MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
DEPARTAMENTO DE ENGENHARIA MECÂNICA

IMPLEMENTAÇÃO DA PSEUDOALEATORIEDADE NAS PROPRIEDADES DE MATERIAL NA TEORIA PERIDINÂMICA ATRAVÉS DO AMBIENTE PERIDIGM

por

Lucas Breda Soares

Monografia apresentada ao Departamento de Engenharia Mecânica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do diploma de Engenheiro Mecânico.

Porto Alegre, Junho de 2018

Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Engenharia Mecânica

IMPLEMENTAÇÃO DA PSEUDOALEATORIEDADE NAS PROPRIEDADES DE MATERIAL NA TEORIA PERIDINÂMICA ATRAVÉS DO AMBIENTE PERIDIGM

por

Lucas Breda Soares

ESTA MONOGRAFIA FOI JULGADA ADEQUADA COMO PARTE DOS REQUISITOS PARA A OBTENÇÃO DO TÍTULO DE
ENGENHEIRO MECÂNICO
APROVADA EM SUA FORMA FINAL PELA BANCA EXAMINADORA DO DEPARTAMENTO DE ENGENHARIA MECÂNICA

Prof. Thamy Cristina Hayashi
Coordenador do Curso de Engenharia Mecânica

Área de Concentração: **Mecânica dos Sólidos**

Orientador: Prof. Ignacio Iturrioz

Comissão de Avaliação:

Prof. Ignacio Iturrioz

Prof. Rogério José Marczak

Prof. Walter Jesus Paucar Casas

Porto Alegre, Junho de 2018.

SOARES, L. B. **Implementação da pseudoaleatoriedade nas propriedades de material na teoria peridinâmica através do ambiente Peridigm**. 2018. 14 folhas. Monografia (Trabalho de Conclusão do Curso em Engenharia Mecânica) – Departamento de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018.

RESUMO

Foi realizado neste trabalho, a simulação da fratura de materiais quase frágeis utilizando uma versão modificada do programa de peridinâmica de código aberto Peridigm. Esta modificação visa a implementação da aleatoriedade nas propriedades do material dentro do programa. Foram simulados dois exemplos, uma placa de concreto submetida a uma carga impulsiva trativa, e um disco de acrílico submetido a uma carga de impacto.
Os resultados foram apresentados em termos de forças e deslocamentos globais, balanços de energia, mapa de dano e configurações finais. Os resultados de ambas simulações foram comparados com outras referências.

PALAVRAS-CHAVE: Peridinâmica; Pseudoaleatoriedade; Fratura

SOARES, L. B. **Implementation of pseudorandomness in material properties in peridynamics theory through Peridigm system**. 2018. 14 folhas. Monografia (Trabalho de Conclusão do Curso em Engenharia Mecânica) – Departamento de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018.

ABSTRACT

It was introduced in this paper the fracture simulation of quasi-fragile materials utilizing the modified version of the open source peridynamics software Peridigm. This modification aims to implement the randomness in the material properties within the software. It was evaluated two distinct scenarios, a concrete plate under tractive prescribed displacement and an acrylic disc under impact load.
The results were presented in terms of global displacement and forces, energy balance, damage map and final configuration. The results of both cases were compared with other sources.

KEYWORDS: Peridynamics; Pseudorandomness, Fracture

ÍNDICE

Pág.

1. INTRODUÇÃO

A mecânica da fratura é a área de mecânica dos sólidos dedicada ao estudo dos mecanismos que governam a nucleação e propagação de trincas em estruturas. O maior desafio para a representação da fratura de materiais é a representação macroscópica de fenômenos que são regidos predominantemente por características mesoscópicas, intrínsecas do material ou do processo de fabricação que o mesmo tenha sido submetido.

A falha de componentes por fratura é amplamente estudada para obtenção de parâmetros que possam descrever o comportamento médio de falha de um determinado material, pois independentemente do nível de controle aplicado na fabricação dos corpos de prova e da realização de testes é virtualmente impossível a obtenção de duas configurações idênticas de ruptura durante a falha de um componente. A motivação desse trabalho se centra na modelagem estrutural de sistemas nos quais o processo de dano está dominado pela aparição de fissuras e finalmente uma configuração de trincas macroscópicas. Nestes casos, a natureza aleatória do material tem que ser considerada para que possa representar corretamente o problema estudado.

Para a simulação da fratura, se pode utilizar o método dos elementos finitos. A maior limitação deste método se encontra na dificuldade em representar a transição entre um meio continuo, hipótese onde o método se fundamenta, e o descontínuo, que se produz quando o material rompe. Isto pode ser realizado através de diversas técnicas, como o método das interfaces coesivas, proposta por Xu e Needleman (1994), onde é feita a inserção de elementos coesivos na interface dos elementos que se separam para modelar a trinca, os quais as forças de tração aplicadas ultrapassam a resistência do material. A principal limitação deste método se deve à restrição da adição de elementos coesivos somente às interfaces entre os elementos. Também é possível utilizar o método de elemento finitos estendido (XFEM), proposto por Belytschko et al. (2003), onde problemas que envolvem pontos de singularidade do meio são modelados através da adição de um conjunto de funções base às funções de interpolação.

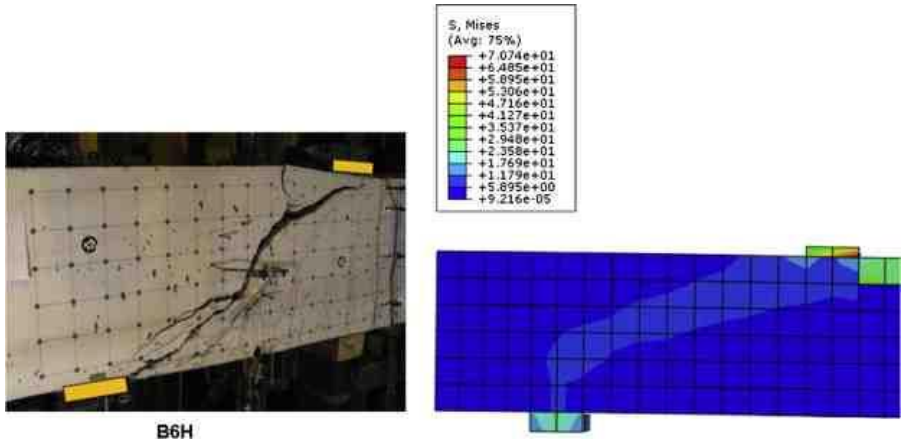
Outra alternativa é utilizar o método de elementos discretos para a simulação deste tipo de problema. Neste caso, o material é representado como um arranjo de nós, onde se concentram as massas, que são vinculadas através de leis de interação não lineares. A aparição de descontinuidades quando a estrutura é excitada pode implicar na degradação destas interações, criando a descontinuidade no material de forma natural. Deste modo, fenômenos como a clasterização de fissuras, fenômeno de localização que transforma fissuras isoladas em macrofissuras, podem ser capturados sem violar as hipóteses básicas da metodologia empregada.

Diversas versões do método dos elementos discretos podem ser citadas, dentre elas as versões de Chiaia et al. (1997), Rinaldi (2007) e Riera (1984), onde as interações intermodais são representadas como barras de treliças ou vigas, dependendo dos graus de liberdade que são considerados em cada caso. Nestes casos arranjos de barras fixos, que podem ser conformes ou não, são empregados

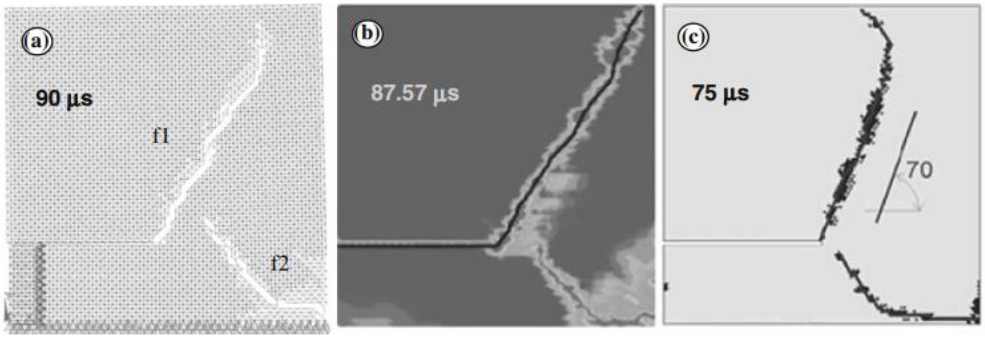
Outra particularização do método de elementos discretos é a Peridinâmica, proposta por Silling (2000), onde um meio é representado por pontos discretos de massa que tem interação definida através de um campo de influência, determinado por uma região esférica de raio δ, denominado horizonte. Os pontos dentro desta esfera, centrada em um nó de referência, são denominados de vizinhança ou família deste nó de referência. Neste método, o dano é computado através de um alongamento crítico entre dois pontos vizinhos, onde passado este alongamento, a força de interação entre estes dois pontos é cessada, sendo redistribuída entre os pontos remanescentes, os tornando mais suscetíveis a falha.

Na figura 1.A se apresenta a simulação da ruptura de uma viga de material quase frágil realizada com elementos finitos, onde a fissura é representada através de uma degradação continua desta região. Na figura 1.B se apresenta uma comparação de um problema de propagação instável de uma fissura, realizado com uma versão do método dos elementos discretos formados por barras, com o método das interfaces coesivas e com a utilização das

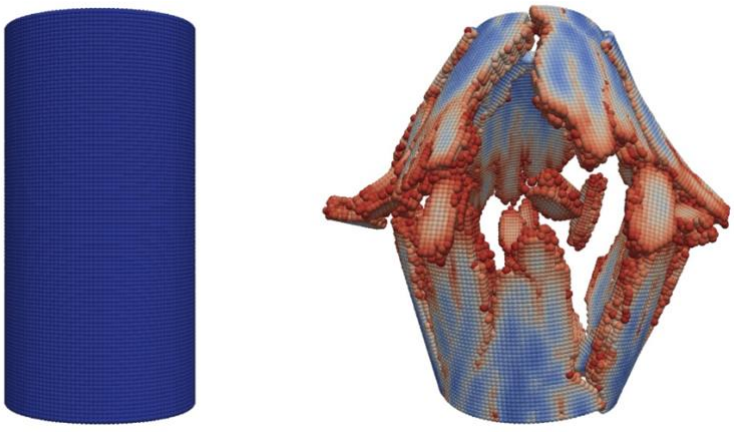
funções de interpolação singulares. Na Fig. 1.C se ilustra a ruptura de uma tubulação frente a um incremento brusco de pressão empregando a peridinâmica.



(A)



(B)



(a) Time=0.0 ms. (b) Time=2.5 ms.

(C)

Figura 1.1: (A) Configuração de uma viga de material quase frágil e sua simulação em elementos finitos, Ibrahim (2015). (B) Simulação da propagação instável de uma trinca utilizando uma versão do método dos elementos discretos formado por barras e duas versões do método dos elementos finitos, Kosteski et al. (2012). (C) Simulação da ruptura de um cilindro de material quase-frágil utilizando a Peridinâmica, Parks (2012).

Desta forma são definidos os objetivos do presente trabalho. **Objetivo geral:** Simular empregando o método dos elementos discretos – peridinâmica – estruturas formadas de materiais quase frágeis. **Objetivos específicos:** (a) Implementar a consideração da aleatoriedade nas propriedades do material na versão do programa de peridinâmica em que se trabalha, Sistema Peridigm de código aberto desenvolvido pela Sandia National Laboratories, Parks et. al. (2012). (b) Simular um corpo de prova submetido a tração até a sua falha utilizando o sistema Peridigm. (c) Simular um teste de impacto em um disco de material polimérico utilizando o Peridigm. (d) Comparar os resultados com outra solução numérica e com resultados experimentais.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. MECÂNICA DA FRATURA

O estado de tensões de um ponto pode ser completamente descrito através do tensor de tensões, que possui informação das tensões em todas as direções. Nos projetos convencionais de componentes mecânicos, utilizam-se as teorias clássicas de resistência dos materiais para obter um único valor de tensão equivalente, denominadas tensão de cálculo, que são utilizadas no comparativo com um valor de tensão de referência do material utilizado, conforme a relação estabelecida na equação 2.1.

$$\sigma_{cálculo}(F, Geometria) = \sigma_{material}(Material, T, \dot{F}) \tag{2.1}$$

A tensão calculada é determinada em função da geometria do componente e das condições de contorno (*F*) aplicadas. Utiliza-se com frequência a tensão equivalente de von Mises para a obtenção da tensão equivalente ao tensor de tensões. A tensão de referência utilizada geralmente é a tensão de escoamento obtida em teste de tração uniaxial, obtida em função do material, sua temperatura (*T*) e da velocidade da carga aplicada ($\dot{F}$).

Um novo comparativo utilizando a mecânica da fratura foi proposto para avaliar componentes com defeitos, representado na equação 2.2. O parâmetro calculado X*calc* é função da geometria do componente, das condições de contorno (*F*) e do tamanho da falha (*a*). O parâmetro que caracteriza o material *Xmat* é função do material, da temperatura (*T*), da velocidade de aplicação da carga ($\dot{F}$) e da espessura analisada (*B*).

$$X_{calc}(F, Geometria, a) = X_{mat}(Material, T, \dot{F}, B) \tag{2.2}$$

Na mecânica da fratura linear elástica, o parâmetro X*mat* é calculado através da energia específica de fratura (*G*), proposto por Griffith (1920) e pelo fator de intensidade de tensão (*K*), proposto por Irwin (1957). O primeiro parâmetro é considerado global por envolver o balanço energético da estrutura em sua totalidade e o segundo é considerado um parâmetro local por considerar somente a distribuição de tensões na região da ponta da trinca.

2.2 MÉTODO DOS ELEMENTOS DISCRETOS FORMADO POR BARRAS

Este método consiste em discretizar um meio contínuo através de um arranjo de barras fixas com massas concentradas em suas extremidades, como a que se apresenta na figura 2.1.a. As rigidezes das barras são determinadas por expressões explícitas que relacionam as barras à rigidez do meio que se deseja representar. A discretização temporal da equação de movimento resultante é realizada aplicando-se o método das diferenças finitas centrais, método de integração direta explícita.

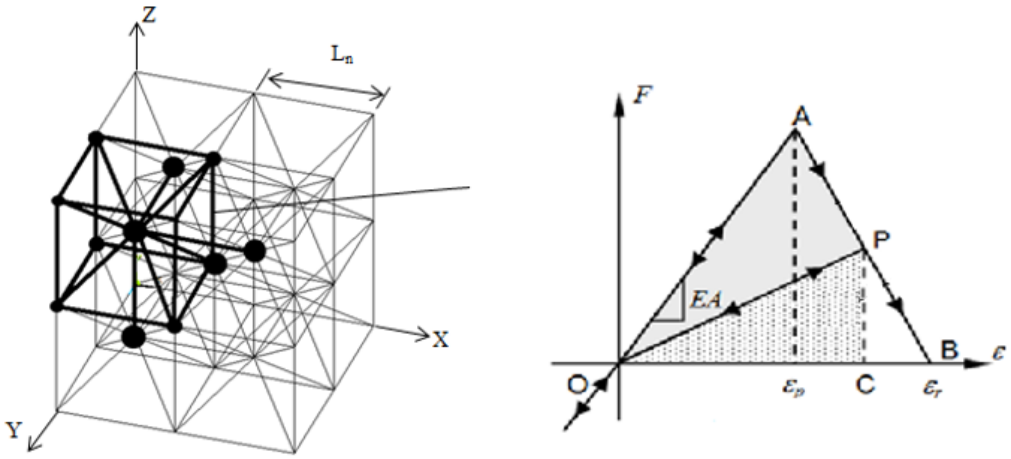


Figura 2.1 a) Arranjo básico em destaque, b) lei constitutiva utilizada em termos de força vs deformação

O dano e a ruptura são computados utilizando-se uma lei bilinear, conforme indicado na figura 2.1.b. Esta lei vincula o limite de deformação elástica às propriedades do material, conforme indicado na equação 2.3.

$$\varepsilon_p = \sqrt{\frac{G_f}{E.d_{eq}}} \tag{2.3}$$

Onde εp é a deformação que indica o limite elástico da barra, *Gf* a energia específica de fratura crítica, *E* o módulo de elasticidade e *deq* um comprimento característico do material. Este último parâmetro se pode interpretar como o tamanho da fissura intrínseca dentro do sólido que se propagaria de forma instável quando a tensão atinge a tensão critica (ε_p*E).

A natureza randômica do material é contemplada considerando εp como um campo randômico. Uma descrição detalhada do modelo, que será utilizado na comparação das aplicações apresentadas, é encontrado em Koteski et. al. (2012).

2.3. PERIDINÂMICA

A teoria da peridinâmica foi proposta por Silling (2000) como um caso particular do método de elementos discretos, tendo como principal diferencial o modo como as interações entre os pontos materiais são realizadas. A conexão entre nós, onde se concentram as massas, se dá a partir de uma esfera de influência de raio δ, denominada horizonte. Tomando como centro um nó de referência e traçando uma esfera que tenha como raio o horizonte, todas os nós dentro da esfera de influência estarão vinculados ao nó de referência, sendo denominados como sua vizinhança, conforme ilustrado na figura 2.3.a. Este procedimento é realizado da mesma forma para todos os nós, construindo a interação entre nós do modelo discretizado.

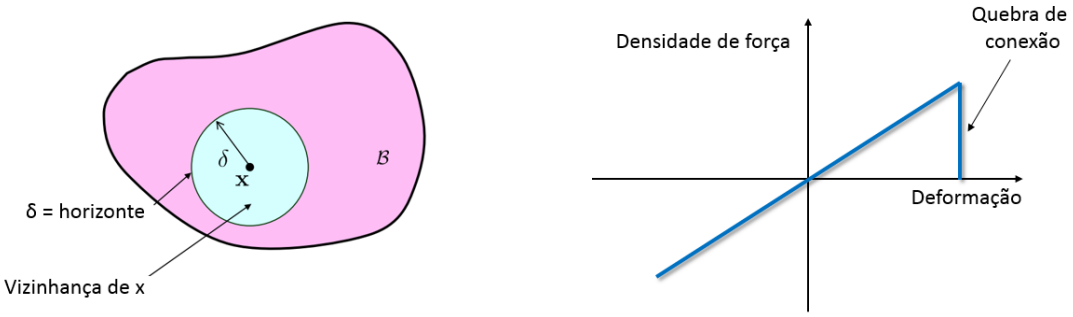


Figura 2.3: (a) Esquema básico para a compreensão da Peridinâmica. (b) Lei constitutiva de cada interação.

A não localidade da teoria da peridinâmica se dá através da inserção do horizonte, que permite que um ponto tenha interação não só com os nós vizinhos imediatos, mas também com os demais pontos da vizinhança, definida pelo horizonte δ. Esta formulação substitui as derivadas espaciais presentes na formulação dos fenômenos físicos por termos integrais, como a equação de equilíbrio de Euler-Lagrange indicada na equação 2.4. Esta substituição é responsável pela habilidade da teoria da peridinâmica em modelar naturalmente componentes com descontinuidades, pois o equacionamento integral é definido em pontos de singularidade, diferentemente do equacionamento diferencial.

$$\int_{H_x} f(q,x)dV_q + b(x,t) = \rho\ddot{u}(x,t) \tag{2.4}$$

Onde ρ representa a densidade do material, $\ddot{u}(x,t)$ aceleração, $f(q,x)$ as forças de pareamento, $b(x,t)$ o campo de forças de corpo e Hx a vizinhança do ponto x.

O método de deformação crítica é utilizado para computar o dano no material. O valor crítico de deformação s_o é estipulado em função do módulo de elasticidade E, energia crítica de fratura G_o e horizonte δ, conforme indicado na equação 2.5. Quando a deformação entre dois pontos ultrapassar o valor crítico de deformação, a conexão entre estes deixa de existir, conforme demonstrado na figura 2.3.b

$$s_o = \sqrt{\frac{5\pi G_o}{9E\delta}} \tag{2.5}$$

Se observa a similaridade entre a expressão (2.3) do método dos elementos discretos formado por barras e a expressão (2.5) da peridinâmica, onde o horizonte seria equivalente ao comprimento característico deq, desta forma fica claro que o comprimento característico e o horizonte são propriedades do material.

3. METODOLOGIA

As análises deste trabalho foram executadas utilizando-se um código modificado baseado no Peridigm, programa de código aberto desenvolvido por Sandia National Laboratories, Parks et. al. (2012), para simulações massivamente-paralelas de multifísica, utilizada primariamente na simulação de falha em sólidos. Programado em C++ com compatibilidade integral ao gerador de malha Cubit e ao pós-processador Paraview, Ayachit (2015).

A customização do código foi realizada no Codeblocks, programa de código aberto para a criação e compilação de códigos C/C++/Fortran. Nele foi possível emular o comportamento isolado das alterações implementadas, permitindo identificar e corrigir erros previamente a integração do código alterado ao código fonte do Peridigm.

Aproveitando-se da escalabilidade do código fonte do Peridigm e dos recursos cedidos pela Escola de Engenharia da Universidade Federal do Rio Grande do Sul, o programa modificado foi compilado no Centro Nacional de Supercomputação (CESUP/UFRGS) em um dos nós de processamento do Cluster SGI Altix (Gauss). Este cluster opera com o Novell SUSE Linux Enterprise e possui 64 nós de processamento. Cada nó disponibiliza dois processadores dodeca-core AMD Opteron 6176 SE de 2.3 GHz em conjunto com 64 Gb de memória RAM e 500 Gb de armazenamento, Meira (2017). As submissões de simulação foram feitas via terminal SSH através do programa gratuito de acesso remoto MobaXterm, www.mobaxterm.mobatek.net acessado em 05/06/2018, desenvolvido por Mobatek. As simulações são realizadas através da submissão de um script, que inicia um *job* que entra em um sistema de filas gerenciado via PBS Pro, onde o tempo de espera é dependente da disponibilidade de recursos.

A compilação efetiva do código permitiu que fosse iniciada a etapa de validação do mesmo. Para tal, foram simulados dois cenários diferentes. O primeiro foi modelado utilizando um gerador de geometrias independente, desenvolvido em linguagem Fortran. As geometrias foram inseridas via arquivo de texto, contendo a coordenada, bloco e volume de cada ponto. O segundo caso, foi modelado utilizando o programa Trelis, Morris (2016), que conta com um sistema de modelamento tridimensional (CAD) simplificado e um gerador de malhas, capaz de exportar arquivos de geometria no formato Exodus, Millis (1988) (extensão g), com compatibilidade integral ao Peridigm, reduzindo drasticamente o pré-processamento quando comparado ao primeiro caso.

As condições de contorno são aplicadas em grupos específicos de pontos, denominados *nodesets*. Estes grupos foram definidos primeiramente utilizando o Microsoft Excel, onde foram aplicados filtros que organizavam os pontos em função de sua localização no espaço, assim foi possível obter a identificação (ID) dos nós desejados, que foram compilados em um arquivo de texto separado para a leitura pelo código. Posteriormente, a utilização do Trelis permitiu a manipulação dos pontos em uma interface gráfica, onde as regras de separação de pontos foram aplicadas de modo rápido e intuitivo. A utilização deste programa eliminou a necessidade da criação de arquivos de texto paralelos para a definição dos *nodesets,* que a partir deste momento estavam embutidos no arquivo de geometria de formato Exodus.

O setup da simulação é realizado através de um arquivo de código específico de extensão *peridigm.* As informações estão dispostas em seções independentes, onde as informações pertinentes a cada seção devem ser inseridas com unidades coerentes às demais. A primeira seção é referente a discretização do modelo, onde são especificados o diretório e o tipo de arquivo de geometria a ser utilizado. O segundo bloco é referente ao material, onde são definidas as propriedades mecânicas e o modelo de material a ser utilizado. Na terceira seção está contida a informação referentes aos modelos de falha, se utilizado. Na quarta seção, são definidos os blocos, onde são definidas as propriedades de material, o modelo de dano e o horizonte. Na quinta seção são definidas as interações entre os diferentes blocos, se aplicável. Na sexta seção, são aplicadas as condições de contornos nos *nodesets* especificados. Na sétima seção estão contidas as informações sobre o *solver* a ser utilizado e o tempo a ser simulado. Finalmente, na oitava seção estão especificadas as variáveis a serem gravadas no arquivo de solução e qual frequência de escrita é desejada.

Os arquivos de geometria e o arquivo de *setup* são enviados ao CESUP via terminal SSH utilizando o MobaXterm e são encaminhados à fila de processamento. Após finalizado, os arquivos de simulação são copiados para uma máquina local e manipulados utilizando o visualizador multi-plataforma Paraview, desenvolvido pela Kitware. O Paraview conta com uma interface robusta para visualização gráfica tridimensional de dados, porém a plotagem de gráficos não se mostrou eficiente. Para contornar este problema, os dados foram exportados em extensão CSV e abertos no Microsoft Excel, onde foram manipulados de forma mais eficiente.

4. IMPLEMENTAÇÕES DA PSEUDOALEATORIEDADE NO PROGRAMA PERIDIGM

A implementação da heterogeneidade do material foi realizada através da substituição de propriedades constantes do material por propriedades que sejam distintas em cada ponto do meio, porém com um comportamento médio que represente com fidelidade o material desejado. Para tal, foi gerado um valor distinto de deformação crítica para cada ponto utilizando a função *std::normal_distribution,* disponível na biblioteca *random* disponível para a linguagem C++. Esta função gera uma série de números de acordo com a distribuição Gaussiana, demonstrada na equação 4.1.

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} . e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.1)$$

Onde, σ representa a média da distribuição Gaussiana e μ o desvio padrão da distribuição Gaussiana.

A pseudoaleatoriedade foi implementada na simulação através da variabilidade da ordem de distribuição dos números que compõem a distribuição normal. A função *std::default_random_engine* foi utilizada para a geração de uma distribuição de números pseudoaleatórios entre 0 e 1 que foram utilizados como argumento na função *std::normal_distribution* para determinar a ordem de distribuição. A função *std::default_random_engine* é denominada geradora de número pseudoaleatórios pois a mesma é dependente de uma entrada do usuário, denominada semente da distribuição, para determinar a ordem com a qual serão gerados os números. Logo, cada valor de semente alimentado pelo usuário resultará em uma distribuição de números em ordem diferente, porém todas distribuições irão resultar em uma distribuição Gaussiana

O código com as alterações para implementação da pseudoaleatoriedade foi compilado no CESUP, de modo que as simulações eram calculadas a partir de um dos nós GAUSS do supercomputador. A submissão de um script via terminal SSH iniciava o pedido de um nó de processamento, que era liberado conforme disponibilidade através de um sistema de filas.

O código fonte do Peridigm é aberto para edição, o que possibilitou a implementação da pseudoaleatoriedade em um dos módulos de cálculo de dano utilizado pelo programa. O módulo de deformação crítica original lê a entrada do usuário de deformação crítica e compara com a deformação de cada um dos pontos em relação aos pontos de sua vizinhança em cada passo de integração de tempo. Se o valor de deformação ultrapassar o valor crítico, a conexão entre os pontos é interrompida de modo irreversível.

A alteração proposta, introduziu valores pseudoaleatórios de deformação crítica, de modo que cada uma das conexões realizadas entre os nós tenham valores distintos de deformação crítica. Para garantir que o comportamento do meio represente o material desejado, os valores pseudoaleatórios são distribuídos conforme uma distribuição gaussiana, onde a média e o desvio padrão são dados de entrada da simulação dependentes do material utilizado.

A introdução da semente de distribuição, dado de entrada da simulação, permite que sejam gerados configurações de ruptura pseduoaleatórias. Cada semente resulta em uma ordem de distribuição distinta dos valores de deslocamento crítico, alterando a localização das conexões mais propícias a falha. Assim, pode-se obter diversas configurações de falha utilizando os mesmos parâmetros de material, condições de contorno e geometria. O módulo de deformação crítica alterado no sistema Peridigm está demonstrado em sua íntegra no apêndice A.

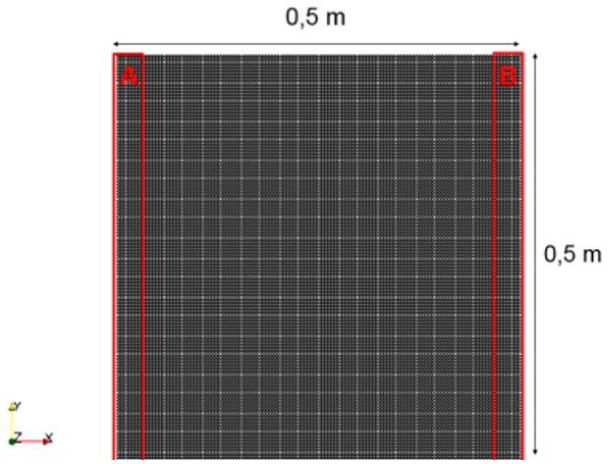
5. APLICAÇÕES

A validação das alterações propostas no código fonte do Peridigm foi feita através da simulação de dois casos onde os resultados eram conhecidos. Assim foi possível realizar a comparação entre os resultados obtidos a partir do código customizado com os resultados

obtidos em ensaios experimentais ou em avaliações com outros métodos de elementos discretos.

5.1 PLACA DE CONCRETO SOB TRAÇÃO

O primeiro caso simulado foi a aplicação de um deslocamento prescrito de tração em uma placa de concreto de 0,5 metros de altura e largura por 0,004 metros de espessura, até sua falha. As condições de contorno impostas foram a restrição de deslocamento nos eixos x e y dos nós em destaque na região A da figura 5.1.a; restrição de deslocamento de todos os nós no eixo z, para simular um estado plano de tensões; aplicação de um deslocamento prescrito com velocidade constante de 8 mm/s no eixo x nos nós em destaque na região B da figura 5.1.a. A placa simulada foi considerada de concreto com módulo de elasticidade de 32 GPa; coeficiente de Poisson de 0,25; densidade de 2400 Kg/m³; horizonte de 11 mm; deformação crítica com média de 4x10^-4 e desvio padrão de 1,5x10^-4.



Módulo de elasticidade	32	GPa
Poisson	0.25	-
Densidade	2400	Kg/m^3
Horizonte	1.10E-02	m
Média de deformação crítica	4.00E-04	-
Desvio padrão de deformação crítica	1.50E-04	-
Número de nós	47.377	-

Figura 5.1: Geometria da placa analisada e resumo dos parâmetros utilizados

Na figura 5.2, o gráfico de tensão versus deformação obtido na simulação da placa de concreto sob tração com o código alterado do Peridigm está comparado com o gráfico obtido com o programa de elementos discretos DEM, proposto por Riera (1984) com a formulação apresentada por Koteski (2012). É possível observar um comportamento muito semelhante entre as duas simulações antes da falha, que ocorre em valores distintos de deformação devido à natureza pseudoaleatória de ambas simulações, onde cada caso simulado resultará em um modo diferente de falha. O módulo de elasticidade resultante da simulação no Peridigm foi cerca de 10% superior ao especificado. Esta diferença foi observada em outros testes e ainda não foi identificado o motivo deste divergência.

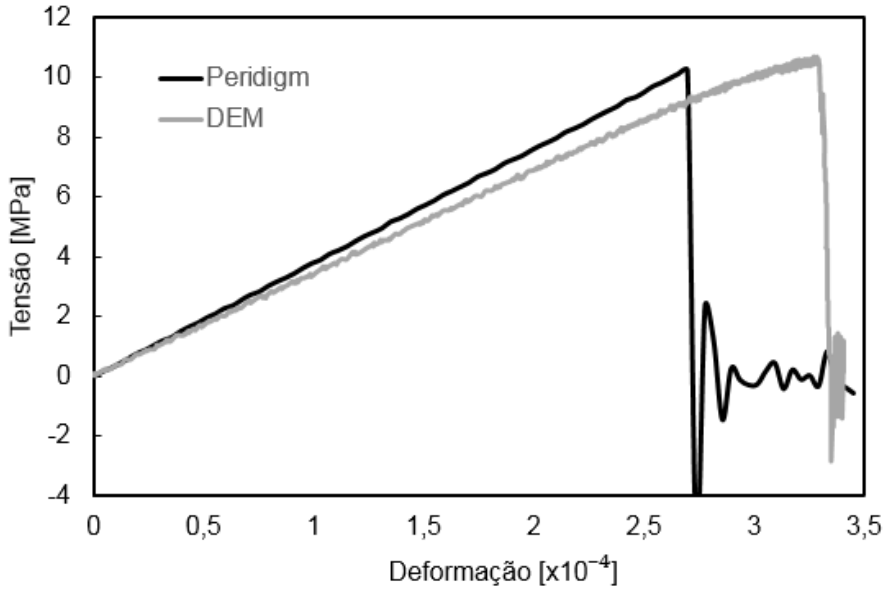


Figura 5.2: Comparação dos gráficos de tensão vs deformação obtidos no Peridigm e DEM

Na figura 5.3 está apresentado o comparativo das configurações finais obtidas com o código modificado do Peridigm e o DEM. Os resultados se mostraram coerentes neste caso também.

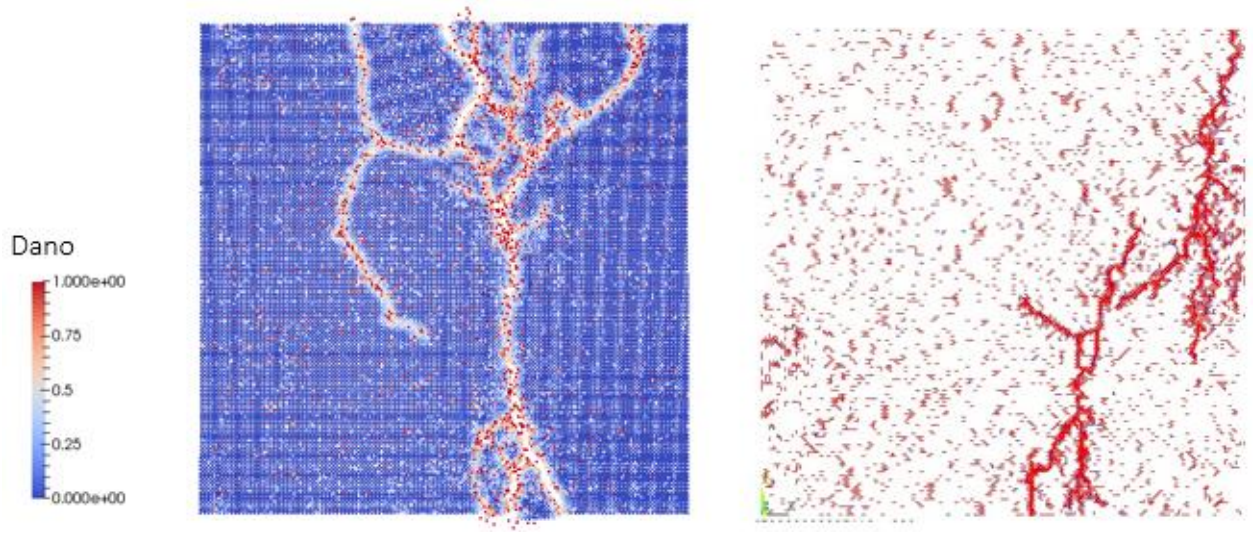


Figura 5.3: Comparação do padrão de trincas previsto pelo a) Peridigm, b) DEM

Na figura 5.4 está representado o trabalho externo aplicado na placa por meio do deslocamento prescrito e a sua decomposição nas energias elástica, cinética e de dano. Nos instantes iniciais, praticamente todo o trabalho aplicado é convertido em energia elástica. Decorridos cerca de 14 milissegundos de simulação, a placa atinge seu limite de deformação e uma trinca se propaga rapidamente levando a placa à falha, onde é possível observar a queda abrupta da energia elástica e crescimento da energia de dano. Após a falha, também é possível observar um aumento na energia cinética, que corresponde ao movimento dos pedaços que se destacaram da placa durante a falha. A energia dissipada no dano foi avaliada através do balanço entre o trabalho externo aplicado na placa e as energias cinética, elástica e de dano. A energia de dano deve ser interpretada como a energia dissipada pelo sistema na forma de dano ou pelo amortecimento que o mesmo possa ter.

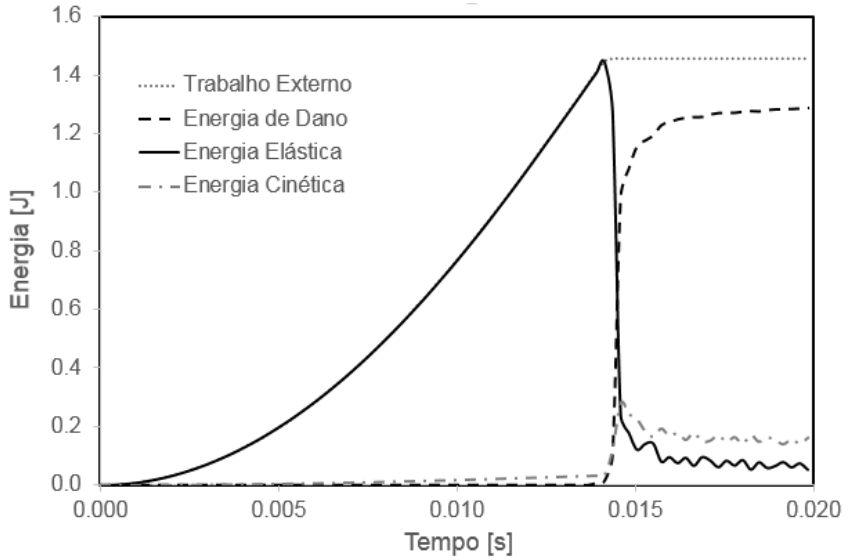


Figura 5.4: Desenvolvimento de energias ao longo do tempo obtidas com o Peridigm

5.2 IMPACTO DE UM MÍSSIL BALÍSTICO EM DISCO DE PMMA

O segundo caso simulado foi o impacto de um projétil de ponta esférica de aço em um disco de polímero fixado em suas extremidades. Os resultados desta simulação foram comparados aos resultados obtidos em testes experimentais e aos resultados obtidos através do DEM, ambos apresentados em Koteski et al. (2015).

O disco de PMMA simulado possui diâmetro externo de 76 mm e 3 mm de espessura, com módulo de elasticidade de 3,3 GPa; coeficiente de Poisson de 0,35 e densidade de 1190 Kg/m³. O projétil utilizado é constituído de aço, com módulo de elasticidade de 200 GPa; coeficiente de Poisson de 0,3; densidade de 7800 Kg/m³.

O disco foi modelado através de uma placa retangular, onde foram engastados os nós em contato com o anel de fixação e os demais nós mantidos livres. Estas condições de contorno foram feitas de modo similar ao DEM, a fim de se obter resultados equivalentes. O projétil teve uma velocidade inicial v_o definida em 1,065 m/s no eixo z; possui uma ponta esferoidal de com diâmetro de 12,7 mm e tem massa de 18,49 Kg. O problema encontra-se esquematizado na figura 5.5.

A média de deformação crítica utilizada foi de 1,328x10^-2 e foi simulado um coeficiente de variação de 25%, logo o desvio padrão foi definido em 3,320x10^-3. O horizonte simulado foi de 3,1x10^-3 m e a distância média entre nós foi de 1x10^-3 m, resultando 25946 pontos.

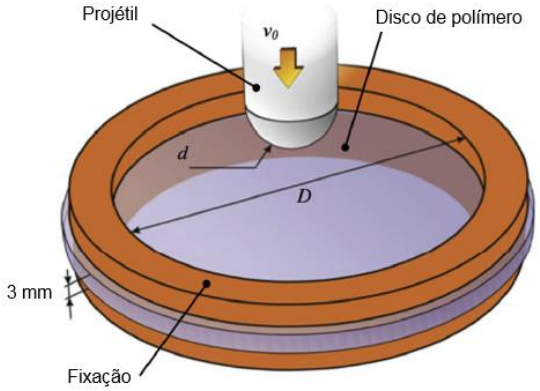


Figura 5.5: Esquematização da geometria do disco e projétil

Os resultados obtidos na simulação do impacto de um míssil balístico em um disco de PMMA utilizando o código alterado do Peridigm foram comparados com resultados numéricos e experimentais publicados por Koteski et al. (2015). Nesta publicação foram apresentados o comparativo da variação da velocidade do projétil em função do tempo e a força de reação do projétil em função do tempo. Os mesmos dados foram computados na simulação do Peridigm e estão apresentados ao lado dos resultados publicados por Kosteski et al. (2015) nas figuras 5.6.e 5.7

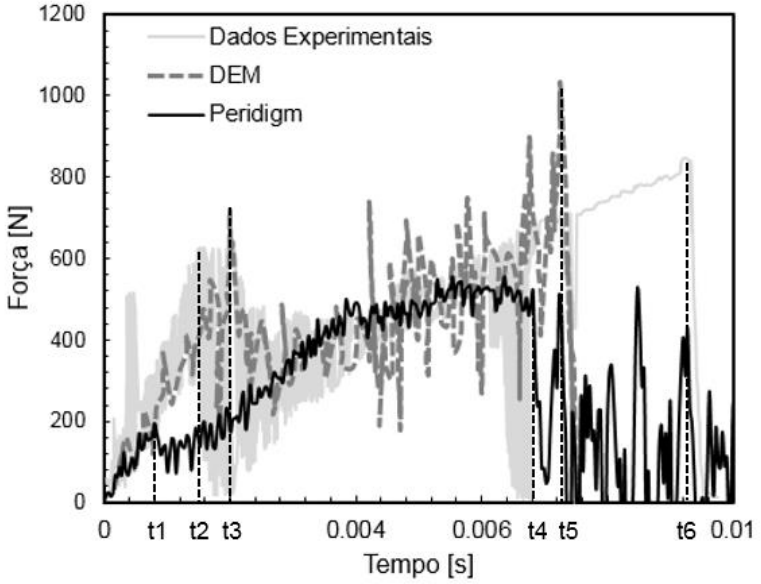


Figura 5.6: Comparação de forças obtidas em testes experimentais, DEM e Peridigm

Na figura 5.6 está apresentado o gráfico de força de reação no suporte do disco em função do tempo de simulação obtidos experimentalmente, com DEM e Peridigm. No teste experimental foi possível observar um pico inicial de força, indicado pelo instante t2, seguido por um decréscimo e posterior ascensão até a falha final no instante t6. Os resultados numéricos conseguiram captar a mesma tendência de comportamento, porém os picos ocorreram em instantes distintos. O primeiro pico foi previsto em t1 pelo Peridigm, ocorrendo um pouco antes ao previsto pelo ensaio experimental. O DEM previu o pico em t3, instantes após ao experimental. No segundo pico, ambas abordagens numéricas obtiveram a falha final em instantes anteriores ao ensaiado, nos instantes t4 e t5, pelo Peridigm e DEM, respectivamente. Estas diferenças se devem à natureza pseudoaleatória utilizada para a representação do material no Peridigm. Para a obtenção de resultados mais próximos aos observados, seria necessária a realização de uma calibração nos parâmetros que definem as características pseudoaleatórias, como o coeficiente de variação e a semente de distribuição. A distância entre nós utilizada na simulação no Peridigm foi em média de 1 mm, enquanto no DEM uma distância de 0.5 mm foi utilizada, o que pode ter influência nas divergências observadas no resultado.

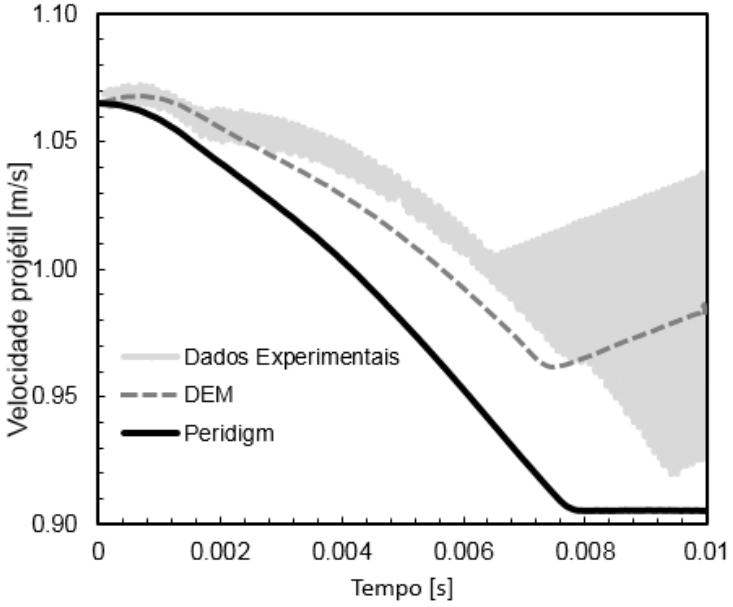


Figura 5.7: Comparação de velocidade do projétil obtida em ensaio experimental, DEM e Peridigm

Na figura 5.7 está apresentado o comparativo de velocidade do projétil obtida em ensaio experimental, DEM e Peridigm. O perfil obtido no Peridigm possui uma desaceleração mais severa do que a observada nos ensaios experimentais e no DEM. Possivelmente se deve ao fato da massa do projétil ter sido concentrada somente na geometria esférica, por meio de incremento de sua densidade, para eliminar a necessidade de modelar um corpo adicional. A diferença das distâncias entre os nós utilizadas no Peridigm e DEM, conforme supracitado, também podem ter influência nesta divergência de resultados.

Os perfis de trincas obtidos com o Peridigm, DEM e em ensaio experimental estão demonstrados na figura 5.8. É possível observar que ambos métodos numéricos foram capazes de representar as trincas radiais e circunferenciais obtidas no ensaio. O número de trincas radiais e o diâmetro das trincas circunferenciais são resultados da calibração dos parâmetros de pseudoaleatoriedade em ambos métodos numéricos. A diferença entre os perfis ocorre devido à calibração destes parâmetros e à característica pseudoaleatória das propriedades de material empregado em ambos métodos.

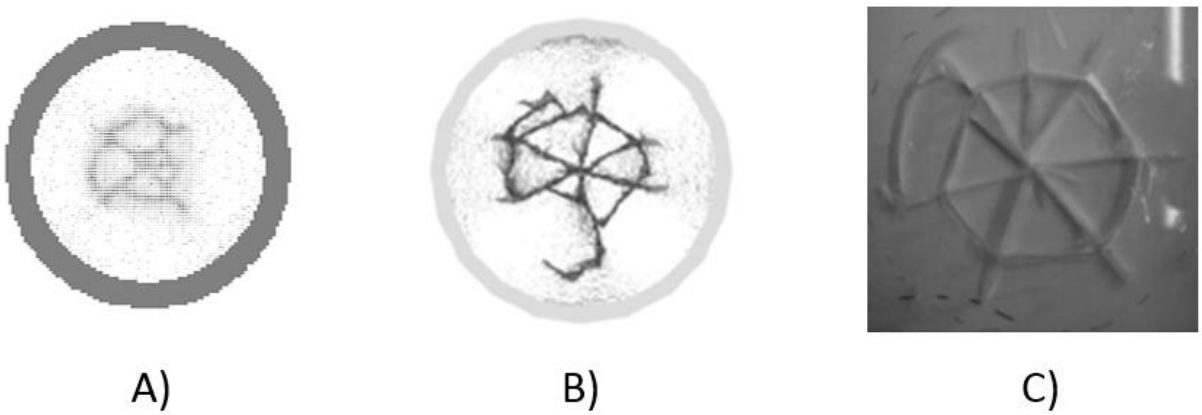


Figura 5.8: Perfis de trincas obtidos com a) Peridigm, b) DEM, c) Testes experimentais

6. CONCLUSÕES

Neste trabalho realizou-se a verificação da validade da implementação da pseudoaleatoriedade de propriedades de materiais na teoria da peridinâmica, uma particularização do método de elementos discretos que permite modelar com naturalidade descontinuidades no meio, como a presença de trincas. Esta modificação tem como objetivo introduzir a natureza aleatória observada na propagação de trincas no ambiente de simulação do Peridigm, programa de código aberto para simulação de peridinâmica.

A validação do novo código implementado no código fonte do Peridigm foi realizada a partir da reprodução de duas situações onde os resultados eram previamente conhecidos. A tração em uma placa de concreto e o impacto em um disco de acrílico. Em ambos casos o sistema foi excitado de forma que levasse a falha.

A partir destas verificações se pôde chegar às seguintes conclusões:

1) A implementação da pseudoaleatoriedade conseguiu introduzir a natureza aleatória que governa a propagação de trincas na teoria da peridinâmica, ao passo que manteve as características de fratura do material.
2) A simulação da placa de concreto sob tração utilizando o código alterado do Peridigm obteve resultados similares aos resultados previstos utilizando o DEM, demonstrando que as características do material e seu comportamento de falha se mantiveram consistentes após a implementação da pseudoaleatoriedade.
3) O balanço de energias obtidos na simulação da placa de concreto se mostrou condizente ao problema simulado
4) A simulação do impacto do projétil em uma placa de polímero utilizando o código alterado do Peridigm obteve resultados similares de desaceleração do projétil e força de reação do engaste, quando comparados aos resultados obtidos em testes experimentais e a partir do DEM, demonstrando a coerência dos resultados. Para uma correlação mais fiel entre os perfil se sugere um trabalho de calibração dos parâmetros que definem a pseudoaleatoriedade do material, coeficiente de variação e semente de distribuição da deformação crítica.
5) A heterogeneidade das propriedades do material possuem papel importante para a variabilidade dos perfis de propagação de trincas, os perfis obtidos em simulação foram coerentes aos obtidos em ensaio experimental, prevendo trincas radiais e circunferenciais. O número de trincas radiais e o diâmetro de trincas circunferenciais são dependentes da heterogeneidade das propriedades do material. Para uma maior correlação entre o observado em ensaio e o simulado, uma calibração dos parâmetros que definem a pseudoaleatoriedade do material, coeficiente de variação e semente de distribuição da deformação crítica.
6) A teoria da peridinâmica se mostrou uma ferramenta robusta para a simulação de componentes submetidos a carregamentos que resultem em sua falha.

REFERÊNCIAS BIBLIOGRÁFICAS

Ayachit, U., The ParaView Guide: A Parallel Visualization Application, Kitware, 2015, ISBN 978-1930934306

Belytschko T., Chen H, Xu J, Zi G. Dynamic crack propagation based on loss of hyperbolicity with a new discontinuous enrichment. Int J Numer Methods Eng 2003;52:1873-905

Chiaia B. et al. Lattice model evaluation of progressive failure in disordered particle composites, Engineering Fracture Mechanics, v. 57, n°. 2/3, p. 301-318, 1997

Gerstle, W., Sau, N. and Silling, S. "Peridynamic Modeling of Concrete Structures," Nuclear Engineering and Design, Vol. 237, No. 12-13, 2007, pp. 1250-1258

Ibrahim A., Mahmood M., “Design of Reinforced Concrete Structures” ISSN ISBN: 624.1834, 2008

Kosteski, L. E.; D’Ambra, R. B.; Iturrioz, I. Crack propagation in elastic solids using the truss-like discrete element method, International Journal of Fracture, vol. 174(2), p. 139–161, 2012

Koteski L E.. Riera JD, Iturrioz I, Singh RK, Kant T. Analysis of reinforced concrete plates subjected to impact employing the truss-like discrete element method. Fatigue Fract Eng Mater Struct 2014

Littlewood D., Parks M., Mitchell J., Silling S.,”The Peridigm Framework for Peridynamic Simulation” 12th U.S. National Congress on Computational Mechanics, SAND2013-5927C, 2013

Koteski L., Iturrioz I., Cisilino A., “A lattice discrete element method to model the falling-weight impact test of PMMA specimens” International Journal of Impact Engineering, Volume 83, p.120-131, 2016

Meira L., “O Cluster SGI Altix: Guia do Usuário”, Centro Nacional de Supercomputação, 2017

Michael P, David L., John M., Stewart S., “Peridigm Users’ Guide v1.0.0”, 2012-7800 Unlimited Release, 2012

Millis W.C., “Exodus: A Finite Element File Format for Pre- and Postprocessing” Sandia Report SAND87 – 2997 UC – 32 Unlimited Release, 1988

Morris R., “Trelis 15.1 – Advanced Meshing For Challenging Simulations” Computational Simulation Software, LLC., American Fork, UT, 2016

Parks M.L., D.J. Littlewood, J.A. Mitchell, and S.A. Silling, Peridigm Users’ Guide, Tech. Report SAND2012-7800, Sandia National Laboratories, 2012

Riera, J.D., Local Effects In Impact Problems In Concrete Structures. Em: proceedings, Conf. on Structural Analysis and Design of Nuclear Power Plants, UFRGS, 1984. Porto Alegre, Rs, Brasil. p. 0-0. 1984

Rinaldi A. E Lai Y.C. Statistical damage theory of 2D lattices: Energetics and physical foundations of damage parameter, International Journal of Plasticity, v. 23, p.1769-1825, 2007

Xu X-P, Needleman A. Numerical simulation of fast crack growth in brittle solids. J Mech Phys Solids 1994;42(9):1397-434

APÊNDICE A

```
#include "Peridigm_CriticalStretchDamageModel.hpp"
#include "Peridigm_Field.hpp"
#include <vector>
#include <stdio.h>
#include <random>

using namespace std;

PeridigmNS::CriticalStretchDamageModel::CriticalStretchDamageModel(const
Teuchos::ParameterList& params)
  : DamageModel(params), m_applyThermalStrains(false), m_modelCoordinatesFieldId(-1),
m_coordinatesFieldId(-1), m_damageFieldId(-1), m_bondDamageFieldId(-1),
m_deltaTemperatureFieldId(-1)
{
  //m_criticalStretch = params.get<double>("Critical Stretch");
    mean = params.get<double>("Critical Stretch Mean");
    stddev = params.get<double>("Critical Stretch Standard Deviation");
    seed = params.get<double>("Critical Stretch Seed");

  if(params.isParameter("Thermal Expansion Coefficient")){
    m_alpha = params.get<double>("Thermal Expansion Coefficient");
    m_applyThermalStrains = true;
  }

  PeridigmNS::FieldManager& fieldManager = PeridigmNS::FieldManager::self();
  m_modelCoordinatesFieldId = fieldManager.getFieldId("Model_Coordinates");
  m_coordinatesFieldId = fieldManager.getFieldId("Coordinates");
  m_damageFieldId = fieldManager.getFieldId(PeridigmNS::PeridigmField::ELEMENT,
PeridigmNS::PeridigmField::SCALAR, PeridigmNS::PeridigmField::TWO_STEP, "Damage");
  m_bondDamageFieldId = fieldManager.getFieldId(PeridigmNS::PeridigmField::BOND,
PeridigmNS::PeridigmField::SCALAR, PeridigmNS::PeridigmField::TWO_STEP,
"Bond_Damage");
  if(m_applyThermalStrains)
    m_deltaTemperatureFieldId = fieldManager.getFieldId(PeridigmField::NODE,
PeridigmField::SCALAR, PeridigmField::TWO_STEP, "Temperature_Change");

  m_fieldIds.push_back(m_modelCoordinatesFieldId);
  m_fieldIds.push_back(m_coordinatesFieldId);
  m_fieldIds.push_back(m_damageFieldId);
  m_fieldIds.push_back(m_bondDamageFieldId);
  if(m_applyThermalStrains)
    m_fieldIds.push_back(m_deltaTemperatureFieldId);
}

PeridigmNS::CriticalStretchDamageModel::~CriticalStretchDamageModel()
{
}

void
PeridigmNS::CriticalStretchDamageModel::initialize(const double dt,
                                                   const int numOwnedPoints,
                                                   const int* ownedIDs,
```

```
                                const int* neighborhoodList,
                                PeridigmNS::DataManager& dataManager) const
{
  double *damage, *bondDamage;
  dataManager.getData(m_damageFieldId,                    PeridigmField::STEP_NP1)-
>ExtractView(&damage);
  dataManager.getData(m_bondDamageFieldId,                PeridigmField::STEP_NP1)-
>ExtractView(&bondDamage);

  // Initialize damage to zero
  int neighborhoodListIndex = 0;
  int bondIndex = 0;
  for(int iID=0 ; iID<numOwnedPoints ; ++iID){
        int nodeID = ownedIDs[iID];
    damage[nodeID] = 0.0;
        int numNeighbors = neighborhoodList[neighborhoodListIndex++];
    neighborhoodListIndex += numNeighbors;
        for(int iNID=0 ; iNID<numNeighbors ; ++iNID){
      bondDamage[bondIndex++] = 0.0;
        }
  }
}

void
PeridigmNS::CriticalStretchDamageModel::computeDamage(const double dt,
                                const int numOwnedPoints,
                                const int* ownedIDs,
                                const int* neighborhoodList,
                                PeridigmNS::DataManager& dataManager) const
{
  double *x, *y, *damage, *bondDamageN, *bondDamageNP1, *deltaTemperature;
  dataManager.getData(m_modelCoordinatesFieldId,          PeridigmField::STEP_NONE)-
>ExtractView(&x);
  dataManager.getData(m_coordinatesFieldId, PeridigmField::STEP_NP1)->ExtractView(&y);
  dataManager.getData(m_damageFieldId,                    PeridigmField::STEP_NP1)-
>ExtractView(&damage);
  dataManager.getData(m_bondDamageFieldId,                  PeridigmField::STEP_N)-
>ExtractView(&bondDamageN);
  dataManager.getData(m_bondDamageFieldId,                PeridigmField::STEP_NP1)-
>ExtractView(&bondDamageNP1);
  deltaTemperature = NULL;
  if(m_applyThermalStrains)
    dataManager.getData(m_deltaTemperatureFieldId,        PeridigmField::STEP_NP1)-
>ExtractView(&deltaTemperature);

  double trialDamage(0.0);
  int neighborhoodListIndex(0), bondIndex(0);
  int nodeId, numNeighbors, neighborID, iID, iNID;
  double nodeInitialX[3], nodeCurrentX[3], initialDistance, currentDistance, relativeExtension,
totalDamage;

  // Set the bond damage to the previous value
  *(dataManager.getData(m_bondDamageFieldId,        PeridigmField::STEP_NP1))       =
*(dataManager.getData(m_bondDamageFieldId, PeridigmField::STEP_N));
```

```
// define vetor m_criticalStretch do tamanho do numero de nos existentes
std::vector<double> m_criticalStretch(numOwnedPoints);
// define sistema gerador de numeros aleatorios dependete da semente
std::default_random_engine generator (seed);
// define distribuicao normal em funcao media e desvio padrao
std::normal_distribution<double> distribution(mean,stddev);

// Update the bond damage
// Break bonds if the extension is greater than the critical extension

for(iID=0 ; iID<numOwnedPoints ; ++iID){
        m_criticalStretch[iID] = distribution(generator);
nodeId = ownedIDs[iID];
        nodeInitialX[0] = x[nodeId*3];
        nodeInitialX[1] = x[nodeId*3+1];
        nodeInitialX[2] = x[nodeId*3+2];
        nodeCurrentX[0] = y[nodeId*3];
        nodeCurrentX[1] = y[nodeId*3+1];
        nodeCurrentX[2] = y[nodeId*3+2];
        numNeighbors = neighborhoodList[neighborhoodListIndex++];
        for(iNID=0 ; iNID<numNeighbors ; ++iNID){
          neighborID = neighborhoodList[neighborhoodListIndex++];
    initialDistance =
      distance(nodeInitialX[0], nodeInitialX[1], nodeInitialX[2],
               x[neighborID*3], x[neighborID*3+1], x[neighborID*3+2]);
    currentDistance =
      distance(nodeCurrentX[0], nodeCurrentX[1], nodeCurrentX[2],
               y[neighborID*3], y[neighborID*3+1], y[neighborID*3+2]);
    if(m_applyThermalStrains)
      currentDistance -= m_alpha*deltaTemperature[nodeId]*initialDistance;
    relativeExtension = (currentDistance - initialDistance)/initialDistance;
    trialDamage = 0.0;
    if(relativeExtension > m_criticalStretch[iID])
      trialDamage = 1.0;
    if(trialDamage > bondDamageNP1[bondIndex]){
      bondDamageNP1[bondIndex] = trialDamage;
    }
    bondIndex += 1;
  }
}

//  Update the element damage (percent of bonds broken)

neighborhoodListIndex = 0;
bondIndex = 0;
for(iID=0 ; iID<numOwnedPoints ; ++iID){
        nodeId = ownedIDs[iID];
        numNeighbors = neighborhoodList[neighborhoodListIndex++];
  neighborhoodListIndex += numNeighbors;
        totalDamage = 0.0;
        for(iNID=0 ; iNID<numNeighbors ; ++iNID){
          totalDamage += bondDamageNP1[bondIndex++];
        }
```

```
        if(numNeighbors > 0)
          totalDamage /= numNeighbors;
        else
          totalDamage = 0.0;
        damage[nodeId] = totalDamage;
    }
}
```