• U C •

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE
ENGENHARIA MECÂNICA

Estudo numérico de propagação de fendas por fadiga em juntas soldadas em T

Dissertação apresentada para a obtenção do grau de Mestre em Engenharia Mecânica na Especialidade de Produção e Projeto

Numerical study of fatigue crack growth in welded T-joints

Autor

André Lopes Almeida

Orientadores

Professor Doutor Fernando Jorge Ventura Antunes

Professor Doutor Ricardo Nuno Madeira Soares Branco

Júri

Presidente	**Professor Doutor José Domingos Moreira da Costa** **Professor Associado com Agregação da Universidade de Coimbra**
Vogais	**Professor Joel Alexandre da Silva de Jesus** **Assistente Convidado da Universidade de Coimbra**
Orientador	**Professor Doutor Ricardo Nuno Madeira Soares Branco** **Professor Adjunto do Instituto Politécnico de Coimbra**

Coimbra, julho, 2016

“Eu tentei 99 vezes e falhei,
mas na centésima tentativa
eu consegui.
Nunca desista dos seus objetivos
mesmo que esses pareçam
impossíveis,
a próxima tentativa
pode ser a vitoriosa.”
Albert Einstein

Aos meus pais e irmãos.

Agradecimentos

A realização da presente dissertação tornou-se possível graças ao contributo e apoio de várias pessoas, auxílio esse fundamental para a consecução deste projeto. Deste modo, não poderia neste momento deixar de dirigir os meus sinceros agradecimentos:

Ao orientador, Professor Doutor Ricardo Nuno Moreira Soares Branco, por toda a disponibilidade demonstrada, por toda a ajuda preciosa que deu de forma incondicional e todo o conhecimento que me transmitiu, facilitando todo o trabalho.

Ao orientador, Professor Doutor Fernando Jorge Ventura Antunes agradeço a disponibilidade e os conselhos dados na escolha do tema da dissertação.

À minha família, especialmente aos meus pais, por toda a motivação, apoio e paciência ao longo desta longa caminhada. Devo-lhes muito do que sou hoje, e sei que sem eles e sem as condições que me disponibilizaram, não seria possível alcançar os meus objetivos.

Aos meus irmãos, Filipe e Daniel, por toda a força e confiança que depositaram em mim, motivando-me sempre para fazer mais e melhor.

Aos meus amigos, que foram um dos pilares mais importantes neste percurso, aconselhando-me, motivando-me e garantindo que sinta orgulho no meu percurso. São todos enormes e especiais.

Aos Bombeiros de Aguiar da Beira, pelo apoio e pela garra e espírito de equipa que me ensinaram a ter, independentemente das adversidades.

À Professora Doutora Marta Cristina Cardoso de Oliveira e Professora Doutora Ana Paula Bettencourt Martins Amaro pela disponibilização do *template* e pela ajuda.

À Fundação para a Ciência e Tecnologia e ao Programa Operacional Temático Fatores de Competitividade (COMPETE), comparticipado pelo fundo comunitário Europeu FEDER (Projeto PTDC/EMS-PRO/1356/2014; COMPETE: T449508144-00019113).

Agradecimentos

Resumo

A fadiga é um processo de alteração estrutural permanente, progressivo e localizado, que resulta de cargas cíclicas aplicadas em componentes, sendo estas responsáveis pela grande parte das falhas de serviço e pela rotura dos materiais. O comportamento à fadiga de juntas soldadas, continua a ser uma área muito investigada. Apesar dos grandes avanços conseguidos na tecnologia da soldadura, as ligações soldadas continuam a ser um ponto crítico e suscetível à iniciação e propagação de defeitos por fadiga. A investigação de fenómenos de fadiga com base em abordagens numéricas, devido ao crescente aumento das velocidades de processamento dos computadores da atualidade, tem-se revelado muito atrativa. De entre as várias técnicas numéricas, destaca-se o método dos elementos finitos, que permite simular geometrias e carregamentos complexos.

Neste trabalho, o principal objetivo consiste no desenvolvimento de um procedimento automático, baseado no método dos elementos finitos, para o estudo de fenómenos de propagação de fendas por fadiga em juntas soldadas em forma de T sujeitas a cargas cíclicas em Modo-I. O procedimento procura implementar uma técnica de remalhagem adaptativa com múltiplos graus de liberdade para a frente de fenda, e é desenvolvido usando o ambiente de programação Visual Basic 2012. As malhas de elementos finitos são constituídas por três regiões: uma região em forma de teia de aranha junto à frente de fenda; uma malha de transição que procura criar uma caixa retangular; e uma malha regular que preenche o resto da peça. Os campos de deslocamentos são obtidos a partir da ferramenta computacional COSMOSM. Os fatores de intensidade de tensão ao longo da frente de fenda são calculados com o método direto da extrapolação com dois pontos. Os avanços da frente de fenda são obtidos através de curvas $da/dN - \Delta K$. As frentes de fenda são definidas usando funções *cubic spline*. O procedimento desenvolvido tem elevada flexibilidade: permite estudar diferentes formas de fendas, incluindo fendas de canto, e fendas superficiais; diferentes geometrias das peças a unir; diferentes formas de cordões e níveis de penetração da soldadura; diferentes tipos de carga, incluindo tração, flexão e tração-flexão; entre outros.

Palavras-chave: Junta soldada em forma de T, Propagação de fenda por fadiga, Propagação automática de fenda, Técnica de remalhagem adaptativa, Método dos elementos finitos.

Abstract

Fatigue is a progressive and localised structural damage caused by cyclic loading, which are largely responsible for the failure of materials. The fatigue behavior of welded joints has been extensively studied over the years. Despite the great advances in welding technology, welded connections are susceptible to fatigue crack initiation and fatigue crack propagation. The research of fatigue phenomena from numerical approaches, due to high processing capacity of today's computers, has proved to be very attractive. Among the various advanced numerical techniques, the finite element method is one of the most efficient because it can handle complex geometries and complex loading.

In this work, the main objective is to develop an automatic procedure based on the finite element method to the study fatigue crack growth phenomena in welded T-joints subject to Mode-I cyclic loading. The procedure implements an adaptive re-meshing technique with multiple degrees of freedom along the crack front using the Visual Basic 2012 programming language. The finite element meshes combine three regions: a spider web mesh near the crack front; a transition mesh whose main aim is to create a rectangular box; and a regular mesh that fills the rest of the body. The displacement fields are obtained from the COSMOSM computational tool. The stress intensity factors along the crack fronts are calculated using the direct method of extrapolation with two points. The advances of the crack front nodes are obtained from $da/dN - \Delta K$ curves. The crack front shapes are defined through cubic spline functions. The procedure developed is very flexible. It allows to study different crack front shapes, including corner cracks and surface cracks; different geometries of the base materials; different welding geometries and penetration levels; different loading scenarios, including tension, bending and both; among others.

Keywords Joint welded T-shaped, Propagation of fatigue cracks, Crack propagation automatic, Adaptive remeshing technique, Finite Element Method.

Abstract

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIMBOLOGIA E SIGLAS

Simbologia

a – Comprimento da fenda

a_f – Tamanho máximo da fenda

a_i – Tamanho mínimo da fenda

b – Largura da fenda

C – Constante da Lei de Paris

d – Diâmetro

da/dN – Velocidade de propagação da fenda

da_i – Incremento local de fenda

E – Módulo de Young

E' – Módulo de Young modificado

$F_{máx}$ – Força máxima

$F_{mín}$ – Força mínima

$f(x_i)$ – Função

$f''(x_i)$ – Derivada de segunda ordem da função $f''(x_i)$

k – Número de nós

K – Fator de intensidade de tensão

K_{Ic} – Tenacidade à fratura

$K_{máx}$ – Fator de intensidade de tensão máximo

$K_{mín}$ – Fator de intensidade de tensão mínima

L_1 – Tamanho radial dos elementos adjacentes à frente de fenda

L_2 – Tamanho radial do segundo anel da malha em teia de aranha

L_3 – Tamanho radial do terceiro anel da malha em teia de aranha

L_4 – Tamanho radial da malha de transição

Lig (1) – Nível de penetração da soldadura do cordão fissurado

Lig (2) – Nível de penetração da soldadura do cordão não-fissurado

m – Expoente da Lei de Paris

n – Número de intervalos

NELi – Variável geométrica que controla a densidade da malha

N_f – Número de ciclos de fadiga

$N^{(j)}$ – Número de ciclos de carga para o incremento j

r – Raio do sistema de coordenadas polares

R – Razão de tensão

r_P – Distância radial entre o nó P e a ponta da fenda

$r^{-\lambda}$ – Singularidade do campo de tensão

t – Largura da peça

T_i – Variável geométrica da malha de transição

v_B – Deslocamento normal do plano da fenda para o ponto B

v_P – Deslocamento normal do plano da fenda para o ponto P

W – Comprimento da peça

x_i – Coordenada x do nó i

Xi – Variável geométrica da peça na direção x

Yi – Variável geométrica da peça na direção y

Y – Fator geométrico

Zi – Variável geométrica da peça na direção z

α_1 – Declive junto à raiz do cordão

$\Delta a^{(j)}$ – Incremento da fenda do passo j

$\Delta a_{máx}^{(j)}$ – Incremento máximo da fenda do passo j

ΔK – Gama do fator de intensidade de tensão

$\Delta K^{(j)}$ – Gama do fator de intensidade de tensão do passo j

ΔK_i – Gama do fator de intensidade de tensão do nó i

ΔK_{lf} – Limiar de propagação de propagação de fenda por fadiga

$\Delta K_{máx}^{(j)}$ – Valo máximo da gama do fator de intensidade de tensão do passo j

θ – Ângulo no sistema de coordenadas polares

σ – Tensão remota aplicada

$\sigma_{máx}$ – Tensão remota máxima

$\sigma_{mín}$ – Tensão remota mínima

κ – Parâmetro elástico

ν – Coeficiente de Poisson

Siglas

EF – Elementos Finitos

MEF – Método dos Elementos Finitos

MFLE – Mecânica da Fratura Linear Elástica

SIF – *Stress Intensity Factor* (Fator de Intensidade de Tensão)

ASCII - *American Standard Code for Information Interchange* (Código Americano Padrão para o Intercâmbio de Informação)

1. INTRODUÇÃO

1.1. Enquadramento

No dia a dia, são várias as máquinas ou equipamentos cujos componentes estão sujeitos a cargas cíclicas. Estas cargas normalmente levam ao fim de um determinado tempo ao dano dos componentes. A ocorrência destes danos, pode causar paragens inesperadas na produção, que não estavam previstas nas tarefas de manutenção, e que podem conduzir a prejuízos económicos e até, em casos extremos, a acidentes de trabalho.

O fenómeno de fadiga é um processo de alteração estrutural permanente, progressivo e localizado, que resulta das cargas anteriormente mencionadas, sendo responsável pela maior parte das falhas de serviço (cerca de 90 %) e pela rotura dos materiais. Este fenómeno de fadiga é dividido em três etapas: i) nucleação e iniciação da fenda; ii) crescimento macroscópico da fenda, onde há uma propagação estável da fenda; iii) e, por fim, verifica-se a rotura do material.

Ao longo dos tempos, foram efetuados diversos testes e experiências, que contribuíram para muitos avanços no conhecimento desta área. No entanto, apesar destes avanços, existem ainda muitos outros focos de investigação em aberto, uma vez que se trata de um fenómeno bastante complexo, afetado por inúmeras variáveis. Um dos assuntos que atualmente continua a ser muito investigado é o comportamento à fadiga de juntas soldadas.

Devido aos avanços conseguidos na tecnologia da soldadura, este método de ligação estrutural tem vindo a ser cada vez mais utilizado. Porém, apesar de todos os avanços observados nas últimas décadas, o cordão de soldadura, devido à inevitável concentração de tensões associada, presença potencial de defeitos, bem como a existência de um campo de tensões residuais, continua a ser um ponto crítico e suscetível à iniciação e propagação de defeitos.

Neste sentido, torna-se muito importante a criação, por parte dos projetistas, de modelos de previsão de vida fiáveis, para que deste modo se possam definir de forma precisa as geometrias mais adequadas para cada caso.

O estudo do comportamento à fadiga dos cordões de soldadura tem sido realizado quer por via experimental, quer numérica. No entanto, devido ao aumento da capacidade de processamento dos computadores, bem como o sucesso verificado na aplicação de métodos numéricos avançados às ciências de engenharia, a abordagem numérica tem-se revelado uma abordagem cada vez mais atrativa.

No que diz respeito à propagação de fendas por fadiga, o desenvolvimento de modelos numéricos baseados na técnica de propagação automática tem sido uma via seguida. Esta técnica tem sido, essencialmente, implementada usando o método dos elementos finitos.

A técnica de propagação automática (Smith e Cooper, 1989) pode dividir-se em cinco etapas principais: (i) desenvolvimento de um modelo de elementos finitos tridimensional representativo; (ii) cálculo do campo de deslocamentos; (iii) cálculo dos fatores de intensidade de tensão efetiva ao longo da frente de fenda; (iv) determinação dos avanços da frente de fenda aplicando uma lei de propagação adequada; (v) definição de uma nova frente de fenda através das posições dos nós obtidos no passo anterior.

Esta técnica necessita do desenvolvimento de um conjunto de subrotinas específicas, de relativa complexidade, que não se encontram disponíveis nos principais *softwares* de elementos finitos utilizados na atualidade (tais como ABAQUS, ANSYS, PATRAN, MARC, etc.).

Embora esta seja uma técnica muito versátil e que tem permitido, com grande sucesso, efetuar estudos muito diversificados ao longo dos últimos 30 anos (Branco *et al*., 2015) a sua aplicação no campo das juntas soldadas tem sido limitada, em certo ponto, devido a complexidade da malha, às exigências geométricas associadas a este tipo de situações e o comportamento local do material.

Assim neste trabalho, pretende-se desenvolver, otimizar e validar um modelo de elementos finitos tridimensional, que será baseado no conceito de propagação automática de fenda, para geometrias soldadas em forma de T, sujeitas a cargas em Modo-I (ver Figura 2.6). Este modelo deve ser suficientemente flexível para acomodar diferentes formas iniciais de fenda, diferentes perfis de cordão de soldadura, e também diferentes geometrias das peças a unir.

1.2. Objetivos

Este trabalho, como foi mencionado anteriormente, tem como objetivo desenvolver um modelo numérico tridimensional baseado na técnica de propagação automática (Smith e Cooper, 1989) para analisar juntas soldadas em forma de T sujeitas a cargas cíclicas em Modo-I.

Pretende-se que este modelo numérico seja bastante flexível de modo a permitir acomodar diferentes formas iniciais de fenda definidas a partir de um número variável de nós. Além disso, o modelo deve permitir simular, quer fendas superficiais, quer fendas passantes. Relativamente ao carregamento, devem ser previstos três cenários principais, nomeadamente tração, flexão e tração-flexão. A necessidade de flexibilidade para acomodar diferentes formas de perfis para cada um dos cordões de soldadura da junta em T também será tida em conta ao longo deste trabalho. Além disso, pretende-se, ainda, que o nível de penetração da junta soldada das peças a unir possa ser variável.

O modelo numérico será desenvolvido a partir do método dos elementos finitos com base numa técnica de remalhagem adaptativa com *n* graus de liberdade que calcula, de forma individual, os avanços locais de todos os nós de canto da frente de fenda (Branco *et al*., 2015). A malha tridimensional de elementos finitos será constituída por três regiões principais: uma malha em forma de teia de aranha, centrada na frente de fenda, e composta por vários anéis concêntricos; uma malha de transição que tem como objetivo efetuar uma transição de uma zona muito refinada junto à frente de fenda para uma região mais grosseira; e uma malha regular que se conecta à anterior e preenche o restante volume do sólido. Os avanços locais dos nós da frente de fenda são calculados a partir da lei de Paris (ver Figura 2.1), e os fatores de intensidade de tensão dos nós da frente de fenda são obtidos através do método direto de extrapolação com 2 pontos.

Relativamente ao desenvolvimento da ferramenta numérica referida, usar-se-á a linguagem de programação Visual Basic que integra a ferramenta informática Microsoft Visual Studio 2012. Nesta fase, este modelo numérico é desenvolvido com o objetivo de garantir compatibilidade com o *software* de elementos finitos COSMOSM que faz parte do *software* comercial SolidWorks 2014.

1.3. Estrutura da dissertação

Esta dissertação estará dividida em cinco capítulos. No primeiro, procura-se enquadrar e definir o problema, de modo a identificar os objetivos a que nos propomos alcançar.

No segundo capítulo, efetua-se uma revisão da literatura com o intuito de estar a par da investigação no que toca a este tema em concreto.

O capítulo seguinte é dedicado à descrição do problema e à forma como é desenvolvido o modelo de propagação automática proposto neste trabalho. Nesse sentido, descrevem-se, de forma sucinta, os modelos físicos e numéricos, bem como as principais abordagens e metodologias selecionadas para a implementação da técnica de remalhagem adaptativa com múltiplos graus de liberdade.

No quarto capítulo são apresentadas, de forma progressiva, as principais fases de desenvolvimento do modelo numérico tridimensional da junta soldada em T, onde se destacam as fases de pré-processamento, processamento, e pós-processamento. De forma complementar à descrição da metodologia implementada, apresenta-se, de forma sucinta, uma explicação do código-fonte criado em Visual Basic 2012 para esta geometria.

Por fim, no último capítulo, será feita uma síntese do trabalho realizado, onde serão explanadas as principais conclusões e apresentadas sugestões para trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

2.1. Fadiga

Os danos causados por fadiga são responsáveis por cerca de 80% a 90% das falhas prematuras de componentes em serviço em engenharia (Branco *et al*., 2012a).

A fadiga é um processo localizado, permanente e progressivo, a que os componentes mecânicos em serviço estão submetidos devido às cargas cíclicas a que estão sujeitos. Estes ciclos de carregamento, com ou sem choque, conduzem à diminuição da secção resistente, levando à rotura mesmo para tensões inferiores ao limite de elasticidade do material.

A falha por fadiga é um processo que compreende três fases distintas. A primeira fase designa-se por iniciação de fenda, e envolve a nucleação e o crescimento microscópico da fenda, sendo este um fenómeno que ocorre normalmente à superfície, em zonas de maior concentração de tensões, e onde, em geral, há maior suscetibilidade de ocorrência de deformação plástica. Na segunda fase, a propagação estável de fenda, verifica-se um crescimento macroscópico da fenda, existindo um aumento progressivo da velocidade de propagação e um aumento da dimensão da fenda. Por fim, depois de atingida uma dimensão crítica da fenda, ocorre a propagação instável que conduz à rotura do componente.

O estudo da propagação de fendas por fadiga tem recorrido à Mecânica da Fratura Linear Elástica (MFLE), iniciada por Irwin (1958), sendo a metodologia que melhor quantifica o período de propagação da fenda e que se baseia no fator de intensidade de tensão, K. A utilização desta ferramenta baseia-se na tolerância ao defeito, ou seja, considera que todos os componentes de engenharia possuem defeitos (Ribeiro, 2011).

Este fator é função da tensão remota aplicada, do modo de deformação da fenda, da dimensão de fenda e da geometria do sólido.

A quantificação da magnitude da concentração de tensões provocada pela existência da fenda num corpo elástico é traduzida pela seguinte expressão (2.1):

$$K = Y\sigma\sqrt{\pi a} \tag{2.1}$$

onde Y é um fator geométrico que considera o efeito da geometria do componente, σ é o valor da tensão remota aplicada e a é o comprimento da fenda.

No momento em que se atinge o valor máximo de K, alcança-se o valor crítico (K_{Ic}) para o qual ocorre rotura instável, e que pode também ser designado como tenacidade à fratura.

Com base nas definições apresentadas anteriormente, é possível relacionar a velocidade de propagação da fenda com a gama do fator de intensidade de tensão, ΔK, através da expressão (2.2):

$$\Delta K = K_{máx} - K_{min} \tag{2.2}$$

onde $K_{máx}$ e $K_{mín}$ representam os valores máximo e mínimo do fator de intensidade de tensão, K, durante um ciclo de carga.

Na Figura 2.1 representa-se uma curva típica $da/dN - \Delta K$, na qual se identificam três zonas com comportamentos distintos, designadas por regimes de propagação I, II e III.

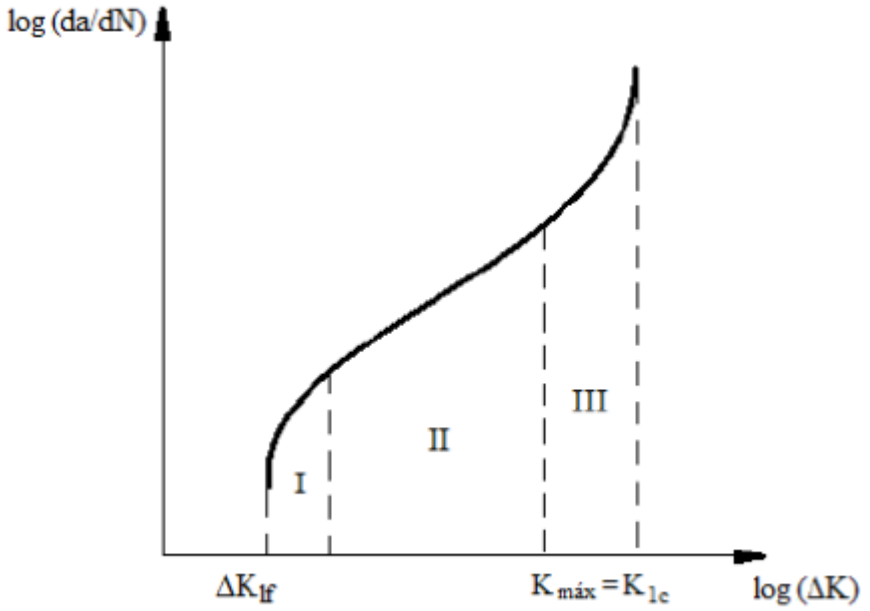

Figura 2.1. Diagrama da curva típica $\boldsymbol{da/dN - \Delta K}$ (adaptada de Paiva, 2015).

O regime I apresenta como valor inferior o limiar de propagação de fendas por fadiga, ΔK_{lf}, em que para valores inferiores não existe propagação. De notar que a propagação inicial é lenta, devido à influência de diversos fatores, entre os quais a microestrutura (fronteiras de grão, inclusões, etc.), a tensão média, ou o meio ambiente.

Num sentido oposto, no regime III, a propagação da fenda ocorre de forma rápida (crescimento instável) até que se dê a rotura. Este fenómeno resulta da aproximação de $K_{máx}$ ao valor crítico, K_{Ic}, que é uma caraterística do material.

Na zona intermédia, regime II, existe um crescimento estável, apresentando uma taxa de propagação de fenda entre valores 10^{-6} e 10^{-3} mm/ciclo para materiais metálicos (Zhao et al., 2008). Nesta zona, verifica-se que, numa escala bi-logarítmica, a velocidade de propagação da fenda apresenta uma relação aproximadamente linear com o fator de intensidade de tensão. Esta relação, proposta por Paris e Erdogan (1963), pode ser escrita na forma (2.3) seguinte:

$$\frac{da}{dN} = C(\Delta K)^m \tag{2.3}$$

onde C e m são constantes determinadas experimentalmente através de técnicas de ajustamento de funções, e que depende da razão de tensão, da tensão média, da frequência, da microestrutura, dos efeitos ambientais, entre outros fatores.

No entanto, esta função apenas faz o ajustamento na zona II, sendo que existem outros modelos que permitem modelar as três regiões. Forman *et al*. (1967) criaram um modelo que é muitas vezes utilizado para ajustar as zonas de crescimento lento e acelerado da fenda. Também Erdogan e Ratwani (1970) propuseram um modelo capaz de incluir as três fases de crescimento da fenda por fadiga.

Com base neste modelo, conhecido por lei de Paris, o número de ciclos de fadiga, N_f, entre dois comprimentos de fenda, é obtido pela seguinte expressão (2.4):

$$N_f = \int_{a_i}^{a_f} \frac{da}{C\,(\Delta K)^m} \tag{2.4}$$

onde a_i e a_f são os limites de integração que representam o tamanho mínimo e máximo da fenda. Em situações práticas, por vezes, existe dificuldade em determinar a solução analítica

da equação 2.4. Nesses casos pode-se recorrer ao algoritmo de integração de Eules, sendo o número de ciclos de carga, dado por:

$$N^{(j+1)} = N^{(j)} + \Delta N^{(j)} <=> N^{(j+1)} = N^{(j)} + \frac{\Delta a^{(j)}}{C[\Delta K(a^{(j)})]^m}, \quad j = 0,1,\dots,n \tag{2.5}$$

onde n representa o número de intervalos, $\Delta a^{(j)}$ é o incremento da fenda do passo j, e $\Delta K^{(j)}$ é a gama do fator de intensidade de tensão do passo j. Realizando uma análise com base em incrementos locais, a lei de Paris é aplicável a qualquer ponto, como se segue:

$$\frac{da_i}{dN} = C(\Delta K_i)^m \tag{2.6}$$

sendo que da_i e ΔK_i representam o incremento local de fenda e a gama do fator de intensidade de tensão do nó i da frente de fenda. Recorrendo à equação 2.6, podemos deduzir as relações seguintes:

$$\Delta a_i^{(j)} = \left(\frac{\Delta K_i^{(j)}}{\Delta K_{máx}^{(j)}}\right)^m \Delta a_{máx}^{(j)} \tag{2.7}$$

$$\Delta N^{(j)} = \frac{\Delta a_{máx}^{(j)}}{C(\Delta K_{máx}^{(j)})^m} \tag{2.8}$$

onde $\Delta a_{máx}^{(j)}$ e $\Delta K_{máx}^{(j)}$ são, respetivamente, o incremento máximo de fenda e o valor máximo da gama do fator de intensidade de tensão do passo j. Estas equações são fortemente dependentes do incremento máximo de fenda (Branco *et al*., 2015). Para que os erros numéricos sejam mínimos, deve usar-se um incremento máximo de fenda relativamente pequeno, mas não exageradamente pequeno, para não penalizar, em demasia, o tempo de computação.

2.2. Técnicas de Remalhagem

As abordagens desenvolvidas a partir do método dos elementos finitos para estudar a evolução da forma de fenda, são, geralmente, assentes no conceito de propagação automática de fenda. Esta abordagem, também conhecida como técnica de remalhagem

adaptativa, provou ser muito eficiente e fiável (Branco *et al.*, 2015). Para além disso, tem a vantagem de poder ser usada na análise de geometrias, condições de fronteira e cenários de carregamento de elevada complexidade, o que não acontece, atualmente, com outras abordagens numéricas.

Como se pode visualizar na Figura 2.2, a técnica de propagação automática pode ser dividida em cinco etapas principais: (i) desenvolvimento de um modelo de elementos finitos tridimensional representativo; (ii) cálculo do campo de deslocamentos; (iii) cálculo dos fatores de intensidade de tensão efetiva ao longo da frente de fenda; (iv) determinação dos avanços da frente de fenda aplicando uma lei de propagação adequada; (v) definição de uma nova frente de fenda através das posições dos nós obtidos no passo anterior.

As duas abordagens que são usadas nos dias de hoje para definir a frente de fenda, são o modelo de dois graus de liberdade e o modelo de múltiplos graus de liberdade. Nos pontos seguintes faz-se uma diferenciação entre as duas abordagens.

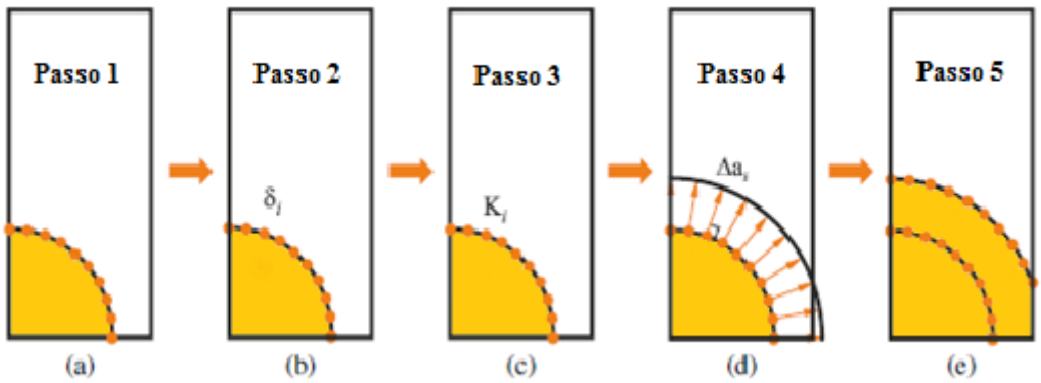

Figura 2.2. Técnica de remalhagem adaptativa (adaptada de Branco *et al.*, 2015).

2.2.1. Modelo de dois graus de liberdade

O modelo de dois graus de liberdade é uma abordagem mais simples em que se assume uma forma de fenda pré-definida (semicircular, semielíptica, etc.), próxima das formas observadas experimentalmente, estudando-se, apenas, a propagação de dois pontos-chave da frente de fenda.

Normalmente, a forma da fenda é estudada a partir dos avanços do ponto superficial e do ponto mais interior da frente de fenda. O avanço destes dois pontos é definido

com recurso a soluções do fator de intensidade de tensão, determinadas previamente e a dados experimentais da taxa de crescimento das fendas por fadiga para material de análise.

Newman e Raju (1981) foram os primeiros a aplicar este modelo de dois graus de liberdade para problemas de fendas por fadiga. As soluções geométricas do fator de intensidade de tensão para as geometrias estudadas foram obtidas a partir de modelos tridimensionais de elementos finitos (Newman e Raju, 1979; 1981; 1983).

2.2.2. Modelo de múltiplos graus de liberdade

Uma abordagem mais versátil, designada por modelo de múltiplos graus de liberdade, consiste na discretização da frente de fenda num conjunto de pontos, sem assumir qualquer forma de fenda teórica. Os pontos considerados, para os quais são calculados os avanços nodais, podem ser ligados através de funções *cubic spline* (tema abordado no subcapítulo 3.2.2) ou, simplesmente, por linhas poligonais. Estas caraterísticas tornam esta abordagem mais precisa e, por isso, usada em situações mais complexas, geralmente associadas a formas de fenda mais precisas e também a fatores de intensidade de tensão mais precisos. Para além disso, tem, ainda, a vantagem de permitir uma total automatização do processo, o que não é possível no caso das linhas poligonais, que requer, por vezes, a intervenção do utilizador para que a simulação prossiga.

Smith e Cooper (1989) introduziram o modelo de múltiplos graus de liberdade na análise dos problemas de propagação de fendas por fadiga. Foram realizadas imensas pesquisas com o intuito de desenvolver a técnica, quer a nível das variáveis numéricas utilizadas (Lin e Smith, 1999a; 1999b; 1999c), quer no que concerne ao estudo do fenómeno de propagação de fendas por fadiga em várias geometrias submetidas a diferentes cenários de carregamento (Lin e Smith, 1995; 1997). No ponto 3.2.2 é abordado em mais detalhe as especificidades desta técnica.

2.3. Técnica de propagação automática da fenda

A técnica de remalhagem adaptativa, como já foi referido no subcapítulo 2.2, encontra-se dividida em cinco etapas principais. Nestas etapas, as variáveis numéricas usadas, podem ser agrupadas em três categorias (Figura 2.3), sendo elas associadas à

aplicação do método dos elementos finitos; associadas ao cálculo do fator de intensidade de tensão; ou associadas ao modelo de propagação da fenda.

Como se pode visualizar na Figura 2.3, a precisão geral da técnica depende da precisão de cada passo. Uma vez que a saída de uma determinada tarefa serve diretamente como entrada da próxima, caso a tarefa à saída possua um erro, esse irá propagar-se ao longo da simulação, sendo por isso desejável que o efeito de cada variável seja conhecido antecipadamente para que se possam obter resultados mais fiáveis.

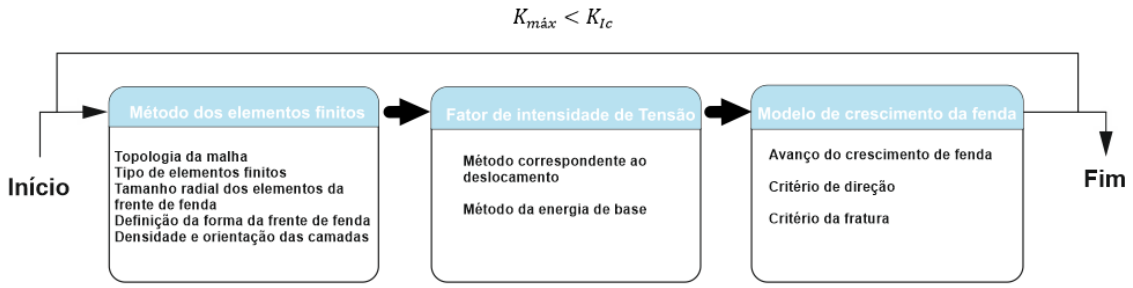

Figura 2.3. Principais variáveis numéricas que afetam a precisão da técnica remalhagem adaptativa (adaptada de Branco *et al.*, 2015).

2.3.1. Modelo dos elementos finitos

Uma das técnicas computacionais mais versáteis para a obtenção de soluções aproximadas das equações diferenciais parciais é o método dos elementos finitos. No âmbito de problemas de fadiga, este permite lidar com uma vasta variedade de geometrias, condições de fronteira, propriedades dos materiais, e cenários de carregamento.

Na análise de peças fissuradas com o MEF, deve-se ter em conta a singularidade das tensões na extremidade da fenda. De notar, que as funções de forma no interior dos elementos não podem alcançar valores infinitos num espaço finito (Branco *et al*., 2015). Por esse motivo, o campo de deslocamentos obtido nos elementos da frente de fenda, não corresponde à distribuição real, ocorrendo, portanto, erros nos deslocamentos nodais. Para colmatar tal facto e melhorar a precisão global do MEF neste tipo de análises, existem algumas soluções práticas que permitem modelar, de forma eficiente, a singularidade da frente de fenda, sem que haja sobrecarga computacional excessiva, tais como o desenvolvimento de uma topologia de malha otimizada e suficientemente refinada que permita capturar a variação do campo de deslocamento r^{λ}, onde λ é igual a 0,5.

O uso de elementos finitos específicos, ou especiais, na frente de fenda ou a definição de um tamanho ótimo para elementos da frente de fenda são outras soluções possíveis para aumentar a precisão do MEF.

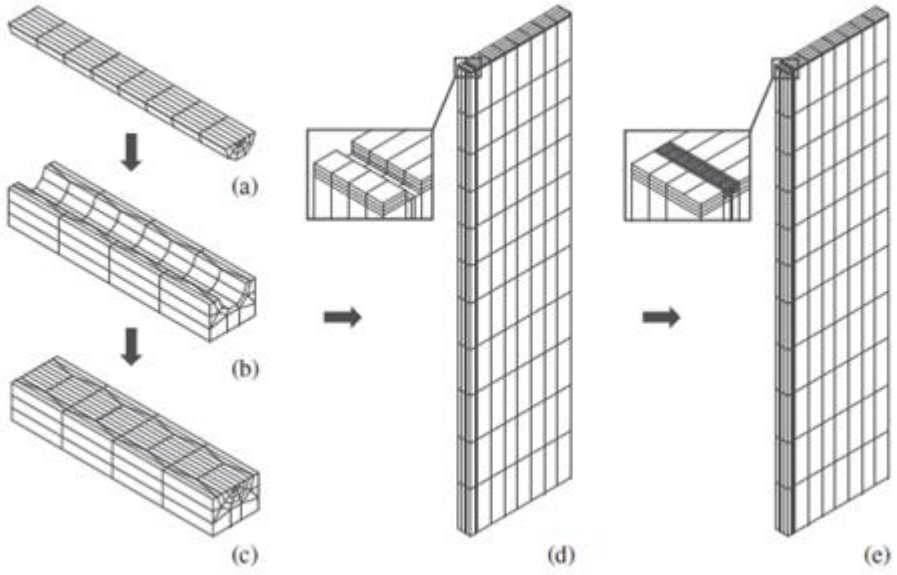

Figura 2.4. Topologia típica de malhas de elementos finitos (Branco *et al.*, 2008): (a) malha em teia de aranha; (b) malha de transição; (c) região da fenda; (d) região sem fenda; (e) Modelo total (adaptada de Branco *et al*., 2015).

2.3.2. Topologia da Malha

Quando é criado um determinado modelo e se aplica uma determinada topologia de malha, esta irá influenciar a precisão dos resultados obtidos. É relativamente consensual entre os investigadores (Carpinteri e Brighenti, 1996; Lin e Smith, 1999a; Branco e Antunes, 2008a) que a melhor topologia de malha deve conter, no mínimo, duas regiões diferentes (Figura 2.4): i) uma malha em forma de teia de aranha, constituída por vários anéis concêntricos (Figura 2.4 (a)) em torno da frente de fenda; (ii) e uma malha mais grosseira, que preenche o restante volume do corpo, e que visa minimizar o esforço computacional (Figura 2.4 (d)). Em determinados casos, pode-se acrescentar uma região adicional, geralmente designada por malha de transição (Figura 2.4 (b)), cujo objetivo principal é diminuir o número total de elementos e promover uma transição mais suave da malha refinada junto à frente de fenda para a malha regular das regiões remotas.

A malha da região da frente da fenda, deve conter entre 2 e 4 anéis concêntricos. Os resultados existentes na literatura sugerem a utilização de pelo menos dois anéis (Guinea *et al*., 2000; Branco *et al*., 2013).

2.3.3. Tipos de elementos finitos

Em análises tridimensionais de componentes com fendas com o método dos elementos finitos são geralmente utilizados elementos isoparamétricos. Por exemplo, a ESIS recomenda a utilização de elementos isoparamétricos com função de forma quadrática (Sedmak *et al*., 1992). Estes elementos permitem representar formas curvas com poucos elementos, uma vez que existe a possibilidade de terem formas distorcidas. Para que tal seja possível, os elementos são definidos em coordenadas locais estabelecendo-se uma relação entre coordenadas locais e globais. Para além disto, os elementos isoparamétricos quadráticos são convergentes e apresentam uma taxa de convergência melhor do que os outros elementos. Há ainda outro aspeto bastante importante que é o facto de estarem bem testados e de se encontrarem disponíveis nos principais *softwares* comerciais de elementos finitos.

A região da malha sem fenda pode ser construída a partir de uma abordagem estruturada (Branco *et al*., 2014; Dhondt, 2014) ou não-estruturada (Fulland *et al*., 2008; Rabold *et al*., 2014). No primeiro caso são, geralmente, usados elementos hexaédricos de 20 nós (Lin e Smith,1999a; Carpinteri e Brighenti, 1996; Branco *et al*., 2014). Neste tipo de elemento, cada nó possui três graus de liberdade, simulando-se variações parabólicas de deslocamento que correspondem as variações lineares de tensão. O seu uso é especialmente adequado para análises elásticas, uma vez que a matriz de rigidez e as forças nodais são calculadas através de integração numérica total. Por vezes, verifica-se que as tensões determinadas nos pontos de integração individual apresentam um comportamento aleatório, e, por isso, utilizam-se, frequentemente, valores médios de tensões de modo a suavizar as variações (Bakker, 1992).

Na proximidade da fenda, a malha deve simular uma tensão elástica linear de $r^{-\lambda}$, onde λ é igual a 0,5, podendo esta ser introduzida na análise utilizando elementos analíticos ou elementos isoparamétricos modificados. Os elementos analíticos são baseados em expressões analíticas da mecânica da fratura linear elástica, cujas formulações contêm o

fator de intensidade de tensão, evitando-se assim o seu cálculo subsequente. Por sua vez, os elementos isoparamétricos modificados são, geralmente, obtidos a partir do colapso dos elementos hexaédricos de 20 nós (Figura 2.5 (c)), ou, em certos casos, utilizando elementos em forma de cunha com 15 nós (Figura 2.5 (d)). Em ambos os casos, deslocam-se os nós intermédios adjacentes à frente de fenda para uma nova posição que corresponde a ¼ da aresta (Figura 2.5 (c) e Figura 2.5 (d)).

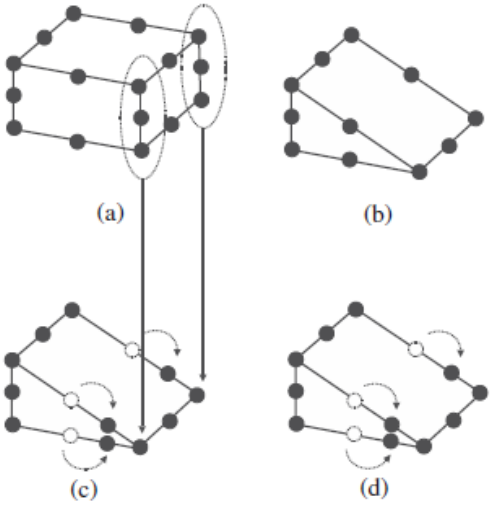

Figura 2.5. Elementos finitos típicos: (a) elemento hexaédrico de 20 nós; (b) elemento em cunha de 15 nós; (c) elemento hexaédrico colapsado com os nós intermédios deslocados para ¼ da aresta; (d) elemento em cunha com os nós intermédios deslocados para ¼ da aresta (adaptada de Branco *et al.*, 2015).

Esta última abordagem tem sido utilizada por diversos investigadores (Lin e Smith, 1999a; Antunes *et al*., 2002; Branco *et al*., 2012b). Trata-se de uma metodologia bastante interessante, pois, por um lado, a ligação destes elementos aos elementos *standard* não é um problema uma vez que não são necessários elementos de transição; e, por outro lado, os elementos *standard* utilizados encontram-se disponíveis em todos os *softwares* comerciais de elementos finitos. É, apenas, necessário atualizar as coordenadas dos nós intermédios.

Para além das abordagens referidas, podem, ainda, utilizar-se elementos singulares de ordem superior, ou elementos singulares para os quais se alteram quer a posição dos nós, quer as funções de forma (Gavete *et al*., 1989).

O efeito da utilização de diferentes elementos na frente de fenda foi investigado por Antunes (1999a). Este estudo abrangeu elementos hexaédricos colapsados e com os nós

intermédios a ¼ da aresta (Figura 2.5 (c)), elementos em cunha com os nós intermédios a ¼ da aresta (Figura 2.5 (d)), e elementos em cunha convencionais (Figura 2.5 (b)). Este estudo permitiu concluir que os melhores resultados eram obtidos com elementos hexaédricos colapsados. Os elementos em cunha convencionais também apresentaram alto desempenho. Não é surpreendente o bom comportamento destes elementos singulares, uma vez que incorporam a singularidade de $r^{-0,5}$ na frente de fenda.

No entanto, podem ocorrer erros nos pontos de canto da frente de fenda, uma vez que a mesma singularidade é implicitamente simulada pelos elementos singulares. Essa singularidade apresenta a forma $r^{-\lambda}$, sendo, geralmente, $\lambda < 0{,}5$. De referir que a singularidade de λ é igual a 0,5 em pontos de canto apenas quando o ângulo de interseção entre a frente de fenda e a superfície livre atinge um valor crítico. Bazant e Estenssoro (1979) mostraram que esse ângulo crítico depende do coeficiente de Poisson do material, não sendo afetado por outra constante de elasticidade. Quando se está perante ângulos de interseção de 90º, e o coeficiente de Poisson assume o valor de 0,3, $\lambda < 0{,}5$ para carregamentos em Modo I (isto é, $K_I = 0$) e $\lambda > 0{,}5$ para carregamentos em Modos II e III (onde K_{II} e K_{III} são infinitos). Na Figura 2.6 apresenta-se os três modos de carregamento referidos anteriormente. Para os pontos de canto, os valores de K obtidos são finitos e possuem a mesma ordem de grandeza daqueles que são obtidos noutras posições ao longo da frente de fenda (Pook, 1994). Tal facto comprova a imprecisão do MEF perto da superfície. Bakker (1992) demonstrou que apenas o valor de λ calculado muito próximo da superfície livre é afetado pelo refinamento da malha no sentido longitudinal da frente de fenda. Nesse sentido, é aconselhável em malhas de elementos finitos efetuar-se um refinamento na direção longitudinal junto aos pontos de canto, mas também na direção da espessura. Na prática, a variabilidade da camada superficial, e da singularidade, torna mais difícil a definição do nível de refinamento da malha.

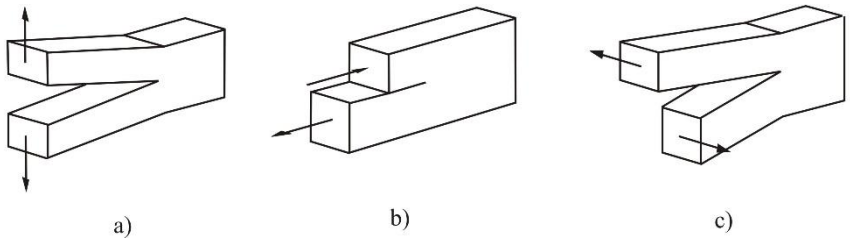

Figura 2.6. Modos de solicitação elementares: a) Modo-I; b) Modo-II; c) Modo-III (adaptada de Branco, 2006).

2.3.4. Tamanho radial dos elementos da frente de fenda

Na maioria dos problemas, o padrão da malha da região da fenda considerado mais eficiente, como já foi referido anteriormente, é um padrão em teia de aranha definido a partir de vários anéis concêntricos em torno da frente de fenda (Figura 2.4 (a)). Geralmente, os elementos do anel mais interior são elementos em cunha, e os outros anéis são modelados através de elementos hexaédricos. Uma vantagem da utilização desta configuração é a simplicidade de transição da malha refinada da frente de fenda para malhas mais grosseiras em posições mais afastadas (Figura 2.4 (e)). De referir que os elementos singulares desempenham um papel importante na qualidade dos resultados numéricos obtidos. Guinea *et al*. (2000) concluíram que a discretização angular mínima dos elementos em torno da ponta da fenda é de 30º. Murti *et al*. (1986) aconselham, em modelos com simetria no plano da fenda, isto é, modelação de apenas uma das faces e a utilização de um mínimo de 5 elementos, mas preferencialmente 6 elementos.

O tamanho radial ótimo dos elementos adjacentes à frente de fenda, L_1, depende da configuração da fenda. Uma vez que não existem tamanhos ideais universais, L_1 é praticamente definido com base em estudos paramétricos específicos procurando-se fixar um limite superior aceitável. Por exemplo, Murti *et al*. (1986) e Nykänen (1996) encontraram tamanhos ótimos dentro das faixas de 15-25% e 1,25-2,75% do comprimento da fenda. Branco *et al*. (2008b) determinaram valores ótimos de L_1 para várias geometrias, tais como provetes normalizados M(T) e C(T) e provetes não normalizados com fenda de canto. De uma forma geral, os valores de L_1 estavam na faixa de 2-6 % do comprimento da frente de fenda.

2.3.5. Malha ortogonal ao longo da frente de fenda

Uma questão de grande importância na precisão global da técnica de remalhagem adaptativa é a ortogonalidade da malha ao longo da frente de fenda. Lin *et al*. (1999a) estudaram a influência do grau de não ortogonalidade da malha em torno da frente de fenda sobre o fator de intensidade de tensão. Esta pesquisa envolveu três malhas com diferentes graus de não ortogonalidade (Malhas B, C e D da Figura 2.7) e uma malha perfeitamente ortogonal (Malha A da Figura 2.7). As frentes de fenda foram implementadas usando funções *cubic spline*, sendo o fator de intensidade de tensão calculado através de um

método direto baseado nas posições dos nós adjacentes a ¼ da aresta (Henshell *et al.*, 1975; Barsoum, 1976) e através de um método energético baseado no integral-J.

Verifica-se que os resultados obtidos através do método direto, representados na Figura 2.7 (a) em termos dos respetivos fatores geométricos, não são idênticos ao longo da frente de fenda para diferentes malhas, havendo um aumento das diferenças com o grau de não ortogonalidade. Porém, para os valores de SIF calculados a partir do método energético, não se verifica o problema retratado anteriormente. Conclui-se, assim, que a ortogonalidade da malha é mais importante quando se usam métodos diretos; e, por outro lado, há uma baixa sensibilidade do integral-J em relação ao grau de não ortogonalidade da malha, mesmo quando esta se apresenta severamente distorcida.

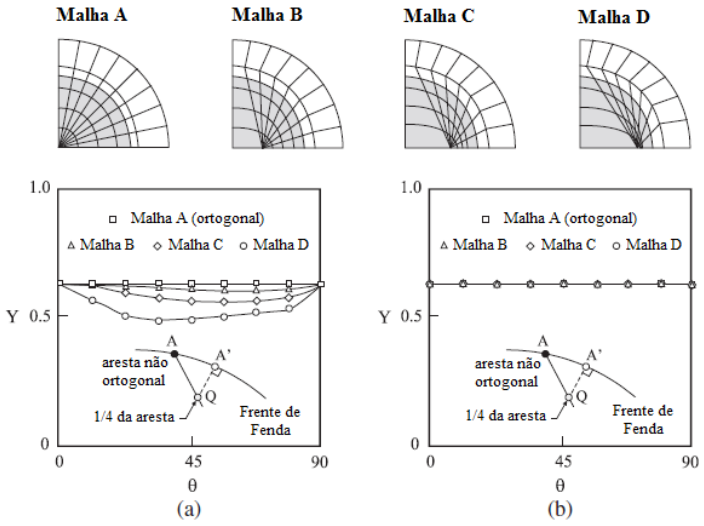

Figura 2.7. Efeito da malha não-ortogonal no fator de intensidade de tensão estimado a partir do método de deslocamento do ponto para ¼ da aresta assumindo: (a) distância Q/A; (b) distância Q/A' (Lin e Smith, 1999a) (adaptada de Branco *et al.*, 2015).

2.3.6. Modelo dos elementos finitos

Em procedimentos numéricos baseados no modelo de múltiplos graus de liberdade (isto é, sem restrições da forma da fenda), a frente de fenda é estabelecida através da ligação dos nós da frente de fenda da malha de elementos finitos. A abordagem mais simples de ligação, representada na Figura 2.8 (a), considera a ligação dos nós de canto através de linhas retas e coloca os nós intermédios da aresta exatamente nos pontos médios dos nós de canto vizinhos (Smith e Cooper, 1989; Nykänen, 1996). Nesta abordagem os

avanços nodais são calculados nos nós de canto, sendo que em determinadas circunstâncias, nomeadamente para grandes alterações da forma da fenda, as distâncias entre os nós de canto tornam-se excessivas, requerendo correção manual das coordenadas destes nós, acabando por reduzir o grau de automação da técnica. Assim, as frentes de fenda acabam por não ser totalmente realistas, uma vez que a precisão do fator de intensidade de tensão ao longo da frente de fenda e a vida de fadiga também são afetadas.

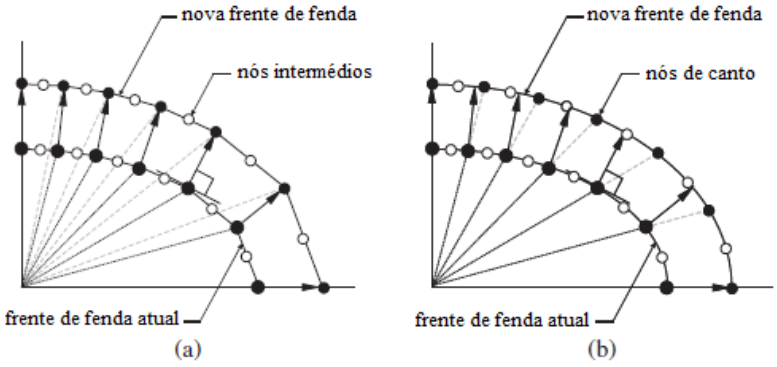

Figura 2.8. Definição da forma da frente de fenda: (a) linha poligonal; (b) curva *cubic spline* (adaptada de Branco *et al*., 2015).

No entanto, é possível adotar uma abordagem mais inovadora (Figura 2.8 (b)) na definição da frente de fenda que consiste no uso de uma função *cubic spline*. Esta função permite reposicionar, quer os nós de canto, quer os nós intermédios. O cálculo dos avanços da frente de fenda é novamente determinado apenas para os nós de canto. Contudo, as posições obtidas não são as posições finais dos nós da nova frente de fenda. Nesta abordagem, essas posições provisórias são utilizadas para ajustar uma curva *cubic spline* que é então aplicada para definir as novas coordenadas dos nós de canto e dos nós intermédios de acordo com um critério pré-definido (por exemplo, manter camadas uniformemente espaçadas ao longo da propagação da fenda). Além disso, esta abordagem tem a vantagem de ser totalmente automatizada; tem, ainda, a vantagem de permitir obter valores mais precisos do fator de intensidade de tensão, o que, consequentemente, leva a resultados mais precisos da vida à fadiga e da forma da fenda.

A influência dos dois métodos anteriormente mencionados sobre o fator de intensidade de tensão foi abordada por Lin *et al*. (1999a) que examinaram uma fenda interior circular num corpo submetido a tensão alternada. Os fatores de intensidade de tensão

calculados através do método do integral-J, em frentes de fenda definidas por *cubic spline*, foram semelhantes, tanto nos nós de canto, como nos nós intermédios (uma diferença inferior de 0,3 % nos nós de canto). Por outro lado, com a definição da frente de fenda com linhas poligonais, os valores de SIF obtidos nos nós intermédios foram superiores cerca de 11 %, relativamente aos nós de canto. Esta diferença é explicada, essencialmente, pelo posicionamento inadequado dos nós intermédios, o que demonstra a sensibilidade dos valores de SIF relativamente à forma de fenda. Assim, mesmo com pequenas diferenças na forma da fenda, podem ocorrer grandes discrepâncias no fator de intensidade de tensão e, por arrasto, na vida de fadiga.

2.3.7. Fator de intensidade de tensão

Existem dois grupos principais de métodos numéricos que permitem extrair o fator de intensidade de tensão, conhecidos por *métodos diretos* e *métodos energéticos*. No primeiro grupo, compara-se o campo de deslocamento obtido através do método dos elementos finitos com o campo de deslocamento analítico que possui a solução do fator de intensidade de tensão na sua formulação (Kanninen *et al*., 1985; Chan *et al*., 1970). Por sua vez, no outro grupo, a força de campo de tensões singulares é definida a partir da taxa de libertação de energia, ou seja, a sensibilidade da energia potencial total para a posição da fenda. Têm sido utilizados vários métodos energéticos, nomeadamente o método da energia total (Antunes *et al*., 1999b), o método baseado na variação de rigidez (Hellen, 1975), o método do integral-J (Murakami *et al*., 1983), o método EDI (Shih *et al*., 1986), e os métodos baseados no incremento virtual de fenda (Branks-Sills *et al*., 2007), entre outros.

2.3.8. Incremento máximo da fenda

Normalmente, a definição do incremento máximo de fenda, $\Delta a_{máx}$, envolve um compromisso entre a precisão dos resultados e o esforço computacional. A utilização de incrementos demasiado pequenos proporciona elevados esforços computacionais, enquanto grandes incrementos levam ao aumento do número de ciclos de carga obtidos para um

determinado comprimento de fenda e, tendem, a causar alguma oscilação dos nós da frente de fenda.

Pode ser visualizado na Figura 2.9 (a) o número de ciclos de fadiga para três diferentes incrementos máximos de fenda para uma placa com uma fenda superficial submetida à tração (Lin *et al*., 1999c). Analisando o gráfico, pode-se verificar que ocorre convergência com a diminuição de $\Delta a_{máx}$. Veja-se que o erro acumulado que existe entre os casos de $\Delta a_{máx} = \mathrm{t}/100$ e $t/167$ é inferior a 2 % quando a fenda atinge a superfície posterior da placa. No entanto, caso o incremento tome o valor de $t/50$, a diferença é, claramente, superior, comprovando que quanto menor o incremento máximo de fenda mais precisos serão os resultados. A influência do incremento máximo de fenda na forma de fenda é demonstrada na Figura 2.9 (b), onde se apresenta a evolução da relação de aspeto da fenda (d/t) com o comprimento de fenda (a/W) para os três incrementos máximos de fenda. Observando a Figura 2.9 (b), as diferenças máximas entre as curvas $\Delta a_{máx} = a/200$ e $a/20$ são inferiores a 3 %, não se verificando o mesmo entre $\Delta a_{máx} = a/200$ e $a/8$ em que a diferença é superior a 10 %. Além disso, esta última apresenta um comportamento oscilatório fisicamente incompreensível para esta situação de propagação. De notar que com a diminuição do incremento máximo de fenda, verifica-se um claro aumento da convergência.

Na literatura, existem duas estratégias diferentes para definir o incremento máximo de fenda adotadas. A mais comum, considera o incremento máximo constante durante toda a simulação, sendo o seu valor definido através de um fator de proporcionalidade entre $\Delta a_{máx}$ e um tamanho característico da geometria, como por exemplo, a espessura (t), a largura (W), o diâmetro (d), ou, em certos casos, assumindo um valor pré-definido, sem qualquer ligação aparente com o tamanho da geometria. Mahmoud (1988; 1990) analisou o crescimento de fendas por fadiga em placas submetidas à tração e à flexão, contendo defeitos iniciais superficiais. Foram utilizados incrementos máximos de fenda iguais a $t/100$, $t/400$ e $t/1000$.

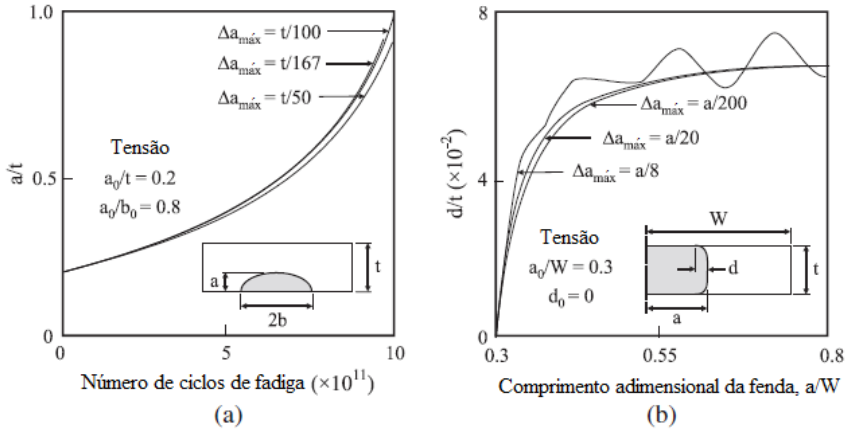

Figura 2.9. Efeito do incremento máximo de crescimento de fenda na: (a) vida de fadiga (Lin e Smith, 1999c); (b) forma de fenda (Branco, 2006**)** (adaptada de Branco *et al*., 2015).

Lin e Smith nos seus estudos sobre o desenvolvimento da forma de fenda recorreram aos valores iguais a $t/167$, $t/100$ e $t/50$ para fendas superficiais em placas de espessura finita (Lin e Smith, 1999c); ou iguais a $t/100$, $t/177$ e $t/200$ para fendas de canto em placas com furos passantes (Lin e Smith, 2001); ou iguais a $d/250$ para fendas superficiais em peças de secção circular, com e sem entalhes, ambas sujeitas à tração e à flexão (Lin e Smith, 1999d). Branco *et al*. (2013) estudaram a extensão da camada superficial em provetes M (T) utilizando incrementos máximos da fenda iguais a $W/300$. O mesmo valor foi utilizado para desenvolver uma geometria entalhada para estudos de fadiga em estado plano de deformação (Branco *et al*., 2014). Uma constante igual a $D/2000$ foi utilizada para obter as constantes da lei de Paris a partir de superfícies de fratura obtidas por fadiga em peças de secção circular utilizando uma técnica numérico-experimental (Branco *et al*., 2012b).

No que diz respeito à estratégia alternativa de definição do incremento máximo de fenda, este assume um valor variável durante a simulação. O seu valor é, geralmente, definido assumindo um número fixo de ciclos de carga; ou estabelecendo uma relação entre o incremento máximo de fenda e o comprimento atual da fenda. Branco *et al*. (2008 a; 2008 b) modelaram a evolução da forma de fenda em provetes M (T) e C (T) utilizando incrementos máximos de crescimento da fenda equivalentes a 1% e 0,1% do comprimento da fenda, respetivamente. Carpinteri *et al.* (2009; 2010) consideraram um número constante de ciclos de carga entre incrementos, mais especificamente 250, na análise de fenómenos de propagação de fenda por fadiga em peças com secção circular sujeitas a carregamento axial

cíclico excêntrico, bem como para estudar o efeito do processo de estampagem a frio no crescimento de fendas por fadiga em peças de secção circular entalhadas. Antunes *et al*. (2002) realizaram uma integração ciclo-a-ciclo para examinar a previsão de vida por fadiga em compósitos de partículas poliméricas.

3. DESCRIÇÃO DO PROBLEMA

Neste capítulo será efetuada uma descrição mais detalhada do problema, mais concretamente em termos geométricos e em termos do modelo numérico de propagação automática de fenda que se pretende desenvolver para estudar a evolução da forma da fenda e a vida de fadiga numa junta soldada em T. O procedimento de propagação automática depende de parâmetros físicos e parâmetros numéricos, que serão abordados neste capítulo. A geometria, a forma inicial da fenda, o material, e o tipo de carregamento serão, também, aspetos abordados neste capítulo.

3.1. Modelo Físico

As geometrias consideradas neste trabalho encontram-se esquematizadas na Figura 3.1. Pretende-se estudar um caso com uma fenda central superficial (Figura 3.1 (a)) e outro caso com uma fenda de canto (Figura 3.1 (b)). Em termos numéricos, ambas são estudadas com o mesmo modelo de elementos finitos, considerando diferentes condições de fronteira.

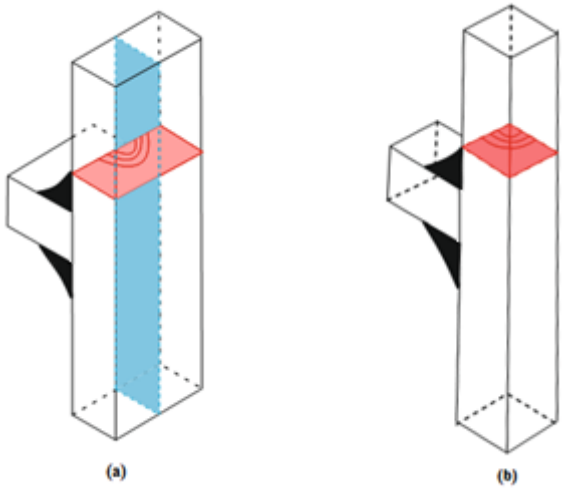

Figura 3.1. Definição da geometria com: (a) fenda central superficial; (b) fenda de canto.

Mais concretamente, devido à simetria da frente de fenda assumida no caso da Figura 3.1 (a) relativamente ao plano médio da peça, o modelo numérico a desenvolver poderá ser apenas metade do modelo físico.

Portanto, em ambos os casos, foi considerada a geometria representada na Figura 3.2, que é fisicamente definida pelas variáveis representadas. Relativamente às variáveis geométricas, embora as normas considerem habitualmente a distância entre apoios e o parâmetro de descentramento, optou-se por definir estas duas variáveis de forma indireta através de Y6 e Y7.

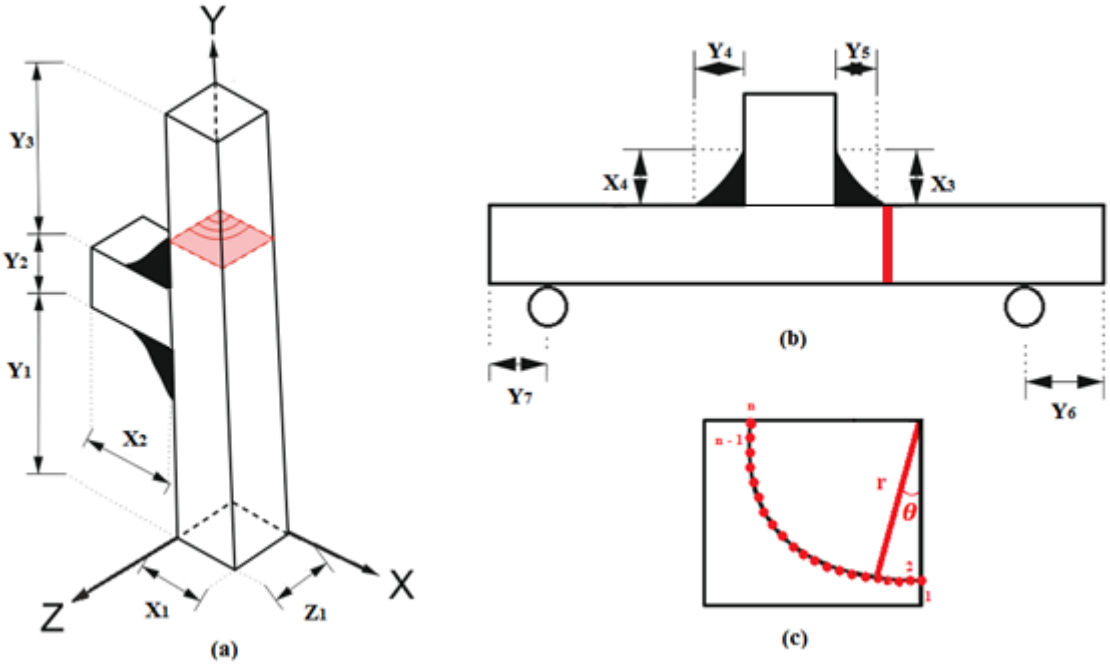

Figura 3.2. Definição das variáveis da geometria: (a) perspetiva tridimensional; (b) vista de lado; (c) secção da frente de fenda.

Analisando a Figura 3.2 (c), verifica-se que a frente de fenda é definida por n nós, introduzidos em coordenadas polares, sendo o valor máximo de n igual a 21. As coordenadas iniciais destes são definidas pelo utilizador que poderá definir os valores dos ângulos (θ) e os respetivos comprimentos (r). Como foi referido anteriormente, o aumento de nós significa um aumento na precisão dos resultados obtidos, mas também um aumento no esforço computacional. Esta abordagem é bastante flexível, permitindo, por um lado, a definição de formas iniciais de fenda arbitrárias com comprimentos variáveis.

A definição dos cordões é efetuada de forma independente, utilizando sistemas de coordenadas cartesianas locais. Os dois perfis são definidos de forma discreta, com número de pontos variável, até um máximo de 100, como se esquematiza na Figura 3.3.

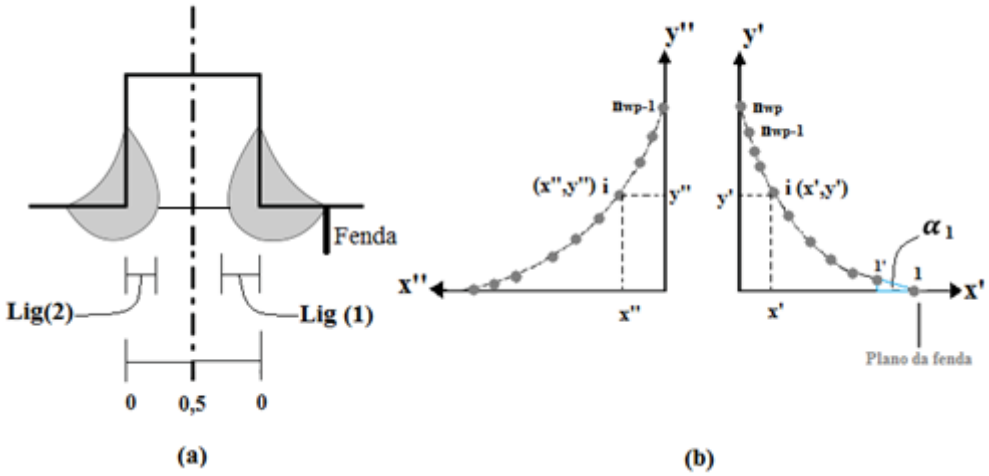

Figura 3.3. Definição dos cordões de soldadura: (a) nível de penetração da soldadura; (b) nós dos cordões.

O perfil do cordão da zona fissurada é definido utilizando o sistema de coordenadas OY'X' e o perfil do cordão da zona não-fissurada é definido, de forma similar, considerando o sistema de coordenadas OY''X''.

Como pode ser visualizado na Figura 3.3 (b), o cordão da zona fissurada apresenta mais um ponto. Isto deve-se ao facto de na parte fissurada ser necessária uma variável adicional para a geração da malha de elementos finitos na zona do defeito, aqui representada por α_1. Essa variável, neste caso, representa o declive junto à raiz do cordão. No que diz respeito ao nível de penetração da soldadura, este é definido pelas variáveis Lig(1) e Lig(2), podendo estes valores ser diferentes para os dois cordões. Caso estas duas variáveis tomem valores iguais a 0,5 estamos perante um caso de penetração total; se ambos forem iguais a 0 estamos perante um caso com ausência de penetração.

A definição do modelo, com base neste número alargado de variáveis geométricas, permite estudar um conjunto considerável de assimetrias, tais como, assimetrias em relação: (i) às posições dos apoios, para carregamentos de flexão em 3 pontos, considerando diferentes valores de Y6 e Y7; (ii) às dimensões dos cordões de soldadura, em termos de altura (X3 diferente de X4), largura (Y4 diferente de Y5); (iii) ao nível de penetração da soldadura (Lig(1) diferente de Lig(2)); (iv) à geometria do provete admitindo

um perfil em T descentrado (Y1 diferente Y3); (v) às formas dos cordões de soldadura, considerando perfis diferentes para os cordões fissurado e não-fissurado.

3.1.1. Carregamento

Para além de toda a versatilidade e flexibilidade que este modelo possui com a possibilidade de se poder alterar diversas varáveis geométricas, foi definido, como objetivo, a criação de uma gama de diferentes tipos de carregamentos, bem como a possibilidade de alterar a razão de tensão, através da variação da força máxima, $F_{máx}$, e força mínima, $F_{mín}$. Nesta fase, pretende-se a simulação de carregamentos cíclicos de amplitude constante, como pode ser visualizado na Figura 3.4.

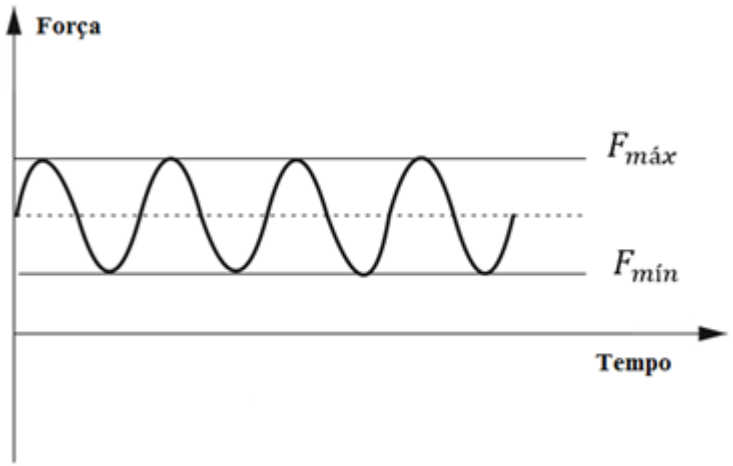

Figura 3.4. Carregamento cíclico de amplitude constante.

A razão de tensão, R, é dada pela razão entre a tensão mínima, $\sigma_{mín}$, e a tensão máxima, $\sigma_{máx}$, e pode ser representada pela equação (3.1). Esta variável, como foi referido no Capítulo 2, apresenta uma grande influência na velocidade de propagação da fenda.

$$R = \frac{\sigma_{mín}}{\sigma_{máx}} \tag{3.1}$$

Os modos de carregamento, todos representativos de Modo-I, simulam esforços de tração, esforços de flexão, e combinações de esforços de tração e flexão. As situações consideradas estão genericamente esquematizadas na Figura 3.5.

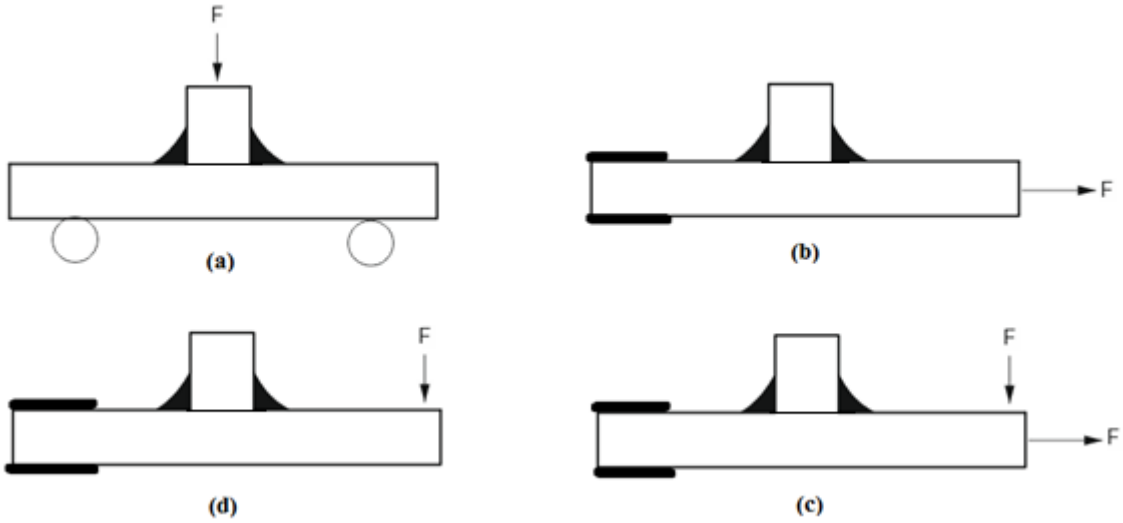

Figura 3.5. Diferentes tipos de carregamento: (a) Flexão típica em 3 pontos; (b) Tração; (c) Flexão; (d) Flexão e tração.

3.1.2. Propriedades do Material

O material considerado no presente estudo foi uma liga de alumínio 6082-T6 (Borrego, 2001). Admitiu-se que este era contínuo, homogéneo, isotrópico e linear elástico. As propriedades mecânicas e as propriedades de propagação deste material são apresentadas nas Tabela 3.1 e Tabela 3.2, respetivamente.

Tabela 3.1. Propriedades do alumínio 6082-T6 (Borrego, 2001).

Tensão de cedência	$\mathbf{307 \pm 2.7\ MPa}$
Tensão de rotura	$330 \pm 2.5\ MPa$
Módulo de elasticidade	$70 \times 10^3\ MPa$
Coeficiente de Poisson	0.33
Dureza Vickers	$100\ kgf/mm^2$
Tenacidade à Fratura	$40.5\ MPa.m^{1/2}$

Tabela 3.2. Constantes da lei de Paris ($\boldsymbol{da/dN - \Delta K[m/ciclo - MPa.m^{1/2}}$]).

Razão de Tensão	C	m	Limites de validade $[\boldsymbol{MPa.m^{1/2}}]$
0.25	8.906×10^{-11}	3.456	$2.7 \leq \Delta K \leq 14$
-0.25	1.900×10^{-11}	3.978	$3.3 \leq \Delta K \leq 15$

3.2. Modelo Numérico

O modelo numérico foi desenvolvido através do ambiente de programação Visual Basic 2012, de modo a que o campo de deslocamentos do modelo numérico seja obtido a partir da ferramenta computacional COSMOSM que integra o *software* comercial SOLIDWORKS 2014.

A exatidão dos resultados numéricos obtidos com este modelo está dependente, entre outros aspetos, da capacidade da malha utilizada para simular, de forma realista, a evolução da fenda. Deste modo, a seleção do tipo e a densidade dos elementos finitos mais adequados são fatores de extrema importância para melhorar a simulação.

A malha a desenvolver deve permitir acomodar diferentes formas de fenda e deve ser constituída por 3 partes distintas: uma malha em teia de arranha junto à fenda, uma malha transição, e uma malha regular. Esta topologia, como foi referido no Capítulo 2, tem inúmeras vantagens, nomeadamente a redução do esforço computacional devido ao uso de uma malha regular pouco refinada, o que é conseguido pelo uso da malha de transição.

A malha em teia de aranha, cujo padrão 2D está esquematizado na Figura 3.6 (a), é composta por três anéis concêntricos, definidos pelas variáveis L_1, L_2, L_3 e por uma discretização angular fixa dos elementos igual a 36º. Este valor, também como já foi referido no Capítulo 2, cumpre as recomendações ótimas para modelos de propagação de fenda baseados no método dos elementos finitos (Guinea *et al*., 2000). Este padrão 2D é aplicado ao longo da frente de fenda que, conforme se referiu anteriormente, pode ser definida até um máximo de 21 nós de canto.

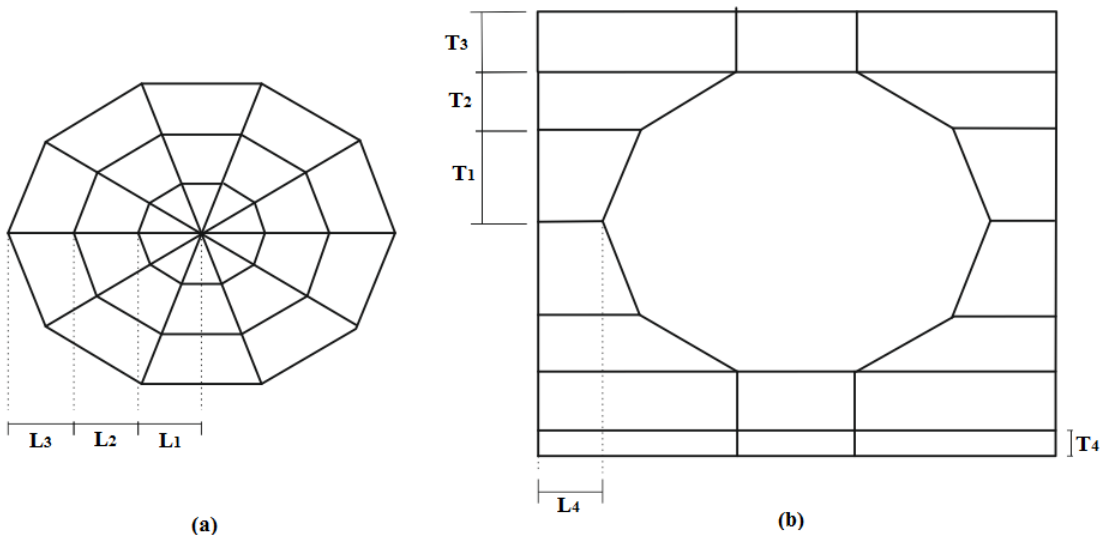

Figura 3.6. Topologia da malha da frente de fenda: (a) Malha em teia de aranha; (b) Malha de transição.

Por sua vez, a malha de transição da Figura 3.6 (b), apresenta um papel importante, uma vez que permite uma transição suave entre a malha mais refinada (malha em teia de aranha) e uma mais grosseira (malha regular). Esta malha é definida pelas variáveis geométricas L_4, T_1, T_2, T_3 e T_4. As variáveis T_1 e T_2 são definidas implicitamente, com base nas definições da malha em teia de aranha, através da equação (3.2):

$$T_1 = T_2 = (L_1 + L_2 + L_3) \sin\theta \tag{3.2}$$

sendo θ igual a 36º e 72º, respetivamente.

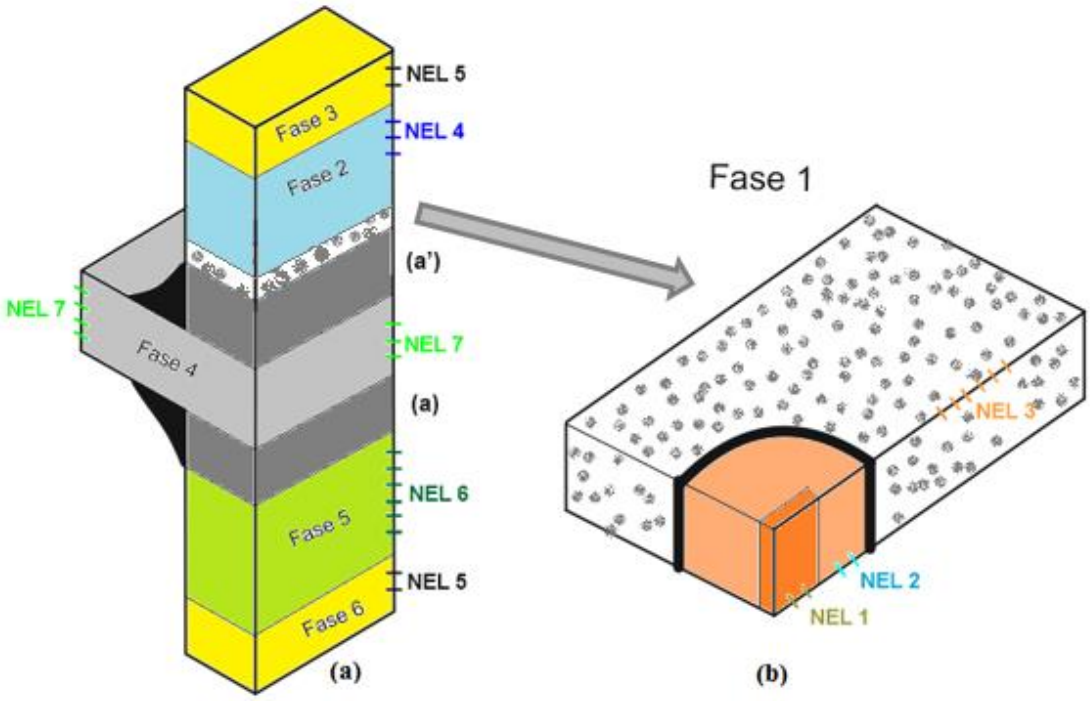

Figura 3.7. Definição da malha regular: (a) malha do corpo do provete; (b) malha do plano da frente de fenda.

A malha regular, como se representa na Figura 3.7, é definida através da união de diferentes regiões, para as quais é possível definir o número de camadas de elementos que a constituem. Esta metodologia é bastante flexível pois permite, por um lado, obter uma densidade adequada e, por outro, controlar de forma efetiva o esforço computacional.

Numa primeira fase, é gerada uma caixa retangular no plano da fenda que é constituída por três partes distintas (Fase 1). Na parte anterior à frente de fenda (Figura 3.7 (b)), representada a laranja, a densidade da malha é controlada pela variável NEL2. Na região posterior à frente de fenda (Figura 3.7 (b)), representada a branco e cinzento, a densidade da malha é controlada por NEL3.

Depois de o plano da fenda estar bem definido, efetua-se a geração do provete na direção ascendente do eixo dos yy, até ao início da zona que define o apoio (Fase 2). O número de camadas de elementos a utilizar nesta zona é definido através da variável NEL4. O restante volume, até ao topo superior (Fase 3), é definido com base no mesmo conceito, recorrendo à variável NEL5.

Seguidamente, é gerada a zona dos cordões de soldadura (Fase 4). As densidades da malha nas zonas dos cordões são definidas implicitamente através dos números de pontos

usados nas definições das formas dos cordões. A densidade da malha na região entre os dois cordões é definida através da variável NEL7.

Finalmente, seguindo uma metodologia idêntica à das Fases 2 e 3, são definidas as zonas inferiores do provete. Foram usadas as variáveis NEL5 e NEL6 para criar o restante volume no sentido descendente do eixo dos yy, sendo o primeiro referente à zona até ao apoio (Fase 5) e o outro à parte da Fase 6.

De referir ainda que nas secções (a) e (a'), a sua densidade depende do número de pontos usados para a definição dos respetivos cordões de soldadura.

3.2.1. Método de Cálculo de K

Os fatores de intensidade de tensão ao longo da frente de fenda são calculados usando o método de extrapolação com dois pontos (Antunes, 1999a). Este método consiste, genericamente, em determinar os valores do fator de intensidade de tensão em dois pontos, demonstrado na Figura 3.8 (a), e extrapolar o valor para a extremidade da fenda ($r = 0$). Neste caso, os pontos utilizados estão representados na seguinte Figura 3.8 (b).

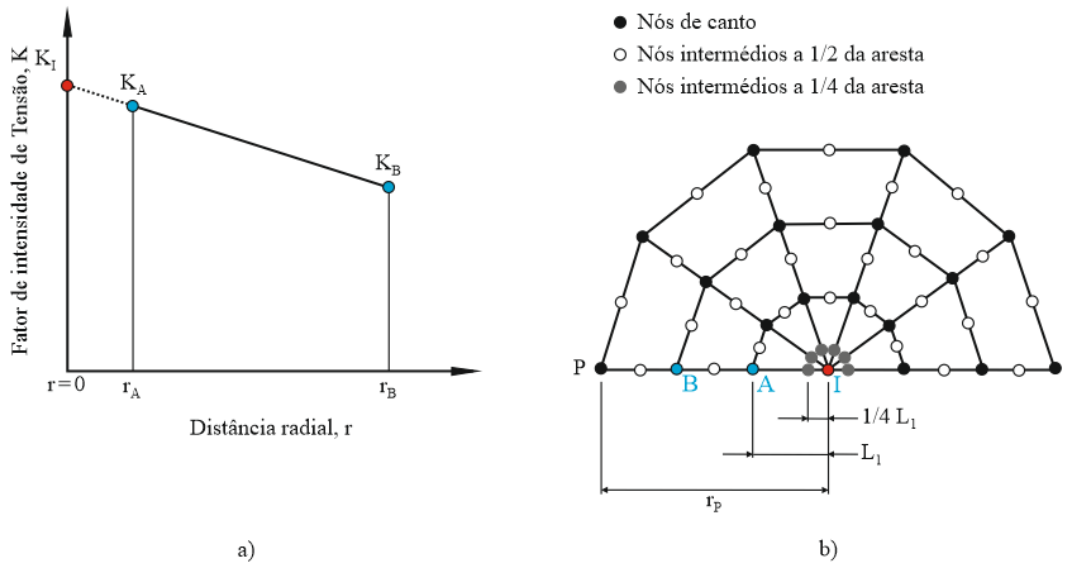

Figura 3.8. (a) Determinação esquemática do cálculo do fator de intensidade de tensão ao longo da frente de fenda usando o método de extrapolação com dois pontos; (b) Identificação dos nós intermédios movidos para ¼ da aresta.

Tal como é aconselhado na literatura (Giunea *et al*., 2000), os valores de K são estimados considerando apenas os dois primeiros termos da expressão (3.3), a partir do deslocamento normal ao plano da fenda.

$$v_B - v_P = K_I \frac{1+\nu}{4E}\sqrt{\frac{2r}{\pi}}\left\{(2\kappa+1)sin\frac{\theta}{2} - sin\frac{3\theta}{2}\right\} + \frac{A_1(1+\nu)r}{E}(\kappa - 3)sin\theta + \frac{A_2(1+\nu)r^{3/2}}{E}\left\{\frac{(2\kappa-1)}{3}sin\frac{3\theta}{2} - sin\frac{\theta}{2}\right\} + \cdots \quad (3.3)$$

onde E é o módulo de Young, ν é o coeficiente de Poisson, κ é um parâmetro elástico igual a $(3-4\nu)$ em estado plano de deformação e $(3+\nu)/(1+\nu)$ em estado plano de tensão, A_i são parâmetros dependentes da geometria e das condições de carregamento, θ e r são as coordenadas polares e v_B e v_P os deslocamentos normais do plano da fenda para estes pontos.

Para um nó P arbitrário, localizado na face superior da fenda, o valor de K é dado pela expressão (3.4):

$$K = \sqrt{\frac{\pi}{8r_P} \times E' \times v_P} \quad (3.4)$$

onde r_P é a distância radial entre o nó P e a ponta da fenda e E' é o módulo de Young modificado, definido por $E' = E/(1-\nu^2)$ para estado plano de deformação, e o $E' = E$ para estado plano de tensão.

No entanto, para simular a singularidade $r^{-0.5}$, os nós intermédios em torno da ponta da fenda são deslocados para ¼ da aresta (Figura 3.8 (b)).

3.2.2. Definição da frente de fenda

No presente estudo, a frente de fenda é aproximada por uma curva c*ubic spline*. Esta é uma função definida por uma curva que passa por um conjunto de pontos (k), normalmente designados por nós. Portanto, para cada intervalo, a *cubic spline* é estabelecida pela aplicação da seguinte expressão (3.5):

$$f_i(x_i) = \frac{f''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 + \left[\frac{f(x_{i-1})}{x_i - x_{i-1}} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6}\right](x_i - x) + \left[\frac{f(x_i)}{x_i - x_{i-1}} - \frac{f''(x_i)(x_i - x_{i-1})}{6}\right](x - x_{i-1}) \quad (3.5)$$

onde x_i e $f(x_i)$ são pares de nós e $f''(x_i)$ é a derivada de segunda ordem. Esta expressão contém apenas duas incógnitas (as derivadas de segunda ordem no final de cada intervalo), que podem ser calculadas pela equação (3.6). Assim, a aplicação da equação anterior, para todos os nós interiores, implica a necessidade de $k-1$ equações em simultâneo e $k+1$ derivadas de segunda ordem desconhecidas. No entanto, devido ao facto de se tratar de uma *cubic spline* natural, as derivadas de segunda ordem nos nós finais são nulas, o que permite reduzir o problema para $k-1$ equações e $k-1$ incógnitas. Além disso, o sistema de equações resultante é tri-diagonal, sendo este particularmente fácil e rápido de resolução.

$$(x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + (x_{i+1} - x_i)f''(x_{i+1}) = \frac{6}{(x_{i+1} - x_i)}[f(x_{i+1}) - f(x_i)] + \frac{6}{(x_i - x_{i-1})}[f(x_{i-1}) - f(x_i)] \quad (3.6)$$

Os incrementos locais da fenda dos nós de canto para carregamentos em Modo-I, seguem a metodologia esquematizada na Figura 2.8, i.e. são calculados assumindo uma propagação na direção normal à frente de fenda.

Para o nó de canto arbitrário i, o incremento normal da fenda para a iteração j, é determinado através da equação (2.7).

Com o objetivo de se obter uma frente de fenda mais realista e valores mais exatos do fator de intensidade de tensão, como foi abordado na secção 2.3.6, os nós de canto provisórios são usados para obter uma *cubic spline* que é posteriormente usada para reposicionar, quer os nós intermédios, quer os nós de canto.

4. DESENVOLVIMENTO DO MODELO DE PROPAGAÇÃO AUTOMÁTICA DE FENDA

No presente capítulo será descrito, de forma detalhada, o modo como foi criado o modelo de propagação automática de fenda, que se baseou no algoritmo apresentado na Figura 4.1. Esta figura apresenta, genericamente, as principais etapas do procedimento desenvolvido ao longo deste trabalho, que consistem, resumidamente, na fase de pré-processamento onde são lidos os dados de entrada, a parte de processamento onde são efetuadas as etapas principais da técnica de remalhagem adaptativa descrita no Capítulo 2, e a parte de pós-processamento que fornece ao utilizador os resultados adquiridos durante a simulação. A Figura 4.2 apresenta, de forma resumida, o fluxo do procedimento automático desenvolvido em termos de subrotinas e respetiva sequência de processamento. Nos subcapítulos seguintes, e com o auxílio dos Apêndices A e B, são descritos, de forma pormenorizada, os passos seguidos para a obtenção do modelo. No ponto 4.1 é analisada a fase de pré-processamento, no ponto 4.2 descreve-se a fase de processamento, e, por fim, no último subcapítulo, aborda-se a fase de pós-processamento. Relativamente aos apêndices, o primeiro apresenta o código integral das principais subrotinas, desenvolvido em Visual Basic 2012; e o segundo apresenta exemplos dos ficheiros de *input* (ficheiros ASCII) utilizados.

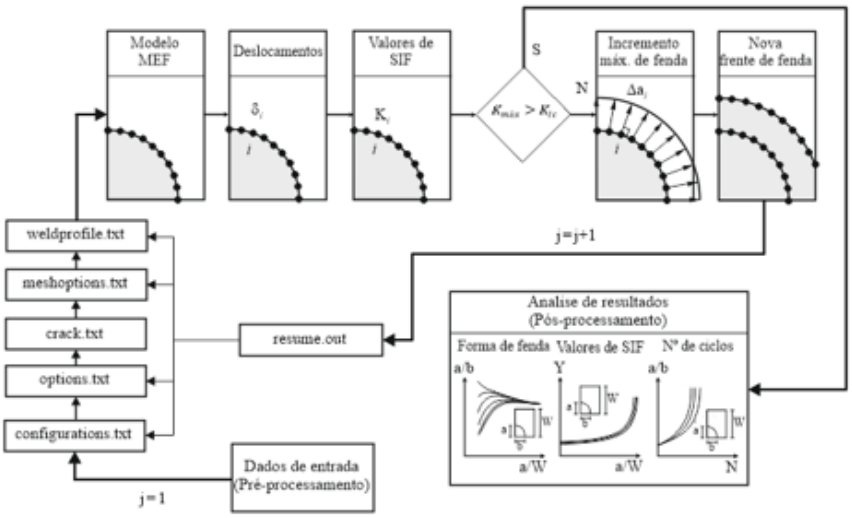

Figura 4.1. Algoritmo do procedimento numérico desenvolvido.

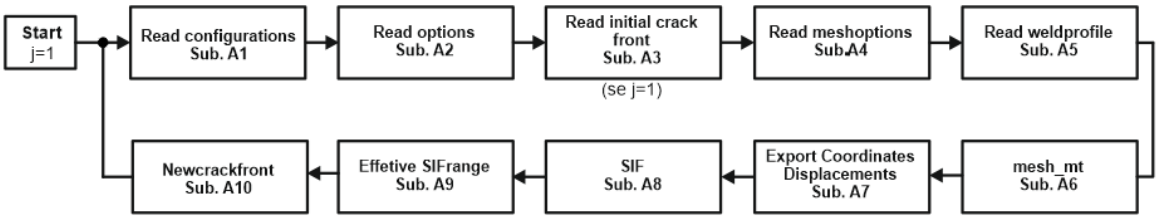

Figura 4.2. Resumo das principais subrotinas criadas para implementar o procedimento numérico desenvolvido.

4.1. Pré-Processamento

Todos os modelos numéricos de elementos finitos necessitam de dados de entrada para que, posteriormente, ocorra a parte do processamento.

Assim, para que este modelo se inicie, o programa tem de importar e ler a informação contida em cinco ficheiros de texto (.txt). Esta tarefa fica ao encargo do módulo *myinput.vb* que importa a informação dos ficheiros *configurations.txt* (Sub. A1), *options.txt* (Sub. A2), *crack.txt* (Sub. A3), *meshoptions.txt* (Sub. A4) e *weldprofile.txt* (Sub. A5). As Figuras B1 a B5 apresentam, respetivamente, exemplos genéricos dos ficheiros ASCII usados atualmente.

O ficheiro *configurations.txt* tem como principal objetivo ler informação geral para interação com o *software* de elementos finitos usado, bem como para fazer a gestão de pastas e ficheiros gerados durante a simulação (Figura B.1). O ficheiro *options.txt* contém as variáveis geométricas (Figura 3.2), o nível de penetração da soldadura (Figura 3.3), o tipo e magnitude do carregamento (Figura 3.5 e Figura 4.9), as constantes do material necessárias para a simulação (Tabela 3.1 e Tabela 3.2) e o incremento máximo de fenda (Figura B.2).

O ficheiro *crack.txt* (Figura B.3) contém as coordenadas polares dos nós da frente de fenda (Figura 3.2 (c)). Muito idêntico ao anterior ficheiro, temos, ainda, o ficheiro *resume.out*, que apresenta a mesma informação, mas guarda, também, o número de ciclos e de iterações já realizados. Este ficheiro substitui o ficheiro *crack.txt* (que é apenas lido no início da simulação) e, por outro lado, permite recomeçar uma simulação que tenha sido interrompida pelo utilizador a partir da última iteração já realizada (Figura B.6). As variáveis que permitem a definição da malha de elementos finitos, mencionadas no subcapítulo 3.2, encontram-se no ficheiro *meshoptions.txt* (Figura B.4).

No ficheiro *weldprofile.txt* está contida a informação sobre as coordenadas dos nós dos perfis dos cordões de soldadura fissurado e não fissurado, com a particularidade mencionada anteriormente, de que o cordão fissurado possui mais um nó, como se pode visualizar na Figura B.5.

4.2. Processamento

A descrição dos principais passos da fase de processamento é apresentada, em detalhe, neste ponto. Com o objetivo de tornar a descrição mais fluida e simples para o leitor, o código-fonte é apresentado no Apêndice A. No que diz respeito à subrotina *mesh_wt* que permite obter o modelo de elementos finitos (vide Figura 4.1), como esta é bastante extensa, as linhas foram numeradas (L1 a L1325) para facilitar a descrição do procedimento desenvolvido.

A primeira etapa da fase de processamento é a geração do modelo numérico. Essa geração é, essencialmente, efetuada através do módulo *mesh.vb* (Sub. A6). Inicialmente, são definidos os tipos elementos finitos e as propriedades elásticas do material (L5 a L8). Depois, são definidas, em coordenadas cartesianas, as coordenadas dos nós de canto da frente de fenda (L9 a L19). Estas coordenadas provêm do ficheiro *crack.txt* onde foram inicialmente inseridas em coordenadas polares. Da leitura deste ficheiro, na subrotina *readinitialcrackfront*, é também determinado o número máximo de nós, definido pelo utilizador, que é armazenado na variável *ncfp*. Este parâmetro, como se verá, terá grande influência no processo de desenvolvimento da malha, pois grande parte dos volumes do sólido a gerar serão indexados a este valor. A seguir, obtém-se os ângulos que definem as direções normais à frente de fenda (L14 a L19).

Após estas primeiras definições, cria-se a malha em teia de aranha, composta por três anéis. Para definir os anéis é necessário criar, de forma sequencial, pontos (L20 a L37; L74 a L91; L128 a L145), linhas (L38 a L54; L92 a L108; L146 a L162), superfícies (L55 a L66; L109 a L120; L163 a L174), e, por fim, volumes (L67 a L73; L121 a L127; L175 a L179). Para melhor compreensão, explica-se, de seguida, a definição do primeiro anel mostrado na Figura 4.3 (a), sendo os outros obtidos do mesmo modo. A criação dos pontos (L20 a L37) resume-se à definição de 10 pontos, dispostos em torno de cada nó da frente de fenda, conforme se representa na Figura 4.2 (a). A numeração indicada corresponde aos

pontos do nó mais interior, sendo nos restantes adotada igual sequência. No caso do nó superficial, junto ao cordão, identificado pelo valor *ncfp*, os pontos são dispostos através de código específico (L28 a L37), uma vez que estes possuem uma variação angular por contactar com o elemento especial do cordão de soldadura. Seguidamente, são definidas as linhas de ligação entre os pontos (L38 a L54), que posteriormente são usadas para criar as superfícies (L55 a L66), que, por fim, dão origem aos volumes do primeiro anel da malha em teia de aranha (L67 a L73). O aspeto final, após a geração das várias entidades mencionadas, encontra-se representado na Figura 4.3 (d). Repetindo o procedimento descrito para o segundo anel (L74 a L127) e, seguidamente, para o terceiro anel (L128 a L179) obtêm-se todos os volumes da malha em teia de aranha, que estão esquematizados na Figura 4.3 (e).

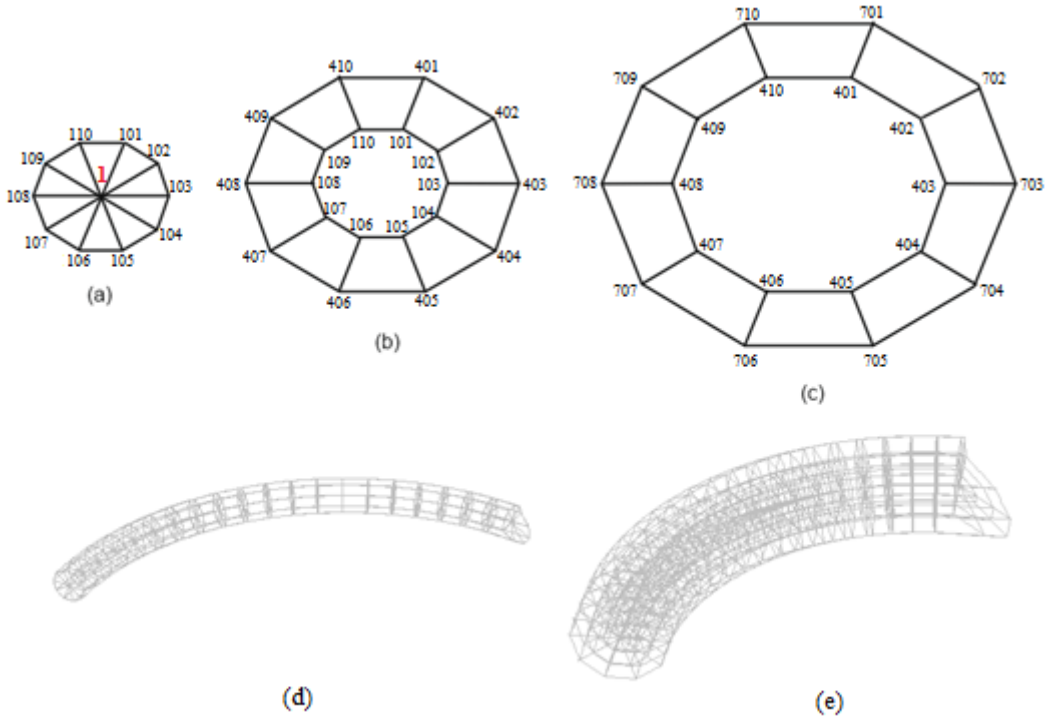

Figura 4.3. Malha em teia de aranha: (a) padrão 2D do 1º anel; (b) padrão 2D do 2º anel; (c) padrão 2D do 3º anel; (d) representação 3D primeiro anel; (e) representação 3D dos três anéis.

Na definição da malha de transição, foi usada uma abordagem idêntica à anterior. Inicialmente foi criado um padrão 2D, disposto ao longo da frente de fenda para cada um dos nós de canto, cujas dimensões são definidas através das variáveis $L_1, L_2, L_3, L_4, T_1, T_2, T_3$ e T_4 (Figura 3.6). Os pontos foram dispostos, para cada camada da malha, segundo a ordem

indicada na Figura 4.4 (a) (L187 a 239). Tal como na malha em teia de aranha, foi previsto um caso especial para a camada superficial, identificada por *ncfp* devido à inclinação do cordão de soldadura (L215 a L239). Seguidamente, criam-se as linhas que unem os pontos (L240 a L283), que permitem obter as superfícies (L284 a L308), a partir das quais se geram os volumes que fazem parte desta malha (L309 a L325). Na Figura 4.4 (b) apresenta-se um exemplo de uma malha de transição na sua versão final. Relativamente à malha de transição convém, ainda, referir que as camadas identificadas a cinzento na Figura 4.4 (a) foram adicionadas para facilitar a geração da malha regular. Estas camadas, como será explicado mais à frente, são sobrepostas umas sobre as outras, com espessuras variáveis (T_3 e T_4) que dependem da densidade de malha definida pelo utilizador no ficheiro *meshoptions.txt*. A camada inferior é usada para geração dos elementos acima da malha de transição (com o comando *elscale*) enquanto a camada superior é usada para gerar os elementos abaixo da malha de transição.

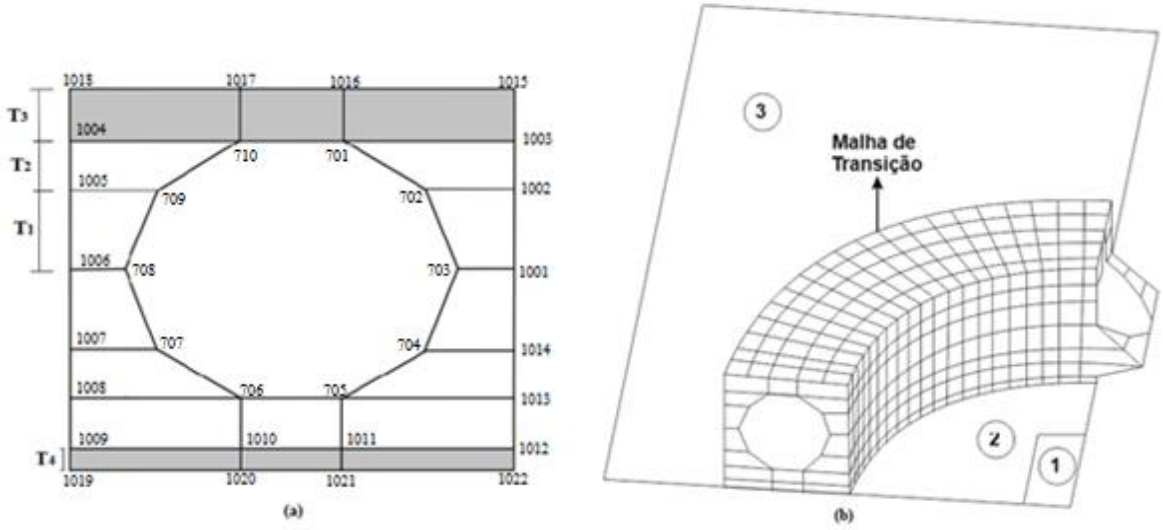

Figura 4.4. (a) Padrão 2D da malha de transição; (b) representação 3D da malha de transição.

No que diz respeito à malha regular, o processo implementado segue a metodologia descrita na Figura 3.7. No plano da fenda, na zona anterior à frente de fenda, são criadas duas zonas distintas, como pode ser visualizado na Figura 4.5 (b).

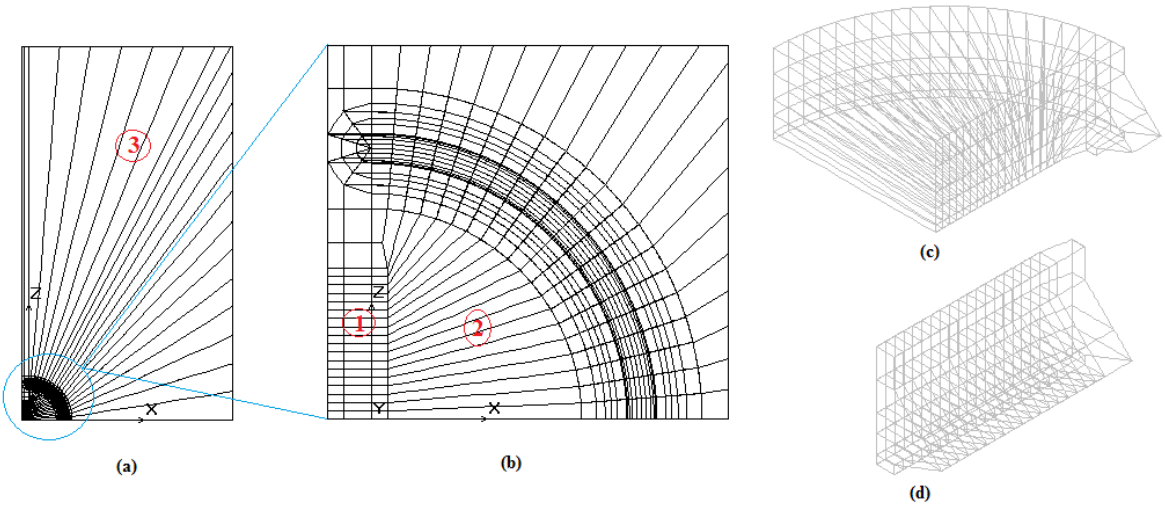

Figura 4.5. (a) Malha no plano da fenda; (b) ampliação da zona junto à frente de fenda; (c) representação 3D da malha da zona 1; (d) representação 3D da malha da zona 2.

Todos os passos, até a obtenção dos volumes desta zona, foram exatamente iguais aos utilizados anteriormente para as outras malhas. Primeiro, foram gerados os pontos ao longo de todo o contorno exterior (L434 a L468), seguiram-se as superfícies (L469 a L488), e, finalmente, os volumes (L489 a L500). Depois de criados todos os volumes do plano da fenda, passou-se a malhagem (L501 a L549). A malha foi criada para a zona em forma de teia de aranha, zona de transição, e malha regular no plano da fenda, utilizando os volumes gerados anteriormente. Posteriormente, criou-se a malha desde a camada superior da Figura 4.4 (a) até ao topo da peça (Fases 2 e 3 da Figura 3.7). A geração da malha a partir da camada inferior da Figura 4.4 (a) foi efetuada posteriormente por uma questão de conveniência no processo de obtenção da geometria.

Como foi referido anteriormente, a geração dos elementos das Fases 2 (L564 a L573) e 3 (L574 a 580) é obtida através da sobreposição de camadas idênticas às da Figura 4.4 (a) mas com espessuras diferentes, que são definidas, respetivamente, com o auxílio das variáveis NEL4 e NEL5, esquematizadas na Figura 3.7, e introduzidas no ficheiro *meshoptions.txt*. Nas Figura 4.6 (a) e (b) são representadas as malhas no final da Fase 2 e da Fase 3, respetivamente.

De modo a facilitar a aplicação das condições de fronteira no modelo de elementos finitos, a compressão (*compress*) e renumeração dos nós (*merge*) foram efetuadas por etapas. Inicialmente, estas tarefas foram executadas no final desta fase (L583 a 589).

Posteriormente foi realizada a malhagem da parte inferior do plano da fenda, através da utilização da camada inferior da Figura 4.4 (a).

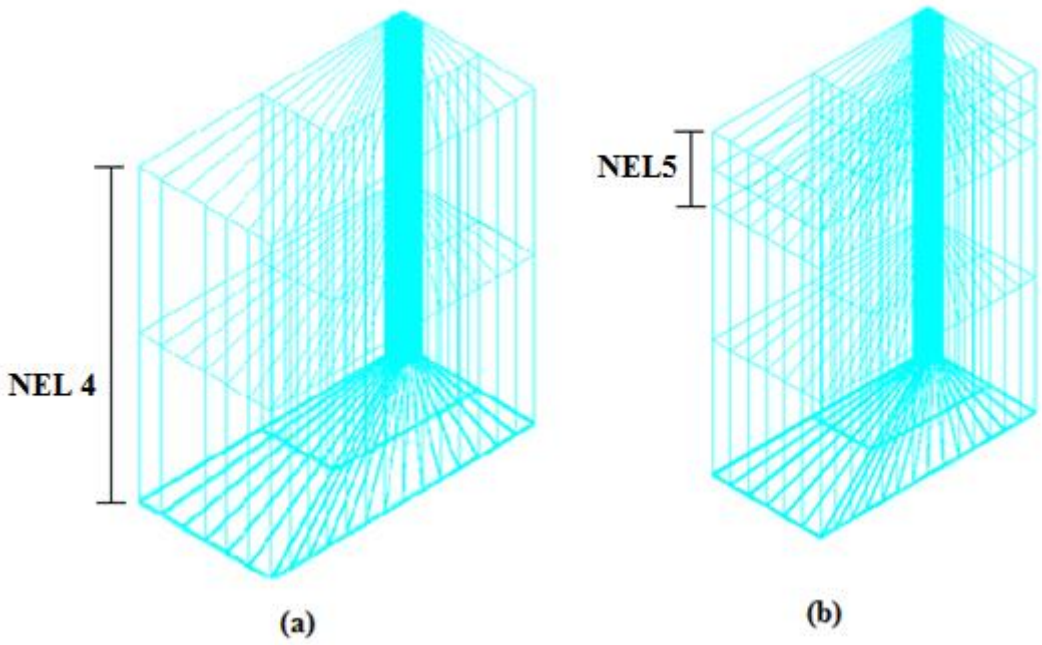

Figura 4.6. Geração da malha: (a) até aos apoios (NEL4=2); (b) até à extremidade superior do provete (NEL5=2).

O procedimento seguiu a filosofia descrita acima (L590 a L635).

A criação da geometria das juntas soldadas, que corresponde à Fase 4 representada na Figura 3.7, é efetuada a partir da linha 636. Como foi representado atrás, na Figura 3.3, o modelo permite definir o nível de penetração, de forma independente, para cada um dos lados do cordão, através das variáveis Lig(1) e Lig(2). Numa primeira abordagem, procura-se definir a geometria ótima (L643 a L662), em função das variáveis inseridas pelo utilizador no ficheiro *options.txt*. Segue-se a definição dos pontos do lado fissurado (L663 a L686), depois os pontos da zona não fissurada (L687 a L699), as superfícies (L700 a L721), e os volumes (L722 a L735) da camada mais interior. Na Figura 4.7 (a) apresenta-se uma vista dos volumes criados nesta fase. A partir dos volumes criados para a camada interior, obtém-se a malha tendo em conta o nível de refinamento pretendido. Nesta região, o refinamento é controlado em função das variáveis geométricas que dependem da geometria ótima do cordão descrita atrás e também da variável NEL7 (Figura 3.7) inserida no ficheiro de *input meshoptions.txt* (L736 a L791). Posteriormente, esta camada de elementos é gerada com base nas definições previamente mencionadas (L792 a 818). Após a geração destes

elementos faz-se uma nova compressão e renumeração dos nós para facilitar a introdução das condições de fronteira da parte não fissurada. Finalmente, os elementos desta camada são utilizados para gerar as restantes camadas, sendo as espessuras de cada camada dependentes das definições assumidas aquando da criação das malhas em teia de aranha, transição, e regular no plano da fenda (Fase 1). No final desta etapa, a malha referente à zona dos cordões tem o aspeto genérico representado na Figura 4.7 (b).

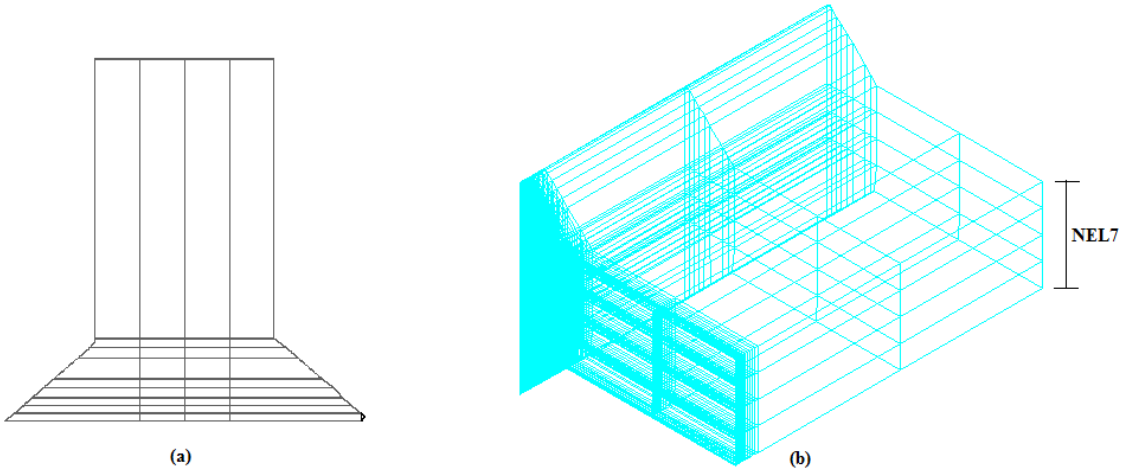

Figura 4.7. (a) Perfil 2D na zona dos cordões de soldadura; (b) malha de elementos finitos na zona dos cordões de soldadura.

Para que a geometria ficasse totalmente definida, foi necessário criar as Fases 5 e 6. O procedimento adotado foi idêntico ao descrito para as Fases 2 e 3. Neste caso, usou-se a camada inferior identificada na Figura 4.4 (a) e foram-se colocando camadas sobrepostas, umas sobre as outras, até à extremidade inferior da peça. Numa primeira fase foram geradas as camadas por baixo da zona criada anteriormente (L828 a L866) onde a espessura de cada uma delas depende da geometria estabelecida anteriormente (Figura 4.7 (b)). Seguidamente, foram geradas as camadas até à zona dos apoios (L883 a L932) e, por fim, as camadas até ao topo inferior da peça (L939 a L943). A densidade da malha nestas duas regiões, tal como se representa na Figura 4.8, é controlada pelas variáveis NEL6 e NEL5, respetivamente. Após esta etapa, que corresponde ao final da geração da malha de elementos, o modelo tem o aspeto genérico representado na Figura 4.8.

Entre as linhas 946 e 953, realizou-se a compressão e renumeração dos nós das geometrias anteriormente criadas, bem como dos nós da região não fissurada da frente de

fenda (L954 a L968) e da região ligada dos cordões de soldadura (L969 a L976). Tal como foi recomendado na literatura, e abordado no Capítulo 2, efetuou-se o reposicionamento radial dos nós intermédios do primeiro anel da frente de fenda para ¼ da aresta (L977 a L997).

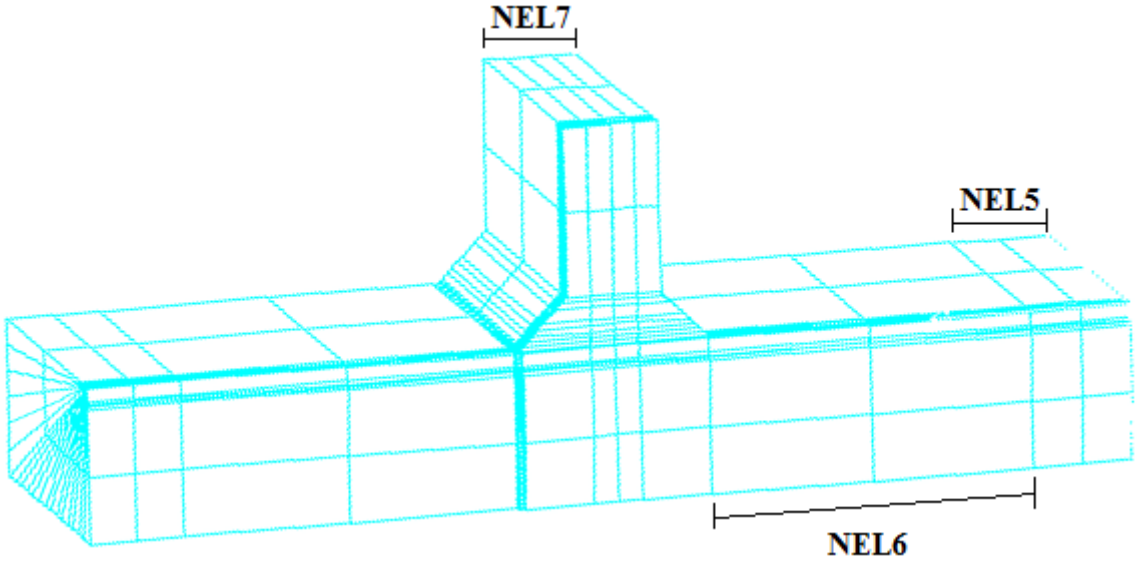

Figura 4.8. Geração da malha até à extremidade inferior do provete (NEL4=2 e NEL6=2).

Seguidamente, passou-se à definição das condições de fronteira e carregamento. Como foi referido no capítulo 3, foram previstos vários tipos de carregamento (tração, flexão e flexão-tração). Foram criados 16 cenários de carregamento implementados no modelo numérico desenvolvido. O código implementado que traduz estes cenários de carregamento encontra-se nas linhas 1015 a 1303, permitindo com que se obtenha os diferentes carregamentos. As variações que se garantem nos diferentes cenários é a possibilidade de se ter cargas aplicadas pontualmente, cargas distribuídas aplicadas sobre arestas ou superfícies. Por outro lado foram definidas para cada cenário as condições de fronteira que permitem cenários de tração simples, flexão-tração, flexão em três pontos e flexão em torno de um ou dois planos. Nesta fase foi ainda efetuada uma otimização da malha regular, de forma a esta ser igual ao longo de todo o corpo do provete. As alterações efetuadas face à Figura 4.8, consistiu na alteração do NEL4 e NEL6 para um valor igual a 4, havendo assim um aumento no número de nós para 25638 e do número de elementos para 5520.

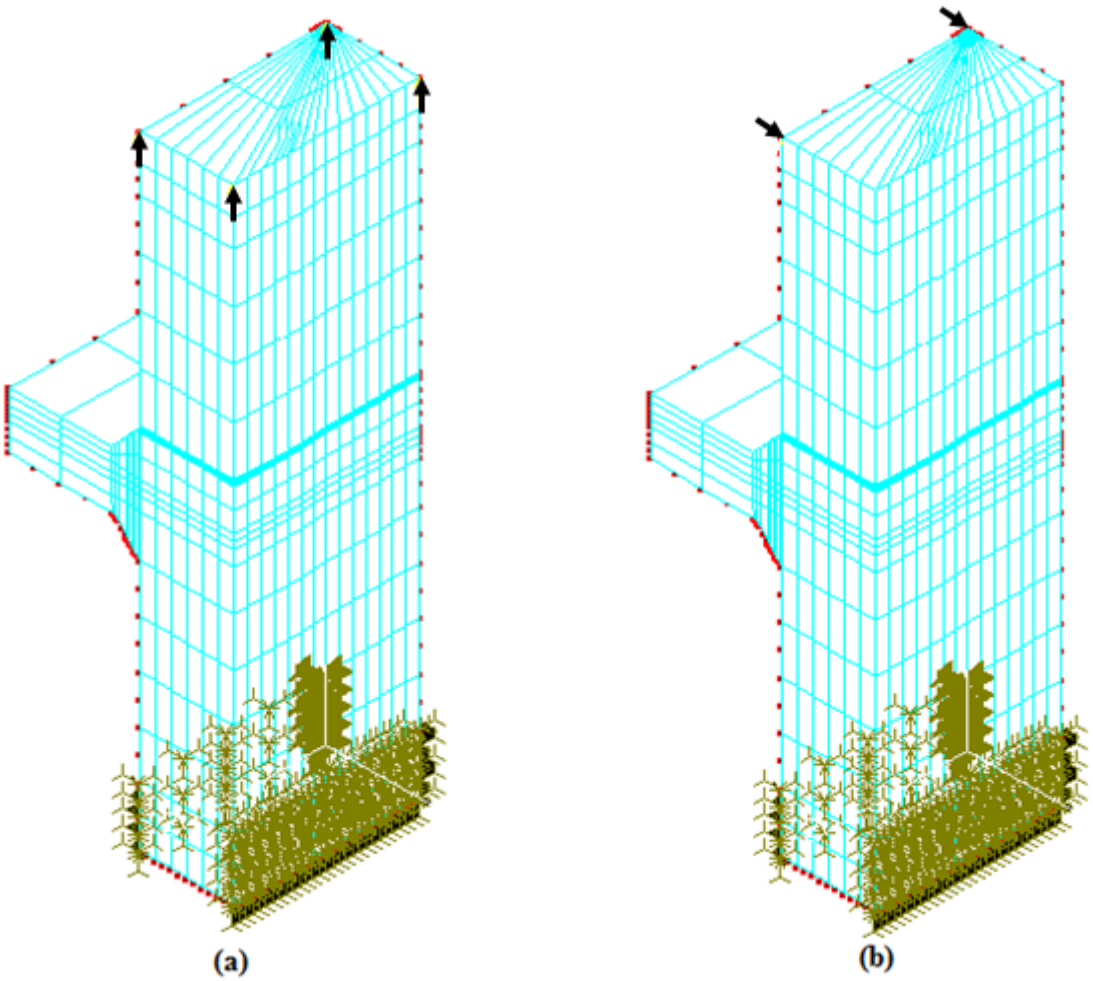

Figura 4.9. Vários cenários de carregamento: (a) fenda de canto sujeita a tração com cargas aplicadas nas extremidades; (b) fenda de canto sujeita a flexão com cargas aplicadas nas extremidades.

Depois da geração do modelo de elementos finitos, o *software* inicia a corrida (L1310). Quando a corrida termina, os deslocamentos e as coordenadas dos nós necessários ao cálculo do fator de intensidade de tensão são exportados para ficheiros ASCII específicos (L1311).

No passo seguinte do algoritmo da Figura 4.2, segue-se a subrotina A8, onde são calculados os valores do fator de intensidade de tensão. O método utilizado para determinar os valores de K, como foi descrito no subcapítulo 3.2.1, é o método da extrapolação com 2 pontos, que é, portanto, implementado nesta subrotina. Após estes cálculos, o *software* verifica se o valor máximo de K excedeu a tenacidade à fratura do material. Caso isso aconteça, a simulação é interrompida; caso contrário prossegue para o cálculo dos valores efetivos do fator de intensidade de tensão para os nós da frente de fenda, o que é efetuado na

subrotina A9, e, seguidamente, define uma nova frente de fenda, seguindo a metodologia descrita no subcapítulo 3.2.2, e que é implementada na subrotina A10.

Após o final da subrotina A10, o procedimento é novamente repetido, seguindo os passos descritos anteriormente. No essencial, há apenas a referir que a nova iteração parte de uma nova frente de fenda, cujas coordenadas foram o resultado da iteração anterior. O procedimento automático é interrompido quando se atinge a tenacidade à fratura do material ou quando o utilizador assim o entender.

4.3. Pós-Processamento

Após a realização da simulação, os resultados obtidos são bastante diversificados. A partir dos ficheiros de sessão (.ses) criados com a subrotina *mesh_wt* (Sub. A6), e com o recurso ao *software* de elementos finitos COSMOSM atualmente em uso, é possível, para cada incremento de fenda, obter informação importante acerca dos campos de deslocamento, de deformações e de tensões. Na Figura 4.10 (a), apresenta-se um exemplo da distribuição de tensões de von Mises na peça fissurada para um carregamento de tração, semelhante ao da Figura 3.5 (b) (as restantes variáveis de *input* consideradas estão listadas nas Figura B.1 a Figura B.4). Na Figura 4.10 (b), apresenta-se uma vista ampliada da zona da fenda com a representação da deformada prevista para a fenda nestas condições de carregamento.

Além disso, tal como representado na Figura 4.1, as variáveis obtidas com as simulações são as formas da frente de fenda, os valores dos fatores de intensidade de tensão, e os números de ciclos de fadiga. Na Figura 4.11 apresentam-se resultados típicos da evolução da frente de fenda obtidos a partir do modelo numérico desenvolvido neste trabalho. Foi considerada uma frente de fenda superficial, com forma semielíptica (aqui representada apenas pela parte modelada, e, por isso, correspondendo a uma forma em quarto-de-círculo), com comprimento inicial igual a 3 mm. A secção do plano da fenda tinha 100 mm ($Z1 = 50$ mm) por 25 mm. Aplicou-se um carregamento de flexão em três pontos correspondente ao caso 14 da Sub. A6 (L1239 a L1257) com razão de tensão $R = 0{,}5$. O incremento máximo de fenda ($\Delta a_{máx}$) utilizado foi igual a 0,05 mm. As malhas utilizadas foram semelhantes às apresentadas na Figura 4.9, exceto no que diz respeito ao carregamento

e às condições de fronteira. As restantes variáveis utilizadas na simulação foram idênticas às apresentadas nas Figura B.1 a Figura B.4 (Apêndice B). De modo a que a evolução da forma de fenda seja mais percetível do ponto de vista gráfico muitas das frentes de fenda não foram representadas na Figura 4.11. Como se pode verificar, a frente de fenda tende a avançar mais junto à superfície, junto ao pé do cordão, do que em pontos mais interiores. Essa diferença pode ser explicada, quer pela concentração de tensões introduzida pela geometria do cordão e o gradiente de tensões. Por outro lado, o estado plano de tensão tende a introduzir o fecho da fenda que causa o retardamento do ponto superficial. Observa-se, também, que a fenda tende para uma forma próxima de quarto-de-elipse (o que significa que a fenda real tende para uma forma aproximadamente semielíptica). Esse facto é também visível na Figura 4.12 que mostra a evolução do parâmetro adimensional a/b com o comprimento de fenda adimensional $b/X1$. Como se constata, há um período inicial caraterizado por uma alteração rápida de valores de a/b próximos de 1 para valores de a/b maiores do que 1. Por outro lado, embora os dados da figura não sejam totalmente conclusivos, à medida que a fenda propaga, a/b parece descrever uma trajetória no sentido de um valor assimptótico, que será, entre outros aspetos, afetado pelo fator de concentração de tensões resultante da geometria do cordão.

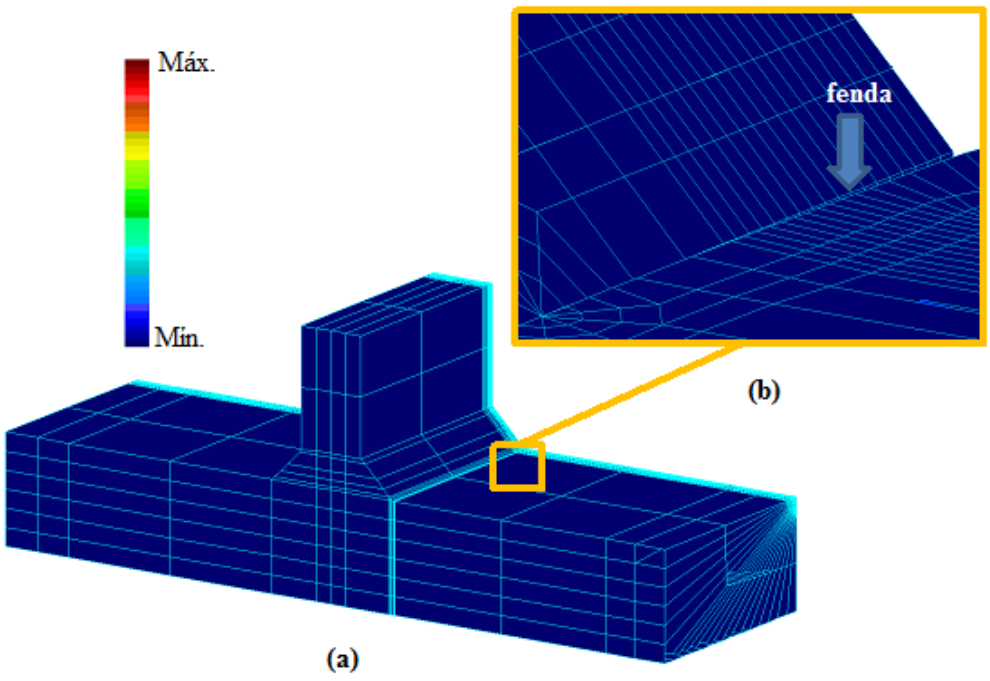

Figura 4.10. (a) Representação das tensões de von Mises na peça; (b) detalhe na região da fenda.

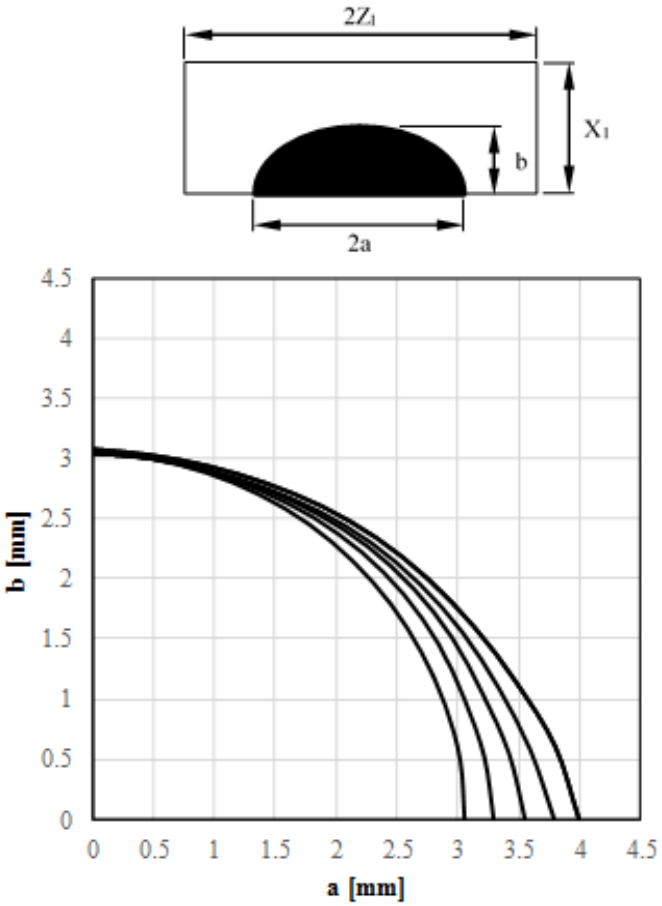

Figura 4.11. Evolução da frente de fenda para uma fenda superficial com forma inicial semicircular com comprimento igual a 3 mm sujeita a flexão em três pontos.

Na Figura 4.13 representa-se a evolução do fator de intensidade de tensão ao longo da frente de fenda (em função de θ, igual a 0º no nó mais interior, e igual a 90º no nó superficial) para as quatro formas de fenda apresentadas na figura anterior. Para facilitar a comparação de resultados, os valores de K foram divididos pelo valor máximo do fator de intensidade de tensão da respetiva frente de fenda. Pode verificar-se que para a frente de fenda inicial, a fenda superficial semicircular, a relação $K_i/K_{máx}$ varia de forma muito acentuada com θ. No entanto, à medida que a fenda se propaga, as diferenças entre os valores máximos deste parâmetro, observadas à superfície, e os valores mínimos, observados no nó mais interior, tendem a diminuir. Para a última frente de fenda (a=3.99 mm), o valor mínimo é superior a 0.6 e o máximo é igual a 1. Para a primeira frente de fenda (a=3.00 mm), o valor mínimo era próximo de 0. Esta evolução parece indiciar uma tendência para um iso-K, i.e. uma relação $K_{mín}/K_{máx}$ próxima de 1.

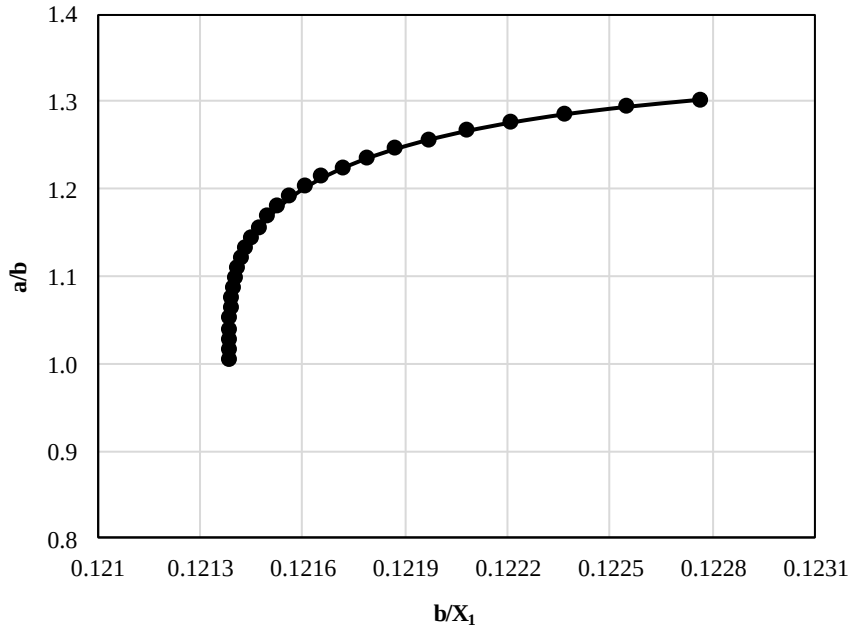

Figura 4.12. Evolução do parâmetro adimensional a/b com o comprimento de fenda adimensional b/X1.

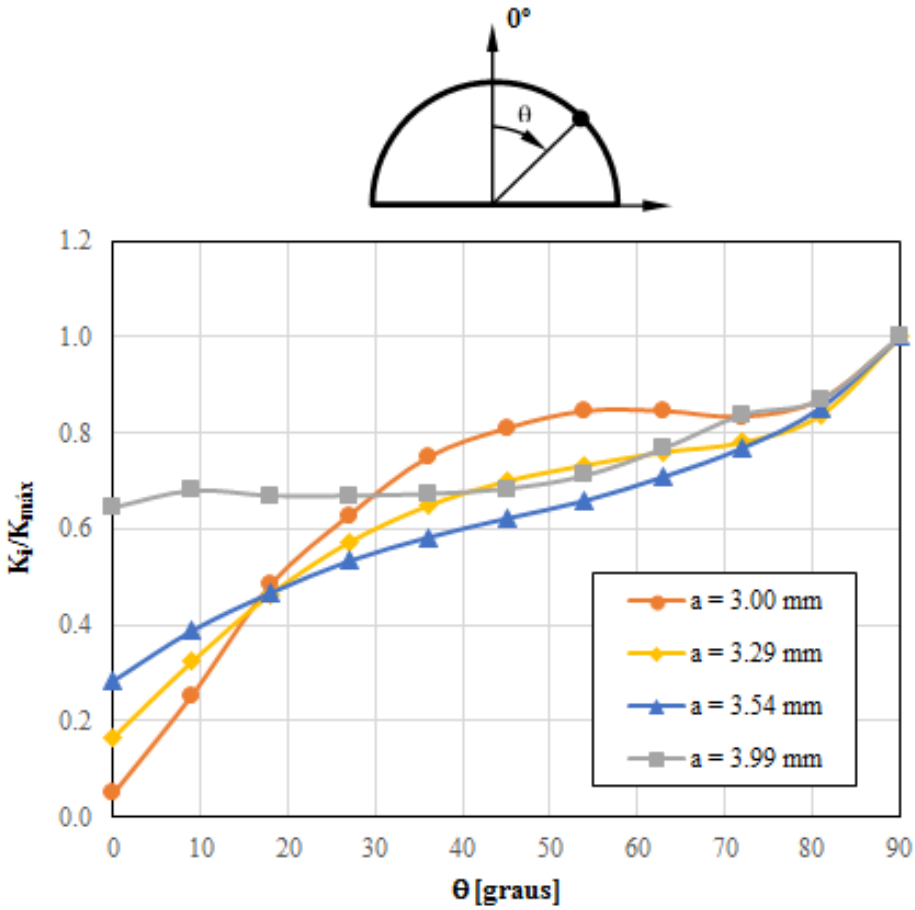

Figura 4.13. Variação do $K_i/K_{máx}$ ao longo da frente de fenda para diferentes comprimentos de fenda.

5. CONCLUSÕES E PERSPETIVAS FUTURAS

O objetivo principal deste trabalho era o desenvolvimento de um procedimento automático para estudo de problemas de propagação de fendas por fadiga em juntas soldadas em forma de T sujeitas a cargas cíclicas de amplitude constante. Pretendia-se que o procedimento a desenvolver utilizasse modelos de elementos finitos tridimensionais e que permitisse simular explicitamente a evolução da forma da fenda.

O procedimento foi desenvolvido através da linguagem de programação Visual Basic 2012 e cumpre totalmente os objetivos delineados inicialmente. A simulação explícita da forma da fenda foi conseguida através da implementação de uma técnica de remalhagem adaptativa que pode ser dividida em cinco passos principais repetidos ciclicamente: (i) desenvolvimento de um modelo de elementos finitos representativo da peça fissurada; (ii) obtenção do campo de deslocamentos; (iii) cálculo das gamas efetivas do fator de intensidade de tensão dos nós da frente de fenda; (iv) determinação dos incrementos nodais aplicando curvas $da/dN - \Delta K$ experimentais; e, por fim, (v) definição da nova frente de fenda.

A malha tridimensional de elementos finitos é constituída por três regiões: uma região em forma de teia de aranha na frente de fenda constituída por três anéis concêntricos; uma malha de transição; e uma malha regular. O campo de deslocamentos do modelo numérico é obtido a partir da ferramenta computacional COSMOSM que integra o *software* comercial SOLIDWORKS 2014. Atualmente, o cálculo dos valores do fator de intensidade de tensão na frente de fenda é efetuado com o método da extrapolação com dois pontos. A definição da frente de fenda é definida com base em funções *cubic spline* que permitem reposicionar os nós de canto e os nós intermédios de forma mais suave contribuindo para valores mais exatos dos fatores de intensidade de tensão, e, consequentemente, para simulações mais realistas.

O modelo de elementos finitos, um dos aspetos cruciais deste trabalho, é bastante flexível e versátil. Podem salientar-se os seguintes aspetos:

i. a frente de fenda é definida a partir de um número variável de nós, o que permite acomodar diferentes dimensões e formas iniciais;

ii. o modelo tanto pode simular fendas de canto como fendas superficiais centrais, neste último caso aplicando condições de simetria;
iii. é possível considerar diferentes tipos de carregamento em Modo-I, incluindo tração, flexão, e tração-flexão;
iv. o perfil do cordão da zona fissurada e o perfil do cordão da zona não fissurada podem ser definidos de forma independente e flexível;
v. podem ser definidos os níveis de penetração da soldadura dos lados fissurado e não fissurado dos cordões;
vi. a densidade da malha pode ser controlada de forma eficiente ao longo de toda a malha regular, incluindo as zonas dos cordões;
vii. o modelo permite estudar diferentes tipos de assimetrias geométricas, nomeadamente no que diz respeito às formas dos perfis dos cordões, às dimensões dos cordões, às dimensões dos apoios, e às percentagens de penetração dos cordões de soldadura; entre outros;

Embora os objetivos iniciais tenham sido atingidos, para que o procedimento automático funcione na sua plenitude, é ainda necessário executar mais algumas tarefas. Nesse sentido, propõe-se como trabalho futuro:

i. otimizar as principais variáveis numéricas, incluindo, entre outros, o comprimento radial dos elementos finitos do primeiro anel da frente de fenda; o valor do incremento máximo de fenda; a densidade da malha, etc.;
ii. validar os resultados numéricos comparando-os com resultados experimentais obtidos para casos idênticos de propagação;
iii. implementar um método energético de cálculo de K, por exemplo o Integral-J, e fazer uma análise de sensibilidade dos valores obtidos com aqueles que se obtêm com o método da extrapolação com 2 pontos;
iv. desenvolver uma interface gráfica, baseada em janelas *Windows*, para facilitar: a introdução de dados (atualmente efetuada a partir de ficheiros ASCII); o acompanhamento em tempo real ao longo da simulação da evolução da forma da fenda, valores do fator de intensidade de tensão, e da vida de fadiga; o tratamento de resultados após términos da simulação.

v. desenvolver *plug-ins* que permitam a leitura e análise dos modelos numéricos tridimensionais a partir de outros softwares comerciais de elementos finitos usados pelos investigadores de fenómenos de fadiga e fratura, tais como, ABAQUS, ANSYS, MARC, entre outros.
vi. Consideração das tensões residuais no modelo.
vii. Simular o comportamento local do material.

REFERÊNCIAS BIBLIOGRÁFICAS

Antunes FV (1999a). "Influence of frequency, stress ratio and stress state on fatigue crack growth in nickel base superalloys at elevated temperature." PhD thesis, Department of mechanical and manufacturing engineering, University of Portsmouth, United Kingdom.

Antunes FV, Ferreira JM, Byrne J (1999b). "Stress intensity factor calculation based on the work of external forces." Int J Fract; 98:1–14.

Antunes FV, Ferreira JM, Costa JD, Capela C (2002). "Fatigue life predictions in polymer particle composites." Int J Fatigue; 24:1095–105.

Banks-Sills L, Wawrzynek PA, Carter BJ, Ingraffea AR, Hershkovitz I (2007). "Methods for calculating stress intensity factors in anisotropic materials: Part II - Arbitrary geometry." Engng Fract Mech; 74:1293–307.

Bakker A (1992). "Three-dimensional constraint effects on stress intensity distributions in plate geometries with through-thickness cracks." Fatigue Fract Engng Mater Struct; 15:1051–69.

Barsoum RS (1979). "On the use of isoparametric finite elements in linear fracture mechanics." Int J Numer Methods Engng; 10:25–37.

Bazant ZP, Estenssoro LF (1979). "Surface singularity and crack propagation." Int J Solids Struct; 15:405–26.

Borrego L. F. P. (2001). Fatigue crack growth under variable amplitude loading in AlMgSi aluminium alloys. PhD thesis, University of Coimbra, Portugal.

Branco R (2006). "Estudo numérico de propagação de fendas por fadiga em provetes M(T)." Tese de Mestrado, Departamento em Engenharia Mecânica, Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Branco R, Antunes FV (2008a). "Finite element modelling and analysis of crack shape evolution in mode-I fatigue middle cracked tension specimens." Engng Fract Mech; 75:3020–37.

Branco R, Antunes FV, Martins RF (2008b). "Modelling fatigue crack propagation in CT specimens." Fatigue Fract Engng Mater Struct; 31:452–65.

Branco C., Ferreira J., Costa J., & Ribeiro A. (2012a). Projecto de Órgãos de Máquinas.

Branco R, Antunes FV, Costa JD, Yang F, Kuang Z (2012b). "Determination of the Paris law constants in round bars from beach marks on fracture surfaces." Engng Fract Mech; 96:96–106.

Branco R, Antunes FV, Costa JD (2013). "Extent of the surface region in notched middle cracked tension specimens." In: Alibadi MH (Ed.), Key engineering materials. Special issue on crack growth modelling; 560:107–27.

Branco R, Antunes FV, Costa JD (2014). "Notched M(T) specimen for plane strain studies." Int J Fatigue; 58:28–39.

Branco R, Antunes FV, Costa JD (2015). "A review on 3D-FE adaptive remeshing techniques for crack growth modelling." Engng Fract Mech; 141:170-195.

Carpinteri A, Brighenti R (1996). "Part-through cracks in round bars under cyclic combined axial and bending loading." Int J Fatigue; 18:33–9.

Carpinteri A, Vantadori S (2009). "Sickle-shaped cracks in metallic round bars under cyclic eccentric axial loading." Int J Fatigue; 31:759–65.

Carpinteri A, Brighenti R, Vantadori S (2010). "Influence of the cold-drawing process on fatigue crack growth of a V-notched round bar." Int J Fatigue; 32:1136–45.

Chan SK, Tuba IS, Wilson WK (1970). "On the finite element method in linear fracture mechanics." Engng Fract Mech; 2:1–17.

Dhondt G (2014). "Application of the finite element method to mixed-mode cyclic crack propagation calculations in specimens." Int J Fatigue; 58:2–11.

Erdogan F, Ratwani M (1970). "Fatigue and fracture of cylindrical shells containing a circumferential crack." Int J Fract Mech; 6:379–92.

Fulland M, Sander M, Kullmer G, Richard HA (2008). "Analysis of fatigue crack propagation in the frame of a hydraulic press." Engng Fract Mech; 75:892–900.

Forman RG, Kearney VE, Engle RM (1967). "Numerical analysis of crack propagation in cyclically loaded structures." Trans ASME. J Basic Engng; 89:459–64.

Gavete L, Michavila F, Díez F (1989). "A new singularity finite element in linear elasticity." Comput Mech; 4:361–71.

Guinea GV, Planas J, Elices M (2000). "KI evaluation by the displacement extrapolation technique." Engng Fract Mech; 66:243–55.

Hellen TK (1975). "On the method of virtual crack extension." Int J Numer Meth Engng; 9:187–207.

Henshell RD, Shaw KG (1975). "Crack tip finite elements are unnecessary." Int J Numer Methods Engng; 9:495–507.

Irwin G. R. (1958). "Fracture in: Encyclopedia of Physics", S. Flugge, Vol. VI, Springer Verlag, 551-590.

Kanninen MF, Popelar CH (1985). "Advanced fracture mechanics." New York: Oxford University Press.

Lin XB, Smith RA (1995). "Numerical prediction of fatigue crack growth of a surface defect." Fatigue Fract Engng Mater Struct; 18:247–56.

Lin XB, Smith RA (1997). "An improved numerical technique for simulating the growth of planar fatigue cracks." Fatigue Fract Engng Mater Struct; 20:1363–73.

Lin XB, Smith RA (1999a). "Finite element modelling of fatigue crack growth of surface cracked plates. Part I: The numerical technique." Engng Fract Mech; 63:503–22.

Lin XB, Smith RA (1999b). "Finite element modelling of fatigue crack growth of surface cracked plates. Part II: Crack shape change." Engng Fract Mech; 63:523–40.

Lin XB, Smith RA (1999c). "Finite element modelling of fatigue crack growth of surface cracked plates. Part III: Stress intensity factor and fatigue crack growth." Engng Fract Mech; 63:541–56.

Lin XB, Smith RA (1999d). "Shape evolution of surface cracks in fatigued round bars with a semicircular circumferential notch." Int J Fatigue; 21:965–73.

Lin XB, Smith RA (2001). "Numerical simulation of fatigue crack growth for corner cracks emanating from fastener holes." Notch effects in fatigue and fracture. NATO Sci Ser; 11:271–87.

Mahmoud MA (1988). "Growth patterns of surface fatigue cracks under cyclic bending – a quantitative analysis." Engng Fract Mech; 31:357–69.

Mahmoud MA (1990). "Surface fatigue crack growth under combined tension and bending loading." Engng Fract Mech; 36:389–95.

Murakami T, Sato T (1983). "Three-dimensional J-integral calculation of part-through surface crack problems." Comput Struct; 17:731–6.

Murti V, Valliappan S (1986). "A universal optimum quarter point element." Engng Fract Mech; 25:237–58.

Newman Jr JC, Raju IS (1979). "Analysis of surface cracks in finite plates under tension and bending loads." NASA TP-1578, National Aeronautics and Space Administration.

Newman Jr JC, Raju IS (1981). "An empirical stress intensity factor equation for the surface crack." Engng Fract Mech; 15:185–92.

Newman Jr JC, Raju IS (1983). "Stress-intensity factor equations for cracks in three-dimensional finite bodies". ASTM 791, American Society for Testing and Materials. DOI:http://dx.doi.org/10.1520/STP37074S.

Nykänen TJ (1996). "Fatigue crack growth simulations based on free front shape development." Fatigue Fract Engng Mater Struct; 19:99–109.

Paiva L. (2015). Propagação de fendas por fadiga: Efeito de subcargas. Dissertação de Mestrado, Universidade de Coimbra, Portugal.

Paris P. C., & Erdogan F. (1963). Journal of Basic Engineering, Trans. ASME, American Society of Mechanical Engineers.

Pook LP (1994). "Some implications of corner point singularities". Engng Fract Mech; 48:367–78.

Rabold F, Kuna M (2014). "Automated finite element simulation of fatigue crack growth in three-dimensional structures with the software system ProCrack". Proc Mater Sci; 3:1099–104.

Ribeiro R. F. (2011). "Efeito da sobrecarga e da espessura na vida em fadiga de componentes de aço API "H grau 50". Monografia, Universidade Federal do Rio Grande Sul, Porto Alegra.

Sedmak A, Savovic N, Pavisic M (1992). ESIS recommendations for use of finite element method in fracture mechanics. In: Sedmak S, Sedmak A, Ruzié A (Eds.), 9th European conference on fracture (ECF9), Bulgaria. Reliability and Structural Integrity of Advanced Materials, EMAS.

Shih CF, Moran B, Nakamura T (1986). "Energy release rate along a three-dimensional crack front in a thermally stresses body". Int J Fract; 30:79–102.

Smith RA, Cooper JF (1989). "A finite element model for the shape development of irregular planar cracks". Int J Press Vessels Pip; 36:315–26.

Zhao, T., Zhang, J. e Jiang, Y. (2008), "A study of fatigue crack growth of 7075-T651 aluminum alloy", International Journal of Fatigue; 30:1169–1180.

APÊNDICE A

```
Sub readconfigurations()

        'find configuration file
 1        configpath = Main.TextBox1.Text

 2        If My.Computer.FileSystem.FileExists(configpath) Then
 3        Else
 4            Main.ListBox1.Items.Add(Now & "  Configuration file not found")
 5            Main.ListBox1.Items.Add(Now & "  Please add file into the configuration
directory")
 6            Main.ListBox1.Items.Add(Now & "  or update configuration path defined in
textbox")
 7            Main.ListBox1.Items.Add(Now & "  Current directory: " &
Main.TextBox1.Text)
 8            Main.ListBox1.Items.Add(Now & "  Simulation has been stopped")
 9            verify = 1
 10            Exit Sub
 11        End If

        'read file
 12        FileOpen(1, configpath, OpenMode.Input)
 13        xC = 0
 14        Do While Not EOF(1)
 15            xC = xC + 1
 16            CD(xC) = LineInput(1)
 17        Loop
 18        FileClose(1)

        'define variables
 19        exepath = CD(2)
 20        execlose = CD(5)
 21        folderpath = CD(8)
 22        deletefolder = CD(11)

 23    End Sub
```

Subrotina A1. Subrotina readconfigurations()

```
Sub readoptions()

        'find options file
 1        If My.Computer.FileSystem.FileExists(folderpath & "\" & "options.txt") Then
 2        Else
 3            Main.ListBox1.Items.Add(Now & "  Options file not found")
 4            Main.ListBox1.Items.Add(Now & "  folderpath " & folderpath)
 5            Main.ListBox1.Items.Add(Now & "  Please add file into the configuration
directory")
 6            Main.ListBox1.Items.Add(Now & "  Current directory: " &
Main.TextBox1.Text)
 7            Main.ListBox1.Items.Add(Now & "  Simulation has been stopped")
 8            verify = 1
 9            Exit Sub
```

```
10        End If

        'define directory
11        ChDir(folderpath)
12        FileOpen(1, "options.txt", OpenMode.Input)
13        xC = 0
14        Do While Not EOF(1)
15            xC = xC + 1
16            opt(xC) = LineInput(1)
17        Loop
18        FileClose(1)

        'define variables
19        vX(1) = opt(2)
20        vX(2) = opt(5)
21        vX(3) = opt(8)
22        vX(4) = opt(11)
23        vY(1) = opt(14)
24        vY(2) = opt(17)
25        vY(3) = opt(20)
26        vY(4) = opt(23)
27        vY(5) = opt(26)
28        vY(6) = opt(29)
29        vY(7) = opt(32)
30        vZ(1) = opt(35)
31        lig(1) = opt(38)
32        lig(2) = opt(41)
33        EX = opt(44)
34        NUXY = opt(47)
35        LDT = opt(50)
36        Fmin = opt(53)
37        Fmax = opt(56)
38        DAmax = opt(59)
39        CCC = opt(62)
40        mmm = opt(65)
41        KIC = opt(68)
42        myname = opt(71)
43        SYM = opt(74)

44    End Sub
```

Subrotina A2. Subrotina readoptions()

```
Sub readinitialcrackfront()

        'find crack file
1         If My.Computer.FileSystem.FileExists(folderpath & "\" & "crack.txt") Then
2         Else
3             Main.ListBox1.Items.Add(Now & "  Crack front file not found")
4             Main.ListBox1.Items.Add(Now & "  Please add file into the configuration
directory")
5             Main.ListBox1.Items.Add(Now & "  Current directory: " &
Main.TextBox1.Text)
6             Main.ListBox1.Items.Add(Now & "  Simulation has been stopped")
7             verify = 1
8             Exit Sub
9         End If

        'define directory
10        ChDir(folderpath)
11        FileOpen(1, "crack.txt", OpenMode.Input)
```

```
        xC = 0
        Do While Not EOF(1)
            xC = xC + 1
            Input(1, cfd(xC))
        Loop
        FileClose(1)
For i = 1 To (xC - 2) / 2
            iAngle(i) = cfd(2 * i + 1)
            iRadius(i) = cfd(2 * i + 2)
        Next

        ncfp = (xC - 2) / 2
        For i = 1 To ncfp
            x(i) = Math.Abs(iRadius(i) * Math.Cos(iAngle(i) * (Math.PI / 180)))
            z(i) = iRadius(i) * Math.Sin(iAngle(i) * (Math.PI / 180))
        Next


   End Sub
```

Subrotina A3. Subrotina readinitialcrackfront()

```
Sub readmeshoptions()

       'find mesh options file
        If My.Computer.FileSystem.FileExists(folderpath & "\" & "meshoptions.txt")
Then
        Else
            Main.ListBox1.Items.Add(Now & "  Mesh options file not found")
            Main.ListBox1.Items.Add(Now & "  Please add file into the configuration
directory")
            Main.ListBox1.Items.Add(Now & "  Current directory: " &
Main.TextBox1.Text)
            Main.ListBox1.Items.Add(Now & "  Simulation has been stopped")
            verify = 1
            Exit Sub
        End If

       'define directory
        ChDir(folderpath)
        FileOpen(1, "meshoptions.txt", OpenMode.Input)
        xC = 0
        Do While Not EOF(1)
            xC = xC + 1
            meshopt(xC) = LineInput(1)
        Loop
        FileClose(1)

       'define variables
        NEL1 = meshopt(14)
        NEL2 = meshopt(17)
        NEL3 = meshopt(20)
        NEL4 = meshopt(23)
        NEL5 = meshopt(26)
        NEL6 = meshopt(29)
        NEL7 = meshopt(32)

        DistL1a = vX(1) / meshopt(2)
        DistL2a = vX(1) / meshopt(5)
        DistL3a = vX(1) / meshopt(8)
        DTrans1 = vX(1) / meshopt(11)
```

```
29    End Sub
```

Subrotina A4. Subrotina readmeshoptions()

```
Sub readweldprofile()

        'find crack file
1         If My.Computer.FileSystem.FileExists(folderpath & "\" & "weldprofile.txt")
Then
2         Else
3             Main.ListBox1.Items.Add(Now & "  Weld profile definitions not found")
4             Main.ListBox1.Items.Add(Now & "  Please add file into the configuration
directory")
5            Main.ListBox1.Items.Add(Now & "  Current directory: " & Main.TextBox1.Text)
6             Main.ListBox1.Items.Add(Now & "  Simulation has been stopped")
7             verify = 1
8             Exit Sub
9         End If

        'define directory
10          ChDir(folderpath)
11          FileOpen(1, "weldprofile.txt", OpenMode.Input)
12          xC = 0
13          Do While Not EOF(1)
14              xC = xC + 1
15              Input(1, wpi(xC))
16          Loop
17          FileClose(1)

        'define coordinates
18          For i = 1 To (xC - 6) / 6
19              xw(i) = wpi(6 * i + 2)
20              yw(i) = wpi(6 * i + 3)
21              xwnf(i) = wpi(6 * i + 11)
22              ywnf(i) = wpi(6 * i + 12)
23          Next

        'define number of crack front nodes
24          nwp = (xC - 6) / 6

        'define weld angle
25          weld_angle(1) = Math.Atan(yw(1) / xw(1))


26    End Sub
```

Subrotina A5. Subrotina readweldprofile()

```
    Sub mesh_wt()

        'create folder and mesh file
1         ChDir(folderpath & "\" & myname)
2         My.Computer.FileSystem.CreateDirectory(it)
3         FileOpen(2, "mesh-" & it & ".ses", OpenMode.Append)
4         Print(2, "C* Welded T-joint (surface crack with symmetry plane)" & vbCrLf)
```

```
        'material and type of element
 5        Print(2, "EGROUP,1,SOLID,0,1,0,0,0,0,0,0" & vbCrLf)
 6        Print(2, "MPROP,1,EX," & EX & vbCrLf)
 7        Print(2, "MPROP,1,NUXY," & NUXY & vbCrLf)
 8        Print(2, "RCONST,1,1,1,9,0,0,0,0,0,0,0,0,0" & vbCrLf)

        'crack front
        'points
 9       Print(2, "C* Crack front coordinates " & vbCrLf)
 10        For i = 1 To ncfp
 11            Print(2, "PT," & i & "," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," &
z(i) & vbCrLf)
 12        Next
        'normal direction to the crack front
 13     angle(1) = 0

 14        For i = 2 To ncfp - 1
 15            aux_angle(1) = Math.Atan((x(i - 1) - x(i)) / (z(i - 1) - z(i))) * (180 /
Math.PI)
 16            aux_angle(2) = Math.Atan((x(i) - x(i + 1)) / (z(i) - z(i + 1))) * (180 /
Math.PI)
 17            angle(i) = (aux_angle(2) + aux_angle(1)) * 0.5
 18        Next
 19        angle(ncfp) = -90


        'first concentric ring
        'points of the first concentric ring
 20        Print(2, " " & vbCrLf)
 21        Print(2, "C* Points of the first concentric ring" & vbCrLf)

 22        For i = 1 To ncfp - 1
 23            Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i)
& ",0," & angle(i) & ",0,0" & vbCrLf)
 24            For j = 1 To 10
 25                Print(2, "PT," & 100 + 10 * (i - 1) + j & "," & DistL1a & "," & 36 *
(j - 1) & "," & 0 & vbCrLf)
 26            Next
 27        Next

 28        For i = ncfp To ncfp
 29            Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i)
& ",0," & angle(i) & ",0,0" & vbCrLf)
 30            For j = 1 To 10
 31                If 36 * (j - 1) > 180 Then
 32                    Print(2, "PT," & 100 + 10 * (i - 1) + j & "," & DistL1a & "," &
36 * (j - 1) & "," & -(DistL1a * Math.Sin(36 * (j - 1) * (Math.PI / 180))) *
Math.Tan(weld_angle(1)) & vbCrLf)
 33                Else
 34                    Print(2, "PT," & 100 + 10 * (i - 1) + j & "," & DistL1a & "," &
36 * (j - 1) & "," & 0 & vbCrLf)
 35                End If
 36            Next
 37        Next

        'curves of the first concentric ring
 38      Print(2, " " & vbCrLf)
 39      Print(2, "C* Curves of the first concentric ring" & vbCrLf)
 40      For j = 0 To ncfp - 1
 41          For i = 0 To 9
 42              Print(2, "CRLINE," & (10 * j) + (i + 1) & "," & j + 1 & "," & (101 + i)
+ (10 * j) & vbCrLf)
```

```
 43          Next
 44      Next

 45      For j = 0 To ncfp - 1
 46          For i = 0 To 8
 47              Print(2, "CRLINE," & (301 + 10 * j) + i & "," & (101 + 10 * j) + i &
"," & (102 + 10 * j) + i & vbCrLf)
 48          Next
 49      Next

 50      For j = 0 To ncfp - 1
 51          For i = 9 To 9
 52              Print(2, "CRLINE," & (301 + 10 * j) + i & "," & (101 + 10 * j) & "," &
(101 + 10 * j) + i & vbCrLf)
 53          Next
 54      Next

        'surfaces of the first concentric ring
 55      Print(2, " " & vbCrLf)
 56      Print(2, "C* Surfaces of the first concentric ring" & vbCrLf)
 57      For j = 0 To ncfp - 1
 58          For i = 0 To 8
 59              Print(2, "SF3CR," & (i + 1) + 10 * j & "," & 1 + (i + 10 * j) & "," & 2
+ (i + 10 * j) & "," & 301 + (i + 10 * j) & ",0" & vbCrLf)
 60          Next
 61      Next

 62      For j = 0 To ncfp - 1
 63          For i = 9 To 9
 64              Print(2, "SF3CR," & (i + 1) + 10 * j & "," & 1 + (10 * j) & "," & 1 +
(i + 10 * j) & "," & 301 + (i + 10 * j) & ",0" & vbCrLf)
 65          Next
 66      Next

        'volumes of the first concentric ring
 67      Print(2, " " & vbCrLf)
 68      Print(2, "C* Volumes of the first concentric ring" & vbCrLf)
 69      For j = 0 To ncfp - 2
 70          For i = 0 To 9
 71              Print(2, "VL2SF," & (i + 1) + 10 * j & "," & (1 + i) + 10 * j & "," &
(11 + i) + 10 * j & ",1" & vbCrLf)
 72          Next
 73      Next
     'second concentric ring
        'points of the second concentric ring
 74       Print(2, " " & vbCrLf)
 75       Print(2, "C* Points of the second concentric ring" & vbCrLf)
 76       For i = 1 To ncfp - 1
 77           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
 78           For j = 1 To 10
 79               Print(2, "PT," & 400 + 10 * (i - 1) + j & "," & DistL1a + DistL2a &
"," & 36 * (j - 1) & "," & 0 & vbCrLf)
 80           Next
 81       Next

 82       For i = ncfp To ncfp
 83           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
 84           For j = 1 To 10
 85               If 36 * (j - 1) > 180 Then
```

```
 86                  Print(2, "PT," & 400 + 10 * (i - 1) + j & "," & DistL1a + DistL2a
& "," & 36 * (j - 1) & "," & -((DistL1a + DistL2a) * Math.Sin(36 * (j - 1) * (Math.PI /
180))) * Math.Tan(weld_angle(1)) & vbCrLf)
 87              Else
 88                  Print(2, "PT," & 400 + 10 * (i - 1) + j & "," & DistL1a + DistL2a
& "," & 36 * (j - 1) & "," & 0 & vbCrLf)
 89              End If
 90          Next
 91      Next

       'curves of the second concentric ring
 92      Print(2, " " & vbCrLf)
 93      Print(2, "C* Curves of the second concentric ring " & vbCrLf)
 94      For j = 0 To ncfp - 1
 95          For i = 0 To 9
 96              Print(2, "CRLINE," & (10 * j) + (i + 1001) & "," & (101 + i) + (10 *
j) & "," & (401 + i) + (10 * j) & vbCrLf)
 97          Next
 98      Next

 99      For j = 0 To ncfp - 1
 100         For i = 0 To 8
 101             Print(2, "CRLINE," & (1301 + 10 * j) + i & "," & (401 + 10 * j) + i &
"," & (402 + 10 * j) + i & vbCrLf)
 102         Next
 103     Next

 104     For j = 0 To ncfp - 1
 105         For i = 9 To 9
 106             Print(2, "CRLINE," & (1301 + 10 * j) + i & "," & (401 + 10 * j) & ","
& (401 + 10 * j) + i & vbCrLf)
 107         Next
 108     Next

       'surfaces of the second concentric ring
 109     Print(2, " " & vbCrLf)
 110     Print(2, "C* Surfaces of the second concentric ring" & vbCrLf)
 111     For j = 0 To ncfp - 1
 112         For i = 0 To 8
 113             Print(2, "SF4CR," & (i + 801) + 10 * j & "," & 1001 + (i + 10 * j) &
"," & 1301 + (i + 10 * j) & "," & 1002 + (i + 10 * j) & "," & 301 + (i + 10 * j) & ",0"
& vbCrLf)
 114         Next
 115     Next

 116     For j = 0 To ncfp - 1
 117         For i = 9 To 9
 118             Print(2, "SF4CR," & (i + 801) + 10 * j & "," & 1001 + (i + 10 * j) &
"," & 1301 + (i + 10 * j) & "," & 1001 + 10 * j & "," & 1301 + (i + 10 * j) & ",0" &
vbCrLf)
 119         Next
 120     Next

       'volumes of the second concentric ring
 121     Print(2, " " & vbCrLf)
 122     Print(2, "C* Volumes of the second concentric ring" & vbCrLf)
 123     For j = 0 To ncfp - 2
 124         For i = 0 To 9
 125             Print(2, "VL2SF," & (i + 201) + 10 * j & "," & (801 + i) + 10 * j &
"," & (811 + i) + 10 * j & ",1" & vbCrLf)
 126         Next
 127     Next
```

```
        'third concentric ring
        'points of the second concentric ring
 128       Print(2, " " & vbCrLf)
 129       Print(2, "C* Points of the third concentric ring" & vbCrLf)
 130       For i = 1 To ncfp - 1
 131           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i)
& ",0," & angle(i) & ",0,0" & vbCrLf)
 132           For j = 1 To 10
 133               Print(2, "PT," & 700 + 10 * (i - 1) + j & "," & DistL1a + DistL2a +
DistL3a & "," & 36 * (j - 1) & "," & 0 & vbCrLf)
 134           Next
 135       Next

 136       For i = ncfp To ncfp
 137           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i)
& ",0," & angle(i) & ",0,0" & vbCrLf)
 138           For j = 1 To 10
 139               If 36 * (j - 1) > 180 Then
 140                   Print(2, "PT," & 700 + 10 * (i - 1) + j & "," & DistL1a + DistL2a
+ DistL3a & "," & 36 * (j - 1) & "," & -((DistL1a + DistL2a + DistL3a) * Math.Sin(36 *
(j - 1) * (Math.PI / 180))) * Math.Tan(weld_angle(1)) & vbCrLf)
 141               Else
 142                   Print(2, "PT," & 700 + 10 * (i - 1) + j & "," & DistL1a + DistL2a
+ DistL3a & "," & 36 * (j - 1) & "," & 0 & vbCrLf)
 143               End If
 144           Next
 145       Next

        'curves of the third concentric ring
 146       Print(2, " " & vbCrLf)
 147       Print(2, "C* Curves of the second concentric ring " & vbCrLf)
 148       For j = 0 To ncfp - 1
 149           For i = 0 To 9
 150               Print(2, "CRLINE," & (10 * j) + (i + 1801) & "," & (401 + i) + (10 *
j) & "," & (701 + i) + (10 * j) & vbCrLf)
 151           Next
 152       Next

 153       For j = 0 To ncfp - 1
 154           For i = 0 To 8
 155               Print(2, "CRLINE," & (2101 + 10 * j) + i & "," & (701 + 10 * j) + i &
"," & (702 + 10 * j) + i & vbCrLf)
 156           Next
 157       Next

 158       For j = 0 To ncfp - 1
 159           For i = 9 To 9
 160               Print(2, "CRLINE," & (2101 + 10 * j) + i & "," & (701 + 10 * j) & ","
& (701 + 10 * j) + i & vbCrLf)
 161           Next
 162       Next

        'surfaces of the third concentric ring
 163       Print(2, " " & vbCrLf)
 164       Print(2, "C* Surfaces of the second concentric ring (+)" & vbCrLf)
 165       For j = 0 To ncfp - 1
 166           For i = 0 To 8
 167               Print(2, "SF4CR," & (i + 1501) + 10 * j & "," & 1801 + (i + 10 * j) &
"," & 2101 + (i + 10 * j) & "," & 1802 + (i + 10 * j) & "," & 1301 + (i + 10 * j) & ",0"
& vbCrLf)
 168           Next
 169       Next
```

```
170       For j = 0 To ncfp - 1
171           For i = 9 To 9
172               Print(2, "SF4CR," & (i + 1501) + 10 * j & "," & 1801 + (i + 10 * j) &
"," & 2101 + (i + 10 * j) & "," & 1801 + 10 * j & "," & 2101 + (i + 10 * j) & ",0" &
vbCrLf)
173           Next
174       Next

       'volumes of the third concentric ring
175       Print(2, " " & vbCrLf)
176       Print(2, "C* Volumes of the second concentric ring" & vbCrLf)
177       For j = 0 To ncfp - 2
178           For i = 0 To 9
179               Print(2, "VL2SF," & (i + 401) + 10 * j & "," & (1501 + i) + 10 * j &
"," & (1511 + i) + 10 * j & ",1" & vbCrLf)
178           Next
179       Next
       'transition mesh
180       aux_angle(3) = (DistL1a + DistL2a + DistL3a) * Math.Cos(2 * 36 * (Math.PI /
180))
181       aux_angle(4) = (DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180))
182       aux_angle(5) = (DistL1a + DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI /
180))
183       aux_angle(6) = (DistL1a + DistL2a + DistL3a + DTrans1)
184       aux_angle(7) = ((DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI /
180))) * Math.Tan(weld_angle(1))
185       aux_angle(8) = ((DistL1a + DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI /
180))) * Math.Tan(weld_angle(1))
186       aux_angle(9) = xw(2)

       'points of transition mesh
187       Print(2, " " & vbCrLf)
188       Print(2, "C* Transition mesh" & vbCrLf)
189       Print(2, "C* Points" & vbCrLf)
190       For i = 1 To ncfp - 1
191           Print(2, "CSYS,4,0," & i & "," & 101 + 10 * (i - 1) & "," & 102 + 10 * (i
- 1) & vbCrLf)
192           Print(2, "PT," & 1001 + 22 * (i - 1) & "," & aux_angle(6) & ",0,0" &
vbCrLf)
193           Print(2, "PT," & 1002 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(4) & ",0" & vbCrLf)
194           Print(2, "PT," & 1003 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(5) & ",0" & vbCrLf)
195           Print(2, "PT," & 1004 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(5) & ",0" & vbCrLf)
196           Print(2, "PT," & 1005 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(4) & ",0" & vbCrLf)
197           Print(2, "PT," & 1006 + 22 * (i - 1) & "," & -aux_angle(6) & ",0,0" &
vbCrLf)
198           Print(2, "PT," & 1007 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(4) & ",0" & vbCrLf)
199           Print(2, "PT," & 1008 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(5) & ",0" & vbCrLf)
200           Print(2, "PT," & 1009 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(6) & ",0" & vbCrLf)
201           Print(2, "PT," & 1010 + 22 * (i - 1) & "," & -aux_angle(3) & "," & -
aux_angle(6) & ",0" & vbCrLf)
202           Print(2, "PT," & 1011 + 22 * (i - 1) & "," & aux_angle(3) & "," & -
aux_angle(6) & ",0" & vbCrLf)
203           Print(2, "PT," & 1012 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(6) & ",0" & vbCrLf)
204           Print(2, "PT," & 1013 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(5) & ",0" & vbCrLf)
```

```
 205           Print(2, "PT," & 1014 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(4) & ",0" & vbCrLf)
 206           Print(2, "PT," & 1015 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(6) & ",0" & vbCrLf)
 207           Print(2, "PT," & 1016 + 22 * (i - 1) & "," & aux_angle(3) & "," &
aux_angle(6) & ",0" & vbCrLf)
 208           Print(2, "PT," & 1017 + 22 * (i - 1) & "," & -aux_angle(3) & "," &
aux_angle(6) & ",0" & vbCrLf)
 209           Print(2, "PT," & 1018 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(6) & ",0" & vbCrLf)
 210           Print(2, "PT," & 1019 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(9) & ",0" & vbCrLf)
 211           Print(2, "PT," & 1020 + 22 * (i - 1) & "," & -aux_angle(3) & "," & -
aux_angle(9) & ",0" & vbCrLf)
 212           Print(2, "PT," & 1021 + 22 * (i - 1) & "," & aux_angle(3) & "," & -
aux_angle(9) & ",0" & vbCrLf)
 213           Print(2, "PT," & 1022 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(9) & ",0" & vbCrLf)
 214       Next

 215       For i = ncfp To ncfp
 216           Print(2, "CSYS,4,0," & i & "," & 101 + 10 * (i - 1) & "," & 102 + 10 * (i
- 1) & vbCrLf)
 217           Print(2, "PT," & 1001 + 22 * (i - 1) & "," & aux_angle(6) & ",0,0" &
vbCrLf)
 218           Print(2, "PT," & 1002 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(4) & ",0" & vbCrLf)
 219           Print(2, "PT," & 1003 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(5) & ",0" & vbCrLf)
 220           Print(2, "PT," & 1004 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(5) & ",0" & vbCrLf)
 221           Print(2, "PT," & 1005 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(4) & ",0" & vbCrLf)
 222           Print(2, "PT," & 1006 + 22 * (i - 1) & "," & -aux_angle(6) & ",0,0" &
vbCrLf)
 223           Print(2, "PT," & 1007 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(4) & "," & aux_angle(7) & vbCrLf)
 224           Print(2, "PT," & 1008 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(5) & "," & aux_angle(8) & vbCrLf)
 225           Print(2, "PT," & 1009 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(6) & ",0" & vbCrLf)
 226           Print(2, "PT," & 1010 + 22 * (i - 1) & "," & -aux_angle(3) & "," & -
aux_angle(6) & ",0" & vbCrLf)
 227           Print(2, "PT," & 1011 + 22 * (i - 1) & "," & aux_angle(3) & "," & -
aux_angle(6) & ",0" & vbCrLf)
 228           Print(2, "PT," & 1012 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(6) & ",0" & vbCrLf)
 229           Print(2, "PT," & 1013 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(5) & "," & aux_angle(8) & vbCrLf)
 230           Print(2, "PT," & 1014 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(4) & "," & aux_angle(7) & vbCrLf)
 231           Print(2, "PT," & 1015 + 22 * (i - 1) & "," & aux_angle(6) & "," &
aux_angle(6) & ",0" & vbCrLf)
 232           Print(2, "PT," & 1016 + 22 * (i - 1) & "," & aux_angle(3) & "," &
aux_angle(6) & ",0" & vbCrLf)
 233           Print(2, "PT," & 1017 + 22 * (i - 1) & "," & -aux_angle(3) & "," &
aux_angle(6) & ",0" & vbCrLf)
 234           Print(2, "PT," & 1018 + 22 * (i - 1) & "," & -aux_angle(6) & "," &
aux_angle(6) & ",0" & vbCrLf)
 235           Print(2, "PT," & 1019 + 22 * (i - 1) & "," & -aux_angle(6) & "," & -
aux_angle(9) & ",0" & vbCrLf)
 236           Print(2, "PT," & 1020 + 22 * (i - 1) & "," & -aux_angle(3) & "," & -
aux_angle(9) & ",0" & vbCrLf)
```

```
237           Print(2, "PT," & 1021 + 22 * (i - 1) & "," & aux_angle(3) & "," & -
aux_angle(9) & ",0" & vbCrLf)
238           Print(2, "PT," & 1022 + 22 * (i - 1) & "," & aux_angle(6) & "," & -
aux_angle(9) & ",0" & vbCrLf)
239       Next


      'curves of transition mesh
240       Print(2, " " & vbCrLf)
241       Print(2, "C* Curves" & vbCrLf)
242       For i = 0 To ncfp - 1
243           Print(2, "CRLINE," & 2551 + 30 * i & "," & (701 + 10 * i) & "," & (1001 +
22 * i) & vbCrLf)
244           Print(2, "CRLINE," & 2552 + 30 * i & "," & (702 + 10 * i) & "," & (1002 +
22 * i) & vbCrLf)
245           Print(2, "CRLINE," & 2553 + 30 * i & "," & (703 + 10 * i) & "," & (1003 +
22 * i) & vbCrLf)
246           Print(2, "CRLINE," & 2554 + 30 * i & "," & (704 + 10 * i) & "," & (1004 +
22 * i) & vbCrLf)
247           Print(2, "CRLINE," & 2555 + 30 * i & "," & (705 + 10 * i) & "," & (1005 +
22 * i) & vbCrLf)
248           Print(2, "CRLINE," & 2556 + 30 * i & "," & (706 + 10 * i) & "," & (1006 +
22 * i) & vbCrLf)
249           Print(2, "CRLINE," & 2557 + 30 * i & "," & (707 + 10 * i) & "," & (1007 +
22 * i) & vbCrLf)
250           Print(2, "CRLINE," & 2558 + 30 * i & "," & (708 + 10 * i) & "," & (1008 +
22 * i) & vbCrLf)
251           Print(2, "CRLINE," & 2559 + 30 * i & "," & (708 + 10 * i) & "," & (1010 +
22 * i) & vbCrLf)
252           Print(2, "CRLINE," & 2560 + 30 * i & "," & (709 + 10 * i) & "," & (1011 +
22 * i) & vbCrLf)
253           Print(2, "CRLINE," & 2561 + 30 * i & "," & (709 + 10 * i) & "," & (1013 +
22 * i) & vbCrLf)
254           Print(2, "CRLINE," & 2562 + 30 * i & "," & (710 + 10 * i) & "," & (1014 +
22 * i) & vbCrLf)
255       Next

256       For i = 0 To ncfp - 1
257           Print(2, "CRLINE," & 2563 + (30 * i) & "," & 1001 + (22 * i) & "," & 1002
+ (22 * i) & vbCrLf)
258           Print(2, "CRLINE," & 2564 + (30 * i) & "," & 1002 + (22 * i) & "," & 1003
+ (22 * i) & vbCrLf)
259           For j = 0 To 9
260               Print(2, "CRLINE," & 2565 + j + (30 * i) & "," & 1004 + j + (22 * i)
& "," & 1005 + j + (22 * i) & vbCrLf)
261           Next
262       Next

263       For i = 0 To ncfp - 1
264           Print(2, "CRLINE," & 2575 + 30 * i & "," & (1014 + 22 * i) & "," & (1001
+ 22 * i) & vbCrLf)
265       Next

      'for generation 1
266       For i = 0 To ncfp - 1
267           Print(2, "CRLINE," & 3181 + 7 * i & "," & (1003 + 22 * i) & "," & (1015 +
22 * i) & vbCrLf)
268           Print(2, "CRLINE," & 3182 + 7 * i & "," & (703 + 10 * i) & "," & (1016 +
22 * i) & vbCrLf)
269           Print(2, "CRLINE," & 3183 + 7 * i & "," & (704 + 10 * i) & "," & (1017 +
22 * i) & vbCrLf)
270           Print(2, "CRLINE," & 3184 + 7 * i & "," & (1004 + 22 * i) & "," & (1018 +
22 * i) & vbCrLf)
```

```
271           Print(2, "CRLINE," & 3185 + 7 * i & "," & (1015 + 22 * i) & "," & (1016 +
22 * i) & vbCrLf)
272           Print(2, "CRLINE," & 3186 + 7 * i & "," & (1016 + 22 * i) & "," & (1017 +
22 * i) & vbCrLf)
273           Print(2, "CRLINE," & 3187 + 7 * i & "," & (1017 + 22 * i) & "," & (1018 +
22 * i) & vbCrLf)
274       Next

      'for generation 2
275       For i = 0 To ncfp - 1
276           Print(2, "CRLINE," & 3351 + 7 * i & "," & (1009 + 22 * i) & "," & (1019 +
22 * i) & vbCrLf)
277           Print(2, "CRLINE," & 3352 + 7 * i & "," & (1010 + 22 * i) & "," & (1020 +
22 * i) & vbCrLf)
278           Print(2, "CRLINE," & 3353 + 7 * i & "," & (1011 + 22 * i) & "," & (1021 +
22 * i) & vbCrLf)
279           Print(2, "CRLINE," & 3354 + 7 * i & "," & (1012 + 22 * i) & "," & (1022 +
22 * i) & vbCrLf)
280           Print(2, "CRLINE," & 3355 + 7 * i & "," & (1019 + 22 * i) & "," & (1020 +
22 * i) & vbCrLf)
281           Print(2, "CRLINE," & 3356 + 7 * i & "," & (1020 + 22 * i) & "," & (1021 +
22 * i) & vbCrLf)
282           Print(2, "CRLINE," & 3357 + 7 * i & "," & (1021 + 22 * i) & "," & (1022 +
22 * i) & vbCrLf)
283       Next

      'surfaces of transition mesh
284       Print(2, " " & vbCrLf)
285       Print(2, "C* Surfaces" & vbCrLf)
286       For i = 0 To ncfp - 1
287           Print(2, "SF4CR," & 2201 + 11 * i & "," & 2551 + 30 * i & "," & 2563 + 30
* i & "," & 2552 + 30 * i & "," & 2101 + 10 * i & ",0" & vbCrLf)
288           Print(2, "SF4CR," & 2202 + 11 * i & "," & 2552 + 30 * i & "," & 2564 + 30
* i & "," & 2553 + 30 * i & "," & 2102 + 10 * i & ",0" & vbCrLf)
289           Print(2, "SF4CR," & 2203 + 11 * i & "," & 2554 + 30 * i & "," & 2565 + 30
* i & "," & 2555 + 30 * i & "," & 2104 + 10 * i & ",0" & vbCrLf)
290           Print(2, "SF4CR," & 2204 + 11 * i & "," & 2555 + 30 * i & "," & 2566 + 30
* i & "," & 2556 + 30 * i & "," & 2105 + 10 * i & ",0" & vbCrLf)
291           Print(2, "SF4CR," & 2205 + 11 * i & "," & 2556 + 30 * i & "," & 2567 + 30
* i & "," & 2557 + 30 * i & "," & 2106 + 10 * i & ",0" & vbCrLf)
292           Print(2, "SF4CR," & 2206 + 11 * i & "," & 2557 + 30 * i & "," & 2568 + 30
* i & "," & 2558 + 30 * i & "," & 2107 + 10 * i & ",0" & vbCrLf)
293           Print(2, "SF4CR," & 2207 + 11 * i & "," & 2558 + 30 * i & "," & 2569 + 30
* i & "," & 2570 + 30 * i & "," & 2559 + 30 * i & ",0" & vbCrLf)
294           Print(2, "SF4CR," & 2208 + 11 * i & "," & 2108 + 10 * i & "," & 2559 + 30
* i & "," & 2571 + 30 * i & "," & 2560 + 30 * i & ",0" & vbCrLf)
295           Print(2, "SF4CR," & 2209 + 11 * i & "," & 2561 + 30 * i & "," & 2560 + 30
* i & "," & 2572 + 30 * i & "," & 2573 + 30 * i & ",0" & vbCrLf)
296           Print(2, "SF4CR," & 2210 + 11 * i & "," & 2562 + 30 * i & "," & 2109 + 10
* i & "," & 2561 + 30 * i & "," & 2574 + 30 * i & ",0" & vbCrLf)
297           Print(2, "SF4CR," & 2211 + 11 * i & "," & 2551 + 30 * i & "," & 2110 + 10
* i & "," & 2562 + 30 * i & "," & 2575 + 30 * i & ",0" & vbCrLf)
298       Next

      'for generation 1
299       For i = 0 To ncfp - 1
300           Print(2, "SF4CR," & 2451 + 3 * i & "," & 3181 + 7 * i & "," & 3185 + 7 *
i & "," & 3182 + 7 * i & "," & 2553 + 30 * i & ",0" & vbCrLf)
301           Print(2, "SF4CR," & 2452 + 3 * i & "," & 3182 + 7 * i & "," & 3186 + 7 *
i & "," & 3183 + 7 * i & "," & 2103 + 10 * i & ",0" & vbCrLf)
302           Print(2, "SF4CR," & 2453 + 3 * i & "," & 3183 + 7 * i & "," & 3187 + 7 *
i & "," & 3184 + 7 * i & "," & 2554 + 30 * i & ",0" & vbCrLf)
303       Next
```

```
        'for generation 2
 304        For i = 0 To ncfp - 1
 305            Print(2, "SF4CR," & 2521 + 3 * i & "," & 3351 + 7 * i & "," & 3355 + 7 *
i & "," & 3352 + 7 * i & "," & 2570 + 30 * i & ",0" & vbCrLf)
 306            Print(2, "SF4CR," & 2522 + 3 * i & "," & 3352 + 7 * i & "," & 3356 + 7 *
i & "," & 3353 + 7 * i & "," & 2571 + 30 * i & ",0" & vbCrLf)
 307            Print(2, "SF4CR," & 2523 + 3 * i & "," & 3353 + 7 * i & "," & 3357 + 7 *
i & "," & 3354 + 7 * i & "," & 2572 + 30 * i & ",0" & vbCrLf)
 308        Next

        'volumes of transition mesh
 309        Print(2, " " & vbCrLf)
 310        Print(2, "C* Volumes of transition mesh" & vbCrLf)
 311        For j = 0 To ncfp - 2
 312            For i = 0 To 10
 313                Print(2, "VL2SF," & (i + 601) + 11 * j & "," & (2201 + i) + 11 * j &
"," & (2212 + i) + 11 * j & ",1" & vbCrLf)
 314            Next
 315        Next

        'for generation 1
 316        For j = 0 To ncfp - 2
 317            For i = 0 To 2
 318                Print(2, "VL2SF," & (i + 821) + 3 * j & "," & (2451 + i) + 3 * j &
"," & (2454 + i) + 3 * j & ",1" & vbCrLf)
 319            Next
 320        Next

        'for generation 2
 321        For j = 0 To ncfp - 2
 322            For i = 0 To 2
 323                Print(2, "VL2SF," & (i + 881) + 3 * j & "," & (2521 + i) + 3 * j &
"," & (2524 + i) + 3 * j & ",1" & vbCrLf)
 324            Next
 325        Next

        'behind the crack
 326        Print(2, " " & vbCrLf)
 327        Print(2, "C* Mesh behing the crack" & vbCrLf)
 328        Print(2, "C* Points (Part 1)" & vbCrLf)
 329        Print(2, "ACTSET,CS,0"& vbCrLf)
 330        Print(2, "PT," & 1702 & ",0," & vY(1) + vY(2) + vY(5) & ",0" & vbCrLf)

 331        aux_dist(1) = z(ncfp - 1)
 332        aux_dist(2) = x(ncfp - 1) - (DistL1a + DistL2a + DistL3a + DTrans1) *
Math.Cos(angle(ncfp - 1) * (Math.PI / 180))
 333        aux_dist(3) = z(ncfp - 1) + (DistL1a + DistL2a + DistL3a + DTrans1) *
Math.Sin(angle(ncfp - 1) * (Math.PI / 180))
 334        aux_dist(5) = z(ncfp) - (aux_dist(3))
 335        aux_dist(6) = aux_dist(3)
 336        aux_dist(7) = aux_dist(6) / (ncfp + 4)
 337        aux_dist(8) = ((DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI /
180))) * Math.Tan(weld_angle(1))
 338        aux_dist(9) = ((DistL1a + DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI /
180))) * Math.Tan(weld_angle(1))

 339        Print(2, "CSYS,5,0,1702,1,1016" & vbCrLf)
 340        For i = 1 To ncfp - 2
 341            aux_dist(10) = 1
 342            Print(2, "PT," & 1507 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & (DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(7)
* (i - 1) & vbCrLf)
```

```
343           Print(2, "PT," & 1508 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & "0" & "," & aux_dist(7) * (i - 1) & vbCrLf)
344           Print(2, "PT," & 1509 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & -(DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," &
aux_dist(7) * (i - 1) & vbCrLf)
345           Print(2, "PT," & 1510 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & -(DistL1a + DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI / 180)) & "," &
aux_dist(7) * (i - 1) & vbCrLf)
346           Print(2, "PT," & 1511 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & -aux_angle(6) & "," & aux_dist(7) * (i - 1) & vbCrLf)
347           Print(2, "PT," & 1512 + 6 * (i - 1) & "," & aux_dist(2) * aux_dist(10) &
"," & -aux_angle(9) & "," & aux_dist(7) * (i - 1) & vbCrLf)
348       Next

349       For i = ncfp - 1 To ncfp - 1
350           If i = ncfp - 1 Then
351               aux_dist(10) = 0.7
352           End If
353           Print(2, "PT," & 1501 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& (DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(7) *
(i + 1) & vbCrLf)
354           Print(2, "PT," & 1502 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& "0" & "," & aux_dist(7) * (i + 1) & vbCrLf)
355           Print(2, "PT," & 1503 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& -(DistL1a + DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(7) *
(i + 1) & vbCrLf)
356           Print(2, "PT," & 1504 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& -(DistL1a + DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI / 180)) & "," &
aux_dist(7) * (i + 1) & vbCrLf)
357           Print(2, "PT," & 1505 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& -aux_angle(6) & "," & aux_dist(7) * (i + 1) & vbCrLf)
358           Print(2, "PT," & 1506 + 6 * (i) & "," & aux_dist(2) * aux_dist(10) & ","
& -aux_angle(9) & "," & aux_dist(7) * (i + 1) & vbCrLf)
359       Next

360       Print(2, "PT," & 1501 + 6 * (ncfp) & ",0," & (DistL1a + DistL2a + DistL3a) *
Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(4) + aux_dist(7) * ncfp & vbCrLf)
361       Print(2, "PT," & 1502 + 6 * (ncfp) & ",0," & "0" & "," & aux_dist(4) +
aux_dist(7) * ncfp & vbCrLf)
362       Print(2, "PT," & 1503 + 6 * (ncfp) & "," & -aux_dist(8) & "," & -(DistL1a +
DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(4) + aux_dist(7) *
ncfp & vbCrLf)
363       Print(2, "PT," & 1504 + 6 * (ncfp) & "," & -aux_dist(9) & "," & -(DistL1a +
DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI / 180)) & "," & aux_dist(4) +
aux_dist(7) * ncfp & vbCrLf)
364       Print(2, "PT," & 1505 + 6 * (ncfp) & ",0," & -aux_angle(6) & "," &
aux_dist(4) + aux_dist(7) * ncfp & vbCrLf)
365       Print(2, "PT," & 1506 + 6 * (ncfp) & ",0," & -aux_angle(9) & "," &
aux_dist(4) + aux_dist(7) * ncfp & vbCrLf)

366       Print(2, " " & vbCrLf)
367       Print(2, "C* Surfaces (Part 1)" & vbCrLf)
368       For j = 0 To ncfp - 2
369           For i = 0 To 4
370               Print(2, "SF4PT," & 3451 + i + 6 * j & "," & 1005 + i + 22 * j & ","
& 1027 + i + 22 * j & "," & 1513 + i + 6 * j & "," & 1507 + i + 6 * j & ",0" & vbCrLf)
371           Next
372           Print(2, "SF4PT," & 3456 + 6 * j & "," & 1019 + 22 * j & "," & 1041 + 22
* j & "," & 1518 + 6 * j & "," & 1512 + 6 * j & ",0" & vbCrLf)
373       Next

374       Print(2, " " & vbCrLf)
375       Print(2, "C* Volumes (Part 1)" & vbCrLf)
```

```
376         Print(2, "VL2SF," & 941 & "," & (3451) & "," & (3452) & ",1" & vbCrLf)

377         For i = 0 To ncfp - 2
378             Print(2, "VL2SF," & 941 + i & "," & (3451) + 6 * i & "," & (3452) + 6 * i
& ",0" & vbCrLf)
379         Next

380         For i = 0 To ncfp - 2
381             Print(2, "VL2SF," & 961 + i & "," & (3455) + 6 * i & "," & (3456) + 6 * i
& ",0" & vbCrLf)
382         Next

383         For j = 0 To 2
384             For i = 0 To ncfp - 2
385                Print(2, "VL2SF," & 981 + i + (ncfp - 1) * j & "," & (3452) + 6 * i +
j & "," & (3453) + 6 * i + j & ",0" & vbCrLf)
386             Next
387         Next

        'Parte 2
388         Print(2, " " & vbCrLf)
389         Print(2, "C* Points (Part 2)" & vbCrLf)
390         For i = 0 To ncfp - 3
391             Print(2, "PT," & 1701 + 6 * (i) & ",0," & (DistL1a + DistL2a + DistL3a) *
Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(4) + aux_dist(7) * (i) & vbCrLf)
392             Print(2, "PT," & 1702 + 6 * (i) & ",0," & "0," & aux_dist(4) +
aux_dist(7) * (i) & vbCrLf)
393             Print(2, "PT," & 1703 + 6 * (i) & "," & -aux_dist(8) & "," & -(DistL1a +
DistL2a + DistL3a) * Math.Sin(36 * (Math.PI / 180)) & "," & aux_dist(4) + aux_dist(7) *
(i) & vbCrLf)
394             Print(2, "PT," & 1704 + 6 * (i) & "," & -aux_dist(9) & "," & -(DistL1a +
DistL2a + DistL3a) * Math.Sin(2 * 36 * (Math.PI / 180)) & "," & aux_dist(4) +
aux_dist(7) * (i) & vbCrLf)
395             Print(2, "PT," & 1705 + 6 * (i) & ",0," & -aux_angle(6) & "," &
aux_dist(4) + aux_dist(7) * (i) & vbCrLf)
396             Print(2, "PT," & 1706 + 6 * (i) & ",0," & -aux_angle(9) & "," &
aux_dist(4) + aux_dist(7) * (i) & vbCrLf)
397         Next

398         Print(2, " " & vbCrLf)
399         Print(2, "C* Surfaces (Part 2)" & vbCrLf)
400         For j = 0 To ncfp - 4
401             For i = 0 To 5
402                 Print(2, "SF4PT," & 3901 + i + 6 * j & "," & 1701 + i + 6 * j & "," &
1507 + i + 6 * j & "," & 1513 + i + 6 * j & "," & 1707 + i + 6 * j & ",0" & vbCrLf)
403             Next
404         Next

405         For i = 0 To 5
406             Print(2, "SF4PT," & 3901 + 6 * (ncfp - 3) + i & "," & 1701 + i + 6 *
(ncfp - 3) & "," & 1507 + i + 6 * (ncfp - 3) & "," & 1513 + i + 6 * (ncfp - 3) & "," &
1519 + i + 6 * (ncfp - 3) & ",0" & vbCrLf)
407         Next

408         Print(2, " " & vbCrLf)
409         Print(2, "C* Volumes (Part 2)" & vbCrLf)
410         For j = 0 To 4
411             For i = 0 To ncfp - 3
412                 Print(2, "VL2SF," & 1041 + 20 * j + i & "," & 3901 + 6 * i + j & ","
& 3902 + 6 * i + j & ",0" & vbCrLf)
413             Next
414         Next
```

```
        'in front of the crack
 415        Print(2, " " & vbCrLf)
 416        Print(2, "C* Mesh in front of the crack" & vbCrLf)
 417        Print(2, "C* Points" & vbCrLf)

 418        aux_dist(10) = 0
 419        For i = 2 To ncfp - 1
 420            aux_dist(11) = z(i) - (DistL1a + DistL2a + DistL3a + DTrans1) *
Math.Sin(angle(i) * (Math.PI / 180))
 421            aux_dist(12) = aux_dist(10) - aux_dist(11)
 422            aux_dist(13) = x(i) + (DistL1a + DistL2a + DistL3a + DTrans1) *
Math.Cos(angle(i) * (Math.PI / 180))
 423            aux_dist(14) = (vX(1) * aux_dist(12)) / aux_dist(13)
 424            ext_coord(i) = aux_dist(10) - aux_dist(14)
 425            Print(2, "C* coord  " & ext_coord(i) & vbCrLf)
 426        Next

 427        ext_coord(ncfp) = 10 * vX(1)
 428        For i = 2 To ncfp - 1
 429            If ext_coord(i) > vZ(1) Then
 430                node_max = i - 1
 431                i = ncfp
 432            End If
 433        Next

 434        Print(2, "CSYS,5,0,1702,1,1016" & vbCrLf)
 435        Print(2, "PT," & 1901 & "," & vX(1) & "," & aux_angle(4) & ",0" & vbCrLf)
 436        Print(2, "PT," & 1902 & "," & vX(1) & ",0,0" & vbCrLf)
 437        Print(2, "PT," & 1903 & "," & vX(1) & "," & -aux_angle(4) & ",0" & vbCrLf)
 438        Print(2, "PT," & 1904 & "," & vX(1) & "," & -aux_angle(5) & ",0" & vbCrLf)
 439        Print(2, "PT," & 1905 & "," & vX(1) & "," & -aux_angle(6) & ",0" & vbCrLf)
 440        Print(2, "PT," & 1906 & "," & vX(1) & "," & -aux_angle(9) & ",0" & vbCrLf)

 441        For i = 2 To node_max
 442            coord_x(i) = vX(1)
 443            coord_z(i) = (vZ(1) / (node_max - 1)) * (i - 1)
 444            Print(2, "PT," & 1901 + 6 * (i - 1) & "," & coord_x(i) & "," &
aux_angle(4) & "," & coord_z(i) & vbCrLf)
 445            Print(2, "PT," & 1902 + 6 * (i - 1) & "," & coord_x(i) & ",0," &
coord_z(i) & vbCrLf)
 446            Print(2, "PT," & 1903 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(4) & "," & coord_z(i) & vbCrLf)
 447            Print(2, "PT," & 1904 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(5) & "," & coord_z(i) & vbCrLf)
 448            Print(2, "PT," & 1905 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(6) & "," & coord_z(i) & vbCrLf)
 449            Print(2, "PT," & 1906 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(9) & "," & coord_z(i) & vbCrLf)
 450        Next

 451        For i = node_max + 1 To ncfp - 1
 452            coord_x(i) = vX(1) - ((vX(1) / (ncfp - node_max)) * (i - node_max))
 453            coord_z(i) = vZ(1)
 454            Print(2, "PT," & 1901 + 6 * (i - 1) & "," & coord_x(i) & "," &
aux_angle(4) & "," & vZ(1) & vbCrLf)
 455            Print(2, "PT," & 1902 + 6 * (i - 1) & "," & coord_x(i) & ",0," & vZ(1) &
vbCrLf)
 456            Print(2, "PT," & 1903 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(4) & "," & vZ(1) & vbCrLf)
 457            Print(2, "PT," & 1904 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(5) & "," & vZ(1) & vbCrLf)
 458            Print(2, "PT," & 1905 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(6) & "," & vZ(1) & vbCrLf)
```

```
459           Print(2, "PT," & 1906 + 6 * (i - 1) & "," & coord_x(i) & "," & -
aux_angle(9) & "," & vZ(1) & vbCrLf)
460       Next

461       Print(2, "PT," & 1901 + 6 * (ncfp - 1) & ",0," & aux_angle(4) & "," & vZ(1) &
vbCrLf)
462       Print(2, "PT," & 1902 + 6 * (ncfp - 1) & ",0,0," & vZ(1) & vbCrLf)
463       Print(2, "PT," & 1903 + 6 * (ncfp - 1) & "," & -aux_angle(7) & "," & -
aux_angle(4) & "," & vZ(1) & vbCrLf)
464       Print(2, "PT," & 1904 + 6 * (ncfp - 1) & "," & -aux_angle(8) & "," & -
aux_angle(5) & "," & vZ(1) & vbCrLf)
465       Print(2, "PT," & 1905 + 6 * (ncfp - 1) & ",0," & -aux_angle(6) & "," & vZ(1)
& vbCrLf)
466       Print(2, "PT," & 1906 + 6 * (ncfp - 1) & ",0," & -aux_angle(9) & "," & vZ(1)
& vbCrLf)
467       coord_x(ncfp) = vZ(1)
468       coord_z(ncfp) = 0

469       Print(2, " " & vbCrLf)
470       Print(2, "C* Surfaces" & vbCrLf)
471       For i = 0 To ncfp - 2
472           Print(2, "SF4PT," & 4301 + i & "," & 1906 + 6 * i & "," & 1022 + 22 * i &
"," & 1044 + 22 * i & "," & 1912 + 6 * i & ",0" & vbCrLf)
473       Next

474       For i = 0 To ncfp - 2
475           Print(2, "SF4PT," & 4321 + i & "," & 1905 + 6 * i & "," & 1012 + 22 * i &
"," & 1034 + 22 * i & "," & 1911 + 6 * i & ",0" & vbCrLf)
476       Next

477       For i = 0 To ncfp - 2
478           Print(2, "SF4PT," & 4341 + i & "," & 1904 + 6 * i & "," & 1013 + 22 * i &
"," & 1035 + 22 * i & "," & 1910 + 6 * i & ",0" & vbCrLf)
479       Next

480       For i = 0 To ncfp - 2
481           Print(2, "SF4PT," & 4361 + i & "," & 1903 + 6 * i & "," & 1014 + 22 * i &
"," & 1036 + 22 * i & "," & 1909 + 6 * i & ",0" & vbCrLf)
482       Next

483       For i = 0 To ncfp - 2
484           Print(2, "SF4PT," & 4381 + i & "," & 1902 + 6 * i & "," & 1001 + 22 * i &
"," & 1023 + 22 * i & "," & 1908 + 6 * i & ",0" & vbCrLf)
485       Next

486       For i = 0 To ncfp - 2
487           Print(2, "SF4PT," & 4401 + i & "," & 1901 + 6 * i & "," & 1002 + 22 * i &
"," & 1024 + 22 * i & "," & 1907 + 6 * i & ",0" & vbCrLf)
488       Next

489       Print(2, " " & vbCrLf)
490       Print(2, "C* Volumes" & vbCrLf)
491       For i = 0 To ncfp - 2
492           Print(2, "VL2SF," & 1141 + i & "," & 4401 + i & "," & 4381 + i & ",0" &
vbCrLf)
493       Next

494       For i = 0 To ncfp - 2
495           Print(2, "VL2SF," & 1161 + i & "," & 4321 + i & "," & 4301 + i & ",0" &
vbCrLf)
496       Next

497       For j = 0 To 2
```

```
498           For i = 0 To ncfp - 2
499               Print(2, "VL2SF," & 1181 + i + (ncfp - 1) * j & "," & 4381 + i - 20 *
j & "," & 4361 + i - 20 * j & ",0" & vbCrLf)
500           Next
        Next

        'meshing (Part 1) from the upper face of the crack to the end of the specimen
501        Print(2, " " & vbCrLf)
502        Print(2, "C* Meshing (Part 1)" & vbCrLf)
503        Print(2, "M_VL,1,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
504        Print(2, "M_VL,2,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
505        Print(2, "M_VL,3,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
506        Print(2, "M_VL,4,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
507        Print(2, "M_VL,5,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
508        num_elements(1) = 15 * (ncfp - 1)
509        Print(2, "C* Number of elements: " & num_elements(1) & vbCrLf)

510        Print(2, "M_VL,601,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
511        Print(2, "M_VL,602,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
512        Print(2, "M_VL,603,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
513        Print(2, "M_VL,604,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
514        num_elements(2) = num_elements(1) + 4 * (ncfp - 1)
515        Print(2, "C* Number of elements: " & num_elements(2) & vbCrLf)

        'transition mesh
516        Print(2, "M_VL,821,880,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
517        num_elements(3) = num_elements(2) + 3 * (ncfp - 1)
518        Print(2, "C* Number of elements: " & num_elements(3) & vbCrLf)

        'behind the crack
519        aux_mesh(1) = 2 * aux_angle(6)
520        aux_mesh(2) = aux_dist(5)
521        aux_NEL(1) = 0.5 * (aux_mesh(2) / aux_mesh(1))
522        If aux_NEL(1) < NEL2 Then
523            NEL2 = aux_NEL(1)
524        End If

525        If NEL2 < 1 Then
526            NEL2 = 1
527        End If

        'generation 1 (interior)
528        Print(2, "M_VL,941,960,1,20," & NEL1 & "," & NEL2 & "," & NEL1 & ",1,1,1" &
vbCrLf)
529        num_elements(4) = num_elements(3) + NEL2 * (ncfp - 1)
530        Print(2, "C* Number of elements: " & num_elements(4) & vbCrLf)

531        Print(2, "M_VL,1041,1059,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
532        num_elements(5) = num_elements(4) + (ncfp - 2)
533        Print(2, "C* Number of elements: " & num_elements(5) & vbCrLf)

        'after the crack
```

```
534         xmax = 0
535         For i = 1 To ncfp
536             If x(i) > xmax Then
537                 xmax = x(i)
538             End If
539         Next

540         aux_NEL(2) = vX(1) / xmax

541         If NEL3 > aux_NEL(2) Then
542             NEL3 = aux_NEL(2)
543         End If

544         If NEL3 < 1 Then
545             NEL3 = 1
546         End If

        'generation 1 (exterior)
547         Print(2, "M_VL,1141,1160,1,20," & NEL3 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
548         num_elements(6) = num_elements(5) + NEL3 * (ncfp - 1)
549         Print(2, "C* Number of elements: " & num_elements(6) & vbCrLf)

        'scalling (Part 1)
550         Print(2, " " & vbCrLf)
551         Print(2, "C* Scalling (Part 1)" & vbCrLf)

        'primeira camada
552         aux_dist(15) = (aux_angle(5) - aux_angle(4)) / aux_angle(4)
553         Print(2, "ELSCALE," & num_elements(3) + 1 & "," & num_elements(4) & ",1,0,1,"
& aux_dist(15) & ",1,0," & aux_angle(4) & ",0" & vbCrLf)
554         Print(2, "ELSCALE," & num_elements(4) + 1 & "," & num_elements(5) & ",1,0,1,"
& aux_dist(15) & ",1,0," & aux_angle(4) & ",0" & vbCrLf)
555         Print(2, "ELSCALE," & num_elements(5) + 1 & "," & num_elements(6) & ",1,0,1,"
& aux_dist(15) & ",1,0," & aux_angle(4) & ",0" & vbCrLf)
556         num_elements(7) = num_elements(6) + (NEL2 * (ncfp - 1)) + ((ncfp - 2)) +
(NEL3 * (ncfp - 1))
557         Print(2, "C* Number of elements: " & num_elements(7) & vbCrLf)

        'segunda camada
558         aux_dist(16) = (aux_angle(6) - aux_angle(5)) / aux_angle(4)
559         Print(2, "ELSCALE," & num_elements(3) + 1 & "," & num_elements(4) & ",1,0,1,"
& aux_dist(16) & ",1,0," & aux_angle(5) & ",0" & vbCrLf)
560         Print(2, "ELSCALE," & num_elements(4) + 1 & "," & num_elements(5) & ",1,0,1,"
& aux_dist(16) & ",1,0," & aux_angle(5) & ",0" & vbCrLf)
561         Print(2, "ELSCALE," & num_elements(5) + 1 & "," & num_elements(6) & ",1,0,1,"
& aux_dist(16) & ",1,0," & aux_angle(5) & ",0" & vbCrLf)
562         num_elements(8) = num_elements(7) + (NEL2 * (ncfp - 1)) + ((ncfp - 2)) +
(NEL3 * (ncfp - 1))
563         Print(2, "C* Number of elements: " & num_elements(8) & vbCrLf)

        'until the support
564         Print(2, "CSYS,6,0,704,703,1017" & vbCrLf)
565         aux_dist(17) = vY(3) - vY(5) - aux_angle(6) - vY(6)
566         aux_dist(18) = aux_dist(17) / NEL4
567         aux_dist(19) = aux_dist(18) / (aux_angle(6) - aux_angle(5))

568         For i = 0 To NEL4 - 1
569             Print(2, "ELSCALE," & num_elements(2) + 1 & "," & num_elements(3) &
",1,0,1," & aux_dist(19) & ",1,0," & (aux_angle(6) - aux_angle(5)) + aux_dist(18) * i &
",0" & vbCrLf)
```

```
570           Print(2, "ELSCALE," & num_elements(7) + 1 & "," & num_elements(8) &
",1,0,1," & aux_dist(19) & ",1,0," & (aux_angle(6) - aux_angle(5)) + aux_dist(18) * i &
",0" & vbCrLf)
571       Next
572       num_elements(9) = num_elements(8) + ((3 * (ncfp - 1)) + (NEL2 * (ncfp - 1)) +
((ncfp - 2)) + (NEL3 * (ncfp - 1))) * NEL4
573       Print(2, "C* Number of elements: " & num_elements(9) & vbCrLf)

        'from the support to the end
574       aux_dist(20) = vY(6) / NEL5
575       aux_dist(21) = aux_dist(20) / (aux_angle(6) - aux_angle(5))
576       aux_dist(22) = (aux_angle(6) - aux_angle(5)) + aux_dist(18) * NEL4

577       For i = 0 To NEL5 - 1
578           Print(2, "ELSCALE," & num_elements(2) + 1 & "," & num_elements(3) &
",1,0,1," & aux_dist(21) & ",1,0," & aux_dist(22) + aux_dist(20) * i & ",0" & vbCrLf)
579           Print(2, "ELSCALE," & num_elements(7) + 1 & "," & num_elements(8) &
",1,0,1," & aux_dist(21) & ",1,0," & aux_dist(22) + aux_dist(20) * i & ",0" & vbCrLf)
580       Next
581       num_elements(10) = num_elements(9) + ((3 * (ncfp - 1)) + (NEL2 * (ncfp - 1))
+ ((ncfp - 2)) + (NEL3 * (ncfp - 1))) * NEL5
582       Print(2, "C* Number of elements: " & num_elements(10) & vbCrLf)

583       Print(2, " " & vbCrLf)
584       Print(2, "C* Merge and compress (1)" & vbCrLf)
585       Print(2, "PARASSIGN,num_nodes0,INT,ndmax" & vbCrLf)
586       Print(2, "NMERGE,1,ndmax,1,0.001,0,1,0" & vbCrLf)
587       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
588       Print(2, "PARASSIGN,num_nodes1,INT,ndmax" & vbCrLf)
589       Print(2, "NMODIFY,1,num_nodes1,1,1,0,20,0,0" & vbCrLf)

        'meshing (Part 2) from the lower face of the crack until the end of the specimen
590       Print(2, " " & vbCrLf)
591       Print(2, "C* Meshing (Part 1)" & vbCrLf)
592       Print(2, "M_VL,6,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
593       Print(2, "M_VL,7,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
594       Print(2, "M_VL,8,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
595       Print(2, "M_VL,9,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
596       Print(2, "M_VL,10,600,10,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
597       num_elements(11) = num_elements(10) + 15 * (ncfp - 1)
598       Print(2, "C* Number of elements: " & num_elements(11) & vbCrLf)

599       Print(2, "M_VL,605,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
600       Print(2, "M_VL,606,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
601       Print(2, "M_VL,607,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
602       Print(2, "M_VL,608,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
603       Print(2, "M_VL,609,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
604       Print(2, "M_VL,610,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
605       Print(2, "M_VL,611,820,11,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
606       num_elements(12) = num_elements(11) + 7 * (ncfp - 1)
607       Print(2, "C* Number of elements: " & num_elements(12) & vbCrLf)
```

```
608       Print(2, "M_VL,881,940,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
609       num_elements(13) = num_elements(12) + 3 * (ncfp - 1)
610       Print(2, "C* Number of elements: " & num_elements(13) & vbCrLf)

        'generation 2 (interior)
611       Print(2, "M_VL,961,980,1,20," & NEL1 & "," & NEL2 & "," & NEL1 & ",1,1,1" &
vbCrLf)
612       num_elements(14) = num_elements(13) + NEL2 * (ncfp - 1)
613       Print(2, "C* Number of elements: " & num_elements(14) & vbCrLf)

614       Print(2, "M_VL,1121,1139,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
615       num_elements(15) = num_elements(14) + (ncfp - 2)
616       Print(2, "C* Number of elements: " & num_elements(15) & vbCrLf)

        'interior (other volumes)
617       Print(2, "M_VL,981,1040,1,20," & NEL1 & "," & NEL2 & "," & NEL1 & ",1,1,1" &
vbCrLf)
618       num_elements(16) = num_elements(15) + 3 * NEL2 * (ncfp - 1)
619       Print(2, "C* Number of elements: " & num_elements(16) & vbCrLf)

620       Print(2, "M_VL,1061,1120,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
621       num_elements(17) = num_elements(16) + 3 * (ncfp - 2)
622       Print(2, "C* Number of elements: " & num_elements(17) & vbCrLf)

        'generation 2 (exterior)
623       Print(2, "M_VL,1161,1180,1,20," & NEL3 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
624       num_elements(18) = num_elements(17) + NEL3 * (ncfp - 1)
625       Print(2, "C* Number of elements: " & num_elements(18) & vbCrLf)

        'other volumes (exterior)
626       Print(2, "M_VL,1181,1240,1,20," & NEL3 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
627       num_elements(19) = num_elements(18) + 3 * NEL3 * (ncfp - 1)
628       Print(2, "C* Number of elements: " & num_elements(19) & vbCrLf)

629       Print(2, " " & vbCrLf)
630       Print(2, "C* Merge and compress (2)" & vbCrLf)
631       Print(2, "PARASSIGN,num_nodes2,INT,ndmax" & vbCrLf)
632       Print(2, "NMERGE,1,ndmax,1,0.001,0,1,0" & vbCrLf)
633       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
634       Print(2, "PARASSIGN,num_nodes3,INT,ndmax" & vbCrLf)
635       Print(2, "NMODIFY,num_nodes1+1,ndmax,1,1," & vX(1) + 20 & ",0,0,0" & vbCrLf)

636       Print(2, " " & vbCrLf)
637       Print(2, "C* Geometry of the welding joints" & vbCrLf)
638       Print(2, "ACTSET,CS,5" & vbCrLf)
639       xw(nwp + 1) = vY(5)
640       yw(nwp + 1) = vX(4)
641       xw(nwp + 2) = vY(5)
642       yw(nwp + 2) = vX(2)

        'análise do ligamento do cordão
643       lig(3) = lig(1)
644       If lig(1) < 0.05 Then
645           lig(3) = 0.05
646       End If

647       If lig(1) > 0.45 Then
```

```
 648           lig(3) = 0.45
 649       End If

 650       If lig(1) = 0.5 Then
 651           lig(3) = 0.25
 652       End If


 653       lig(4) = lig(2)
 654       If lig(2) < 0.05 Then
 655           lig(4) = 0.05
 656       End If

 657       If lig(2) > 0.45 Then
 658           lig(4) = 0.45
 659       End If

 660       If lig(2) = 0.5 Then
 661           lig(4) = 0.25
 662       End If


 663       Print(2, " " & vbCrLf)
 664       Print(2, "C* Points" & vbCrLf)
 665       For i = 2 To nwp + 2
 666           Print(2, "PT," & 2041 + 2 * (i - 2) & "," & -yw(i) & "," & -xw(i) & ",0"
& vbCrLf)
 667           Print(2, "PT," & 2042 + 2 * (i - 2) & "," & -yw(i) & "," & -xw(i) & "," &
aux_dist(7) & vbCrLf) 'novo 4
 668       Next

 669       For i = 2 To nwp + 2
 670           Print(2, "PT," & 2541 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) -
lig(3) * vY(2) & ",0" & vbCrLf)
 671           Print(2, "PT," & 2542 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) -
lig(3) * vY(2) & "," & aux_dist(7) & vbCrLf) 'novo 4
 672       Next
 673       Print(2, "PT," & 2539 & ",0," & -vY(5) - lig(3) * vY(2) & ",0" & vbCrLf)
 674       Print(2, "PT," & 2540 & ",0," & -vY(5) - lig(3) * vY(2) & "," & aux_dist(7) &
vbCrLf) 'novo 4

 675       For i = 2 To nwp + 2
 676           Print(2, "PT," & 3041 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) - 0.5 *
vY(2) & ",0" & vbCrLf)
 677           Print(2, "PT," & 3042 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) - 0.5 *
vY(2) & "," & aux_dist(7) & vbCrLf) 'novo 4
 678       Next
 679       Print(2, "PT," & 3039 & ",0," & -vY(5) - 0.5 * vY(2) & ",0" & vbCrLf)
 680       Print(2, "PT," & 3040 & ",0," & -vY(5) - 0.5 * vY(2) & "," & aux_dist(7) &
vbCrLf) 'novo 4

 681       For i = 2 To nwp + 2
 682           Print(2, "PT," & 3541 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) - (1 -
lig(4)) * vY(2) & ",0" & vbCrLf)
 683           Print(2, "PT," & 3542 + 2 * (i - 2) & "," & -yw(i) & "," & -vY(5) - (1 -
lig(4)) * vY(2) & "," & aux_dist(7) & vbCrLf) 'novo 4
 684       Next
 685       Print(2, "PT," & 3539 & ",0," & -vY(5) - (1 - lig(4)) * vY(2) & ",0" &
vbCrLf)
 686       Print(2, "PT," & 3540 & ",0," & -vY(5) - (1 - lig(4)) * vY(2) & "," &
aux_dist(7) & vbCrLf) 'novo 4

       'additional points of the uncracked welded joint
```

```
 687       Print(2, " " & vbCrLf)
 688       Print(2, "C* Points of the uncracked joint" & vbCrLf)
 689       xwnf(nwp) = vY(4)
 690       ywnf(nwp) = vX(3)
 691       xwnf(nwp + 1) = vY(4)
 692       ywnf(nwp + 1) = vX(2)
 693       aux_dist(23) = vY(4) + vY(2) + vY(5)
 694       Print(2, "PT," & 4039 & ",0," & -aux_dist(23) & ",0" & vbCrLf)
 695       Print(2, "PT," & 4040 & ",0," & -aux_dist(23) & "," & aux_dist(7) & vbCrLf)
 696       For i = 1 To nwp + 1
 697           Print(2, "PT," & 4041 + 2 * (i - 1) & "," & -ywnf(i) & "," & -
aux_dist(23) + xwnf(i) & ",0" & vbCrLf)
 698           Print(2, "PT," & 4042 + 2 * (i - 1) & "," & -ywnf(i) & "," & -
aux_dist(23) + xwnf(i) & "," & aux_dist(7) & vbCrLf) 'novo 4
 699       Next

       'surfaces
 700       Print(2, " " & vbCrLf)
 701       Print(2, "C* Surfaces" & vbCrLf)
 702       Print(2, "SF4PT,4801,1704,1705,1706,2041,0" & vbCrLf)
 703       Print(2, "SF4PT,4802,1710,1711,1712,2042,0" & vbCrLf)
 704       Print(2, "SF4PT,4803,2041,1706,2539,2541,0" & vbCrLf)
 705       Print(2, "SF4PT,4804,2042,1712,2540,2542,0" & vbCrLf)

 706       For i = 0 To nwp - 1
 707           Print(2, "SF4PT," & 4805 + 2 * i & "," & 2043 + 2 * i & "," & 2041 + 2 *
i & "," & 2541 + 2 * i & "," & 2543 + 2 * i & ",0" & vbCrLf)
 708           Print(2, "SF4PT," & 4806 + 2 * i & "," & 2044 + 2 * i & "," & 2042 + 2 *
i & "," & 2542 + 2 * i & "," & 2544 + 2 * i & ",0" & vbCrLf)
 709       Next

 710       For i = 0 To nwp
 711           Print(2, "SF4PT," & 5203 + 2 * i & "," & 2541 + 2 * i & "," & 2539 + 2 *
i & "," & 3039 + 2 * i & "," & 3041 + 2 * i & ",0" & vbCrLf)
 712           Print(2, "SF4PT," & 5204 + 2 * i & "," & 2542 + 2 * i & "," & 2540 + 2 *
i & "," & 3040 + 2 * i & "," & 3042 + 2 * i & ",0" & vbCrLf)
 713       Next

 714       For i = 0 To nwp
 715           Print(2, "SF4PT," & 5703 + 2 * i & "," & 3041 + 2 * i & "," & 3039 + 2 *
i & "," & 3539 + 2 * i & "," & 3541 + 2 * i & ",0" & vbCrLf)
 716           Print(2, "SF4PT," & 5704 + 2 * i & "," & 3042 + 2 * i & "," & 3040 + 2 *
i & "," & 3540 + 2 * i & "," & 3542 + 2 * i & ",0" & vbCrLf)
 717       Next

 718       For i = 0 To nwp
 719           Print(2, "SF4PT," & 6203 + 2 * i & "," & 3541 + 2 * i & "," & 3539 + 2 *
i & "," & 4039 + 2 * i & "," & 4041 + 2 * i & ",0" & vbCrLf)
 720           Print(2, "SF4PT," & 6204 + 2 * i & "," & 3542 + 2 * i & "," & 3540 + 2 *
i & "," & 4040 + 2 * i & "," & 4042 + 2 * i & ",0" & vbCrLf)
 721       Next

       'volumes
 722       Print(2, " " & vbCrLf)
 723       Print(2, "C* Volumes" & vbCrLf)
 724       For i = 0 To nwp + 1
 725           Print(2, "VL2SF," & 1241 + i & "," & 4801 + 2 * i & "," & 4802 + 2 * i &
",0" & vbCrLf) 'novo 3
 726       Next

 727       For i = 0 To nwp
 728           Print(2, "VL2SF," & 1341 + i & "," & 5203 + 2 * i & "," & 5204 + 2 * i &
",0" & vbCrLf)
```

```
729         Next

730         For i = 0 To nwp
731             Print(2, "VL2SF," & 1441 + i & "," & 5703 + 2 * i & "," & 5704 + 2 * i &
",0" & vbCrLf)
732         Next

733         For i = 0 To nwp
734             Print(2, "VL2SF," & 1541 + i & "," & 6203 + 2 * i & "," & 6204 + 2 * i &
",0" & vbCrLf)
735         Next

        'Meshing (Part 2)
736         aux_NEL(3) = 0
737         aux_NEL(4) = 0
738         If lig(1) < 0.05 Then
739             aux_NEL(3) = 0.05 / lig(1)
740             aux_NEL(4) = 0.8 * NEL6
741         ElseIf lig(1) > 0.45 Then
742             aux_NEL(4) = 0.05 / (lig(1) - 0.45)
743             aux_NEL(3) = 0.8 * NEL6
744         ElseIf lig(1) >= 0.05 And lig(1) <= 0.45 Then
745             aux_NEL(3) = (lig(3) / 0.5) * NEL6
746             aux_NEL(4) = NEL6 - aux_NEL(3)
747         End If

748         If aux_NEL(3) < 1 Then
749             aux_NEL(3) = 1
750         End If

751         If aux_NEL(3) > 10 Then
752             aux_NEL(3) = 10
753         End If

754         If aux_NEL(4) < 1 Then
755             aux_NEL(4) = 1
756         End If

757         If aux_NEL(4) > 10 Then
758             aux_NEL(4) = 10
759         End If

760         If lig(1) = 0.5 Then
761             aux_NEL(3) = 0.5 * NEL6
762             aux_NEL(4) = 0.5 * NEL6
763         End If

764         aux_NEL(5) = 0
765         aux_NEL(6) = 0
766         If lig(2) < 0.05 Then
767             aux_NEL(6) = 0.05 / lig(2)
768             aux_NEL(5) = 0.8 * NEL6
769         ElseIf lig(2) > 0.45 Then
770             aux_NEL(5) = 0.05 / (lig(2) - 0.45)
771             aux_NEL(6) = 0.8 * NEL6
772         ElseIf lig(2) >= 0.05 And lig(2) <= 0.45 Then
773             aux_NEL(6) = (lig(4) / 0.5) * NEL6
774             aux_NEL(5) = NEL6 - aux_NEL(6)
775         End If

776         If aux_NEL(5) < 1 Then
777             aux_NEL(5) = 1
778         End If
```

```
779         If aux_NEL(5) > 10 Then
780             aux_NEL(5) = 10
781         End If

782         If aux_NEL(6) < 1 Then
783             aux_NEL(6) = 1
784         End If

785         If aux_NEL(6) > 10 Then
786             aux_NEL(6) = 10
787         End If

788         If lig(2) = 0.5 Then
789             aux_NEL(5) = 0.5 * NEL7
790             aux_NEL(6) = 0.5 * NEL7
791         End If

792         Print(2, " " & vbCrLf)
793         Print(2, "C* Meshing (Part 2)" & vbCrLf)
794         Print(2, "M_VL,1241,1241,1,20," & NEL1 & "," & NEL1 & "," & NEL1 & ",1,1,1" &
vbCrLf)
795         num_elements(20) = num_elements(19) + 1
796         Print(2, "C* Number of elements: " & num_elements(20) & vbCrLf)

797         Print(2, "M_VL,1242," & 1241 + nwp & ",1,20," & NEL1 & "," & aux_NEL(3) & ","
& NEL1 & ",1,1,1" & vbCrLf)
798         num_elements(21) = num_elements(20) + nwp * aux_NEL(3)
799         Print(2, "C* Number of elements: " & num_elements(21) & vbCrLf)

800         Print(2, "M_VL," & 1241 + nwp + 1 & "," & 1241 + nwp + 1 & ",1,20," & NEL7 &
"," & aux_NEL(3) & "," & NEL1 & ",1,1,1" & vbCrLf)
801         num_elements(22) = num_elements(21) + aux_NEL(3) * NEL7
802         Print(2, "C* Number of elements: " & num_elements(22) & vbCrLf)

803         Print(2, "M_VL,1341," & 1341 + nwp - 1 & ",1,20," & NEL1 & "," & aux_NEL(4) &
"," & NEL1 & ",1,1,1" & vbCrLf)
804         num_elements(23) = num_elements(22) + nwp * aux_NEL(4)
805         Print(2, "C* Number of elements: " & num_elements(23) & vbCrLf)

806         Print(2, "M_VL," & 1341 + nwp & "," & 1341 + nwp & ",1,20," & NEL7 & "," &
aux_NEL(4) & "," & NEL1 & ",1,1,1" & vbCrLf)
807         num_elements(24) = num_elements(23) + aux_NEL(4) * NEL7
808         Print(2, "C* Number of elements: " & num_elements(24) & vbCrLf)

809         Print(2, "M_VL,1441," & 1441 + nwp - 1 & ",1,20," & NEL1 & "," & aux_NEL(5) &
"," & NEL1 & ",1,1,1" & vbCrLf)
810         num_elements(25) = num_elements(24) + nwp * aux_NEL(5)
811         Print(2, "C* Number of elements: " & num_elements(25) & vbCrLf)

812         Print(2, "M_VL," & 1441 + nwp & "," & 1441 + nwp & ",1,20," & NEL7 & "," &
aux_NEL(5) & "," & NEL1 & ",1,1,1" & vbCrLf)
813         num_elements(26) = num_elements(25) + aux_NEL(5) * NEL7
814         Print(2, "C* Number of elements: " & num_elements(26) & vbCrLf)

815         Print(2, "M_VL,1541," & 1541 + nwp - 1 & ",1,20," & NEL1 & "," & aux_NEL(6) &
"," & NEL1 & ",1,1,1" & vbCrLf)
816         num_elements(27) = num_elements(26) + nwp * aux_NEL(6)
817         Print(2, "C* Number of elements: " & num_elements(27) & vbCrLf)

818         Print(2, "M_VL," & 1541 + nwp & "," & 1541 + nwp & ",1,20," & NEL7 & "," &
aux_NEL(6) & "," & NEL1 & ",1,1,1" & vbCrLf)
819         num_elements(28) = num_elements(27) + aux_NEL(6) * NEL7
```

```
820       Print(2, "C* Number of elements: " & num_elements(28) & vbCrLf)

821       Print(2, " " & vbCrLf)
822       Print(2, "C* Merge and compress (3)" & vbCrLf)
823       Print(2, "PARASSIGN,num_nodes4,INT,ndmax" & vbCrLf)
824       Print(2, "NMERGE,1,ndmax,1,0.0001,0,1,0" & vbCrLf)
825       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
826       Print(2, "PARASSIGN,num_nodes5,INT,ndmax" & vbCrLf)
827       Print(2, "NMODIFY,num_nodes3+1,ndmax,1,1,-20,0,0,0" & vbCrLf)

      'scalling (Part 2)
828       Print(2, " " & vbCrLf)
829       Print(2, "C* Scalling (Part 2)" & vbCrLf)
830       Print(2, "CSYS,7,0,1712,1718,2042" & vbCrLf)

831       aux_dist(24) = aux_dist(7)
832       For i = 1 To ncfp - 4
833           Print(2, "ELSCALE," & num_elements(19) + 1 & "," & num_elements(28) &
",1,0,1,1,1," & aux_dist(24) * i & ",0,0" & vbCrLf)
834       Next
835       num_elements(29) = num_elements(28) + (num_elements(28) - num_elements(19)) *
(ncfp - 4)
836       Print(2, "C* Number of elements: " & num_elements(29) & vbCrLf)

      'elemento diferente do linha de elementos na parte interior junto ao cordão
837       Print(2, "CSYS,8,0,1702,1708,1701" & vbCrLf)

838       aux_dist(25) = 3 * aux_dist(7)
839       aux_dist(26) = aux_dist(25) / aux_dist(24)
840       Print(2, "ELSCALE," & num_elements(19) + 1 & "," & num_elements(28) & ",1,0,"
& aux_dist(26) & ",1,1," & aux_dist(7) * (ncfp - 3) & ",0,0" & vbCrLf)
841       num_elements(30) = num_elements(29) + (num_elements(28) - num_elements(19))
842       Print(2, "C* Number of elements: " & num_elements(30) & vbCrLf)

      'parte entre a malha de transição e a linha de elementos junto ao cordão
843       aux_dist(26) = z(ncfp) - aux_angle(6) - (aux_dist(7) * ncfp)
844       aux_dist(27) = (aux_dist(26) / aux_dist(7)) / NEL2
845       aux_dist(28) = aux_dist(7) * ncfp
846       For i = 0 To NEL2 - 1
847           Print(2, "ELSCALE," & num_elements(19) + 1 & "," & num_elements(28) &
",1,0," & aux_dist(27) & ",1,1," & aux_dist(28) + aux_dist(26) * i & ",0,0" & vbCrLf)
848       Next
849       num_elements(31) = num_elements(30) + (num_elements(28) - num_elements(19))
850       Print(2, "C* Number of elements: " & num_elements(31) & vbCrLf)

851       aux_dist(29) = aux_angle(6) - aux_angle(3)
852       aux_dist(30) = 2 * aux_angle(3)
853       aux_dist(31) = aux_angle(6) - aux_angle(3)
854       aux_dist(32) = z(ncfp) - aux_angle(6)
855       aux_dist(33) = z(ncfp) - aux_angle(3)
856       aux_dist(34) = z(ncfp) + aux_angle(3)
857       For i = 0 To 2
858           Print(2, "ELSCALE," & num_elements(19) + 1 & "," & num_elements(28) &
",1,0," & aux_dist(29 + i) / aux_dist(7) & ",1,1," & aux_dist(32 + i) & ",0,0" & vbCrLf)
859       Next
860       num_elements(32) = num_elements(31) + (num_elements(28) - num_elements(19)) *
3
861       Print(2, "C* Number of elements: " & num_elements(32) & vbCrLf)

862       aux_dist(35) = z(ncfp) + aux_angle(6)
863       aux_dist(36) = (vZ(1) - aux_dist(35)) / NEL3
864       For i = 0 To NEL3 - 1
```

```
865           Print(2, "ELSCALE," & num_elements(19) + 1 & "," & num_elements(28) &
",1,0," & aux_dist(36) / aux_dist(7) & ",1,1," & aux_dist(35) + aux_dist(36) * i &
",0,0" & vbCrLf)
866       Next
867       num_elements(33) = num_elements(32) + (num_elements(28) - num_elements(19)) *
NEL3
868       Print(2, "C* Number of elements: " & num_elements(33) & vbCrLf)

869       Print(2, " " & vbCrLf)
870       Print(2, "C* Merge and compress (4)" & vbCrLf)
871       Print(2, "PARASSIGN,num_nodes6,INT,ndmax" & vbCrLf)
872       Print(2, "NMERGE,1,ndmax,1,0.0001,0,1,0" & vbCrLf)
873       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
874       Print(2, "PARASSIGN,num_nodes7,INT,ndmax" & vbCrLf)
875       Print(2, "NMODIFY,num_nodes1+1,num_nodes3,1,1," & -vX(1) - 20 & ",0,0,0" &
vbCrLf)

      'scalling (Part 3)
876       Print(2, " " & vbCrLf)
877       Print(2, "C* Scalling (Part 3)" & vbCrLf)
878       Print(2, "CSYS,7,0,1011,1012,1021" & vbCrLf)
879       aux_dist(37) = aux_angle(9) - aux_angle(6)
880       aux_dist(38) = (vY(5) + lig(3) * vY(2)) - aux_angle(9)
881       aux_dist(39) = aux_dist(38) / aux_NEL(3)
882       aux_dist(40) = aux_dist(39) / aux_dist(37)
883       For i = 0 To aux_NEL(3) - 1
884           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(40) & ",1,0," & aux_dist(37) + aux_dist(39) * i & ",0" & vbCrLf)
'novo
885           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(40) & ",1,0," & aux_dist(37) + aux_dist(39) * i & ",0" & vbCrLf)
'novo
886           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(40) & ",1,0," & aux_dist(37) + aux_dist(39) * i & ",0" & vbCrLf)
'novo
887       Next
888       num_elements(34) = (num_elements(13) - num_elements(12)) + (num_elements(15)
- num_elements(13)) + (num_elements(18) - num_elements(17))
889       num_elements(35) = num_elements(33) + num_elements(34) * aux_NEL(3)
890       Print(2, "C* Number of elements: " & num_elements(35) & vbCrLf)

891       aux_dist(41) = (vY(5) + 0.5 * vY(2)) - (vY(5) + lig(3) * vY(2))
892       aux_dist(42) = aux_dist(41) / aux_NEL(4)
893       aux_dist(43) = aux_dist(42) / aux_dist(37)
894       aux_dist(44) = aux_dist(37) + aux_dist(38)
895       For i = 0 To aux_NEL(4) - 1
896           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(43) & ",1,0," & aux_dist(44) + aux_dist(42) * i & ",0" & vbCrLf)
897           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(43) & ",1,0," & aux_dist(44) + aux_dist(42) * i & ",0" & vbCrLf)
898           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(43) & ",1,0," & aux_dist(44) + aux_dist(42) * i & ",0" & vbCrLf)
899       Next
900       num_elements(36) = num_elements(35) + num_elements(34) * aux_NEL(4)
901       Print(2, "C* Number of elements: " & num_elements(36) & vbCrLf)

902       aux_dist(55) = (vY(5) + (1 - lig(4)) * vY(2)) - (vY(5) + 0.5 * vY(2))
903       aux_dist(56) = aux_dist(55) / aux_NEL(5)
904       aux_dist(57) = aux_dist(56) / aux_dist(37)
905       aux_dist(58) = aux_dist(44) + aux_dist(41)
906       For i = 0 To aux_NEL(5) - 1
907           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(57) & ",1,0," & aux_dist(58) + aux_dist(56) * i & ",0" & vbCrLf)
```

```
908           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(57) & ",1,0," & aux_dist(58) + aux_dist(56) * i & ",0" & vbCrLf)
909           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(57) & ",1,0," & aux_dist(58) + aux_dist(56) * i & ",0" & vbCrLf)
910       Next
911       num_elements(37) = num_elements(36) + num_elements(34) * aux_NEL(5)
912       Print(2, "C* Number of elements: " & num_elements(37) & vbCrLf)

913       aux_dist(59) = aux_dist(23) - (vY(5) + (1 - lig(4)) * vY(2))
914       aux_dist(60) = aux_dist(59) / aux_NEL(6)
915       aux_dist(61) = aux_dist(60) / aux_dist(37)
916       aux_dist(62) = aux_dist(58) + aux_dist(56) * aux_NEL(5)
917       For i = 0 To aux_NEL(6) - 1
918           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(61) & ",1,0," & aux_dist(62) + aux_dist(60) * i & ",0" & vbCrLf)
919           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(61) & ",1,0," & aux_dist(62) + aux_dist(60) * i & ",0" & vbCrLf)
920           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(61) & ",1,0," & aux_dist(62) + aux_dist(60) * i & ",0" & vbCrLf)
921       Next
922       num_elements(38) = num_elements(37) + num_elements(34) * aux_NEL(6)
923       Print(2, "C* Number of elements: " & num_elements(38) & vbCrLf)

924       aux_dist(63) = vY(1) - vY(4) - vY(7)
925       aux_dist(64) = aux_dist(63) / NEL4
926       aux_dist(65) = aux_dist(64) / aux_dist(37)
927       aux_dist(66) = aux_dist(62) + aux_dist(60) * aux_NEL(6)
928       For i = 0 To NEL4 - 1
929           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(65) & ",1,0," & aux_dist(66) + aux_dist(64) * i & ",0" & vbCrLf)
930           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(65) & ",1,0," & aux_dist(66) + aux_dist(64) * i & ",0" & vbCrLf)
931           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(65) & ",1,0," & aux_dist(66) + aux_dist(64) * i & ",0" & vbCrLf)
932       Next
933       num_elements(39) = num_elements(38) + num_elements(34) * NEL4
934       Print(2, "C* Number of elements: " & num_elements(39) & vbCrLf)

935       aux_dist(67) = vY(7)
936       aux_dist(68) = aux_dist(67) / NEL5
937       aux_dist(69) = aux_dist(68) / aux_dist(37)
938       aux_dist(70) = aux_dist(66) + aux_dist(64) * NEL4
939       For i = 0 To NEL5 - 1
940           Print(2, "ELSCALE," & num_elements(12) + 1 & "," & num_elements(13) &
",1,0,1," & aux_dist(69) & ",1,0," & aux_dist(70) + aux_dist(68) * i & ",0" & vbCrLf)
941           Print(2, "ELSCALE," & num_elements(13) + 1 & "," & num_elements(15) &
",1,0,1," & aux_dist(69) & ",1,0," & aux_dist(70) + aux_dist(68) * i & ",0" & vbCrLf)
942           Print(2, "ELSCALE," & num_elements(17) + 1 & "," & num_elements(18) &
",1,0,1," & aux_dist(69) & ",1,0," & aux_dist(70) + aux_dist(68) * i & ",0" & vbCrLf)
943       Next
944       num_elements(40) = num_elements(39) + num_elements(34) * NEL5
945       Print(2, "C* Number of elements: " & num_elements(40) & vbCrLf)

946       Print(2, " " & vbCrLf)
947       Print(2, "C* Merge and compress (5)" & vbCrLf)
948       Print(2, "PARASSIGN,num_nodes8,INT,ndmax" & vbCrLf)
949       Print(2, "NMERGE,1,ndmax,1,0.001,0,1,0" & vbCrLf)
950       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
951       Print(2, "PARASSIGN,num_nodes9,INT,ndmax" & vbCrLf)
952       Print(2, "NMODIFY,1,num_nodes1,1,1,0,-20,0,0" & vbCrLf)
953       Print(2, "NMODIFY,num_nodes3+1,num_nodes7,1,1,20,0,0,0" & vbCrLf)

954      Print(2, "C* " & vbCrLf)
```

```
 955       Print(2, "C* Merge and compress nodes of the uncracked region on the crack
plane " & vbCrLf)
 956       Print(2, "INITSEL,ND,1,1" & vbCrLf)
 957       Print(2, "SELREF,ND,VL,1,200,10" & vbCrLf)
 958       Print(2, "SELREF,ND,VL,10,200,10" & vbCrLf)
 959       Print(2, "SELREF,ND,VL,201,400,10" & vbCrLf)
 960       Print(2, "SELREF,ND,VL,210,400,10" & vbCrLf)
 961       Print(2, "SELREF,ND,VL,401,600,10" & vbCrLf)
 962       Print(2, "SELREF,ND,VL,410,600,10" & vbCrLf)
 963       Print(2, "SELREF,ND,VL,601,940,11" & vbCrLf)
 964       Print(2, "SELREF,ND,VL,611,940,11" & vbCrLf)
 965       Print(2, "SELREF,ND,VL,1141,1160,1" & vbCrLf)
 966       Print(2, "SELREF,ND,VL,1181,1200,1" & vbCrLf)
 967       Print(2, "NMERGE,1,ndmax,1,0.001,0,1,0" & vbCrLf)
 968       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)

 969       Print(2, "C* " & vbCrLf)
 970       Print(2, "C* Merge and compress nodes of the bonded region of the welding " &
vbCrLf)
 971       Print(2, "ACTSET,CS,0" & vbCrLf)
 972       Print(2, "INITSEL,ND,1,1" & vbCrLf)
 973       Print(2, "SELRANGE,ND,0,1,1,1,-0.0001,0.0001," & vY(1) + vY(2) - lig(1) *
vY(2) - 0.001 & "," & vY(1) + vY(2) + vY(5) - 0.005 & ",-0.001" & "," & vZ(1) + 0.001 &
",1" & vbCrLf)
 974       Print(2, "SELRANGE,ND,0,1,1,1,-0.0001, 0.0001," & vY(1) - vY(4) - 0.001 & ","
& vY(1) + lig(2) * vY(2) + 0.001 & ",-0.001" & "," & vZ(1) + 0.001 & ",1" & vbCrLf)
 975       Print(2, "NMERGE,1,ndmax,1,0.001,0,1,0" & vbCrLf)
 976       Print(2, "NCOMPRESS,1,ndmax" & vbCrLf)
      'mid-side radial nodes moved to quarter-point positions
 977       Print(2, " " & vbCrLf)
 978       Print(2, "C* Mid-side radial nodes moved to quarter-point positions " &
vbCrLf)
 979       aux_dist(72) = (DistL1a / Math.Cos(weld_angle(1)))
 980       Print(2, "C* aux_dist(72)  " & aux_dist(72) & vbCrLf)
 981       For i = 1 To ncfp - 1
 982           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 983           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i)
& ",0," & angle(i) & ",0,0" & vbCrLf)
 984           Print(2, "SELRANGE,ND,3,1,1,1," & 0.5 * DistL1a - 0.0001 & "," & 0.5 *
DistL1a + 0.0001 & ",0,360,-0.0001,0.0001,1" & vbCrLf)
 985           Print(2, "NMODIFY," & 1 & ",ndmax,1,1," & -0.25 * DistL1a & ",0,0,3" &
vbCrLf)
 986       Next

 987       For i = 1 To 10
 988           Print(2, "CSYS,10,0," & ncfp & "," & 100 + 10 * (ncfp - 1) + i & "," &
(ncfp - 1) & vbCrLf)
 989           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 990           If 36 * (i - 1) > 180 Then
 991               Print(2, "SELRANGE,ND,10,1,1,1," & 0.4 * aux_dist(72) & "," & 0.6 *
aux_dist(72) & ",-0.001,0.001,-0.001,0.001,1" & vbCrLf)
 992               Print(2, "NMODIFY," & 1 & ",ndmax,1,1," & -0.25 * aux_dist(72) &
",0,0,10" & vbCrLf)
 993           Else
 994               Print(2, "SELRANGE,ND,10,1,1,1," & (0.5 * DistL1a) - 0.001 & "," &
(0.5 * DistL1a) + 0.001 & ",-0.001,0.001,-0.001,0.001,1" & vbCrLf)
 995               Print(2, "NMODIFY," & 1 & ",ndmax,1,1," & -0.25 * DistL1a & ",0,0,10"
& vbCrLf)
 996           End If
 997       Next

       'seleção dos nós da zona a ligar (ajuda)
 998       Print(2, "C* nós da zona a ligar" & vbCrLf)
```

```
 999        Print(2, "c* INITSEL,ND,1,1" & vbCrLf)
 1000       Print(2, "c* SELRANGE,ND,0,1,1,1,-0.0001,0.0001," & vY(1) - vY(4) - 0.001 &
"," & vY(1) + vY(2) + vY(5) + 0.001 & ",-0.001" & "," & vZ(1) + 0.001 & ",1" & vbCrLf)

        'seleção do plano da fenda (ajuda)
 1001       Print(2, "c* nós do plano da fenda" & vbCrLf)
 1002       Print(2, "c* INITSEL,ND,1,1" & vbCrLf)
 1003       Print(2, "c* SELRANGE,ND,0,1,1,1,-0.0001," & vX(1) + 0.0001 & "," & vY(1) +
vY(2) + vY(5) - 0.001 & "," & vY(1) + vY(2) + vY(5) + 0.001 & ",-0.001" & "," & vZ(1) +
0.001 & ",1" & vbCrLf)

        'seleção de todo o modelo (ajuda)
 1004       Print(2, "C* nós de todo o modelo" & vbCrLf)
 1005       Print(2, "c* INITSEL,all,1,1" & vbCrLf)
 1006       Print(2, "c* SELRANGE,all,0,1,1,1," & -vX(2) - 0.0001 & "," & vX(1) + 0.0001
& ",-0.001," & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.001" & "," & vZ(1) + 0.001 & ",1" &
vbCrLf)

        'symmetry plane
 1007       If SYM = 1 Then
 1008           Print(2, "C* Symmetry plane (surface crack centred on the cross-section
" & vbCrLf)
 1009           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1010           Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & vX(1) + 0.0001
& ",-0.001," & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.001" & "," & 0.001 & ",1" & vbCrLf)
 1011           Print(2, "DND,1,SZ,0,NDMAX,1, ;" & vbCrLf)
 1012       Else
 1013           Print(2, "C* Corner crack (no symmetry plane) " & vbCrLf)

 1014       End If

        'loanding and boundary conditions
 1015       Print(2, "ACTSET,CS,0" & vbCrLf)
 1016       If LDT = 1 Then
 1017           Print(2, " " & vbCrLf)
 1018           Print(2, "C* Load case 1 " & vbCrLf)
 1019           Print(2, "C* Boundary conditions " & vbCrLf)
 1020           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1021           Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1022           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1023           Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1024           appForce = Fmax / 4
 1025           Print(2, " " & vbCrLf)
 1026           Print(2, "C* Loading " & vbCrLf)
 1027           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1028           Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1029           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-
0.0001,0.0001,1" & vbCrLf)
 1030           Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1031           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," &
vZ(1) - 0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1032           Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

 1033       ElseIf LDT = 2 Then
```

```
1034            Print(2, " " & vbCrLf)
1035            Print(2, "C* Load case 2 " & vbCrLf)
1036            Print(2, "C* Boundary conditions " & vbCrLf)
1037            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1038            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1039            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1040            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

1041            appForce = Fmax / ((2 * NEL5 + 1) * 4)
1042            Print(2, " " & vbCrLf)
1043            Print(2, "C* Loading definitions " & vbCrLf)
1044            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1045            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) -
0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
1046            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-
0.0001,0.0001,1" & vbCrLf)
1047            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 &
"," & vZ(1) - 0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
1048            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 &
",-0.0001,0.0001,1" & vbCrLf)
1049            Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

1050        ElseIf LDT = 3 Then
1051            Print(2, " " & vbCrLf)
1052            Print(2, "C* Load case 3 " & vbCrLf)
1053            Print(2, "C* Boundary conditions " & vbCrLf)
1054            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1055            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1056            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1057            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

1058            appForce = Fmax / 4
1059            Print(2, " " & vbCrLf)
1060            Print(2, "C* Loading " & vbCrLf)
1061            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1062            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
1063            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
1064            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

1065        ElseIf LDT = 4 Then
1066            Print(2, " " & vbCrLf)
1067            Print(2, "C* Load case 4 " & vbCrLf)
1068            Print(2, "C* Boundary conditions " & vbCrLf)
1069            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1070            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1071            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1072            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

1073            appForce = Fmax / ((2 * NEL5 + 1) * 2)
```

```
 1074            Print(2, " " & vbCrLf)
 1075            Print(2, "C* Loading definitions " & vbCrLf)
 1076            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1077            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) -
0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1078            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-
0.0001,0.0001,1" & vbCrLf)
 1079            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1080        ElseIf LDT = 5 Then
 1081            Print(2, " " & vbCrLf)
 1082            Print(2, "C* Load case 5 " & vbCrLf)
 1083            Print(2, "C* Boundary conditions " & vbCrLf)
 1084            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1085            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1086            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1087            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1088            appForce = Fmax / 4
 1089            Print(2, " " & vbCrLf)
 1090            Print(2, "C* Loading " & vbCrLf)
 1091            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1092            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1093            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1094            Print(2, "FND,1,FZ," & appForce & ",NDMAX,1" & vbCrLf)

 1095        ElseIf LDT = 6 Then
 1096            Print(2, " " & vbCrLf)
 1097            Print(2, "C* Load case 6 " & vbCrLf)
 1098            Print(2, "C* Boundary conditions " & vbCrLf)
 1099            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1100            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1101            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1102            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1103            appForce = Fmax / ((2 * NEL5 + 1) * 2)
 1104            Print(2, " " & vbCrLf)
 1105            Print(2, "C* Loading definitions " & vbCrLf)
 1106            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1107            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) -
0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1108            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 &
"," & vZ(1) - 0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1109            Print(2, "FND,1,FZ," & appForce & ",NDMAX,1" & vbCrLf)

 1110        ElseIf LDT = 7 Then
 1111            Print(2, " " & vbCrLf)
 1112            Print(2, "C* Load case 7 " & vbCrLf)
 1113            Print(2, "C* Boundary conditions " & vbCrLf)
 1114            Print(2, "INITSEL,ND,1,1" & vbCrLf)
```

```
 1115            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1116            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1117            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1118            appForce = Fmax / 2
 1119            Print(2, " " & vbCrLf)
 1120            Print(2, "C* Loading " & vbCrLf)
 1121            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1122            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1123            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-
0.0001,0.0001,1" & vbCrLf)
 1124            Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

 1125        ElseIf LDT = 8 Then
 1126            Print(2, " " & vbCrLf)
 1127            Print(2, "C* Load case 8 " & vbCrLf)
 1128            Print(2, "C* Boundary conditions " & vbCrLf)
 1129            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1130            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1131            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1132            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1133            appForce = Fmax / 2
 1134            Print(2, " " & vbCrLf)
 1135            Print(2, "C* Loading " & vbCrLf)
 1136            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1137            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1138            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1139            Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

 1140        ElseIf LDT = 9 Then
 1141            Print(2, " " & vbCrLf)
 1142            Print(2, "C* Load case 9 " & vbCrLf)
 1143            Print(2, "C* Boundary conditions " & vbCrLf)
 1144            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1145            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 & "," & 0.001 & "," & -0.001 &
"," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1146            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1147            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1148            appForce = Fmax
 1149            Print(2, " " & vbCrLf)
 1150            Print(2, "C* Loading " & vbCrLf)
 1151            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1152            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1153            Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

 1154        ElseIf LDT = 10 Then
 1155            Print(2, " " & vbCrLf)
```

```
 1156            Print(2, "C* Load case 10 (3-point bending) " & vbCrLf)
 1157            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
 1158            Print(2, "ACTSET,CS,0" & vbCrLf)
 1159            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1160            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1161            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
 1162            Print(2, "C* Simple support" & vbCrLf)
 1163            Print(2, "ACTSET,CS,0" & vbCrLf)
 1164            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1165            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1166            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

 1167            appForce = Fmax / 4
 1168            Print(2, " " & vbCrLf)
 1169            Print(2, "C* Loading definitions " & vbCrLf)
 1170            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1171            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + 0.001 & ",-0.0001,0.0001,1" & vbCrLf)
 1172            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) + vY(2) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1173            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + 0.001 & "," & vZ(1) - 0.0001 & "," & vZ(1) +
0.0001 & ",1" & vbCrLf)
 1174            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) + vY(2) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1175            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1176        ElseIf LDT = 11 Then
 1177            Print(2, " " & vbCrLf)
 1178            Print(2, "C* Load case 11 (3-point bending) " & vbCrLf)
 1179            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
 1180            Print(2, "ACTSET,CS,0" & vbCrLf)
 1181            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1182            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1183            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
 1184            Print(2, "C* Simple support" & vbCrLf)
 1185            Print(2, "ACTSET,CS,0" & vbCrLf)
 1186            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1187            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1188            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

 1189            appForce = Fmax / ((2 * ((ncfp - 1) + 3 + NEL3) + 1) * 2)
 1190            Print(2, " " & vbCrLf)
 1191            Print(2, "C* Loading definitions " & vbCrLf)
 1192            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1193            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + 0.001 & "," & -0.0001 & "," & vZ(1) + 0.0001 &
",1" & vbCrLf)
 1194            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) + vY(2) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & -0.0001 & "," &
vZ(1) + 0.0001 & ",1" & vbCrLf)
 1195            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1196        ElseIf LDT = 12 Then
 1197            Print(2, " " & vbCrLf)
```

```
 1198            Print(2, "C* Load case 12 (3-point bending) " & vbCrLf)
 1199            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
 1200            Print(2, "ACTSET,CS,0" & vbCrLf)
 1201            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1202            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1203            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
 1204            Print(2, "C* Simple support" & vbCrLf)
 1205            Print(2, "ACTSET,CS,0" & vbCrLf)
 1206            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1207            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1208            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

 1209            appForce = Fmax / ((2 * (aux_NEL(3) + aux_NEL(4) + aux_NEL(5) +
aux_NEL(6)) + 1) * 2)
 1210            Print(2, " " & vbCrLf)
 1211            Print(2, "C* Loading definitions " & vbCrLf)
 1212            Print(2, "C* aux_NEL(3) " & aux_NEL(3) & vbCrLf)
 1213            Print(2, "C* aux_NEL(4) " & aux_NEL(4) & vbCrLf)
 1214            Print(2, "C* aux_NEL(5) " & aux_NEL(3) & vbCrLf)
 1215            Print(2, "C* aux_NEL(6) " & aux_NEL(4) & vbCrLf)

 1216            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1217            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & -0.0001 & "," & 0.0001 &
",1" & vbCrLf)
 1218            Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & vZ(1) - 0.0001 & "," &
vZ(1) + 0.0001 & ",1" & vbCrLf)
 1219            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1220        ElseIf LDT = 13 Then
 1221            Print(2, " " & vbCrLf)
 1222            Print(2, "C* Load case 13 (3-point bending) " & vbCrLf)
 1223            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
 1224            Print(2, "ACTSET,CS,0" & vbCrLf)
 1225            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1226            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1227            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
 1228            Print(2, "C* Simple support" & vbCrLf)
 1229            Print(2, "ACTSET,CS,0" & vbCrLf)
 1230            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1231            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1232            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

 1233            appForce = Fmax / ((((2 * (aux_NEL(3) + aux_NEL(4) + aux_NEL(5) +
aux_NEL(6)) + 1)) * ((2 * ((ncfp - 1) + 3 + NEL3) + 1))))
 1234            Print(2, " " & vbCrLf)
 1235            Print(2, "C* Loading definitions " & vbCrLf)
 1236            Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1237            Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 - vX(2) & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & -0.0001 & "," & vZ(1) +
0.0001 & ",1" & vbCrLf)
 1238            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1239        ElseIf LDT = 14 Then

 1240            Print(2, " " & vbCrLf)
```

```
1241            Print(2, "C* Load case 14 (3-point bending) " & vbCrLf)
1242            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
1243            Print(2, "ACTSET,CS,0" & vbCrLf)
1244            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1245            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1246            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
1247            Print(2, "C* Simple support" & vbCrLf)
1248            Print(2, "ACTSET,CS,0" & vbCrLf)
1249            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1250            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1251            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

1252            appForce = Fmax / (node_max + node_max - 1)
1253            Print(2, " " & vbCrLf)
1254            Print(2, "C* Loading " & vbCrLf)
1255            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1256            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + 0.5 * vY(2) - 0.001 & ", " & vY(1) + 0.5 * vY(2) + 0.001 & ",-0.0001," &
vZ(1) + 0.0001 & ",1" & vbCrLf)
1257            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

1258        ElseIf LDT = 15 Then

1259            Print(2, " " & vbCrLf)
1260            Print(2, "C* Load case 15 (3-point bending) " & vbCrLf)
1261            Print(2, "C* Fixed support (translations restrained)" & vbCrLf)
1262            Print(2, "ACTSET,CS,0" & vbCrLf)
1263            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1264            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(7) - 0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1265            Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)
1266            Print(2, "C* Simple support" & vbCrLf)
1267            Print(2, "ACTSET,CS,0" & vbCrLf)
1268            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1269            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & vY(1) + vY(2) + vY(3) - vY(6) - 0.001 & "," & vY(1) + vY(2) + vY(3) - vY(6) +
0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
1270            Print(2, "DND,1,UX,0,NDMAX,1, ;" & vbCrLf)

1271            appForce = Fmax / 2
1272            Print(2, " " & vbCrLf)
1273            Print(2, "C* Loading " & vbCrLf)
1274            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1275            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + 0.5 * vY(2) - 0.001 & ", " & vY(1) + 0.5 * vY(2) + 0.001 & ",-0.0001," &
0.0001 & ",1" & vbCrLf)
1276            Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + 0.5 * vY(2) - 0.001 & ", " & vY(1) + 0.5 * vY(2) + 0.001 & "," & vZ(1) -
0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
1277            Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

1278        ElseIf LDT = 16 Then

1279            Print(2, " " & vbCrLf)
1280            Print(2, "C* Load case 14 " & vbCrLf)
1281            Print(2, "C* Boundary conditions " & vbCrLf)
1282            Print(2, "INITSEL,ND,1,1" & vbCrLf)
1283            Print(2, "SELRANGE,ND,0,1,1,1," & -0.001 - vX(2) & "," & -vX(2) + 0.001
& "," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
```

```
 1284           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.001 & "," & vX(1) + 0.001 &
"," & -0.001 & "," & vY(7) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 1285           Print(2, "DND,1,AU,0,NDMAX,1, ;" & vbCrLf)

 1286           appForce = Fmax / 4
 1287           Print(2, " " & vbCrLf)
 1288           Print(2, "C* Loading " & vbCrLf)
 1289           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1290           Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1291           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & ",-
0.0001,0.0001,1" & vbCrLf)
 1292           Print(2, "SELRANGE,ND,0,1,1,1," & -0.0001 & "," & 0.001 & ", " & vY(1) +
vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1293           Print(2, "SELRANGE,ND,0,1,1,1," & vX(1) - 0.0001 & "," & vX(1) + 0.001 &
", " & vY(1) + vY(2) + vY(3) - 0.001 & ", " & vY(1) + vY(2) + vY(3) + 0.001 & "," &
vZ(1) - 0.0001 & "," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1294           Print(2, "FND,1,FY," & appForce & ",NDMAX,1" & vbCrLf)

 1295           Print(2, " " & vbCrLf)
 1296           Print(2, "C* Loading definitions " & vbCrLf)
 1297           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 1298           Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + 0.001 & ",-0.0001,0.0001,1" & vbCrLf)
 1299           Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) + vY(2) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & ",-0.0001,0.0001,1" &
vbCrLf)
 1300           Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) - 0.001 & ", " & vY(1) + 0.001 & "," & vZ(1) - 0.0001 & "," & vZ(1) +
0.0001 & ",1" & vbCrLf)
 1301           Print(2, "SELRANGE,ND,0,1,1,1," & -vX(2) - 0.0001 & "," & -vX(2) + 0.001
& ", " & vY(1) + vY(2) - 0.001 & ", " & vY(1) + vY(2) + 0.001 & "," & vZ(1) - 0.0001 &
"," & vZ(1) + 0.0001 & ",1" & vbCrLf)
 1302           Print(2, "FND,1,FX," & appForce & ",NDMAX,1" & vbCrLf)

 1303       End If

            'select all entities
 1304           Print(2, " " & vbCrLf)
 1305           Print(2, "C* Select all entities " & vbCrLf)
 1306           Print(2, "INITSEL,ALL,1, 1" & vbCrLf)
 1307           Print(2, "SELRANGE,ALL,0,1,1,1," & -0.001 - vX(2) & "," & vX(1) + 0.001
& ",-0.001," & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" &
vbCrLf)
            'run static analysis
 1308           Print(2, " " & vbCrLf)
 1309           Print(2, "C* Run Static analysis" & vbCrLf)
 1310           Print(2, "R_Static" & vbCrLf)

            'export displacements and node coordinates
 1311           Call mygeofiles.ExportCoordinatesDisplacements()

            'create end file
 1312           Print(2, " " & vbCrLf)
 1313           Print(2, "ACTSET,CS,0" & vbCrLf)
 1314           Print(2, "C* End of analysis" & vbCrLf)
 1315            Print(2, "LISTLOG,1,end.lis,0" & vbCrLf)
 1316           Print(2, "LISTLOG,0" & vbCrLf)

            'close mesh file
```

```
            FileClose(2)
            Main.ListBox1.Items.Add(Now & "  Mesh has been created ")

        'run simulation
            ChDir(folderpath & "\" & myname & "\" & it)
            FileOpen(1, "run.bat", OpenMode.Append)
            Print(1, " " & exepath & " " & folderpath & "\" & myname & "\" & it &
"\it.gen " & folderpath & "\" & myname & "\" & "mesh-" & it & ".ses")
            FileClose(1)

            Shell(folderpath & "\" & myname & "\" & it & "\" & "run.bat", 6)

        'activate timer
            Main.Timer1.Enabled = True
            Main.ListBox1.Items.Add(Now & "  FE software has been launched ")

    End Sub
```

Subrotina A6. Subrotina mesh_wt()

```
Sub ExportCoordinatesDisplacements()

        'define coordinate system in the crack plane similar to the one used to define
the crack front coordinates
        Print(2, "ACTSET,CS,0" & vbCrLf)

        'coordinates
        Print(2, " " & vbCrLf)
        Print(2, "C* Coordinates A " & vbCrLf)
        Print(2, "LISTLOG,1,CoordinatesA.lis,0" & vbCrLf)
        For i = 1 To ncfp
            Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
            Print(2, "INITSEL,ND,1,1" & vbCrLf)
            Print(2, "SELRANGE,ND,3,1,1,1," & (DistL1a - 0.001) & "," & (DistL1a +
0.001) & ",179,181,-0.001,0.001,1" & vbCrLf)
            Print(2, "NLIST,1,ndmax,1,0" & vbCrLf)
        Next
        Print(2, "LISTLOG,0" & vbCrLf)

        Print(2, "" & vbCrLf)
        Print(2, "C* Coordinates B" & vbCrLf)
        Print(2, "LISTLOG,1,CoordinatesB.lis,0" & vbCrLf)
        For i = 1 To ncfp
            Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
            Print(2, "INITSEL,ND,1,1" & vbCrLf)
            Print(2, "SELRANGE,ND,3,1,1,1," & (DistL1a + DistL2a) - 0.0001 & "," &
(DistL1a + DistL2a) + 0.0001 & ",179,181,-0.001,0.001,1" & vbCrLf)
            Print(2, "NLIST,1,ndmax,1,0" & vbCrLf)
        Next
        Print(2, "LISTLOG,0" & vbCrLf)

        'displacements
        Print(2, " " & vbCrLf)
        Print(2, "C* Displacements A " & vbCrLf)
        Print(2, "LISTLOG,1,DisplacementsA.lis,0" & vbCrLf)
        For i = 1 To ncfp
            Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
            Print(2, "INITSEL,ND,1,1" & vbCrLf)
```

```
 30           Print(2, "SELRANGE,ND,3,1,1,1," & DistL1a + -0.0001 & "," & DistL1a +
0.0001 & ",179,181,-0.001,0.001,1" & vbCrLf)
 31           Print(2, "DISLIST,1,1,1,ndmax,1,0" & vbCrLf)
 32       Next
 33      Print(2, "LISTLOG,0" & vbCrLf)

 34       Print(2, " " & vbCrLf)
 35       Print(2, "C* Displacements B " & vbCrLf)
 36       Print(2, "LISTLOG,1,DisplacementsB.lis,0" & vbCrLf)
 37       For i = 1 To ncfp
 38           Print(2, "CSANGL,3,1," & x(i) & "," & vY(1) + vY(2) + vY(5) & "," & z(i) &
",0," & angle(i) & ",0,0" & vbCrLf)
 39           Print(2, "INITSEL,ND,1,1" & vbCrLf)
 40           Print(2, "SELRANGE,ND,3,1,1,1," & (DistL1a + DistL2a) - 0.0001 & "," &
(DistL1a + DistL2a) + 0.0001 & ",179,181,-0.001,0.001,1" & vbCrLf)
 41           Print(2, "DISLIST,1,1,1,ndmax,1,0" & vbCrLf)
 42       Next
 43       Print(2, "LISTLOG,0" & vbCrLf)

        'forces
 44       Print(2, " " & vbCrLf)
 45       Print(2, "C* Applied forces" & vbCrLf)
 46       Print(2, "LISTLOG,1,Forces.lis,0" & vbCrLf)
 47       Print(2, "ACTSET,CS,0" & vbCrLf)
 48       Print(2, "INITSEL,ALL,1, 1" & vbCrLf)
 49       Print(2, "SELRANGE,ALL,0,1,1,1," & -0.001 - vX(2) & "," & vX(1) + 0.001 & ",-
0.001," & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)
 50       Print(2, "FLIST,1,ndmax,1" & vbCrLf)
 51       Print(2, "LISTLOG,0" & vbCrLf)

        'select all entities
 52       Print(2, "C* Select all entities " & vbCrLf)
 53       Print(2, "INITSEL,ALL,1, 1" & vbCrLf)
 54       Print(2, "SELRANGE,ALL,0,1,1,1," & -0.001 - vX(2) & "," & vX(1) + 0.001 & ",-
0.001," & vY(1) + vY(2) + vY(3) + 0.001 & ",-0.001," & vZ(1) + 0.001 & ",1" & vbCrLf)

 55   End Sub
```

Subrotina A7. Subrotina ExportCoordinatesDisplacements()

```
  Sub sif(x, y, z, coorA, coorB, dispA, dispB, rpA, rpB, FITA, FITB, FITp)

 1      For i = 1 To ncfp

            'radial distances
 2           rpA(i) = ((x(i) - coorA(i, 1)) ^ 2 + (y(i) - coorA(i, 2)) ^ 2 + (z(i) -
coorA(i, 3)) ^ 2) ^ 0.5
 3           rpB(i) = ((x(i) - coorB(i, 1)) ^ 2 + (y(i) - coorB(i, 2)) ^ 2 + (z(i) -
coorB(i, 3)) ^ 2) ^ 0.5

            'stress intensity factors
 4           FITA(i) = ((2 * Math.PI) ^ 0.5) * EX * dispA(i, 2) / (4 * ((rpA(i)) ^ 0.5)
* (1 - NUXY ^ 2))
 5           FITB(i) = ((2 * Math.PI) ^ 0.5) * EX * dispB(i, 2) / (4 * ((rpB(i)) ^ 0.5)
* (1 - NUXY ^ 2))

            'extrapolation for r=0
 6           FITp(i) = FITA(i) - rpA(i) * (FITB(i) - FITA(i)) / (rpB(i) - rpA(i))

            'correction for plane stress state at surface
 7           If i = ncfp Then
```

```
 8              FITp(i) = FITp(i) * (1 - NUXY ^ 2)
 9           End If

 10           If SYM = 0 Then 'caso sem simetria (fenda de canto com simetria nas duas
extremidades da fenda)
 11               FITp(1) = FITp(1) * (1 - NUXY ^ 2)
 12           End If


 13       Next


        'save values
 14       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 15       Print(3, "iteration " & it & vbCrLf)
 16       Print(3, "i   Ax  Ay  Az  Bx  By  Bz   AUx  AUy AUz  BUx   BUy   BUz   rpA
rpB   FITA   FITB   FIT " & vbCrLf)
 17       For i = 1 To ncfp
 18           Print(3, i & " " & coorA(i, 1) & " " & coorA(i, 2) & " " & coorA(i, 3) & "
" & coorB(i, 1) & " " & coorB(i, 2) & " " & coorB(i, 3) & " " & dispA(i, 1) & " " &
dispA(i, 2) & " " & dispA(i, 3) & " " & dispB(i, 1) & " " & dispB(i, 2) & " " & dispB(i,
3) & " " & rpA(i) & " " & rpB(i) & " " & FITA(i) & " " & FITB(i) & " " & FITp(i) &
vbCrLf)
 19       Next
 20       FileClose(3)

 21   End Sub
```

Subrotina A8. Subrotina SIF()

```
Sub effectiveSIFrange(FIT, deltaFIT)

        'stress intensity factor range
 1       For i = 1 To ncfp
 2           deltaFIT(i) = FIT(i) * (1 - (Fmin / Fmax))
 3       Next

        'save values
 4       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 5       Print(3, "stress intensity factor ranges " & it & vbCrLf)
 6       Print(3, "i   value " & vbCrLf)
 7       For i = 1 To ncfp
 8           Print(3, i & " " & deltaFIT(i) & vbCrLf)
 9       Next
 10       FileClose(3)

 11   End Sub
```

Subrotina A9. Subrotina effectiveSIF()

```
Sub newcrackfront(deltaFIT, x)

        'calculate second order derivatives of current crack front nodes
 1       Call mycubicspline.SecondDerivatives(iAngle, iRadius, ncfp, D2)
 2       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 3       Print(3, "Second order derivatives " & vbCrLf)
```

```
 4       For i = 1 To ncfp
 5           Print(3, "Node " & i & " " & D2(i) & vbCrLf)
 6       Next
 7       FileClose(3)

        'crack propagation direction
 8       Call mycubicspline.crackpropagationdirection(iAngle, iRadius, ncfp, D2, deriv)
 9       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 10       Print(3, "Crack propagation direction (first order derivatives) " & vbCrLf)
 11       For i = 1 To ncfp
 12           Print(3, "Node " & i & " " & deriv(i) & vbCrLf)
 13       Next
 14       FileClose(3)

        'find maximum effective stress intensity factor range
 15       DFITmax = deltaFIT(1)
 16       For i = 1 To ncfp
 17           If deltaFIT(i) > DFITmax Then
 18               DFITmax = deltaFIT(i)
 19           End If
 20       Next
 21       Kmax = DFITmax * (Fmax / (Fmax - Fmin))
 22       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 23       Print(3, "Maximum stress intensity factor " & vbCrLf)
 24       Print(3, Kmax & vbCrLf)
 25       FileClose(3)

        'calculate number of fatigue cycles
 26       DeltaN = (DAmax * 0.001) / (CCC * ((0.001 ^ 0.5) * DFITmax) ^ mmm)
'(m/cycle; MPa m0.5)
 27       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 28       Print(3, "Number of loading cycles " & vbCrLf)
 29       Print(3, DeltaN & vbCrLf)
 30       FileClose(3)

        'calculate crack increments
 31       For i = 1 To ncfp
 32           deltaA(i) = DAmax * ((deltaFIT(i) / DFITmax) ^ mmm)
 33       Next
 34       FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 35       Print(3, "Nodal crack advances " & vbCrLf)
 36       For i = 1 To ncfp
 37           Print(3, i & " " & deltaA(i) & vbCrLf)
 38       Next
 39       FileClose(3)

        'calculate new nodal positions (in cartesian and polar coordinates)
 40       For i = 1 To ncfp
 41           pX(i) = x(i) + deltaA(i) * (Math.Cos(Math.Atan(deriv(i))))
 42           pY(i) = 0
 43           pZ(i) = z(i) - deltaA(i) * (Math.Sin(Math.Atan(deriv(i))))
            'corrige se a fenda anda para tráz
 44           If pZ(i) < z(i) Then
 45               pX(i) = x(i) + deltaA(i) * (Math.Cos(Math.Atan(-deriv(i))))
 46               pY(i) = 0
 47               pZ(i) = z(i) - deltaA(i) * (Math.Sin(Math.Atan(-deriv(i))))
 49           End If
 50       Next
```

```
 51        FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 52        For i = 1 To ncfp
 53            pRadius(i) = ((pX(i) ^ 2) + (pZ(i) ^ 2)) ^ 0.5
 54            pAngle(i) = Math.Atan((pZ(i)) / pX(i))
 55            Print(3, i & " " & pAngle(i) & " " & pRadius(i) & vbCrLf)
 56        Next

 57        Print(3, "graus" & vbCrLf)
 58        For i = 1 To ncfp
 59            pAngle(i) = pAngle(i) * (180 / Math.PI)
 60        Next
 61        FileClose(3)

 62        FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 63        Print(3, "New crack front positions in cartesian coordiantes (not optimised) "
& vbCrLf)
 64        Print(3, "i    X      Y      Z " & vbCrLf)
 65        For i = 1 To ncfp
 66            Print(3, i & " " & pX(i) & " " & pY(i) & " " & pZ(i) & vbCrLf)
 67        Next
 68        Print(3, "New crack front positions in polar coordiantes (not optimised) " &
vbCrLf)
 69        Print(3, "i    teta(rad)  teta(graus)     r " & vbCrLf)
 70        For i = 1 To ncfp
 71            Print(3, i & " " & pAngle(i) & "      " & pRadius(i) & vbCrLf)
 72        Next
 73        FileClose(3)

        'calculate second order derivatives of provisional crack front nodes
        'Call mycubicspline.SecondDerivatives(pZ, pX, ncfp, D2)
 74        Call mycubicspline.SecondDerivatives(pAngle, pRadius, ncfp, D2)
 75        FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 76        Print(3, "Second order derivatives of the new crack front (for optimisation)"
& vbCrLf)
 77        For i = 1 To ncfp
 78            Print(3, i & " " & D2(i) & vbCrLf)
 79        Next
 80        FileClose(3)

 81        ai = 1
 82        af = ncfp - 1

 83        FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
        'first node
 84        Print(3, "iAngle(1) " & iAngle(1) & vbCrLf)
 85        Print(3, "pAngle(1) " & pAngle(1) & vbCrLf)
 86        If iAngle(1) < pAngle(1) Then
 87            fRadius(1) = pRadius(1)
 88            ai = 2
 89        End If
 90        Print(3, "pRadius(1) " & pRadius(1) & vbCrLf)
 91        Print(3, "fRadius(1) " & fRadius(1) & vbCrLf)

        'last node
 92        Print(3, "iAngle(ncfp) " & iAngle(ncfp) & vbCrLf)
 93        Print(3, "pAngle(ncfp) " & pAngle(ncfp) & vbCrLf)
 94        If (iAngle(ncfp) > pAngle(ncfp)) Then
 95            fRadius(ncfp) = pRadius(ncfp)
 96            af = ncfp - 1
```

```
 97        End If
 98        Print(3, "pRadius(ncfp) " & pRadius(ncfp) & vbCrLf)
 99        Print(3, "fRadius(ncfp) " & fRadius(ncfp) & vbCrLf)

 100        nint = 1
 101        For i = ai To af
 102            nint = 1
 103            Do While Not ((iAngle(i) >= pAngle(nint)) And (iAngle(i) <= pAngle(nint +
1)))
 104                nint = nint + 1
 105                If nint + 1 > 100 Then
 106                End If
 107            Loop
 108            Term1 = D2(nint) * (pAngle(nint + 1) - iAngle(i)) ^ 3 / (6 * (pAngle(nint
+ 1) - pAngle(nint)))
 109            Term2 = D2(nint + 1) * (iAngle(i) - pAngle(nint)) ^ 3 / (6 * (pAngle(nint
+ 1) - pAngle(nint)))
 110            Term3 = (pRadius(nint) / (pAngle(nint + 1) - pAngle(nint)) - D2(nint) *
(pAngle(nint + 1) - pAngle(nint)) / 6) * (pAngle(nint + 1) - iAngle(i))
 111            Term4 = (pRadius(nint + 1) / (pAngle(nint + 1) - pAngle(nint)) - D2(nint
+ 1) * (pAngle(nint + 1) - pAngle(nint)) / 6) * (iAngle(i) - pAngle(nint))
 112            fRadius(i) = Term1 + Term2 + Term3 + Term4
 113        Next

        'surface node
 114        For i = 1 To ncfp
 115            fAngle(i) = iAngle(i)
 116        Next
 117        FileClose(3)

        'save final crack front coordinates
 118        FileOpen(3, folderpath & "\" & myname & "\output-" & myname & ".out",
OpenMode.Append)
 119        Print(3, "Final coordinates of the new crack front (polar coordinates) " &
vbCrLf)
 120        Print(3, "i   angle       radius " & vbCrLf)
 121        For i = 1 To ncfp
 122            Print(3, i & "    " & fAngle(i) & "     " & fRadius(i) & vbCrLf)
 123        Next
 124        FileClose(3)

 125    End Sub
```

Subrotina A10. Subrotina newcrackfront()

APÊNDICE A

APÊNDICE B

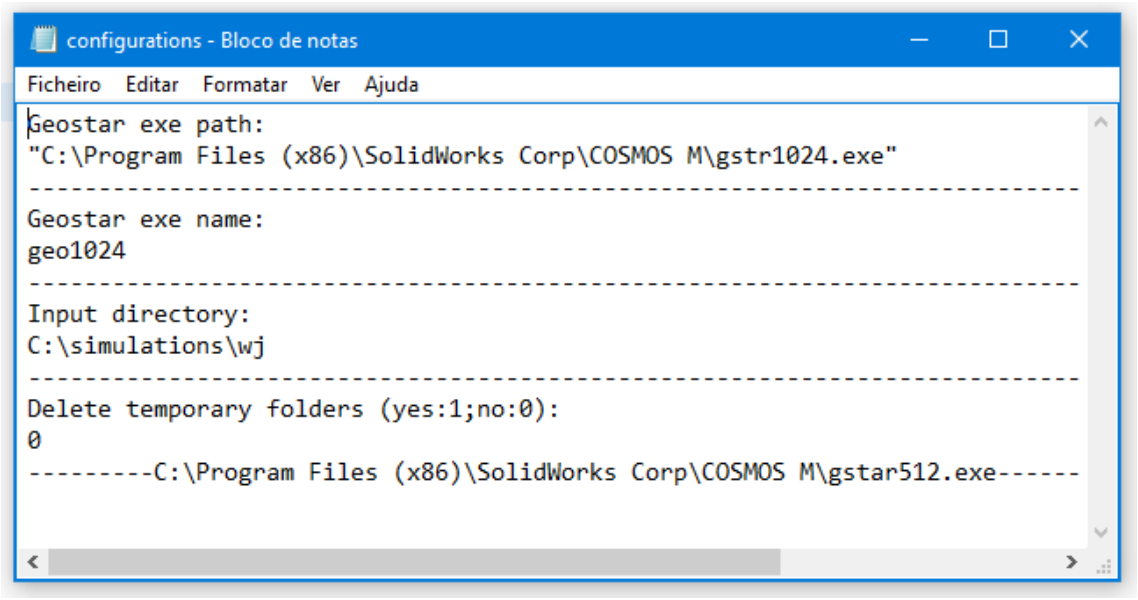

Figura B.1. Ficheiro ASCII para abertura do programa de elementos finitos (*configurations.txt*).

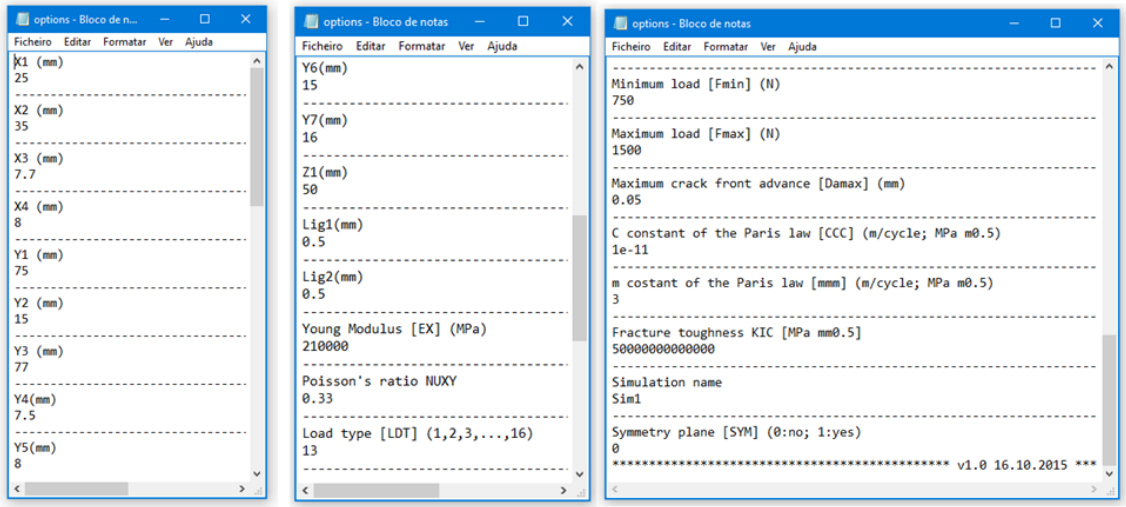

Figura B.2. Ficheiro ASCII para definição da geometria do provete em T, do nível de penetração dos cordões de soldadura, dos tipos e magnitude dos carregamentos e das propriedades do material (*options.txt*).

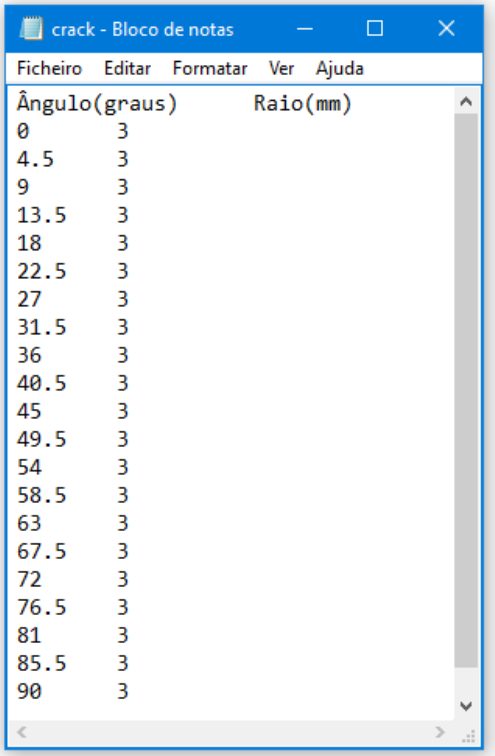

Figura B.3. Ficheiro ASCII para definição da frente de fenda inicial (*crack.txt*).

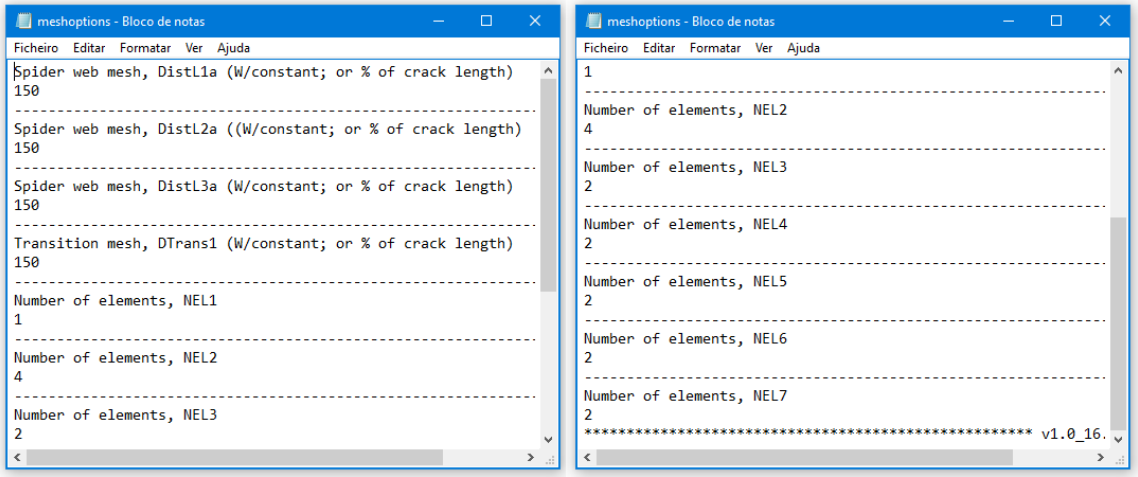

Figura B.4. Ficheiro ASCII para definição das variáveis L_1, L_2, L_3 e L_4 (correspondentes as primeiras 4 variáveis do ficheiro) e das variáveis NEL (*meshoptions.txt*).

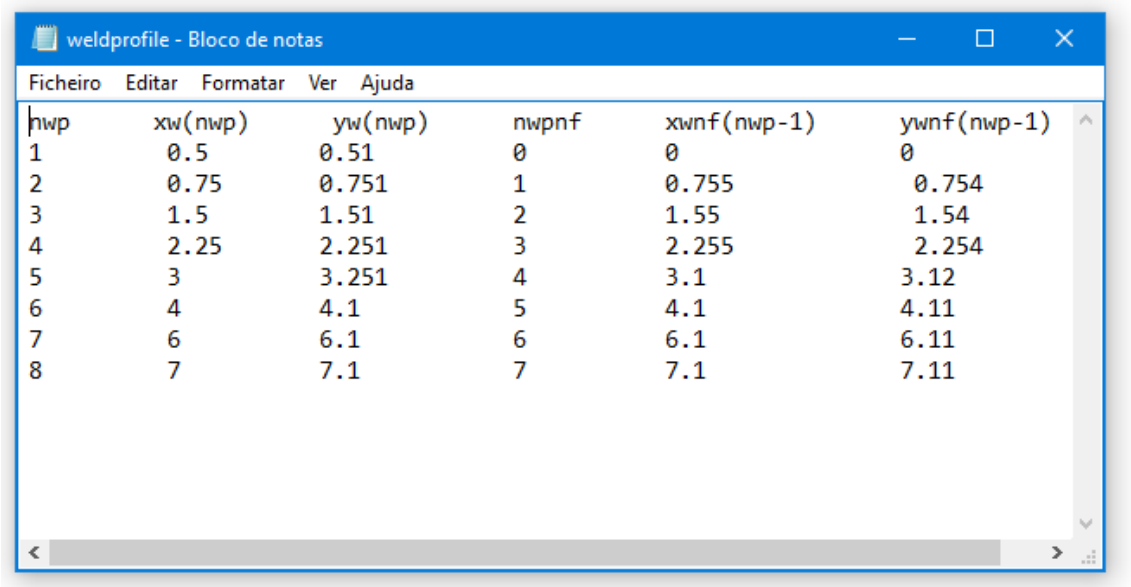

Figura B.5. Ficheiro ASCII para definição dos perfis de soldadura fissurados (3 colunas da esquerda) e não fissurados (3 colunas da direita) (*weldprofile.tx*t).

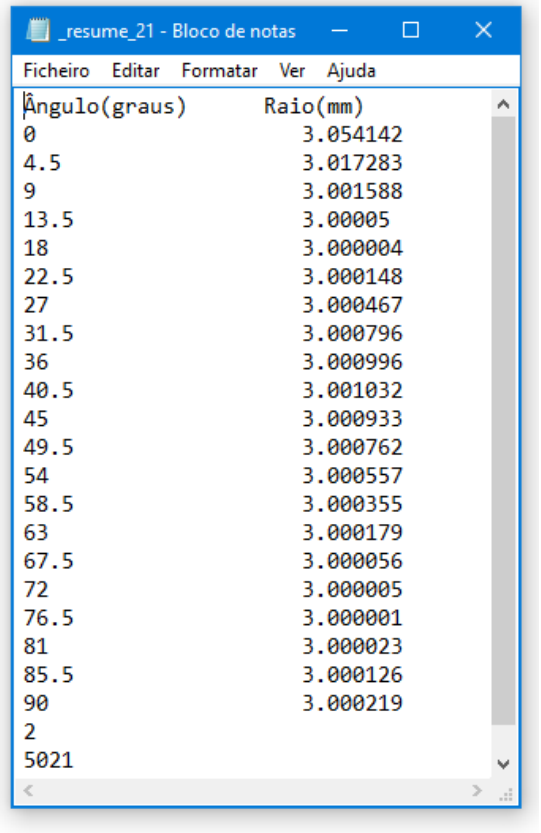

Figura B.6. Ficheiro ASCII para definição da frente de fenda inicial, nº de iterações e nº de corridas realizadas (*resume.out*).