJacobn[2,3])/0.2;
                   end;
                if iContador=4 then
                   begin
                     aJacobn[3,1]:=(fNdtemp+aJacobn[3,1])/0.2;
                     aJacobn[3,2]:=(fMx+aJacobn[3,2])/0.2;
                     aJacobn[3,3]:=(fMy+aJacobn[3,3])/0.2;
                      // Para comecar a iteracao, preciso comparar os valores iniciais, que foram
obtidos no iContador=1, repare que salvei-os nas variaveis fpNNow, fpMxNow e fpMyNow,
vou agora devolve-las para liberar as variaveis fNdtemp e as suas analogas para momento, os
valores foram salvos no iContador=1
                       fNdtemp:=fpNNow;
                       fMx:=fpMxNow;
                       fMy:=fpMyNow;
                       fpEpsilonC:=0.1;
```

```
                    fpDeltaEpsX:=0.0;
                    fpDeltaEpsY:=0.0;
                  end;
              end;
            // Agora, pode-se fazer o processo iterativo
            // O pascal nao faz divisao de matrizes, entao o Jacobiano deve ser invertido antes
            // O jacobiano invertido sera mantido na variavel Jacoboinv
            varTemp:=aJacobn[1,1]*(aJacobn[2,2]*aJacobn[3,3]-aJacobn[2,3]*aJacobn[3,2])-
aJacobn[1,2]*(aJacobn[3,3]*aJacobn[2,1]-
aJacobn[2,3]*aJacobn[3,1])+aJacobn[1,3]*(aJacobn[2,1]*aJacobn[3,2]-
aJacobn[2,2]*aJacobn[3,1]);
            aJacobInverso[1,1]:=(aJacobn[2,2]*aJacobn[3,3]-
aJacobn[2,3]*aJacobn[3,2])/varTemp;
            aJacobInverso[2,1]:=-(aJacobn[2,1]*aJacobn[3,3]-
aJacobn[2,3]*aJacobn[3,1])/varTemp;
            aJacobInverso[3,1]:=(aJacobn[2,1]*aJacobn[3,2]-
aJacobn[2,2]*aJacobn[3,1])/varTemp;
            aJacobInverso[1,2]:=-(aJacobn[1,2]*aJacobn[3,3]-
aJacobn[1,3]*aJacobn[3,2])/varTemp;
            aJacobInverso[2,2]:=(aJacobn[1,1]*aJacobn[3,3]-
aJacobn[1,3]*aJacobn[3,1])/varTemp;
            aJacobInverso[3,2]:=-(aJacobn[1,1]*aJacobn[3,2]-
aJacobn[1,2]*aJacobn[3,1])/varTemp;
            aJacobInverso[1,3]:=(aJacobn[1,2]*aJacobn[2,3]-
aJacobn[1,3]*aJacobn[2,2])/varTemp;
            aJacobInverso[2,3]:=-(aJacobn[1,1]*aJacobn[2,3]-
aJacobn[1,3]*aJacobn[2,1])/varTemp;
            aJacobInverso[3,3]:=(aJacobn[1,1]*aJacobn[2,2]-
aJacobn[1,2]*aJacobn[2,1])/varTemp;
            fpEpsilonCPrevious:=0.1;
            fpDeltaEpsXPrevious:=0.2;
            fpDeltaEpsYPrevious:=0.2;
            iContador:=0; // Agora o iContador ira contar o numero de iteracoes - caso exceder
20000 a iteracao ira parar
            while
abs(fpEpsilonCPrevious+fpDeltaEpsXPrevious+fpDeltaEpsYPrevious)>=fEpsilonErroELS
do
              begin
                fpNNow:=fNdtemp-fpN;
                fpMxNow:=fMx-fpMxx;
                fpMyNow:=fMy-fpMyy;
                iContador:=iContador+1;
                //Agora, a divisao de matrizes (produto do inverso, na verdade)

fpEpsilonCPrevious:=(aJacobInverso[1,1]*fpNNow+aJacobInverso[1,2]*fpMxNow+aJacobI
nverso[1,3]*fpMyNow)/2;
                fpEpsilonC:=fpEpsilonC-fpEpsilonCPrevious;

fpDeltaEpsXPrevious:=(aJacobInverso[2,1]*fpNNow+aJacobInverso[2,2]*fpMxNow+aJacob
Inverso[2,3]*fpMyNow)/2;
```

```
                fpDeltaEpsX:=fpDeltaEpsX-fpDeltaEpsXPrevious;

fpDeltaEpsYPrevious:=(aJacobInverso[3,1]*fpNNow+aJacobInverso[3,2]*fpMxNow+aJacob
Inverso[3,3]*fpMyNow)/2;
                fpDeltaEpsY:=fpDeltaEpsY-fpDeltaEpsYPrevious;

                fpCurvaturaX:=fpDeltaEpsX/fphx;
                fpCurvaturaY:=fpDeltaEpsY/fphy;
                if fpCurvaturaY<0 then
                  begin
                     if                fpCurvaturaX>0                then
falfatemp:=arctan(fpCurvaturaX/fpCurvaturaY)*180/pi+180;
                     if                fpCurvaturaX<0                then
falfatemp:=arctan(fpCurvaturaX/fpCurvaturaY)*180/pi+180;
                     if fpCurvaturaX=0 then falfatemp:=180;
                  end;
                if fpCurvaturaY>0 then
                  begin
                     if                fpCurvaturaX>0                then
falfatemp:=arctan(fpCurvaturaX/fpCurvaturaY)*180/pi;
                     if                fpCurvaturaX<0                then
falfatemp:=arctan(fpCurvaturaX/fpCurvaturaY)*180/pi+360;
                     if abs(fpCurvaturaX)<=0.00001 then falfatemp:=0;
                  end;
                if abs(fpCurvaturaY)<=0.00001 then
                  begin
                     if fpCurvaturaX>0 then falfatemp:=90;
                     if fpCurvaturaX<0 then falfatemp:=270;
                     if abs(fpCurvaturaX)<=0.00001 then falfatemp:=0;
                  end;

fpCurvatura:=sqrt(fpCurvaturaX*fpCurvaturaX+fpCurvaturaY*fpCurvaturaY);
                RotacaoDePontos(falfatemp,CoordSecT,CoordSec);
                PontosMaximoMinimo(CoordSec,fvmax,fvmin);
                fhalfa:=fvmax-fvmin;
                fpDeltaEpsAlfa:=fpCurvatura*fhalfa;
                fEpsilon1:=fpEpsilonC+fpDeltaEpsAlfa/2;
                fEpsilon2:=fpEpsilonC-fpDeltaEpsAlfa/2;


                //Obtencao N e Mx e My
                RotacaoDePontos(falfatemp,CoordSecT,CoordSec); //inicio, obtencao de N,
Mxx e Myy a partir de falfatemp, fEpsilon1 e fEpsilon2;
                if                 fbArmPas                then
RotacaoDePontos(falfatemp,CoordArmPasT,CoordArmPas);
                if                fbArmAtiv                then
RotacaoDePontos(falfatemp,CoordArmAtivT,CoordArmAtiv);
                PontosMaximoMinimo(CoordSec,fvmax,fvmin);
                fhalfa:=fvmax-fvmin;
                if abs(fEpsilon1)<0.000000001 then fEpsilon1:=0;
```

```
                if abs(fEpsilon2)<0.000000001 then fEpsilon2:=0;
                if fEpsilon1=fEpsilon2 then
                  begin
                     fxEpsilonC2:=10e6;
                     fxalfa:=fxEpsilonC2*fEpsilon1/fEpsilonC2;
                  end;
                if fEpsilon1<>fEpsilon2 then
                  begin
                     fxalfa:=fhalfa*fEpsilon1/(fEpsilon1-fEpsilon2);
                     fxEpsilonC2:=fhalfa*fEpsilonC2/(fEpsilon1-fEpsilon2);      //Explicacoes
na secao 4.1.2
                  end;

fNdtemp:=ForcaConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

fMtempx:=MomentoXXConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

fMtempy:=MomentoYYConcreto(CoordSec,ffcd,fxalfa,fxEpsilonC2,fNTensaoConcreto);

                if fbArmPas then
                  begin
                     for fTemp:=0 to Length(CoordArmPasT)-1 do // Inicio força do aço
                       begin
                          ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                          aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;

fvarMtemp:=aAs[fTemp]*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd); // Calcula-se a força
na barra e ela é salva numa variavel temporaria
                          fNdtemp:=fNdtemp+fvarMtemp;
                          fMtempx:=fMtempx+fvarMtemp*CoordArmPas[fTemp].yi;
                          fMtempy:=fMtempy+fvarMtemp*CoordArmPas[fTemp].xi;
                       end; //fim força do aço
                  end;

                if fbArmAtiv then
                   begin
                     for fTemp:=0 to Length(CoordArmAtivT)-1 do // Inicio força do aço
                       begin
                          ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                          aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-
fEpsilon2)/fhalfa;
                          varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];

fvarMtemp:=aAp[fTemp]*TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp);   //
Calcula-se a força na barra e ela é salva numa variavel temporaria
                          fNdtemp:=fNdtemp+fvarMtemp;
                          fMtempx:=fMtempx+fvarMtemp*CoordArmAtiv[fTemp].yi;
                          fMtempy:=fMtempy+fvarMtemp*CoordArmAtiv[fTemp].xi;
                       end; //fim força do aço
```

```
                    end;

                if abs(fMtempx)<0.000001 then fMtempx:=0;
                if abs(fMtempy)<0.000001 then fMtempy:=0;

                fMx:=fMtempx*cos(-falfatemp*3.14159265359/180)+fMtempy*sin(-
falfatemp*3.14159265359/180); //Girando de U para X
                fMy:=-fMtempx*sin(-falfatemp*3.14159265359/180)+fMtempy*cos(-
falfatemp*3.14159265359/180); //Girando de V para Y

                if abs(fMx)<0.000001 then fMx:=0;
                if abs(fMy)<0.000001 then fMy:=0;

                //Fim da obtencao N Mx e My
                if iContador=20000 then
                   begin
                     fpEpsilonCPrevious:=0;
                     fpDeltaEpsXPrevious:=0;
                     fpDeltaEpsYPrevious:=0;
                   end;

              end;


    end;
          //while acaba aqui
          fCurvatura:=(fEpsilon1-fEpsilon2)/fhalfa/10;
          MemoProcurar2.Append('Eps1 = '+FloatToStrF(fEpsilon1,ffGeneral,5,4)+'; Eps2 =
'+FloatToStrF(fEpsilon2,ffGeneral,5,4)+';                    Angulo                    =
'+FloatToStrF(falfatemp,ffGeneral,5,4)+';          Curv          1/r          (1/m)          =
'+FloattostrF(fCurvatura,ffGeneral,6,4));
          MemoProcurar2.Append('Numero de iteracoes = '+FloatToStr(iContador));
          MemoProcurar2.Append('');
          MemoProcurar2.Append('ESTADO-LIMITE DE SERVIÇO: ABERTURA DE
FISSURAS');
          MemoProcurar2.Append('');

          //Agora vou incluir as tensões e abertura de fissura em cada armadura
          if fbArmPas then
            begin
              MemoProcurar2.Append('   Abertura de fissuras na armadura passiva:');
              for fTemp:=0 to Length(CoordArmPas)-1 do // Inicio momento do aço
                 begin
                   ahsi[fTemp]:=fvmax-CoordArmPas[fTemp].yi;
                   aEpsilonsi[fTemp]:=fEpsilon1-ahsi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                   fvarMtemp:=0;
                   if                    aAs[fTemp]>0                    then
fvarMtemp:=exp(ln(aAs[fTemp]*4/3.14159265359)*0.5)*10; //Bitola (mm) - calculada a
partir da area
                   fvarMtemp:=fvarMtemp/(12.5*2.25); //Primeiro termo
```

```
                    fvarMtemp:=fvarMtemp*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd)/fEs;
// Segundo termo

fvarMtemp:=fvarMtemp*3*TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd)/(0.1*0.3*exp(ln(ffck
)*0.666667)); // Terceiro termo
                    if aEpsilonsi[fTemp]>0 then fvarMtemp:=0;
                    MemoProcurar2.Append('                    Barra    nr.:
'+FloatToStrF(fTemp+1,ffGeneral,6,4)+';          Tensao          (KN/cm²):
'+floattostrF(TensaoArmPas(aEpsilonsi[fTemp],fEs,ffyd),ffGeneral,6,4)+';  wk,1  (mm)  :
'+floattostrF(fvarMtemp,ffGeneral,6,4));
                  end; //fim da abertura de fissuras da armadura passiva
              end;
            if fbArmPas=false then
              begin
                MemoProcurar2.Append('   Nao ha armadura passiva para calculo da abertura
de fissura');
              end;

            if fbArmAtiv then
              begin
                MemoProcurar2.Append('');
                MemoProcurar2.Append('   Abertura de fissuras na armadura ativa:');
                for fTemp:=0 to Length(CoordArmAtiv)-1 do
                  begin
                    ahpi[fTemp]:=fvmax-CoordArmAtiv[fTemp].yi;
                    aEpsilonpsi[fTemp]:=fEpsilon1-ahpi[fTemp]*(fEpsilon1-fEpsilon2)/fhalfa;
                    varTemp:=aEpsilonpsi[fTemp]+aEpsilonprealong[fTemp];
                    fvarMtemp:=0;
                    fvarMtemp:=aDiametroAp[fTemp]/(12.5*fEtaP1);  //Primeiro  termo  -
Bitola (mm) - deve ser fornecida pelo usuario

fvarMtemp:=fvarMtemp*(TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp)-
TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,aEpsilonprealong[fTemp]))/fEp;  //
Segundo termo

fvarMtemp:=fvarMtemp*3*(TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp)-
TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,aEpsilonprealong[fTemp]))/(0.1*0.3*exp
(ln(ffck)*0.666667)); // Terceiro termo
                    if aEpsilonpsi[fTemp]>0 then fvarMtemp:=0;
                    MemoProcurar2.Append('                    Barra    nr.:
'+FloatToStrF(fTemp+1,ffGeneral,6,4)+';          Tensao          (KN/cm²):
'+floattostrF((TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,varTemp)-
TensaoArmAtiv(fEpsilonpyd,fEpsilonpu,ffpyd,ffptd,aEpsilonprealong[fTemp])),ffGeneral,6,4
)+'; wk,1 (mm) : '+floattostrF(fvarMtemp,ffGeneral,6,4));
                  end; //fim da abertura de fissuras da armadura passiva
              end;
            if fbArmAtiv=false then
              begin
                MemoProcurar2.Append('   Nao ha armadura ativa para calculo da abertura de
fissura');
```

```
            end;


            MemoProcurar2.Append('');
            MemoProcurar2.Append('');
            MemoProcurar2.Append('ESTADO-LIMITE DE FORMACAO DE FISSURAS');
            MemoProcurar2.Append('Menor deformacao que ocorre na secao =
'+FloatToStrF(fEpsilon2,ffGeneral,5,4));
            MemoProcurar2.Append('Menor deformacao permitida = '+FloatToStrF(-
fEpsilonELSF,ffGeneral,5,4));
            if fEpsilon2>=-fEpsilonELSF then MemoProcurar2.Append('ELS-F:
ATENDIDA.');
            if fEpsilon2<-fEpsilonELSF then MemoProcurar2.Append('ELS-F: NAO
ATENDIDA.');
            MemoProcurar2.Append('');
            MemoProcurar2.Append('');
            MemoProcurar2.Append('ESTADO-LIMITE DE DESCOMPRESSAO');
            MemoProcurar2.Append('Menor deformacao que ocorre na secao =
'+FloatToStrF(fEpsilon2,ffGeneral,5,4));
            MemoProcurar2.Append('Menor deformacao permitida = 0');
            if fEpsilon2>=0 then MemoProcurar2.Append('ELS-D: ATENDIDA.');
            if fEpsilon2<0 then MemoProcurar2.Append('ELS-D: NAO ATENDIDA.');
            MemoProcurar2.Append('');
            if iContador=20000 then MemoProcurar2.Clear;
            if iContador=20000 then MemoProcurar2.Append('Processo travou - resultados
desconhecidos');
            if iContador=20000 then MemoProcurar2.Append('As solicitacoes podem nao
atender o ELU!');
end;


end.
```

ARQUIVO OUTRO

```
unit leoufscarfoc;

{$mode objfpc}{$H+}

interface

type
  Tcoord = Record
    npol:integer;
    xi:real;
    yi:real;
  end;
```

```
function Ncd1(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;
function Mcd1xx(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;
function Mcd1yy(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;
function Ncd2(x1,y1,x2,y2,y,fcd:Real):Real;
function Mcd2xx(x1,y1,x2,y2,y,fcd:Real):Real;
function Mcd2yy(x1,y1,x2,y2,y,fcd:Real):Real;
procedure ObterCaractGeomet(const Coord:array of Tcoord; var fpAc,fpxcg, fpycg:real);
procedure TranslacaoDePontos(const Coord:array of Tcoord; var fpxcg,fpycg:real;var
CoordT:array of Tcoord);
procedure RotacaoDePontos(const Angulo:real;Coord:array of Tcoord; var CoordR:array of
TCoord);
procedure PontosMaximoMinimo(const Coord:array of Tcoord;var fpvmax,fpvmin:real);
function TensaoArmPas(var varEpsilonS,varEs,varfyd:Real):real;
function                                                    TensaoArmAtiv(var
varEpsilonpyd,varEpsilonpu,varfpyd,varfptd,varEpsilont:Real):real;
function                          ForcaConcreto(Poligono:array                    of
Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;
function                       MomentoXXConcreto(Poligono:array                   of
Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;
function                       MomentoYYConcreto(Poligono:array                   of
Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;

implementation

function Ncd1(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;
var
ffuncc,ffuncd,ffunce:real;

begin
  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar
uma faixa de segurança, y1/y2>0,0001, por ex
     begin
       ffuncc:=((y1*x2-y2*x1)/(y1-y2));
       ffuncd:=((x1-x2)/(y1-y2));
     end;
  if y1=y2 then // Todas as condições já excluem a chance de y1=y2, mesmo assim, é melhor
prevenir, criando uma certa seguranca y1/y2>0,0001, por ex
    begin
       ffuncc:=0;
       ffuncd:=0;
     end;
   if (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)<=0 then ffunce:=0; // Isto é, logaritmo neperiano
de um numero extremamente pequeno - mas maior e diferente de zero!
   if  (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)>0  then  ffunce:=exp(ln(((ymax-x)+xEpsilonC2-
y)/xEpsilonC2)*(ffcnNconcreto+1)); // Rodar a função!
```

```
  Result:=-0.85*fcd*(-(xEpsilonC2*ffunce*(ffuncc*(ffcnNconcreto+2)+ffuncd*((ymax-
x)+xEpsilonC2+ffcnNconcreto*y+y)))
  /((ffcnNconcreto+1)*(ffcnNconcreto+2))
    -ffuncc*y-(ffuncd*y*y)/2);
end;


function Mcd1xx(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;

var
ffuncc,ffuncd,ffunce:real;

begin
  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar
uma faixa de segurança, y1/y2>0,0001, por ex
     begin
       ffuncc:=((y1*x2-y2*x1)/(y1-y2));
       ffuncd:=((x1-x2)/(y1-y2));
     end;
  if y1=y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma
faixa de segurança, y1/y2>0,0001, por ex
    begin
       ffuncc:=0;
       ffuncd:=0;
     end;
   if (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)<=0 then ffunce:=0; // Isto é, logaritmo neperiano
de um numero extremamente pequeno - mas maior e diferente de zero!
   if  (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)>0  then  ffunce:=exp(ln(((ymax-x)+xEpsilonC2-
y)/xEpsilonC2)*(ffcnNconcreto+1)); // Rodar a função!


  Result:=(0.85*fcd*(3*ffuncc*((1+ffcnNconcreto)*(2+ffcnNconcreto)*
        (3+ffcnNconcreto)*y*y+2*xEpsilonC2*ffunce*(((ymax-x)+xEpsilonC2)*
        (3+ffcnNconcreto)+(3+4*ffcnNconcreto+ffcnNconcreto*ffcnNconcreto)*y))+
        2*ffuncd*((1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto)*y*y*y+
        3*xEpsilonC2*ffunce*(2*(ymax-x)*(ymax-x)+2*xEpsilonC2*xEpsilonC2+
        2*xEpsilonC2*(1+ffcnNconcreto)*y+
        (2+3*ffcnNconcreto+ffcnNconcreto*ffcnNconcreto)*y*y
        +2*(ymax-x)*(2*xEpsilonC2+(1+ffcnNconcreto)*y)))))
        /(6*(1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto))
        ;
end;


function Mcd1yy(x1,y1,x2,y2,ymax,x,y,fcd,xEpsilonC2,ffcnNconcreto:Real):Real;
var
ffuncc,ffuncd,ffunce:real;

begin
```

```
  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=((y1*x2-y2*x1)/(y1-y2));
      ffuncd:=((x1-x2)/(y1-y2));
    end;
 if y1=y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
   begin
      ffuncc:=0;
      ffuncd:=0;
    end;
  if (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)<=0 then ffunce:=0; // Isto é, logaritmo neperiano de um numero extremamente pequeno - mas maior e diferente de zero!
  if  (((ymax-x)+xEpsilonC2-y)/xEpsilonC2)>0  then  ffunce:=exp(ln(((ymax-x)+xEpsilonC2-y)/xEpsilonC2)*(ffcnNconcreto+1)); // Rodar a função!

 Result:=(0.85*fcd*(3*ffuncc*ffuncc*(xEpsilonC2*
 (6+5*ffcnNconcreto+ffcnNconcreto*ffcnNconcreto)*ffunce+
 (1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto)*y)+
 3*ffuncc*ffuncd*((1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto)*y*y+
 2*xEpsilonC2*ffunce*(((ymax-x)+xEpsilonC2)*(3+ffcnNconcreto)+
 (3+4*ffcnNconcreto+ffcnNconcreto*ffcnNconcreto)*y))+
 ffuncd*ffuncd*((1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto)*y*y*y+
 3*xEpsilonC2*ffunce*(2*(ymax-x)*(ymax-x)+
 2*xEpsilonC2*xEpsilonC2+2*xEpsilonC2*(1+ffcnNconcreto)*y+
 (2+3*ffcnNconcreto+ffcnNconcreto*ffcnNconcreto)*y*y+
 2*(ymax-x)*(2*xEpsilonC2+(1+ffcnNconcreto)*y)))))/
 (6*(1+ffcnNconcreto)*(2+ffcnNconcreto)*(3+ffcnNconcreto))

 ;


end;

function Ncd2(x1,y1,x2,y2,y,fcd:Real):Real;
var
ffuncc,ffuncd:real;

begin

  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=((y1*x2-y2*x1)/(y1-y2));
      ffuncd:=((x1-x2)/(y1-y2));
    end;
 if y1=y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
   begin
```

```
      ffuncc:=0;
      ffuncd:=0;
    end;
  Result:=0.85*fcd*y*(ffuncc+ffuncd*y/2);


end;

function Mcd2xx(x1,y1,x2,y2,y,fcd:Real):Real;
var
ffuncc,ffuncd:real;

begin

  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=((y1*x2-y2*x1)/(y1-y2));
      ffuncd:=((x1-x2)/(y1-y2));
    end;
  if y1=y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=0;
      ffuncd:=0;
    end;
  Result:=0.85*fcd*(ffuncc*y*y/2+ffuncd*y*y*y/3)
  ;


end;

function Mcd2yy(x1,y1,x2,y2,y,fcd:Real):Real;
var
ffuncc,ffuncd:real;

begin

  if y1<>y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=((y1*x2-y2*x1)/(y1-y2));
      ffuncd:=((x1-x2)/(y1-y2));
    end;
  if y1=y2 then // Todas as condições já excluem a chance de y1=y2, aqui é possível criar uma faixa de segurança, y1/y2>0,0001, por ex
    begin
      ffuncc:=0;
      ffuncd:=0;
    end;
```

```
  Result:=0.85*fcd*(ffuncc*ffuncc*y+ffuncc*ffuncd*y*y+ffuncd*ffuncd*y*y*y/3)/2
  ;


end;

procedure ObterCaractGeomet(const Coord:array of Tcoord; var fpAc,fpxcg, fpycg:real);
var
fSxx,fSyy:real;
fContador:integer;

begin
    fpAc:=0;
    fSxx:=0;
    fSyy:=0;

    for fContador:=0 to Length(Coord)-2 do
    begin
       fpAc:=fpAc+Coord[fContador].xi*Coord[fContador+1].yi-
Coord[fContador+1].xi*Coord[fContador].yi;
       fSxx:=fSxx+(Coord[fContador].xi*Coord[fContador+1].yi-
Coord[fContador+1].xi*Coord[fContador].yi)*(Coord[fContador].yi+Coord[fContador+1].yi)
;
       fSyy:=fSyy+(Coord[fContador].xi*Coord[fContador+1].yi-
Coord[fContador+1].xi*Coord[fContador].yi)*(Coord[fContador].xi+Coord[fContador+1].xi)
;
    end;

    fpAc:=fpAc/2;
    fSxx:=fSxx/6;
    fSyy:=fSyy/6;
    fpxcg:=fSyy/fpAc;
    fpycg:=fSxx/fpAc;
end;

procedure TranslacaoDePontos(const Coord:array of Tcoord; var fpxcg,fpycg:real;var
CoordT:array of Tcoord);
var
  fContador:integer;

begin
    for fContador:=0 to Length(Coord)-1 do
    begin
       CoordT[fContador].xi:=Coord[fContador].xi-fpxcg;
       CoordT[fContador].yi:=Coord[fContador].yi-fpycg;
       CoordT[fContador].npol:=Coord[fContador].npol;

    end;
end;
```

```
procedure RotacaoDePontos(const Angulo:real;Coord:array of Tcoord; var CoordR:array of
TCoord);
var
iContador:integer;

  begin
    for iContador:=0 to Length(Coord)-1 do
       begin
          CoordR[iContador].xi:=Coord[iContador].xi*cos(Angulo*pi/180)-
Coord[iContador].yi*sin(Angulo*pi/180);

CoordR[iContador].yi:=Coord[iContador].xi*sin(Angulo*pi/180)+Coord[iContador].yi*cos(
Angulo*pi/180);
          CoordR[iContador].npol:=Coord[iContador].npol;
       end;

  end;

procedure PontosMaximoMinimo(const Coord:array of Tcoord;var fpvmax,fpvmin:real);
var
iContador:integer;

 begin
   fpvmax:=Coord[0].yi;
   fpvmin:=Coord[0].yi;
   for iContador:=1 to Length(Coord)-1 do
      begin
         if Coord[iContador].yi>fpvmax then
           fpvmax:=Coord[iContador].yi;
         if Coord[iContador].yi<fpvmin then
           fpvmin:=Coord[iContador].yi;
      end;
 end;

function TensaoArmPas(var varEpsilonS,varEs,varfyd:Real):real;
begin
   if varEpsilonS<=varfyd/varEs*1000 then  //Trabalhando apenas antes da ruptura por
compressão
      begin
      TensaoArmPas:=-varfyd;
       if varEpsilonS>=-varfyd/varEs*1000 then
         begin
          TensaoArmPas:=varEs*varEpsilonS/1000;
         end;

      end;

   if varEpsilonS>=varfyd/varEs*1000 then
      begin
```

```
        TensaoArmPas:=varfyd;
      end;
end;

function TensaoArmAtiv(var varEpsilonpyd,varEpsilonpu,varfpyd,varfptd,varEpsilont:Real):real;


begin
  Result:=0;
  if varEpsilont<0 then //Lembrar: negativo e tração; e positivo compressao
    begin
      if varEpsilont>=varEpsilonpyd then
        begin
          Result:=varEpsilont*varfpyd/varEpsilonpyd;
        end;
      if varEpsilont<varEpsilonpyd then
        begin
          Result:=varfpyd+(varfptd-varfpyd)/(varEpsilonpu-varEpsilonpyd)*(varEpsilont-
varEpsilonpyd);
        end;
    end;
end;

function ForcaConcreto(Poligono:array of Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;
var
fcnConcContador:integer;
ffcnvmax,ffcnvmin:real;

begin
  Result:=0;
  ffcnvmax:=0;
  ffcnvmin:=0;
  PontosMaximoMinimo(Poligono,ffcnvmax,ffcnvmin);
  for fcnConcContador:=0 to Length(Poligono)-2 do // Inicio do calculo das forças no
concreto
          begin
             if abs(Poligono[fcnConcContador].yi-Poligono[fcnConcContador+1].yi)>0.000001
then // Medida para evitar travar o programa com (Y1 = Y2) que acarreta ( Denominador =
Zero )
               begin

                 if Poligono[fcnConcContador].yi<Poligono[fcnConcContador+1].yi then //
Primeiro ponto MENOR que segundo

                       begin
                          if Poligono[fcnConcContador].yi<ffcnvmax-ffcnxalfa then //Primeiro
Ponto Em tração
                            begin
```

```
                    if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa then
                      begin
                        // Se segundo ponto em tração, fazer nada
                      end;


                    if  Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa  then  //
Segundo ponto numa área comprimida
                      begin
                        if                  Poligono[fcnConcContador+1].yi<=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                          begin
                            Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador+
1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                          end;
                        if                  Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then   // Segundo ponto na denominada area de compressão 2
                          begin
                            Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                        -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                          end;
                      end;




                  end;
                //Primeiro Ponto na primeira área de compressão
                    if                              Poligono[fcnConcContador].yi<ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then
```

```
                    begin
                      if Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa then
                        begin

                          if                    Poligono[fcnConcContador+1].yi<=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                          begin
                            Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador+
1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador].
yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                          end;

                          if                    Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 2
                            begin
                            Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador].
yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                    -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                            end;


                        end;
                    end;

                  if    Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then//Primeiro Ponto na segunda área de compressão
                   begin
                                Result:=Result

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
```

```
                                    -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);


                        end;

                    end; // Fim de (Primeiro ponto Menor que segundo ponto)

                    if Poligono[fcnConcContador+1].yi<Poligono[fcnConcContador].yi then //
Segundo ponto MENOR que primeiro ponto
                     begin

                      if Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa then //Primeiro
Ponto Em tração
                       begin
                         // Nada a fazer
                       end;

                       if Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na primeira área de compressão
                        begin

                          if Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa then
//Primeiro ponto na primeira area de compressão - condição 2
                           begin

                            if Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa then //
Segundo ponto na denominada area 1
                             begin
                              Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador+
1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador].
yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                             end;

                            if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa then //
Segundo ponto na area de tração
                             begin
                              Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
```

```
                                    -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador].
yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                  end;



                               end;
                            end;

                          if    Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na segunda área de compressão
                            begin

                              if                          Poligono[fcnConcContador+1].yi>=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na segunda área de compressão
                              begin
                                Result:=Result

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                    -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);


                              end;

                              if                          Poligono[fcnConcContador+1].yi<ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na primeira área de compressão
                              begin
                                if    Poligono[fcnConcContador+1].yi>=ffcnvmax-ffcnxalfa    then
//Segundo ponto na primeira área de compressão - condição 2
                                begin
                                  Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador+
1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                    -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);
```

```
                    end;
                  end;

                if Poligono[fcnConcContador+1].yi<ffcnvmax-ffcnxalfa then //
Segundo ponto na area tracionada
                  begin
                    Result:=Result

+Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Ncd1(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCont
ador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                        -
Ncd2(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcConta
dor+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);
                    end;

                end;

              end; // Final dos calculos com segundo ponto menor que o primeiro para
compressão do concreto
            end;



          end; // Fim do calculo das forças no concreto
end;
// Força concreto - Fim

function                    MomentoXXConcreto(Poligono:array                    of
Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;
var
fcnConcContador:integer;
ffcnvmax,ffcnvmin:real;

begin

        Result:=0;
        ffcnvmax:=0;
        ffcnvmin:=0;

        PontosMaximoMinimo(Poligono,ffcnvmax,ffcnvmin);
```

```
          for fcnConcContador:=0 to Length(Poligono)-2 do // Inicio do calculo do momento
xx no concreto
              begin
                if                                          abs(Poligono[fcnConcContador].yi-
Poligono[fcnConcContador+1].yi)>0.000001 then // Medida para evitar travar o programa
com (Y1 = Y2) que acarreta ( Denominador = Zero )
                 begin

                    if Poligono[fcnConcContador].yi<Poligono[fcnConcContador+1].yi then //
Primeiro ponto MENOR que segundo

                     begin
                       if Poligono[fcnConcContador].yi<ffcnvmax-ffcnxalfa then //Primeiro
Ponto Em tração
                         begin
                           if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa then
                            begin
                              // Se segundo ponto em tração, fazer nada
                            end;


                           if Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa then //
Segundo ponto numa área comprimida
                             begin
                               if                    Poligono[fcnConcContador+1].yi<=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                                 begin
                                   Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                                -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                 end;
                               if                    Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then   // Segundo ponto na denominada area de compressão 2
                                 begin
                                   Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                                -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
```

```
+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                        -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                                    end;
                                end;




                            end;
                        //Primeiro Ponto na primeira área de compressão
                            if                              Poligono[fcnConcContador].yi<ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then
                             begin
                              if Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa then
                               begin

                                if                      Poligono[fcnConcContador+1].yi<=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                                begin
                                 Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                end;

                                if                      Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 2
                                 begin
                                 Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
```

```
                                    -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                                  end;


                                end;
                              end;

                          if    Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then//Primeiro Ponto na segunda área de compressão
                           begin
                                        Result:=Result

+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                    -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);


                           end;

                        end; // Fim de (Primeiro ponto Menor que segundo ponto)

                        if  Poligono[fcnConcContador+1].yi<Poligono[fcnConcContador].yi  then  //
Segundo ponto MENOR que primeiro ponto
                           begin

                            if  Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa  then  //Primeiro
Ponto Em tração
                             begin
                               // Nada a fazer
                             end;

                            if  Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na primeira área de compressão
                             begin

                                 if         Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa         then
//Primeiro ponto na primeira area de compressão - condição 2
                                  begin

                                    if   Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa   then   //
Segundo ponto na denominada area 1
                                     begin
                                      Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
```

```
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                  end;

                                 if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa then //
Segundo ponto na area de tração
                                  begin
                                    Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                                -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                  end;



                              end;
                            end;

                          if  Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na segunda área de compressão
                            begin

                              if                    Poligono[fcnConcContador+1].yi>=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na segunda área de compressão
                              begin
                                Result:=Result

+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                          -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);



                              end;

                              if                    Poligono[fcnConcContador+1].yi<ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na primeira área de compressão
                              begin
                                if  Poligono[fcnConcContador+1].yi>=ffcnvmax-ffcnxalfa  then
//Segundo ponto na primeira área de compressão - condição 2
```

```
                                begin
                                  Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                                -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                                -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);

                                  end;
                                end;

                              if  Poligono[fcnConcContador+1].yi<ffcnvmax-ffcnxalfa  then  //
Segundo ponto na area tracionada
                                begin
                                  Result:=Result

+Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                                -
Mcd1xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                                -
Mcd2xx(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);
                                end;

                            end;

                          end; // Final dos calculos com segundo ponto menor que o primeiro para
compressão do concreto
                        end;
                      end;// Fim do calculo do momento Mxx no concreto

end;
```

```
function MomentoYYConcreto(Poligono:array of Tcoord;ffcnfcd,ffcnxalfa,ffcnxEpsilonC2,ffcnNconcreto:real):real;
var
fcnConcContador:integer;
ffcnvmax,ffcnvmin:real;

begin

    Result:=0;
    ffcnvmax:=0;
    ffcnvmin:=0;

    PontosMaximoMinimo(Poligono,ffcnvmax,ffcnvmin);
    for fcnConcContador:=0 to Length(Poligono)-2 do // Inicio do calculo do momento yy no concreto
            begin

              if abs(Poligono[fcnConcContador].yi-Poligono[fcnConcContador+1].yi)>0.000001 then // Medida para evitar travar o programa com (Y1 = Y2) que acarreta ( Denominador = Zero )
                begin
                  if Poligono[fcnConcContador].yi<Poligono[fcnConcContador+1].yi then // Primeiro ponto MENOR que segundo

                    begin
                      if Poligono[fcnConcContador].yi<ffcnvmax-ffcnxalfa then //Primeiro Ponto Em tração
                        begin
                          if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa then
                            begin
                              // Se segundo ponto em tração, fazer nada
                            end;


                          if Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa then // Segundo ponto numa área comprimida
                            begin
                              if Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                                begin
                                  Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcContador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContador+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcContador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                                end;
```

```
                    if                    Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then   // Segundo ponto na denominada area de compressão 2
                      begin
                        Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                    -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                      end;
                  end;




              end;
            //Primeiro Ponto na primeira área de compressão
              if  Poligono[fcnConcContador].yi<ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then
               begin
                 if Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa then
                  begin

                    if                    Poligono[fcnConcContador+1].yi<=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 1
                      begin
                        Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                    -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                      end;

                    if                    Poligono[fcnConcContador+1].yi>ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then // Segundo ponto na denominada area 2
                      begin
                      Result:=Result
```

```
+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                        -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                        -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd);
                                    end;


                                  end;
                               end;

                        if      Poligono[fcnConcContador].yi>=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then//Primeiro Ponto na segunda área de compressão
                         begin
                                        Result:=Result

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
                                        -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);


                         end;

                      end; // Fim de (Primeiro ponto Menor que segundo ponto)

                      if  Poligono[fcnConcContador+1].yi<Poligono[fcnConcContador].yi  then  //
Segundo ponto MENOR que primeiro ponto
                       begin

                         if   Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa   then   //Primeiro
Ponto Em tração
                          begin
                            // Nada a fazer
                          end;

                          if    Poligono[fcnConcContador].yi<=ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na primeira área de compressão
                           begin
```

```
                    if Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa then  //Primeiro
ponto na primeira area de compressão - condição 2
                     begin

                      if  Poligono[fcnConcContador+1].yi>ffcnvmax-ffcnxalfa  then  //
Segundo ponto na denominada area 1
                       begin
                        Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                  -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                       end;

                      if  Poligono[fcnConcContador+1].yi<=ffcnvmax-ffcnxalfa  then  //
Segundo ponto na area de tração
                       begin
                        Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                  -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContado
r].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto);
                       end;



                     end;
                    end;

                   if   Poligono[fcnConcContador].yi>ffcnvmax-ffcnxalfa+ffcnxEpsilonC2
then //Primeiro Ponto na segunda área de compressão
                    begin

                     if                  Poligono[fcnConcContador+1].yi>=ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na segunda área de compressão
                      begin
                       Result:=Result

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador+1].yi,ffcnfcd)
```

```
                                -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);


                          end;

                        if                          Poligono[fcnConcContador+1].yi<ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2 then //Segundo ponto na primeira área de compressão
                          begin
                            if     Poligono[fcnConcContador+1].yi>=ffcnvmax-ffcnxalfa     then
//Segundo ponto na primeira área de compressão - condição 2
                            begin
                              Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,Poligono[fcnConcContad
or+1].yi,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);

                            end;
                          end;

                        if     Poligono[fcnConcContador+1].yi<ffcnvmax-ffcnxalfa     then     //
Segundo ponto na area tracionada
                          begin
                            Result:=Result

+Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)
                                -
Mcd1yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax,ffcnxalfa,ffcnvmax-
ffcnxalfa+ffcnxEpsilonC2,ffcnfcd,ffcnxEpsilonC2,ffcnNconcreto)

+Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcC
ontador+1].xi,Poligono[fcnConcContador+1].yi,ffcnvmax-ffcnxalfa+ffcnxEpsilonC2,ffcnfcd)
                                -
Mcd2yy(Poligono[fcnConcContador].xi,Poligono[fcnConcContador].yi,Poligono[fcnConcCo
ntador+1].xi,Poligono[fcnConcContador+1].yi,Poligono[fcnConcContador].yi,ffcnfcd);
```

```
                    end;

                end;

            end; // Final dos calculos com segundo ponto menor que o primeiro para compressão do concreto
        end;

    end; // Fim do calculo do momento yy no concreto
end;


end.
```