UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

RUDIMAR BAESSO ALTHOF

ANÁLISE DE INCERTEZAS DE USINAS EÓLICAS UTILIZANDO O MÉTODO UNSCENTED TRANSFORM

Joinville
2017

RUDIMAR BAESSO ALTHOF

ANÁLISE DE INCERTEZAS DE USINAS EÓLICAS UTILIZANDO O MÉTODO UNSCENTED TRANSFORM

Trabalho apresentado como requisito para obtenção do título de bacharel no Curso de Graduação em Engenharia Mecatrônica, na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Prof. Dr. Moisés Ferber de Vieira Lessa

Joinville
2017

RUDIMAR BAESSO ALTHOF

ANÁLISE DE INCERTEZAS DE USINAS EÓLICAS UTILIZANDO O MÉTODO UNSCENTED TRANSFORM

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de bacharel em Engenharia Mecatrônica, na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Joinville, 12 de Junho de 2017.

_____

Prof. Dr. Diego Greff
Coordenador do Curso

Banca examinadora:

_____

Prof. Dr. Moisés Ferber de Vieira Lessa
Universidade Federal de Santa Catarina
Orientador

_____

Prof. Dr. Diego Greff
Universidade Federal de Santa Catarina

_____

Prof. Dr. Milton Evangelista de Oliveira Filho
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Aos meus pais Nabor e Maria de Lourdes por terem me apoiado e me incentivado nos estudos, pela sua determinação de trabalhar arduamente na lavoura com o objetivo maior de poder dar estudos aos filhos. Vocês conseguiram!

Ao meu padrinho, Jacob, por ter acompanhado todas as etapas desta caminhada, sempre me aconselhando e vibrando com as minhas realizações.

Às minhas irmãs, Rosiléia e Silvana, por acreditarem no meu potencial.

À minha amada Daniela, pelo seu companheirismo, carinho e compreensão nesta etapa final da graduação.

Ao CNPq pelo apoio financeiro na concessão de bolsa de pesquisa.

Ao meu orientador, Moisés, pela sua paciência e disposição durante o andamento deste trabalho.

Aos professores e à UFSC pela oportunidade de aprendizado proporcionada.

Aos meus amigos de graduação Fredi, Eduardo, Rodrigo, Philipe Leonardo, Janaína e Anna pelos momentos compartilhados durante estes anos.

Ao meu amigo de infância Michel, pela continuação desta amizade mesmo que vivendo longe.

À empresa Perfil Térmico pela oportunidade de estágio e a todos meus colegas e amigos que lá conheci.

A todos familiares e colegas que me apoiaram e incentivaram nos estudos.

RESUMO

Soluções em energias sustentáveis estão sendo cada vez mais discutidas e diversas fontes de energias renováveis vêm sendo utilizadas. Entre elas encontra-se a energia eólica que inicialmente foi usada para bombear água ou triturar grãos. O uso de turbinas eólicas tem como grande vantagem vantagem a baixa emissão de CO_2 durante a sua vida útil. Quanto ao seu projeto, uma característica de sistemas de energia eólica é que seus projetos são altamente dependente de simulações numéricas para o dimensionamento e a obtenção de parâmetros importantes de seu funcionamento. Devido ao fato da velocidade do vento ser um parâmetro incerto e ter um papel crucial na geração de energia, este fator deve ser levado em consideração no momento das simulações para que se obtenha resultados confiáveis. Neste contexto, aplicam-se métodos de quantificação de incertezas com o objetivo de verificar o comportamento do sistema dadas as características dos parâmetros de entrada, como por exemplo as funções densidade de probabilidade. Um método comum na quantificação de incertezas é o método de Monte Carlo, que requer milhares de simulações para produzir resultados precisos. Devido ao grande número de simulações requeridas pelo método de Monte Carlo, tem-se buscado métodos mais eficientes para a quantificação de incertezas. Um destes métodos é o Unscented Transform, utilizado na obtenção de médias e desvios padrão das respostas de sistemas, dados as médias e os desvios padrão das entradas. Neste trabalho, ambos os métodos foram implementados e simulados em um modelo de sistema eólico e seus resultados foram comparados.

Palavras-chave: simulação numérica, análise de incertezas, Unscented Transform, Monte Carlo.

ABSTRACT

Solutions in sustainable energies are being increasingly discussed and several sources of renewable energy are being used. Among them, there is the wind energy that was initially used for pumping water or grinding grains. The use of wind turbines has the advantage of a low emission of CO_2 throughout its entire lifetime. A characteristic of wind energy systems is that their design are highly dependent on numerical simulations for sizing and obtaining important parameters of operation. Due to the fact that wind speed is an uncertain parameter and plays a crucial role in the generation of energy, this factor must be taken into account at the simulations in order to obtain reliable results.
In this scenario, methods of uncertainty quantification are applied in order to verify the behavior of the system given the probability density functions of the input parameters. One commonly used method is the Monte Carlo method, which requires thousands of simulations to produce accurate results. Due to this low convergence rate, more efficient methods have been sought for the quantification of uncertainties. One of these methods is the Unscented Transform, used to obtain means and standard deviations of the system responses, given the means and standard deviations of the inputs. In this work, both methods were implemented and simulated in a wind system model and their results were compared.

Keywords: numerical simulation, uncertainty analysis, Unscented Transform, Monte Carlo.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CO_2	dióxido de carbono
BTU	Unidades Térmicas Britânica
DFIG	gerador de indução duplamente alimentado
EDO	Equação Diferencial Ordinária
IGBT	transistor bipolar de porta isolada
MC	método de Monte Carlo
ME2P	método de estimação de dois pontos
OECD	Organização para a Cooperação e Desenvolvimento Econômico
PDF	função densidade de probabilidade
UT	Unscented Transform

SUMÁRIO

1 INTRODUÇÃO

Com base nas políticas atuais, tecnologias e tendências demográficas, espera-se um aumento de 48% no consumo de energia elétrica mundial, partindo de 549 quadrilhões de Unidades Térmicas Britânicas (BTUs) em 2012 para 815 quadrilhões em 2040. Segundo Conti et al. (2016), uma grande parte deste crescimento na demanda de energia elétrica ocorre entre os países em desenvolvimento não participantes da Organização para a Cooperação e Desenvolvimento Econômico (OECD). O consumo de energia elétrica cresce juntamente com o aumento salarial das famílias pobres, que consequentemente estimula a compra de produtos elétricos, gerando a necessidade de conexão com a rede elétrica. Além do consumo destes bens, o consumo da sua manufatura também contribui em uma grande parte da demanda desta energia. (WOLFRAM; SHELEF; GERTLER, 2012)

A população mundial está cada vez mais preocupada com relação à degradação ambiental e suas causas a curto e longo prazos, tais como aquecimento global, degradação da camada de ozônio, emissões radioativas e poluição da água e do ar. Neste contexto, soluções sustentáveis estão atraindo a atenção da indústria de energia a décadas. Dincer (2000) afirma que os consumidores têm adotado a responsabilidade pela poluição causada pela geração de energia, gerando o interesse no uso de fontes de energias renováveis e suas tecnologias. Existem várias fontes de energias renováveis: solar, hídrica, biomassa, geotérmica, marés, eólica, entre outras. Este trabalho foca na energia eólica, mas a ideia apresentada pode ser utilizada para qualquer outro sistema de energias renováveis.

A energia eólica já vem sendo utilizada por no mínimo 3000 anos por moinhos de vento para a trituração de grãos e bombeamento de água. Se considerarmos os navios movidos à vela, esta fonte de energia tem sido utilizada por um tempo ainda maior. (BURTON et al., 2001).

O baixo nível de emissões de CO_2 durante a vida útil de uma turbina eólica é a principal razão pelo seu uso (BURTON et al., 2001). Além disso, a energia eólica tem benefícios econômicos, pois reduz a dependência de combustíveis fósseis e sua volatilidade nos preços, especialmente para países compradores de combustível que importam de áreas politicamente instáveis. (KROHN; MORTHORST; AWERBUCH, 2009)

A análise e o projeto de sistemas de energia eólica são fortemente dependentes de simulações numéricas. Estas simulações permitem o estudo de características elétricas cruciais de um dado modelo a um custo relativamente baixo, visto que não se se faz necessário o uso de protótipos físicos nas etapas iniciais do projeto. Existem diferentes categorias de modelos que podem ser simulados, com diferentes objetivos e

técnicas. Por exemplo, em uma simulação do domínio do tempo, o passo de simulação pode ser tão pequeno quanto $5\mu s$ para avaliar efeitos da comutação dos conversores de eletrônica de potência ou tão grandes quanto $500\mu s$ para avaliar o comportamento médio destes conversores. Burton et al. (2001) citam alguns exemplos de características analisadas: fluxo de potência, efeitos de faltas, estabilidade transiente e simulações eletromagnéticas. Estas simulações de sistemas eólicos podem ser feitas por pacotes de software tais como MATLAB/Simulink (GAGNON et al., 2005), PSS/E (SLOOTWEG et al., 2003), e PSCAD/EMTDC (KIM; KIM, 2007; ROBINSON; JOVCIC; JOÓS, 2010).

Estes pacotes de software disponíveis são capazes de simular sistemas de potência em larga escala e fornecer resultados precisos. Porém os sistemas renováveis são altamente afetados por entradas incertas, como por exemplo a velocidade do vento em fazendas eólicas ou incidência solar em painéis solares. Como a velocidade do vento desempenha o papel principal em sistemas de energia eólica, a sua incerteza deve ser levada em consideração a fim de executar uma simulação numérica confiável.

O método mais comum para o tratamento de incertezas é o método de Monte Carlo (MC) que baseia-se em amostragem aleatórias das entradas do modelo e sua simulação, obtendo-se os parâmetros desejados a partir dos resultados gerados.

Diversos autores utilizaram o método de Monte Carlo em soluções de problemas envolvendo energias renováveis. Conti e Raiti (2007) tratam do problema de cálculos de fluxo de carga no estudo de níveis de penetração em sistemas distribuídos com painéis fotovoltaicos. A partir das incertezas de disponibilidade de energia solar e variações de cargas, estes autores utilizam modelos para a previsão da potência ativa produzida pelos sistemas fotovoltaicos e da potência absorvida pelas cargas. Utilizando o método MC e incorporando a distribuição de probabilidades do fluxo de carga, Conti e Raiti (2007) discutem através de um estudo de caso o pico máximo de potência que pode ser instalada na rede de distribuição sem que as limitações de corrente e tensão sejam violadas.

Marmidis, Lazarou e Pyrgioti (2008) aplicam o método Monte Carlo para encontrar uma solução para o posicionamento ótimo de turbinas em um parque eólico, buscando possíveis localizações com os critérios de produção máxima de energia e custo mínimo de instalação. Neste estudo, uma área quadrada foi dividida em 100 locais para possíveis instalações de turbinas e o arranjo com o menor custo por unidade de energia produzida foi encontrado, baseado na seguinte função objetivo:

$$objetivo = \frac{custo}{P_{tot}} \tag{1.1}$$

Na Equação 1.1, P_{tot} representa a potência total produzida e $custo$ é o custo anual de uma fazenda eólica com N turbinas que é dado por:

$$custo = N(\tfrac{2}{3} + \tfrac{1}{3}e^{(-0.00174N^2)}) \tag{1.2}$$

Também utilizando o método de Monte Carlo, Silva et al. (2010) propõem uma metodologia para avaliar os requisitos de reservas de sistemas de geração de energia que possuem grandes dependências de fontes de energias renováveis, utilizando uma abordagem probabilística para estudar índices de performance baseados no planejamento a longo prazo.

No entanto, o método MC tem baixa taxa de convergência e por isso tipicamente requer dezenas ou centenas de milhares de simulações em problemas típicos de engenharia elétrica a fim de produzir resultados precisos.

Este grande número de amostras requeridas pelo método de Monte Carlo tornam-o um método de lenta convergência, consequentemente atraindo a atenção para o estudo de métodos mais eficientes. O método Unscented Transform é um método que vem sendo utilizado como alternativa ao método Monte Carlo e foi criado para superar os inconvenientes associados com técnicas de linearização. Além disso, a facilidade na codificação e a simplicidade do método tem causado o aparecimento de várias extensões deste método. O método UT obtém parâmetros estatísticos de uma variável aleatória de saída provindas de entradas submetidas a um conjunto de transformações não lineares. A etapa mais importante desse método consiste na geração de amostras que possam manter informações suficientes sobre as funções densidades de probabilidade de suas entradas. (AIEN; FOTUHI-FIRUZABAD; AMINIFAR, 2012)

Aien, Rashidinejad e Firuz-Abad (2015) implementaram a aplicação do método Unscented Transform em um modelo híbrido, com geração de energia através de turbinas eólicas e painéis solares, objetivando verificar sua eficiência no estudo probabilístico de fluxo de potência. Os autores consideraram a correlação entre as variáveis e compararam o método UT com o método de Monte Carlo e o método de estimação de dois pontos (ME2P). O método UT apresentou ótimos ganhos em relação ao Monte Carlo e em relação a precisão quando comparado ao método ME2P utilizando entradas correlacionadas.

Também com objetivo de validar a eficiência do método, Oke et al. (2011) utilizaram o método Unscented Transform para estudos de fluxo de potência em sistemas de distribuição com geração energia eólica. O estudo de caso consiste de um sistema com três barramentos e duas variáveis de entrada: a velocidade do vento e a carga do consumidor. A carga foi representada como uma função de probabilidade normal e a variação da velocidade do vento foi representada utilizando uma distribuição Weibull de três parâmetros. Os resultados das simulações foram comparados com os resultados gerados pelo método de Monte Carlo com 5000 simulações e mostram ótima precisão.

Dupré et al. (2014) utilizam uma abordagem alternativa do método para o estudo de compatibilidade eletromagnética, chamada de Unscented Transform adaptativa, possibilitando uma maior taxa de convergência. Esta abordagem consiste na classificação dos parâmetros de entrada de acordo com sua influência na variável de saída.

Aplica-se o método UT primeiramente na variável mais importante e sucessivamente aumenta-se o número de variáveis até que o critério de convergência seja alcançado.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho propõe a implementação dos métodos de Monte Carlo e Unscented Transform e suas aplicações em um modelo de fazenda eólica, comparando a eficiência entre os métodos.

1.1.2 Objetivos Específicos

- Compreender a teoria do método de Monte Carlo;
- Implementar, em linguagem MATLAB, o método de Monte Carlo;
- Validar o método de Monte Carlo;
- Compreender a teoria do método Unscented Transform;
- Implementar, em linguagem MATLAB, o método Unscented Transform;
- Comparar os métodos em um modelo de parque eólico relevante.

2 DEFINIÇÃO DO PROBLEMA

Considere um sistema de energia eólica arbitrário. Este sistema inclui subsistemas de geração, transmissão e distribuição que podem ser modelados utilizando componentes lineares e não lineares, tais como: resistores, indutores, capacitores, MOSFET's, geradores e fontes controladas de corrente e tensão. Este modelo é essencialmente um sistema não-linear de equações diferenciais ordinárias (EDOs).

O problema considerado neste trabalho consiste em determinar a média (μ) e o desvio padrão (σ) de qualquer corrente ou tensão do modelo, dados μ e σ de cada um dos parâmetros incertos.

Os parâmetros incertos de um sistema, tais como a velocidade do vento, a capacitância das linhas de transmissão por unidade de comprimento e o nível da falta, são representados pelo vetor de variáveis aleatórias $\vec{X} = [X_1, X_2, ..., X_{N-1}, X_N]$, com dimensão N. Cada um destes parâmetros pode possui uma distribuição estatística diferente, dependentes ou não entre si. Os vetores $\vec{\mu_X} = [\mu_{X_1}, \mu_{X_2}, ..., \mu_{X_{N-1}}, \mu_{X_N}]$ e $\vec{\sigma_X} = [\sigma_{X_1}, \sigma_{X_2}, ..., \sigma_{X_{N-1}}, \sigma_{X_N}]$ são os vetores de médias e desvios padrão de $\vec{X}$, respectivamente. As variáveis de saída, também chamadas de quantidades de interesse, são qualquer tensão ou corrente no sistema de energia eólica e são representadas pelo vetor $\vec{Y} = [Y_1, Y_2, ..., Y_{M-1}, Y_M]$, de dimensão M. Os vetores $\vec{\mu_y} = [\mu_{y_1}, \mu_{y_2}, ..., \mu_{y_{M-1}}, \mu_{y_M}]$ e $\vec{\sigma_y} = [\sigma_{y_1}, \sigma_{y_2}, ..., \sigma_{y_{M-1}}, \sigma_{y_M}]$ são os vetores de médias e desvios padrão de $\vec{Y}$, respectivamente.

O modelo do sistema pode ser descrito como:

$$\vec{Y}(t) = G(\vec{X}, t) \tag{2.1}$$

Onde G representa o sistema simulado. Nota-se que $\vec{Y}(t)$ é um vetor dependente do tempo que possui distribuição estatística gerada a partir das entradas do modelo. Portanto, o problema consiste em obter $\vec{\mu_y}$ e $\vec{\sigma_y}$. Entre os métodos utilizados para este fim estão os métodos de Monte Carlo e Unscented Transform, apresentados a seguir.

3 REVISÃO TEÓRICA

Neste capítulo, presenta-se o equacionamento dos métodos de Monte Carlo e Unscented Transform que serão posteriormente utilizados na quantificação de incertezas do modelo de fazenda eólica escolhido.

3.1 Método de Monte Carlo

Na execução do método de Monte Carlo um grande número de amostras (N_{MC}) é gerado de acordo com a Função Densidade de Probabilidades (PDF) das variáveis de entrada. Para cada amostra uma simulação numérica do modelo considerado é executada produzindo um conjunto de N_{MC} resultados. A partir destes resultados, os valores de interesse tais como a média e os desvios padrão da variável de saída são obtidos.

No método de Monte Carlo o valor esperado da saída é obtido através de uma simples média aritmética. O equacionamento matemático do método para o modelo G, que tem como entrada as variáveis aleatórias $\vec{U}$ com N_{MC} amostras, representadas por $\vec{X}$, pode ser vista na Equação 3.1.

$$\bar{G} = E\left\{G(\bar{U} + \hat{u})\right\} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} G(X_i) \tag{3.1}$$

Na equação acima, $\bar{G}$ é a média da variável de saída sendo analisada, G representa o modelo simulado, E é o operador esperança que associa o valor esperado à variável aleatória de entrada, X_i representa a i-ésima amostra da variável de entrada x, $\bar{U}$ representa a média do parâmetro de entrada e $\hat{u}$ representa o vetor de incertezas da variável de entrada do sistema de média nula e função densidade de probabilidades conhecida. Desta forma, $G(X_i)$ representa a saída do modelo quando o mesmo é avaliado para a i-ésima entrada do sistema e consequentemente $E\left\{G(\bar{U} + \hat{u})\right\}$ representa o valor esperado da variável de saída do sistema dado a média das entradas e suas incertezas.

Após o cálculo da média, obtém-se a variância do parâmetro de saída a partir da Equação 3.2.

$$\sigma_G^2 = E\left\{(G(\bar{U} + \hat{u}) - \bar{G})^2\right\} = \frac{1}{N_{MC} - 1} \sum_{i=1}^{N_{MC}} (G(X_i) - \bar{G})^2 \tag{3.2}$$

O valor σ_G^2 representa a variância do parâmetro observado que é obtido a partir de um somatório do quadrado da diferença entre os valores de cada simulação $(G(X_i)$

) e a média ($\bar{G}$) obtida através da Equação 3.1, divididos pelo número de amostras menos um.

3.1.1 Convergência do método de Monte Carlo

Conforme apresentado por Jarosz (2008) para se saber o quão rápido a estimativa do método de Monte Carlo converge para uma solução suficientemente precisa é necessário determinar a taxa de convergência da variância do estimador. A estimativa de uma função F que é função de X aproximada pelo método de Monte Carlo com N amostras é dada por $\left\langle F^N \right\rangle$ e possui um desvio padrão proporcional a:

$$\sigma[\left\langle F^N \right\rangle] \propto \frac{1}{\sqrt{N}} \tag{3.3}$$

A Equação 3.3 prova a lenta convergência do método, indicando que deve-se quadruplicar o número de amostras a fim de reduzir o erro pela metade. (JAROSZ, 2008)

3.1.2 Vantagens e desvantagens método de Monte Carlo

As vantagens e desvantagens do método de Monte Carlo, apresentadas por Genest (2012) são:

- O método não requer propriedades de diferenciação da função sendo analisada;
- Facilidade de implementação do método, necessitando apenas da geração de amostras aleatórias;
- O estimador não é polarizado;
- O erro na estimativa pode ser controlado pelo Teorema do Limite Central, permitindo a obtenção de um intervalo de confiança;
- A velocidade da convergência é independente da dimensão do problema.

A limitação do método é a sua convergência, visto que muitas simulações devem ser executadas para a obtenção de uma estimativa correta, requerendo elevado tempo de computação.

3.2 Método Unscented Transform

Existem diferentes abordagens para a geração de pontos sigma e pesos das equações. Uma delas, apresentada por Menezes et al. (2008) e utilizada neste trabalho, é apresentada abaixo.

3.2.1 Obtenção dos parâmetros de interesse

Definindo os vetores $\vec{S} = [S_1, S_2, ..., S_{N_S}]$ e $\vec{w} = [w_0, w_1, w_2, ..., w_{N_S}]$ como os vetores de pontos sigma e os pesos, respectivamente, tem-se que a dimensão de cada um destes vetores é dada por N_S. Definido o modelo G, o método para obter $\bar{G}$ a partir das variáveis de entrada $\bar{U}$ apresentado por Menezes et al. (2008) é dado como:

$$\bar{G} = E\left\{G(\bar{U} + \hat{u})\right\} = w_0 G(\bar{U}) + \sum_{i=1}^{N_{SP}} w_i G(\bar{U} + S_i) \tag{3.4}$$

Após o cálculo da média pode-se obter a variância σ_G^2 dada pela equação 3.5.

$$\sigma_G^2 = E\left\{(G(\bar{U} + \hat{u}) - \bar{G})^2\right\} = w_0 (G(\bar{U} + \hat{u}) - \bar{G})^2 + \sum_{i=1}^{N_{SP}} w_i (G(\bar{U} + S_i) - \bar{G})^2 \tag{3.5}$$

A metodologia utilizada para a obtenção dos pontos sigma e dos pesos é apresentada na seção a seguir.

3.2.2 Obtenção dos Pontos Sigma e Pesos

A derivação do método UT é apresentada por Menezes et al. (2008) conforme representado na equação 3.6, soma dos pesos deve ser igual a 1.

$$w_0 + \sum_i \omega_i = 1 \tag{3.6}$$

O cálculo dos pontos sigma é baseado em um conjunto geral de pontos sigma para n_{rv} variáveis aleatórias. O primeiro conjunto de pontos sigma representando as coordenadas das bordas do cubo n_{rv}-dimensional possui $2^{n_{rv}}$ valores que são dados por:

$$(\pm 1, ..., \pm 1)\frac{\sqrt{n_{rv} + 2}}{\sqrt{n_{rv}}} \tag{3.7}$$

O ultimo conjunto de pontos sigma, composto por $2 \cdot n_{rv}$ valores, representando as coordenadas dos eixos do cubo n_{rv}-dimensional são dados por:

$$(\pm 1, ..., 0)\sqrt{n_{rv} + 2} \tag{3.8}$$

De acordo com as equações 3.7 e 3.8, o número de pontos sigma fornecidos por este esquema é $2^{n_{rv}} + 2 \cdot n_{rv}$. Os pesos são diferentes para o primeiro e o segundo conjunto de pontos sigma. Para o primeiro conjunto, com $2^{n_{rv}}$ valores, os pesos são:

$$w_1 = \frac{n_{rv}^2}{2^{n_{rv}}(n_{rv} + 2)^2} \tag{3.9}$$

Para o segundo conjunto de pontos sigma com $2 \cdot n_{rv}$ valores, os pesos são:

$$w_2 = \frac{1}{(n_{rv}+2)^2} \tag{3.10}$$

Com os pesos w_1 e w_2 calculados, é possível obter o valor de w_0 com base na equação 3.6, que produz:

$$w_0 = 1 - (2 \cdot n_{rv} \cdot w_1 + 2^{n_{rv}} \cdot w_2) \tag{3.11}$$

Os pontos sigma gerados por este esquema são normalizados. Desta forma, para obter os valores corretos a serem aplicados na simulação é necessário multiplicar os pontos sigma pelos valores conhecidos de desvio padrão das variáveis de entrada.

Por exemplo, ao considerarmos um caso com duas variáveis aleatórias onde o vetor de entradas é dado por $\vec{\mu} = [\mu_1, \mu_2]$ que possuem desvios padrão $\vec{\sigma} = [\sigma_1, \sigma_2]$ tem-se o número de variáveis aleatórias sendo $n_{rv} = 2$ e portanto utiliza-se $2^{n_{rv}} + 2n_{rv} = 8$ pontos sigma para a realização das simulações. Os pontos sigma para este exemplo são dados por:

$$S_{8,2} = \begin{bmatrix} 1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_1 & 1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_2 \\ 1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_1 & -1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_2 \\ -1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_1 & 1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_2 \\ -1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_1 & -1 \cdot \frac{\sqrt{4}}{\sqrt{2}} \cdot \sigma_2 \\ 1 \cdot \sqrt{4} \cdot \sigma_1 & 0 \\ -1 \cdot \sqrt{4} \cdot \sigma_1 & 0 \\ 0 & 1 \cdot \sqrt{4} \cdot \sigma_2 \\ 0 & -1 \cdot \sqrt{4} \cdot \sigma_2 \end{bmatrix} \tag{3.12}$$

Com os pontos sigma apresentados na equação 3.12 é possível realizar as 9 simulações (oito utilizando os pontos sigma e uma simulações com os valores médios das entradas) necessárias para os cálculos de média e desvios padrão das variáveis de saída. Cada linha da matriz representa um conjunto de entradas para o modelo, desta forma os resultados da primeira simulação seriam dados como $Y_1 = G(S_{1,1}, S_{1,2})$.

A Tabela 1 apresenta os números de pontos sigma e pesos para diferentes números de variáveis de entrada para o método Unscented Transform com aproximação de segunda ordem abordado.

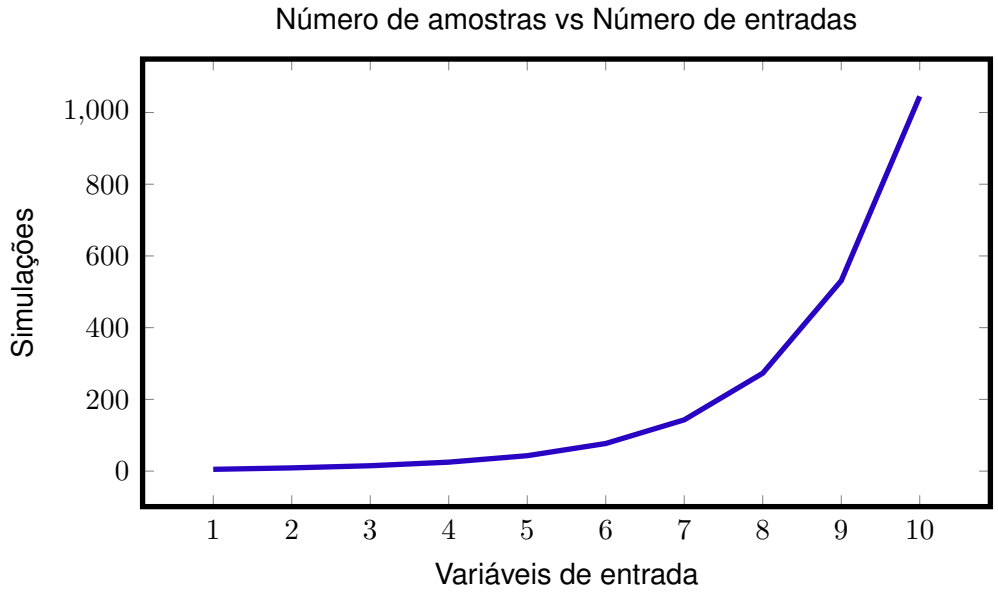
Tabela 1 – Relação entre o número de variáveis aleatórias e parâmetros de simulação.

Variáveis Aleatórias	**Número de Simulações**	**Pesos** 1	2
1	5	1/ 18	1/ 9
2	9	1/ 16	1/ 16
3	15	9/ 200	1/ 25
4	25	1/ 36	1/ 36
5	43	25/1568	1/ 49
6	77	9/1024	1/ 64
7	143	49/10368	1/ 81
8	273	1/400	1/100
9	531	121/61962	1/121
10	1045	1/1042	1/144

Fonte: Menezes et al. (2008)

Observa-se que o crescimento do número de pontos se dá de forma exponencial em relação ao número de entradas. Para facilitar a visualização a Figura 1 apresenta esta informação de forma gráfica. Este número aumenta consideravelmente para aproximações de maiores ordens, reduzindo a eficiência do método para para um sistema com muitas entradas aleatórias e aproximações de maior ordem.

Figura 1 – Relação entre o número de variáveis aleatórias e o número de simulações.



Fonte: Autor (2017)

4 ESTUDO DE CASO

Neste capítulo, apresenta-se o modelo utilizado nas simulações e a função densidade de probabilidade da variável de entrada é apresentada.

4.1 Simulink/Simscape

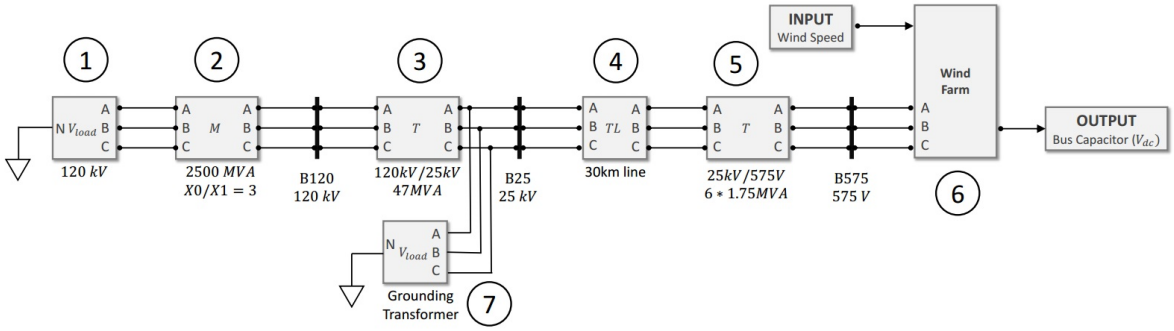
O pacote Simulink é um ambiente de diagramas para simulações de múltiplos domínios e de projetos baseados em modelo, pertencente ao software MATLAB. Este ambiente permite a construção de modelo, simulação, análise dos dados, gerenciamento de projeto e conexão com hardware. O Simscape é a ferramenta que permite a criação de modelos físicos dentro do ambiente Simulink através de diagramas de blocos. (MATHWORKS, 2017)

4.2 Modelo de Fazenda Eólica

O modelo utilizado neste trabalho consiste de um exemplo do Simscape Power Systems que reproduz uma fazenda eólica de 9MW. Conforme descrito em Mathworks (2016), as características deste modelo, que pode ser observado na Figura 2, são:

- Seis turbinas eólicas de 1.5MW;
- O parque eólico é conectado a um sistema de distribuição de 25 kV e distribui potência para uma rede de 120 kV através de uma linha de distribuição de 30 km;
- As turbinas utilizam um gerador de indução duplamente alimentado (Doubly fed induction generator, DFIG) composto por um gerador de indução com rotor bobinado e um conversor AC/DC/AC PWM baseado em transistores bipolares de porta isolada (IGBTs);
- Um controlador de torque mantém a velocidade do rotor a 1.2 por unidade (p.u.);
- A potência reativa é regulada a 0 MVar;
- Uma falta com uma queda de tensão de 0.5 p.u. é simulada no tempo $t = 0.03s$;

Figura 2 – Modelo de usina eólica consistindo de: (1) sistema de 120kV com falha no tempo t=0.03s, (2) impedância trifásica com acoplamento mútuo entre fases, (3) transformador 120/25 kV, (4) linhas de transmissão de 30 km, (5) sistema de distribuição de 25 kV, (6) turbina eólica DFIG e (7) transformador fornecendo um neutro para o sistema trifásico.



Fonte: Mathworks (2016)

4.2.1 Obtenção das Condições Iniciais

No modelo original da fazenda eólica a velocidade do vento é mantida em 15 m/s e a falta na carga ocorre no momento em que o sistema de transmissão encontra-se em regime permanente. Antes de simular o sistema com qualquer modificação dos parâmetros, como por exemplo a velocidade do vento, é necessário obter-se os novos estados dos sistema para estas condições. As etapas executadas para configurar a simulação a fim de obter os estados iniciais são dadas por:

1. Modificar os parâmetros desejados para a nova simulação;
2. Desativar o parâmetro *LoadInitialState* da simulação;
3. Desabilitar o degrau de tensão aplicado na fonte de tensão trifásica de 120 kV que representa a carga, configurando o parâmetro *VariationEntity* para *none*;
4. Dividir a inércia das turbinas por 10, temporariamente, para diminuir o tempo de simulação;
5. Modificar o tempo de simulação para 5 segundos;
6. Mudar o modo de simulação de *Normal* para *Accelerator*;
7. Ativar o parâmetro de simulação *SaveFinalState*;
8. Rodar a simulação e salvar os estados obtidos no seu término;

Neste ponto, os estados salvos representam os estados iniciais da simulação com os novos parâmetros e a simulação da falta pode ser executada. Para isto, é necessário restaurar as configurações do sistema a partir das seguintes modificações:

9. Ativar o parâmetro *LoadInitialState* da simulação;
10. Carregar os estados salvos no passo 8 como estados iniciais;
11. Restabelecer a inércia das turbinas para os seus valores originais;
12. Habilitar o degrau de tensão na fonte de tensão trifásica de 120 k,V modificando o parâmetro *VariationEntity* para *Amplitude*;
13. Definir o parâmetro *Amplitudes* da carga de 120 kV como [1 0.5 1.0] e o parâmetro *TimeValues* como [0 0.03 0.13]. Isto irá configurar a queda de tensão de 0.5 p.u. no tempo $t = 0.03s$ e sua recuperação para 1 p.u. no tempo $t = 0.13s$;
14. Mudar o tempo de parada da simulação para 0.2s e seu modo novamente como *Normal*;
15. Salvar o modelo e executar a simulação.

Conforme citado anteriormente, estas etapas são repetidas a cada nova simulação. O tempo de execução de uma simulação completa é de aproximadamente 38.8s, levando em consideração o tempo de obtenção dos estados iniciais e da simulação da falta. Este e outros detalhes são apresentados na Tabela 2.

Tabela 2 – Detalhes da simulação.

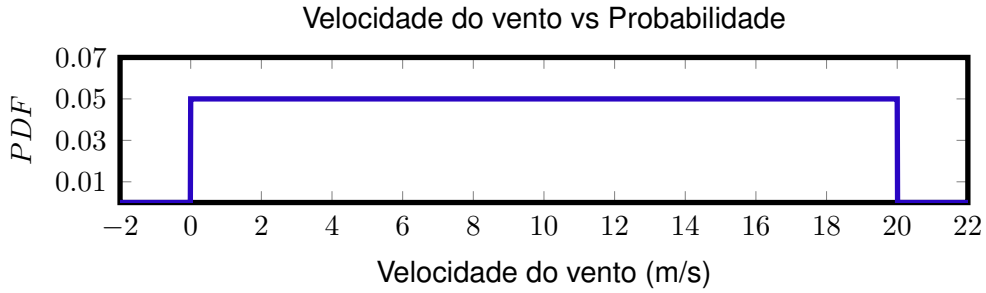
Parâmetro	**Valor**
Solver	Passo fixo, Discreto
Passo	$5\mu s$
Tempo de uma simulação	38.8s
Computador utilizado	Intel® Core^{TM} i7-4700MQ CPU @ 2.40GHz

Fonte: Autor (2017)

4.3 Parâmetro de simulação

A variável aleatória considerada neste trabalho foi a velocidade do vento. Conforme a função densidade de probabilidade apresentada na Figura 3, a velocidade do vento foi considerada uniforme, variando entre 0 e $20m/s$, com média $\mu = 10m/s$ e desvio padrão $\sigma = 5.75m/s$.

Figura 3 – Função densidade de probabilidade da velocidade do vento, uniforme, de média μ = 10m/s e desvio padrão $\sigma = 5,7735m/s$.



Fonte: Autor (2017)

Conforme os dados de entrada apresentados, esta simulação possui vetores de entradas de dimensão unitária, sendo eles:

Variável Aleatória: $\vec{U}$ = [Velocidade do Vento]

Vetor de Médias: $\bar{U}$=[Média velocidade do vento]

Vetor de Desvios Padrões: $\vec{\sigma_u}$=[Desvio padrão da velocidade do vento]

5 SIMULAÇÃO

Este capítulo apresenta os algoritmos criados para possibilitar a implementação dos métodos de Monte Carlo e Unscented Transform.

5.1 Implementação do método de Monte Carlo

As etapas descrevendo a sequência de execução do método de Monte Carlo são apresentadas abaixo. As etapas 1, 2 e 3 consistem na caracterização do modelo. O método de Monte Carlo propriamente dito consiste das etapas 4, 5, 6, 7 e 8 na qual o método é aplicado e os resultados são obtidos. A etapa 9 consiste na utilização dos dados para os seus devidos fins, tais como comparação com outros métodos e análises do sistema simulado. A implementação é descrita pelos seguintes passos:

1. Carregar o modelo para o sistema;
2. Definir os valores médios e incertezas em porcentagem para cada entrada do sistema;
3. Obter os valores mínimos e máximos para cada entrada do sistema;
4. Simular com o valor nominal;
5. Gerar um número de n_a de amostras para as simulações baseadas nas funções densidade de probabilidades das entradas;
6. Simular as n_a amostras e armazenar os resultados das simulações;
7. Calcular os valores máximos e mínimos de cada etapa de tempo;
8. Obter as médias e desvios padrão para cada etapa de tempo;
9. Interpretar/salvar os dados obtidos.

Os etapas descritas acima foram implementadas em uma rotina principal que utiliza funções importadas de outros arquivos. A seguir, apresenta-se os algoritmos referentes a cada passo apresentado acima. O algoritmo completo pode ser visto no Apêndice A.

O Algoritmo 5.1 apresenta a parte do código referente às três primeiras etapas. A linha 2 carrega do modelo para o espaço de trabalho, a linha 6 define o vetor com as médias das entradas e as linhas 12 e 13 definem os vetores de mínimos e máximos que serão utilizados na geração de amostras aleatórias. respectivamente.

Algoritmo 5.1 – Definição dos parâmetros de simulação para o método MC

```
1  model='WIND_FARM';
2  load_system(model);
3  n_samples_MC=50; %Number of simulations
4  %valores nominais
5  WS = 10;
6  nominal_input_MC = [WS];
7  %Components uncertainty in percentage
8  incWS= 50;
9  %Displaying maximum and minimum values associated with each circuit's component
10
11 disp('Components minimum values:')
12 xmin = [WS-WS*incWS/100]
13
14 disp('Components maximum values:')
15 xmax = [WS+WS*incWS/100]
```

Fonte: Autor (2017)

As etapas 4, 5, 6, 7 e 8 são apresentadas no Algoritmo 5.2. A linha 2 obtém os resultados da simulação com os parâmetros nominais através da função MAIN_INITAL_CONDITIONS (Apêndice C), que obtém as condições iniciais do sistema, simula e retorna o vetor com os resultados.

Algoritmo 5.2 – Aplicação do método MC.

```
1  %Simulation with nominal values
2  [nominal_output_MC, time_vector_MC] = MAIN_INITIAL_CONDITIONS(nominal_input_MC);
3
4  %Monte Carlo simulation
5  [simulation_input_MC,simulation_output_MC]=montecarlo(@MAIN_INITIAL_CONDITIONS, xmin, ...
       xmax, n_samples_MC); %retorna dados de simulacao, saida das simulacoes
6
7
8  elapsed_time_MC=toc %Stops the chronometer
9
10 upper_bound, lower_bound]=worstcase_analisys(simulation_output_MC, n_samples_MC); ...
       %encontrando limites e plotando graficos
11 [mean_vector_MC, std_vector_MC] = statistical_analysis(simulation_output_MC,n_samples_MC);
```

Fonte: Autor (2017)

Finalizando a execução do método, os dados foram salvos em arquivos para posterior interpretação e comparação com outros métodos. O Algoritmo 5.3 apresenta a rotina de códigos que realiza esta etapa.

Algoritmo 5.3 – Armazenamento dos resultados gerados pelo método MC.

```
1  n_outs = length(simulation_output_MC{1}(1,:));
2  filename = 'outputMC.mat';
3  save(filename,'n_samples_MC');
4  save(filename,'n_outs', '-append');
5  save(filename,'simulation_output_MC','-append')
6  save(filename,'nominal_output_MC','-append')
```

```
save(filename,'nominal_input_MC', '-append')
save(filename,'simulation_input_MC', '-append')
save(filename,'std_vector_MC','-append')
save(filename,'mean_vector_MC','-append')
save(filename,'time_vector_MC','-append')
save(filename,'elapsed_time_MC','-append')
save(filename,'upper_bound','-append')
save(filename,'lower_bound','-append')
```

Fonte: Autor (2017)

Realizaram-se 1000 simulações com o método de Monte Carlo que foram divididas em 20 lotes de 50 simulações. O processo para dividir as simulações em lotes e reunir os dados é brevemente explicado abaixo.

5.1.0.1 Execução do método de Monte Carlo em lotes

Com o objetivo de permitir e execução do método em etapas ou até mesmo adicionar simulações no conjunto de dados previamente obtidos, incrementando o número de simulações na qual as quantidades de interesse tais como média e desvio padrão são obtidos, modificou-se os códigos para salvar e carregar resultados de múltiplos arquivos.

Desta forma, esta nova implementação permite que o método de Monte Carlo seja executado por etapas. Por exemplo, pode-se separar uma longa simulação que durariam 18 horas em 3 etapas de 6 horas enquanto o computador esteja ocioso.

Este algoritmo equivale ao algoritmo anterior, com uma diferença: ao invés de fazer a análise de dados salva-se os resultados da simulação em arquivos que equivalem ao número da fornada. A Figura 4 apresenta os arquivos criados ao executar este arquivo 5 vezes consecutivas.

Figura 4 – Arquivos gerados por 5 lotes.

Fonte: Autor (2017).

Os algoritmo utilizado para a simulação do método de Monte Carlo em etapas pode ser visto no Apêndice H e o algoritmo utilizado para realizar a junção destes dados é apresentado no Apêndice I.

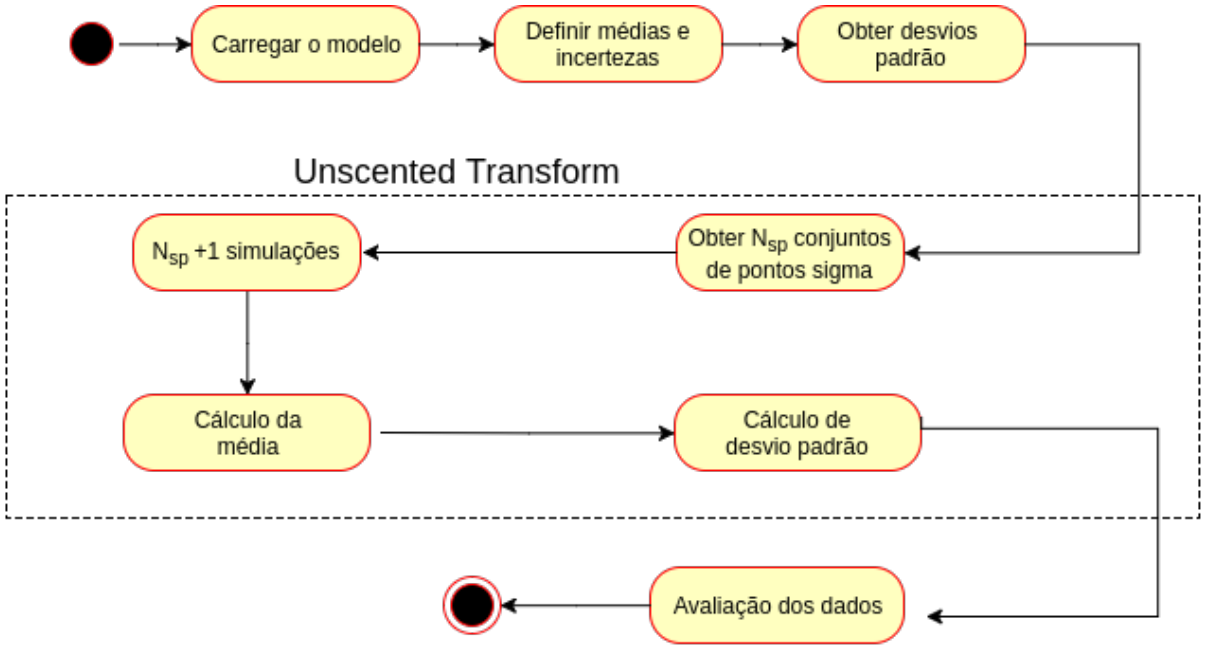
5.2 Implementação do método Unscented Transform

A implementação do método Unscented Transform seguem os mesmos processos apresentados no método de Monte Carlo: caracterização dos sistema, simulação e análise dos resultados. A implementação deste método foi desenvolvida conforme os passos abaixo:

1. Carregar o modelo para o sistema;
2. Definir os valores médios e incertezas para cada entrada do sistema;
3. Obter os desvios padrões das entradas do sistema;
4. Obter os pontos sigma que serão utilizados como entrada nas simulações do modelo;
5. Realizar uma simulação para cada conjunto de pontos sigma gerados e para o valor nominal das entradas;
6. Calcular as médias a partir da soma ponderada dos resultados de cada simulação efetuada;
7. Calcular os desvios padrão a partir das médias obtidas na etapa anterior e dos resultados das simulações;
8. Interpretar/salvar os dados obtidos.

Similar ao método de Monte Carlo, as etapas 1, 2 e 3 consistem da caracterização do problema, definindo os parâmetros do modelo. O método Unscented Transform consiste das etapas de número 4, 5, 6 e 7 que implementam os algoritmos de execução do método. A avaliação dos resultados consiste da etapa 8. O diagrama de sequências exemplificando as etapas apresentadas acima pode ser observado na Figura 5.

Figura 5 – Diagrama de sequências para a implementação do método Uncented Transform.



Fonte: Autor (2017)

As etapas 1, 2 e 3 são apresentadas no Algoritmo 5.4. O modelo é carregado para o ambiente de simulação na linhas 1 e 2, as incertezas são definidas nas linhas 5 a 9 e os desvios padrão enquanto que os valores máximos e mínimos da variável de entrada obtidos na linha 12 através da função *standard_deviations* (Apêndice F).

Algoritmo 5.4 – Definição dos parâmetros de simulação para o método UT

```
1  model='WIND_FARM';
2  load_system(model);
3
4  %Nominal Values
5  WS=10; %Wins Speed
6  nominal_values = [WS];
7  %Components uncertainty in percentage
8  incWS = 100;
9  uncertainties = [incWS];
10
11 %Generating the standard deviation according to the distribuition
12 [std_vec, xmin, xmax] = standard_deviations(nominal_values, uncertainties, 'uniform');
```

Fonte: Autor (2017)

Após definidos os parâmetros de simulação, inicia-se o método. O Algoritmo 5.5 apresenta a implementação do método Unscented Transform propriamente dito, representado pelas etapas 4, 5, 6 e 7.

A linha 4 realiza a chamada da função que calcula os pontos sigma (Apêndice D). A linha 7 chama a função *simulate_UT* (Apêndice E) que obtém os resultados das simulações geradas pelos pontos sigma através da função *MAIN_INITIAL_CONDITIONS* passada como argumento. A partir dos resultados obtidos a função *UT_statistical_analisys* (Apêndice F) obtém as médias e desvios padrões para cada passo de tempo a partir das equações apresentadas anteriormente.

Algoritmo 5.5 – Aplicação do método UT.

```
1  %Simulation with nominal values
2  [nominal_output_UT, time_vector_UT] = MAIN_INITIAL_CONDITIONS(nominal_values);
3  %Obtaining the sigma points
4  [sp_vec, w_vec]=sigma_points(std_vec);
5  n_samples = length(sp_vec);
6  %Unscented Transform simulations
7  [simulation_output] = simulate_UT(@MAIN_INITIAL_CONDITIONS, nominal_values, sp_vec);
8  %Statistical analysis
9  [mean_vector_UT, std_vector_UT] = ...
       UT_statistical_analysis(nominal_output_UT,simulation_output,w_vec,sp_vec);
10 std_vector_UT = sqrt(std_vector_UT);
```

Fonte: Autor (2017)

Finalizando a implementação do método, a etapa 8 realiza o armazenamento dos dados para posterior comparação e é apresentada no Algoritmo 5.6.

Algoritmo 5.6 – Armazenamento dos resultados gerados pelo método UT.

```
1  %Saving data for later comparison
2  save('outputUT.mat','nominal_output_UT')
3  save('outputUT.mat','mean_vector_UT','-append')
4  save('outputUT.mat','time_vector_UT','-append')
5  save('outputUT.mat','elapsed_time_UT','-append')
6  save('outputUT.mat','std_vector_UT','-append')
```

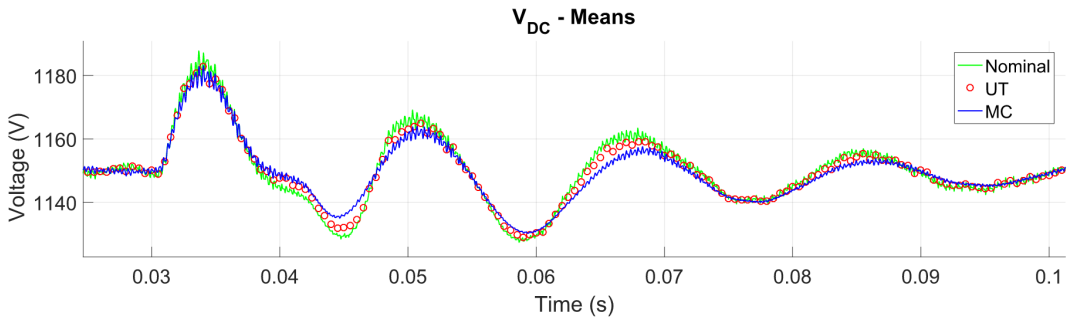
Fonte: Autor (2017)

O algoritmo completo que implementa a rotina principal do método Unscented Transform pode ser visto no Apêndice B.

6 RESULTADOS

A comparação entre os métodos (Apêndice G) consiste na apresentação gráfica dos resultados gerados pelos dois métodos, possibilitando a sua inspeção visual. O primeiro gráfico apresenta as médias obtidas através de cada método, para cada passo de tempo, juntamente com o valor nominal. O parâmetro de saída analisado por esta simulação foi a tensão no capacitor do barramento CC do conversor de potência. A Figura 6 apresenta a tensão obtida no capacitor para a simulação com os valores nominais e as tensões médias obtidas para cada etapa para ambos os métodos Monte Carlo e Unscented Transform.

Figura 6 – Médias obtidas através dos métods Monte Carlo e Unscented Transform. Voltagem nominal: 1150V; Valor máximo: $\simeq$ 1185V; Valor mínimo: $\simeq$ 1130V.



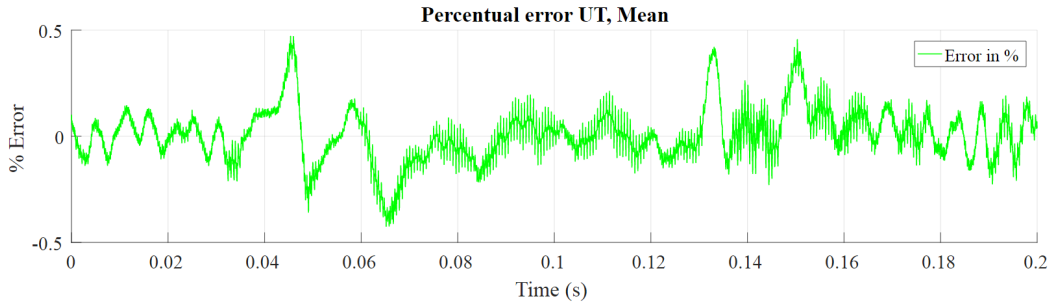
Fonte: Autor (2017)

Os erros percentuais dos valores médios obtidos através do método Unscented Transform ($\mu_{UT}(t)$) em comparação com os valores obtidos pelo método de Monte Carlo ($\mu_{MC}(t)$) foram obtidos a partir da seguinte equação:

$$e_{\mu}(t) = \frac{\mu_{MC}(t) - \mu_{UT}(t)}{\mu_{MC}(t)} \tag{6.1}$$

Os resultados são apresentados na Figura 7, onde observa-se que o maior erro obtido é de aproximadamente 0,5%.

Figura 7 – Erros percentuais do método Unscented Transform para as médias obtidas, em relação ao método Monte Carlo com 1000 amostras.



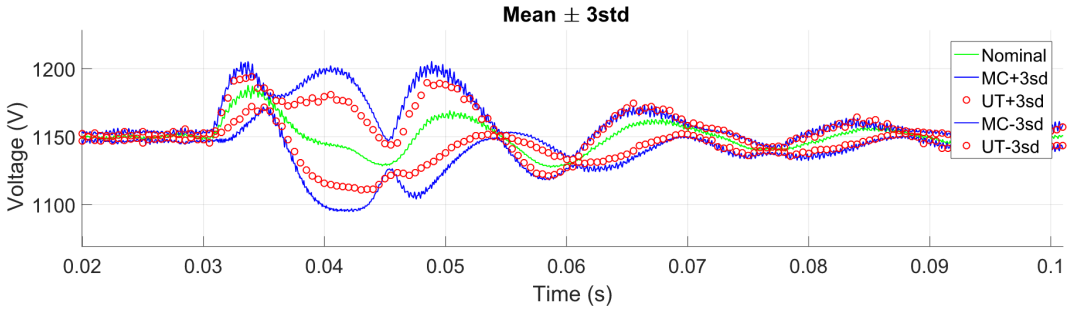
Fonte: Autor (2017)

Após a obtenção das médias, obteve-se os desvios padrão para os métodos Monte Carlo e Unscented Transform. A partir dos desvios padrão obteve-se os valores máximos e mínimos de tensões atingidas pelo capacitor para qualquer velocidade de vento no intervalo especificado com 0,01% de confiança, calculado conforme as seguintes equações:

$$\begin{aligned} max_{MC}(t) &= \mu_{MC}(t) + 3 \cdot \sigma_{MC}(t) \\ min_{MC}(t) &= \mu_{MC}(t) - 3 \cdot \sigma_{MC}(t) \\ max_{UT}(t) &= \mu_{UT}(t) + 3 \cdot \sigma_{UT}(t) \\ min_{UT}(t) &= \mu_{UT}(t) - 3 \cdot \sigma_{UT}(t) \end{aligned} \tag{6.2}$$

No qual $max_{MC}(t)$,$min_{MC}(t)$,$max_{UT}(t)$ e $min_{UT}(t)$ representam os valores máximos e mínimos de tensão para cada passo de simulação para os métodos Monte Carlo e Unscented Transform, respectivamente. Os valores μ_{MC} e μ_{UT} são as médias obtidas para cada passo e os valores σ_{MC} e σ_{UT} representam os desvios padrão, também para cada passo da simulação. O resultado desta comparação é apresentado na Figura 8.

Figura 8 – Limites máximos e mínimos com 99% de confiança. $1100V < V_{DC} < 1200V$.

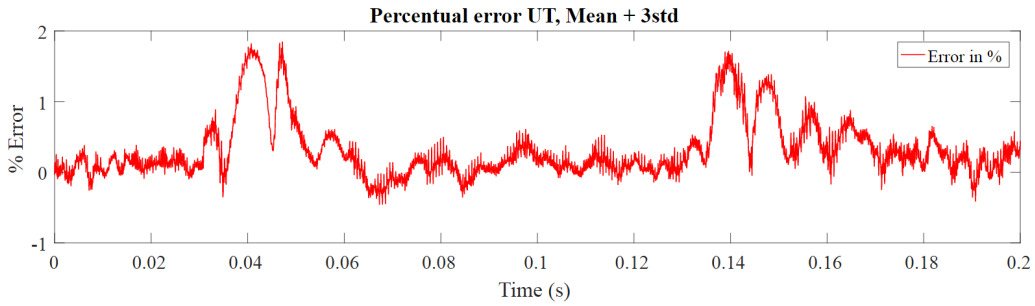


Fonte: Autor (2017)

Observa-se que o método Unscented Transform apresentou resultados similares ao método de Monte Carlo na maioria das etapas de simulação, exceto nos passos de simulação próximos aos instantes de tempo t=0.04s. Isto ocorre pois o método UT, apresentado por Menezes et al. (2008) no qual esta implementação se baseia, aproxima o sistema a equações de segunda ordem. Portanto, melhores resultados seriam obtidos na utilização do método Unscented Transform com maior ordem.

Os erros percentuais também foram obtidos para os limites superiores de tensão calculados e são apresentados na Figura 9. Observa-se que o erro máximo obtido para estes valores é e de aproximadamente 1.8%.

Figura 9 – Erros percentuais do método Unscented Transform dos limites superiores com 99% de confiança, em relação ao método Monte Carlo com 1000 amostras.



Fonte: Autor (2017)

Em relação ao tempo de execução de cada método, o método Unscented Transforma apresentou uma redução de aproximadamente 143 vezes em relação ao tempo de execução do método de Monte Carlo, com mil amostras. A Tabela 3 apresenta estas informações.

Tabela 3 – Número de simulações e tempos de execução dos métodos MC e UT.

Método	**Simulações**	**Tempo**
Monte Carlo	1000	6 horas e 36 minutos
Unscented Transform	5	2 minutos 46 segundos

Fonte: Autor (2017)

7 CONCLUSÃO

Este trabalho foi desenvolvido com o objetivo de implementar um método eficiente de quantificação de incertezas, o método Unscented Transform, realizando sua comparação com o método de Monte Carlo. Com o intuito de possibilitar o entendimento deste trabalho, apresentou-se ao leitor a metodologia utilizada na implementação de ambos os métodos, Monte Carlo e Unscented Transform. Ambos os métodos foram implementados no software MATLAB.

Para avaliar a eficiência e precisão do método sugerido utilizou-se um modelo de fazenda eólica fornecida como exemplo do pacote de simulação utilizado. Este modelo representa uma fazenda eólica de 9MW de potência com 6 turbinas de 1.5MW cada, com a ocorrência de uma falta no tempo t=0.03s.

Executou-se mil simulações para o método de Monte Carlo e 5 simulações para o método Unscented Transform. A partir destes resultados calculou-se as médias e os desvios padrão de cada passo de tempo e comparou-se os resultados obtidos, que avaliam o comportamento da tensão no capacitor do barramento de corrente contínua dos conversores de potência.

A partir dos resultados obtidos observou-se que o método Unscented Transform apresentou precisão similar ao método Monte Carlo e redução no tempo de computação de aproximadamente 143 vezes. Adicionalmente, a implementação deste trabalho mostra a importância de considerar a propagação de incertezas na especificação de componentes, visto que pode-se observar tensões superiores acima da tensão nominal do barramento avaliado.

A apresentação deste trabalho estimula a implementação de trabalhos futuros com múltiplas entradas e múltiplas saídas. Embora o método Unscented Transform foi aplicado na análise de um sistema eólico, o mesmo método pode ser utilizado para outras fontes de energias renováveis, bem como qualquer outro modelo matemático, efetuando a análise das médias e desvios padrão das variáveis de saída.

REFERÊNCIAS

AIEN, M.; FOTUHI-FIRUZABAD, M.; AMINIFAR, F. Probabilistic load flow in correlated uncertain environment using unscented transformation. **IEEE Transactions on Power systems**, IEEE, v. 27, n. 4, p. 2233–2241, 2012.

AIEN, M.; RASHIDINEJAD, M.; FIRUZ-ABAD, M. F. Probabilistic optimal power flow in correlated hybrid wind-pv power systems: A review and a new approach. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 41, p. 1437–1446, 2015.

BURTON, T.; SHARPE, D.; JENKINS, N.; BOSSANYI, E. **Wind energy handbook**. [S.l.]: John Wiley & Sons, 2001.

CONTI, J.; HOLTBERG, P.; DIEFENDERFER, J.; LAROSE, A.; TURNURE, J.; WESTFALL, L. Annual energy outlook 2016-with projections to 2040, us energy information administration. **Report Number**, 2016.

CONTI, S.; RAITI, S. Probabilistic load flow using monte carlo techniques for distribution networks with photovoltaic generators. **Solar Energy**, Elsevier, v. 81, n. 12, p. 1473–1481, 2007.

DINCER, I. Renewable energy and sustainable development: a crucial review. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 4, n. 2, p. 157–175, 2000.

DUPRÉ, L.; CREVECOEUR, P. D. G.; FERBER, M.; VOLLAIRE, C.; KRÄHENBÜHL, L.; VASCONCELOS, J. A. Adaptive unscented transform for uncertainty quantification in emc large-scale systems. **COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering**, Emerald Group Publishing Limited, v. 33, n. 3, p. 914–926, 2014.

GAGNON, R.; SYBILLE, G.; BERNARD, S.; PARÉ, D.; CASORIA, S.; LAROSE, C. Modeling and real-time simulation of a doubly-fed induction generator driven by a wind turbine. In: **Proceedings of the International Conference on Power systems Transients (IPST'05)**. [S.l.: s.n.], 2005. p. 19–23.

GENEST, A. G. Introduction to the monte carlo methods. 2012.

JAROSZ, W. **Efficient Monte Carlo Methods for Light Transport in Scattering Media**. Tese (Doutorado) — UC San Diego, September 2008.

KIM, S.-K.; KIM, E.-S. Pscad/emtdc-based modeling and analysis of a gearless variable speed wind turbine. **IEEE Transactions on Energy Conversion**, IEEE, v. 22, n. 2, p. 421–430, 2007.

KROHN, S.; MORTHORST, P.-E.; AWERBUCH, S. **The economics of wind energy**. [S.l.]: EWEA, 2009.

MARMIDIS, G.; LAZAROU, S.; PYRGIOTI, E. Optimal placement of wind turbines in a wind park using monte carlo simulation. **Renewable energy**, Elsevier, v. 33, n. 7, p. 1455–1460, 2008.

MATHWORKS. **Simscape Power Systems**. 2016. <https://www.mathworks.com/help/physmod/sps/ug/product-description.html>.

MATHWORKS. **Simulink - Simulation and Model-Based Design**. [S.l.]: SimPowerSystems, 2017. <https://www.mathworks.com/products/simulink.html>.

MENEZES, L. d.; AJAYI, A.; CHRISTOPOULOS, C.; SEWELL, P.; BORGES, G. A. Efficient computation of stochastic electromagnetic problems using unscented transforms. **IET Science, Measurement & Technology**, v. 2, n. 2, p. 88, 2008.

OKE, O. A.; THOMAS, D. W.; ASHER, G. M.; MENEZES, L. R. de. Probabilistic load flow for distribution systems with wind production using unscented transform method. In: IEEE. **Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES**. [S.l.], 2011. p. 1–7.

ROBINSON, J.; JOVCIC, D.; JOÓS, G. Analysis and design of an offshore wind farm using a mv dc grid. **IEEE Transactions on Power Delivery**, IEEE, v. 25, n. 4, p. 2164–2173, 2010.

SILVA, A. M. L. L. da; SALES, W. S.; MANSO, L. A. da F.; BILLINTON, R. Long-term probabilistic evaluation of operating reserve requirements with renewable sources. **IEEE Transactions on Power Systems**, IEEE, v. 25, n. 1, p. 106–116, 2010.

SLOOTWEG, J.; HAAN, S. D.; POLINDER, H.; KLING, W. General model for representing variable speed wind turbines in power system dynamics simulations. **IEEE Transactions on power systems**, IEEE, v. 18, n. 1, p. 144–151, 2003.

WOLFRAM, C.; SHELEF, O.; GERTLER, P. How will energy demand develop in the developing world? **The Journal of Economic Perspectives**, American Economic Association, v. 26, n. 1, p. 119–137, 2012.

APÊNDICE A – ROTINA PRINCIPAL DO MÉTODO DE MONTE CARLO

```
%This example applies the Monte Carlo method for a wind farm model

model='WIND_FARM';
load_system(model);

n_samples_MC=50; %Number of simulations

%valores nominais
WS = 10;
nominal_input_MC = [WS];

%Components uncertainty in percentage
incWS= 50;

%Displaying maximum and minimum values associated with each circuit's component
format short eng

disp('Components minimum values:')
xmin = [WS-WS*incWS/100]

disp('Components maximum values:')
xmax = [WS+WS*incWS/100]

format %restores original data printing format

%####################### SIMULATING ##########################
tic %Starts chronometer for simulation time measurement

%Simulation with nominal values
[nominal_output_MC, time_vector_MC] = MAIN_INITIAL_CONDITIONS(nominal_input_MC);

%Monte Carlo simulation
[simulation_input_MC,simulation_output_MC]=montecarlo(@MAIN_INITIAL_CONDITIONS, xmin, ...
    xmax, n_samples_MC); %retorna dados de simulacao, saida das simulacoes

elapsed_time_MC=toc %Stops the chronometer

%####################### DATA ANALISYS ##########################
[upper_bound, lower_bound]=worstcase_analisys(simulation_output_MC, n_samples_MC); ...
    %encontrando limites e plotando graficos
[mean_vector_MC, std_vector_MC] = statistical_analysis(simulation_output_MC,n_samples_MC);

%####################### DATA SAVING ##########################
n_outs = length(simulation_output_MC{1}(1,:));
filename = 'outputMC.mat';
save(filename,'n_samples_MC');
save(filename,'n_outs', '-append');
save(filename,'simulation_output_MC','-append')
save(filename,'nominal_output_MC','-append')
```

```
save(filename,'nominal_input_MC', '-append')
save(filename,'simulation_input_MC', '-append')
save(filename,'std_vector_MC','-append')
save(filename,'mean_vector_MC','-append')
save(filename,'time_vector_MC','-append')
save(filename,'elapsed_time_MC','-append')
save(filename,'upper_bound','-append')
save(filename,'lower_bound','-append')

%###################### GRAPHICAL INTERPRETATION ##############
n_outs= 2
titles=['Voltage on the DC bus  ';
        'Rotor speed in per unit'];

plot_labels = ['V_{DC}          ',
               '\omega_r (pu)   '];
%maximum and minimun values
for i=1:n_outs
    disp('Plots');
    ax1=subplot(n_outs,1,i);
    hold all;  grid on;
    plot(time_vector_MC,nominal_output_MC(:,i),'g','linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i),'r','linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i)+3*std_vector_MC(:,i),'b', 'linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i)-3*std_vector_MC(:,i),'c', 'linewidth',1);
    %plot(time_vector_MC,lower_bound(:,i),'k','linewidth',1);
    %plot(time_vector_MC,upper_bound(:,i),'y','linewidth',1);
    xlabel('Time (s)','fontsize',16);
    ylabel(plot_labels(i,:),'fontsize',16);
    set(gca,'fontsize',18);
    %l=legend(ax1,'Nominal-MC', 'Mean-MC', 'Mean+2std','Mean-2std', 'Lower Bound', ...
        'Upper Bound');
    l=legend(ax1,'Nominal-MC', 'Mean-MC', 'Mean+3std','Mean-3std');
    l.Location='northeast';
    title(titles(i,:));
end
```

APÊNDICE B – ROTINA PRINCIPAL DO MÉTODO UNSCENTED TRANSFORM

```
clear

%################################## PARAMETERS ########################################

%This example applies the Unscented Transform method for a WindFarm with the following ...
    uncertainties:
% WindSpeed: 10ms +- 100% [0,20]

model='WIND_FARM';
load_system(model);

%Nominal Values
WS=10; %Wins Speed
nominal_values = [WS];
n_variables=1;

%Components uncertainty in percentage
incWS = 100;
uncertainties = [incWS];

%Displaying maximum and minimum values associated with each circuit's component
format short eng

%Generating the standard deviation according to the distribuition
[std_vec, xmin, xmax] = standard_deviations(nominal_values, uncertainties, 'uniform');
disp('Variables minimum values:')
xmin
disp('Variables maximum values:')
xmax
disp('Variables standard deviations:')
std_vec


format                                                                  %restores ...
    original data printing format

%################################## SIMULATING ########################################
tic                                                                     %Starts ...
    chronometer for simulation time measurement
[nominal_output_UT, time_vector_UT] = MAIN_INITIAL_CONDITIONS(nominal_values); ...
            %Simulation with nominal values
[sp_vec, w_vec]=sigma_points(std_vec);                                  %obtaining ...
    the sigma points
n_samples = length(sp_vec);

%Unscented Transform simulations
[simulation_output] = simulate_UT(@MAIN_INITIAL_CONDITIONS, nominal_values, sp_vec);

```

```
elapsed_time_UT = toc                                                    %Stops ...
    the chronometer

%################################### DATA ANALISYS ######################################
[mean_vector_UT, std_vector_UT] = ...
    UT_statistical_analysis(nominal_output_UT,simulation_output,w_vec,sp_vec);
std_vector_UT = sqrt(std_vector_UT);

%################################ DATA SAVING ###########################################
%Saving data for later comparison
save('outputUT.mat','nominal_output_UT')
save('outputUT.mat','mean_vector_UT','-append')
save('outputUT.mat','time_vector_UT','-append')
save('outputUT.mat','elapsed_time_UT','-append')
save('outputUT.mat','std_vector_UT','-append')


%################################### GRAPHICAL INTERPRETATION ############################
n_outs= 2
titles=['Voltage on the DC bus  ';
        'Rotor speed in per unit'];
%maximum and minimun values
for i=1:n_outs
    disp('Plots');
    ax1=subplot(n_outs,1,i);
    hold all;  grid on;
    plot(time_vector_UT,nominal_output_UT(:,i),'g','linewidth',1);
    plot(time_vector_UT,mean_vector_UT(:,i),'r','linewidth',1);
    plot(time_vector_UT,mean_vector_UT(:,i)+3*std_vector_UT(:,i),'b', 'linewidth',1);
    plot(time_vector_UT,mean_vector_UT(:,i)-3*std_vector_UT(:,i),'c', 'linewidth',1);
    xlabel('Time (s)','fontsize',16);
    ylabel('Voltage (V)','fontsize',16);
    set(gca,'fontsize',18);
    l=legend(ax1,'Nominal-UT', 'Mean-UT', 'Mean+3std','Mean-3std');
    l.Location='northeast';
    title(titles(i,:));
end
```

APÊNDICE C – FUNÇÃO MAIN_INITIAL_CONDITIONS

```
function [out, time_vector] =MAIN_INITIAL_CONDITIONS(input_variables)

display('Starting..')

WindSpeed=input_variables(1)
% Set wind speed
set_param('WIND_FARM/Wind Speed','Value',num2str(WindSpeed));

% In the Simulation/Configuration Parameters menu, uncheck the "Initial
% state" parameter.
set_param('WIND_FARM','LoadInitialState','off');

% In the 120 kV Three-phase Voltage Source menu, disable the source
% voltage step by setting the "Time variation of " parameter to "none".
set_param('WIND_FARM/120 kV','VariationEntity','none');

% In order to shorten the time required to reach steady-state, you will
% have to temporarily decrease the inertia of the turbine-generator group.
% Open the DFIG Wind Turbine menu and in the Drive train data and Generator
% data, divide the H inertia constants by 10.
set_param('WIND_FARM/DFIG Wind Turbine','mec', ...
    '[0.0685 0.01 3]'); % Original: [0.685 0.01 3]

% Change the Simulation Stop Time to 5 seconds. Note that in order to
% generate initial conditions coherent with the 60 Hz voltage source phase
% angles, the Stop Time must be an integer number of 60 Hz cycles.
set_param('WIND_FARM','StopTime','5');

% Change the Simulation Mode from "Normal" to "Accelerator".
set_param('WIND_FARM','SimulationMode','Accelerator');

% Make sure the final states are saved
set_param('WIND_FARM','SaveFinalState','on');
set_param('WIND_FARM','FinalStateName','xFinal');


% Start simulation. When Simulation is completed, verify that steady state
% has been reached by looking at waveforms displayed on the Scope. The
% final states which have been saved in the "xFinal" structure with time
% can be used as initial states for future simulations.
set_param('WIND_FARM','SaveTime','off');
save_system('WIND_FARM');
disp('Getting initial conditions..')
sim('WIND_FARM');

% Executing the next two commands copies these final conditions in
% "xInitial" and saves this variable in a new file (myModel_init.mat).
xInitial=xFinal;
save myModel_init xInitial;

%clear
```

```
%load('myModel_init');

% In the Simulation/Configuration Parameters menu, check "Initial state".
% (And make sure the initial state name is correct)
set_param('WIND_FARM','LoadInitialState','on');
set_param('WIND_FARM','InitialState','xInitial');

% In the Wind Turbine Generator and Drive train data, reset the inertia
% constants H back to their original values.
set_param('WIND_FARM/DFIG Wind Turbine','mec', ...
    '[0.685 0.01 3]'); % Reduced: [0.0685 0.01 3]

% In the 120 kV Three-phase voltage source menu, set the "Time variation
% of" parameter back to "Amplitude".
set_param('WIND_FARM/120 kV','VariationEntity','Amplitude');

% (Amplitudes) Amplitude values:[1 0.5 1.0]
set_param('WIND_FARM/120 kV','Amplitudes','[1 0.5 1.0]');

% (TimeValues) Time values: [   0 0.03 0.13]
set_param('WIND_FARM/120 kV','TimeValues','[0 0.03 0.13]');

% Change the Simulation Stop Time and Simulation Mode back to their
% original values (0.2 seconds, Normal).
set_param('WIND_FARM','StopTime','0.2');
set_param('WIND_FARM','SimulationMode','Accelerator');

%save the time vector
set_param('WIND_FARM','SaveTime','on');

% Simulate model with Initial State vector (freshly computed)
save_system('WIND_FARM');
disp('Fault simulation..')
simOut = sim('WIND_FARM');

%out = [Vdc.Data, wr_pu.Data];
out = [Vdc.Data];
time_vector=simOut;

end
```

APÊNDICE D – FUNCÃO PARA OBTENÇÃO DOS PONTOS SIGMA

```
function [s_points, weights] = sigma_points(std_vec)
    % generates the sigma points and weights for a generic number of
    % varibles
    % input:
    %   std_vec = vector of standard deviation for each random variable
    % output:
    %   s_points = set of sigma points
    %   weigths = vector with the weights
    %nrv = number of varibles/random values
    nrv = length(std_vec);

    %nsp = number of sigma points
    edge_points = 2^nrv;
    axis_points = 2*nrv;
    nsp = edge_points + axis_points;

    w1 = nrv^2/(2^nrv*(nrv+2)^2);
    w2 = 1/(nrv+2)^2;
    w0 = 1-(edge_points*w1+axis_points*w2);

    weights = [w0, w1, w2];

    edges_value = sqrt(nrv+2)/sqrt(nrv);
    axis_value = sqrt(nrv+2);

    %creates a matrix with NSP lines and NRV columns
    %in which each line store a sigma point set
    s_points = ones(nsp-1,nrv);

    % fills the s_points array with the sigma points for the edge
    % that is the upper part of the array, which is multiplied
    % by the weight w1
    %disp('Creating spoints set')
    value = 1;
    for j=1:nrv %iterate throug the columns
        counter_max = 2^(nrv-j);
        counter=0;
        for i=1:2^nrv %iterate through the lines
            counter=counter+1;
            if(counter≤counter_max)
                value = 1;
            else
                value = -1;
                if(counter==2*counter_max)
                    counter = 0;
                end
            end
            s_points(i,j)=value*edges_value;
        end
```

```
50      end
51
52      % terminates filling the s_points array with the
53      % sigma points for the axis, the lower part of the array which is
54      % multiplied by the weight w2
55      for j=1:nrv %iterate throug the columns
56          for i=2^nrv+1:nsp %iterate through the lines
57               s_points(i,j)=0;
58          end
59      end
60
61      for j=1:nrv %iterate throug the columns
62          for i=1:2*nrv %iterate through the lines
63              if(i+1 == 2*j)
64                  s_points(i+2^nrv,j)=1*axis_value;
65                  s_points(i+2^nrv+1,j)=-1*axis_value;
66              end
67          end
68      end
69
70      %multiplying by the standard deviation to get the actual sigma points
71      for i=1:nsp
72          for j=1:nrv
73              s_points(i,j)=s_points(i,j)*std_vec(j);
74          end
75      end
76  end
```

APÊNDICE E – FUNÇÃO PARA SIMULAÇÃO DO MODELO UTILIZANDO OS PONTOS SIGMA

```
1
2  function [sim_output, x] = simulate_UT(func, nominal_values, sp_vec)
3      % simulates the function 'func' using the nominal_values + sigma_points
4      % for each sigma point set
5      %inputs
6      %    - func = the function to simulate
7      %    - nominal_values = nominal values of each componente used in the
8      %    simulation
9      %    - sp_vec = vector with the sigma points set used in each simulation
10
11     sim_output = 0;
12
13     n_coeffs = length(nominal_values); %Get the number of input parameters
14     sample_number = length(sp_vec); %Number of simulations
15     fprintf('Unscented Transform Simulation with %d simulations.', sample_number);
16     x = zeros(sample_number,n_coeffs);
17     % Generation of the simulation input parameters according to the Normalized
18     % Sigma Points and the Standard Deviations
19     for i=1:sample_number
20         for j=1:n_coeffs
21            x(i,j) = nominal_values(j)+sp_vec(i,j);
22         end
23     end
24
25     % Size of the generated input values matrix
26     n_lines = length(x);
27     n_columns= length(x(1,:));
28
29     %##################### SIMULATION ####################
30     fprintf('\n\nStarting simulation, please be patient... \n\n')
31
32     sim_output = cell(1,n_lines);
33
34     for j=1:sample_number;
35         %Evals each line as a simulation and stores the output data in
36         %sim_output
37         fprintf('Simulation number %d/%d \n', j, sample_number);
38         sim_output{j}=func(x(j,:));
39     end
40
41     disp('The simulation is done...');
42 end
```

APÊNDICE F – FUNÇÃO PARA A OBTENÇÃO DAS MÉDIAS E DESVIOS PADRÃO

```
1
2  function [mean_, std_] = ...
       UT_statistical_analysis(nominal_output,simulation_output,w_vec, sp_vec)
3      %returns the output mean and standard deviation obtained from the UT
4      %simulations
5      %inputs
6      %    - nominal_output = simulation vector using the nominal inputs
7      %    - simulation_output = cell containing every simulation with the
8      %    sigma points set
9      %    - w_vec = vector containing the weigths
10     %    - sp_vec = vector with the sigma points set used in each simulation
11
12     disp('Starting Statistical Analysis...');
13
14     %the actual transformation from cell to matrix
15     simulation_vector=cell2mat(simulation_output)';
16
17     %represents the number of steps
18     n_spoints = length(sp_vec);
19     n_variables = length(sp_vec(1,:));
20     n_steps = length(nominal_output);
21     n_outputs = length(nominal_output(1,:));
22
23     %create the arrays, one mean vector and one std vector for each output
24     mean_ = zeros(n_outputs, n_steps);
25     std_ = zeros(n_outputs, n_steps);
26
27     nominal_output = nominal_output';
28
29     %%%%%%%%%%%%%%% Mean Vector %%%%%%%%%%%%%%%%%%
30     %obtaining the means for each step of the simulation
31     for out_var = 1:n_outputs %iterate through the outputs
32         for j=1:n_steps %iterate throug the lines
33             weighted_sum = 0;
34             for i=out_var:n_outputs:(n_spoints*n_outputs) %1, 4, 7... for 3 outputs
35                 if(i<=2^n_variables*n_outputs) %%summation of w1 G(U+Si)
36                     weighted_sum = weighted_sum + w_vec(2)*simulation_vector(i,j);
37                 else %%summation of w2 G(U+Si)
38                     weighted_sum = weighted_sum + w_vec(3)*simulation_vector(i,j);
39                 end
40             end
41             mean_(out_var, j)=w_vec(1)*nominal_output(out_var,j)+weighted_sum;
42         end
43     end
44
45
46        %obtaining the standard deviations for each step of the simulation
47     for out_var = 1:n_outputs %iterate through the outputs
48         for j=1:n_steps
```

```
            weighted_sum = 0;
            for i=out_var:n_outputs:(n_spoints*n_outputs) %1, 4, 7... for 3 outputs
                if(i≤2^n_variables*n_outputs) %summation of w1 G(U+Si)
                    weighted_sum = weighted_sum + ...
                        w_vec(2)*(simulation_vector(i,j)-mean_(out_var,j))^2;
                else %%summation of w2 G(U+Si)
                    weighted_sum = weighted_sum + ...
                        w_vec(3)*(simulation_vector(i,j)-mean_(out_var,j))^2;
                end
            end
            std_(out_var,j)=w_vec(1)*(nominal_output(out_var,j)-mean_(out_var,j))^2 + ...
                weighted_sum;
        end
    end

    mean_ = mean_';
    std_ = std_';
    disp('Done!');
end
```

APÊNDICE G – FUNÇÃO PARA COMPARAÇÃO DOS MÉTODOS

```
load('outputMC');
load('outputUT');

title1 = sprintf('V_{DC} - Means',n_samples_MC);
title2 = 'Mean \pm 3stds';
title3 = 'Percentual error UT, Mean'
title4 = 'Percentual error UT, Mean + 3std'

plot_labels = ['Voltage (V)      ',
               '\omega_r (pu)    '];

nlines=length(mean_vector_MC(:,1));
t=1:100:nlines;
%maximum and minimun values

%x = [time_vector_UT, nominal_output_UT(:,1)];
%save test2.dat x -ascii
%return

for i=1:1

    %% MEAN PLOTS
    figure;
    ax1=subplot(2,1,1);
    hold all;  grid on;
    plot(time_vector_UT,nominal_output_UT(:,i),'g','linewidth',1);
    plot(time_vector_UT(t),mean_vector_UT(t,i),'ro', 'linewidth',1);
    %plot(time_vector_UT,mean_vector_UT(:,i),'r','linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i),'b','linewidth',1);
    %plot(time_vector_UT,mean_vector_UT(:,i)+3*std_vector_UT(:,i),'b', 'linewidth',1);
    %plot(time_vector_UT,mean_vector_UT(:,i)-3*std_vector_UT(:,i),'c', 'linewidth',1);
    xlabel('Time (s)','fontsize',16);
    ylabel(plot_labels(i,:),'fontsize',16);
    set(gca,'fontsize',18);
    l=legend(ax1,'Nominal', 'UT', 'MC');
    l.Location='northeast';
    title(title1);

    %% MEAN + 3 SD PLOTS
    ax2=subplot(2,1,2);
    hold all;  grid on;
    plot(time_vector_MC,nominal_output_UT(:,i),'g','linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i)+3*std_vector_MC(:,i),'b', 'linewidth',1);
    plot(time_vector_UT(t),mean_vector_UT(t,i)+3*std_vector_UT(t,i),'ro', 'linewidth',1);
    %plot(time_vector_UT(:),mean_vector_UT(:,i)+3*std_vector_UT(:,i),'r', 'linewidth',1);
    plot(time_vector_MC,mean_vector_MC(:,i)-3*std_vector_MC(:,i),'b', 'linewidth',1);
    plot(time_vector_UT(t),mean_vector_UT(t,i)-3*std_vector_UT(t,i),'ro', 'linewidth',1);

    %plot(time_vector_MC,lower_bound(:,i),'k','linewidth',1);
    %plot(time_vector_MC,upper_bound(:,i),'y','linewidth',1);
    xlabel('Time (s)','fontsize',16);
```

```
    ylabel(plot_labels(i,:),'fontsize',16);
    set(gca,'fontsize',18);

    l=legend(ax2,'Nominal', 'MC+3sd', 'UT+3sd', 'MC-3sd','UT-3sd');
    l.Location='northeast';
    title(title2);


    %% ERROR - MEAN
    figure
    ax3=subplot(2,1,1);
    hold all;  grid on;
    mean_error = mean_vector_MC(:,i)-mean_vector_UT(:,i);
    for j=1:nlines
        mean_error(j,i)=100*mean_error(j,i)/mean_vector_MC(j,i);
    end
    plot(time_vector_MC,mean_error(:,i),'g','linewidth',1);

    l=legend(ax3,'Error in %');
    l.Location='northeast';
    xlabel('Time (s)','fontsize',16);
    ylabel('% Error','fontsize',16);
    set(gca,'fontsize',18);
    title(title3);


    %% ERROR - MEAN+3SD
    ax4=subplot(2,1,2);
    std_error = mean_vector_MC(:,i)+3*std_vector_MC(:,i)- ( ...
        mean_vector_UT(:,i)+3*std_vector_UT(:,i))
    for j=1:nlines
        std_error(j,i)=100*std_error(j,i)/(mean_vector_MC(j,i)+3*std_vector_MC(j,i));
    end
    plot(time_vector_MC,std_error(:,i),'r','linewidth',1);
        l=legend(ax4,'Error in %');
    l.Location='northeast';
    xlabel('Time (s)','fontsize',16);
    ylabel('% Error','fontsize',16);
    set(gca,'fontsize',18);
    title(title4);
end
```

APÊNDICE H – ALGORITMO PARA A EXECUÇÃO DO MÉTODO DE MONTE CARLO EM LOTES

```
disp(sprintf('\n\n ---------- Example ---------- '))

%This example applies the Monte Carlo method for a buck converter with the
%following components and uncertainties:
% Resistor: 50R +- 10%
% Inductor: 230mH +- 5%
% Capacitor: 100uF +- 10%

model='WIND_FARM';
load_system(model);

n_batches = length(dir('MC_batch_output_*.mat')) %checks the number of previous batches
n_samples=50; %Number of simulations

%valores nominais
WS = 10;
nominal_input_MC = [WS];

%Components uncertainty in percentage
incWS= 100;

%Displaying maximum and minimum values associated with each circuit's component
format short eng

disp('Components minimum values:')
xmin = [WS-WS*incWS/100]

disp('Components maximum values:')
xmax = [WS+WS*incWS/100]

format %restores original data printing format

%####################### SIMULATING ##########################
tic %Starts chronometer for simulation time measurement

%Simulation with nominal values only if it was not simulated yet
if(n_batches == 0)
    disp('Simulation nominal value');
    [nominal_output_MC, time_vector_MC] = MAIN_INITIAL_CONDITIONS(nominal_input_MC);
end
%Monte Carlo simulation
[simulation_input_MC,simulation_output_MC]=montecarlo(@MAIN_INITIAL_CONDITIONS, xmin, ...
    xmax, n_samples); %retorna [dados de simulacao, saida das simulacoes


elapsed_time_MC=toc %Stops the chronometer

%########################### DATA ANALISYS ##########################
%This is left for the 'FINAL_BATCH_ANALISYS' scritp
```

```
%Execute it when all the batches are complete

%############################## DATA SAVING ############################
n_outs = length(simulation_output_MC{1}(1,:));
filename=sprintf('MC_batch_output_%d.mat', n_batches+1) %create a filename for this batch

%saving the data
save(filename,'n_samples');
save(filename,'n_outs', '-append');
save(filename,'simulation_output_MC', '-append')
save(filename,'simulation_input_MC', '-append')

save(filename,'nominal_input_MC', '-append')
save(filename,'elapsed_time_MC','-append')
if(n_batches==0)
    save(filename,'time_vector_MC','-append')
    save(filename,'nominal_output_MC', '-append')
end
```

APÊNDICE I – ALGORITMO PARA JUNÇÃO DOS RESULTADOS OBTIDOS A PARTIR DAS FORNADAS DO MÉTODO DE MONTE CARLO

```
function [mean_,std_, max_, min_] = analyze_batch(name)
    %Reads all batches outputs, performs data analysis and plots the results
    %Saves the result in the file 'outputCM.mat'
    %This function also plots the inputs in a order graph

    disp(name);
    n_batches = length(dir(name)); %count the number of files to analyse
    files = dir(name); %get the files name

    %simulations per batch
    load(files(1).name, 'n_samples')
    load(files(1).name, 'n_outs')
    batch_sims = n_samples;

    n_inputs = 1;

    %cell to put all the outputs together
    total_outputs = cell(1, (batch_sims*n_batches)+1);%+1 is the nominal output

    %cell to put all the inputs together
    total_inputs = zeros((batch_sims*n_batches)+1,n_inputs);%+1 is the nominal output

    %variable for counting the total time in all batches
    total_time = 0;

    %load all batches in one single cell
    for i=1:n_batches
       load(files(i).name);
       total_time = total_time + elapsed_time_MC;
       total_inputs((i-1)*batch_sims+1:(i-1)*batch_sims+batch_sims,:) = ...
           simulation_input_MC(1,:);
       for j=1:batch_sims
          total_outputs{(i-1)*batch_sims+j} = simulation_output_MC{j};
       end
       %nominal outputs at the end - last element
       total_outputs{n_batches*(batch_sims)+1} = nominal_output_MC; %+1 is the nominal ...
           output
       %nominal outputs at the end - last element
       total_inputs(n_batches*(batch_sims)+1,:) = nominal_input_MC; %+1 is the nominal ...
           output
    end

    n_samples_MC = n_batches*batch_sims+1

    %###################### DATA ANALISYS ########################
    %performing the worst case analysis as if it was one big simulation
    [upper_bound, lower_bound]=worstcase_analisys(total_outputs, n_samples_MC); ...
        %encontrando limites e plotando graficos
```

```
45      %performing the statistical analysis normally
46      [mean_vector_MC, std_vector_MC] = statistical_analysis(total_outputs,n_samples_MC);
47
48      %######################## DATA SAVING ########################
49      elapsed_time_MC = total_time
50
51  %######################## DATA SAVING ########################
52      n_outs = length(simulation_output_MC{1}(1,:));
53      filename = 'outputMC.mat';
54
55      save(filename,'n_samples_MC');
56      save(filename,'n_outs', '-append');
57      save(filename,'simulation_output_MC','-append')
58      save(filename,'nominal_output_MC','-append')
59      save(filename,'nominal_input_MC', '-append')
60      save(filename,'simulation_input_MC', '-append')
61      save(filename,'std_vector_MC','-append')
62      save(filename,'mean_vector_MC','-append')
63      save(filename,'time_vector_MC','-append')
64      save(filename,'elapsed_time_MC','-append')
65      save(filename,'upper_bound','-append')
66      save(filename,'lower_bound','-append')
67
68      %######################## GRAPHICAL INTERPRETATION ###############
69      titles=['Voltage on the DC bus  ';
70              'Rotor speed in per unit'];
71
72      plot_labels = ['Voltage (V)      ',
73                     '\omega_r (pu)    '];
74
75      %maximum and minimun values
76      figure
77      title('Ordered input data');
78      plot(sort(total_inputs(:,1)),'o','linewidth',1);
79      xlabel('Sample number','fontsize',16);
80      ylabel('Wind speed   ','fontsize',16);
81
82      figure
83      for i=1:n_outs
84          ax1=subplot(n_outs,1,i);
85          hold all;  grid on;
86          plot(time_vector_MC,nominal_output_MC(:,i),'g','linewidth',1);
87          plot(time_vector_MC,mean_vector_MC(:,i),'r','linewidth',1);
88          plot(time_vector_MC,mean_vector_MC(:,i)+3*std_vector_MC(:,i),'b', 'linewidth',1);
89          plot(time_vector_MC,mean_vector_MC(:,i)-3*std_vector_MC(:,i),'c', 'linewidth',1);
90          %plot(time_vector_MC,lower_bound(:,i),'k','linewidth',1);
91          %plot(time_vector_MC,upper_bound(:,i),'y','linewidth',1);
92          xlabel('Time (s)','fontsize',16);
93          ylabel(plot_labels(i,:),'fontsize',16);
94          set(gca,'fontsize',18);
95          %l=legend(ax1,'Nominal-MC', 'Mean-MC', 'Mean+2std','Mean-2std', 'Lower Bound', ...
                'Upper Bound');
96          l=legend(ax1,'Nominal-MC', 'Mean-MC', 'Mean+3std','Mean-3std');
97          l.Location='northeast';
98          title(titles(i,:));
99      end
100 end
```

Efficient Uncertainty Analysis of Wind Farms in the Time Domain using the Unscented Transform

Rudimar Baesso Althof[1] and Moises Ferber[2]

***Abstract*—In this paper, an efficient method of uncertainty analysis, the so-called Unscented Transform, is applied to a wind farm numerical model. Particularly, the effect of an uncertain wind speed on the DC bus voltage of the power converter, as a fault occurs, is considered. We show that the DC bus voltage is strongly influenced by the wind speed. Moreover, the proposed uncertainty analysis method computes the results with similar accuracy than the Monte Carlo method, but with a significant computational effort reduction.**

I. INTRODUCTION

Sustainable solutions are attracting the attention of the energy industry for decades already. The idea that the consumers take responsibility for pollution caused by energy generation is increasingly adopted, leading to a surge in the use of renewable energy sources and technologies [1]. There are several sources of renewable energy: solar, hydro, biomass, geothermal, tides, wind and many others. In this work, we focus on wind energy, even if the presented ideas may be applied to any renewable energy system.

The wind energy has being used for at least 3000 years by windmills for grinding grains or pumping water. If we consider its use in sailing ships this source of power has been used for even longer. The very low level of CO_2 emission over the entire life of a wind turbine is the main reason for its use. Moreover, the wind power has economical benefits since it reduces the fossil fuel dependency and its price volatility, specially for fuel importing countries which import from politically unstable areas [1].

Nowadays, the analysis and design of wind energy systems are strongly dependent on numerical simulations. These simulations allow the assessment of crucial electrical characteristics of a given wind energy system model at a relatively low cost. There are different categories of models that can be simulated, with different goals and techniques. For instance, in a time-domain simulation, the time step can be as low as $5\mu s$ to capture switching effects of the power electronics. Some examples of these relevant analysis tools are: power flow, fault effects, transient stability and electromagnetic simulations. These numerical simulations of wind energy systems can be performed by software packages, such as MATLAB/Simulink, PSS/E, and PSCAD/EMTDC.

*This work was supported by a grant PIBIC from CNPq, Brazil

[1]R. B. Althof is an undergraduate in Mechatronics Engineering, Department of Mobility Engineering, Federal University of Santa Catarina, Joinville, SC 89218-035, Brazil rualthof at grad.ufsc.br

[2]M. Ferber is with the Mechatronics Group, Department of Mobility Engineering, Federal University of Santa Catarina, Joinville, SC 89218-035, Brazil moises.ferber at ufsc.br

978-1-5090-5339-1/17/$31.00 © 2017 IEEE

Although the software packages available today are capable of simulating large-scale power systems and provide accurate results, renewable systems are highly affected by uncertain inputs, such as the wind speed for wind farms and solar incidence in solar panels. As the wind speed plays a major role in wind energy systems, its uncertainty must be taken into account in order to carry out a reliable numerical simulation.

The most common method for treating uncertainties is the Monte Carlo (MC) method. The Monte Carlo method can be briefly explained as follows: a large number of samples (N_{MC}) is generated according to the multivariate input Probability Density Function (PDF). For each sample, a numerical simulation of the considered model is carried out, producing a collection of N_{MC} results. Then, the quantity of interest, such as the average and standard deviation or the output PDF is computed from the collection. The MC method has a low rate of conversion and therefore it requires typically tens or hundreds of thousands of simulations in typical electrical engineering problems in order to produce accurate results. For example, if the time required to carry out one simulation is 1 minute, then the MC method would require approximately 7 days to complete, if 10000 samples are considered.

In this context, there has been a great effort to develop alternative Uncertainty Quantification (UQ) methods that are more efficient and require less simulations for accurate results. Some examples can be found in [2]–[5]. One example of an efficient UQ method that has been used lately is the Unscented Transform (UT) [6].

The Unscented Transform method has being applied for uncertainty quantification in [7], [8] as an alternative to the Monte Carlo method. The UT approximates a nonlinear mapping by a set of points, called *sigma points*, which allow one to obtain both the expected value and the variance from a weighted average of the simulation outputs generated by the sigma points [7].

In this paper, we apply the UT to a wind farm model and show that: (1) it requires significant less time to compute the results; (2) its accuracy is similar to the MC method and (3) the wind speed uncertainty must be taken into account for components specification.

II. PROBLEM STATEMENT

Consider an arbitrary wind energy system. This wind farm system includes generation, transmission and distribution systems which can be modeled using linear and nonlinear components such as resistors, inductors, capacitors, IGBT's,

generators, controlled current and voltage sources. This model is essentially a system of nonlinear ODE's.

The problem considered in this paper consists of determining the average (μ) and standard deviation (σ) of any current or voltage of the model, given the μ and σ of each uncertain parameter.

The mathematical formulation is given as follows. The uncertain parameters of the system, such as the wind speed, the transmission line capacitance per unit length and fault level, are represented by the vector of real random variables $\vec{U} = [U_1, U_2, ..., U_{N-1}, U_N]$, with dimension N. Each of these parameters may have a different statistical distribution and, in this work, it is assumed statistical independence between them. The vectors $\vec{\mu_u} = [\mu_{u_1}, \mu_{u_2}, ..., \mu_{u_{N-1}}, \mu_{u_N}]$ and $\vec{\sigma_u} = [\sigma_{u_1}, \sigma_{u_2}, ..., \sigma_{u_{N-1}}, \sigma_{u_N}]$ are the vectors of averages and standard deviations of $\vec{U}$, respectively. The output variable, or the so-called quantity of interest, which is any voltage or current of the wind energy system, is represented by Y. The quantities μ_y and σ_y are the average and standard deviation of Y, respectively.

The system model (G) is described as:

$$Y(t) = G(\vec{U}, t) \tag{1}$$

Note that $Y(t)$ is a time-dependent random variable. Thus, the problem consists in computing μ_y and σ_y.

III. METHODOLOGY

Let $\vec{S} = [S_1, S_2, ..., S_{N_S}]$ and $\vec{w} = [w_0, w_1, w_2, ..., w_{N_S}]$ be the so-called sigma points and the weights, respectively. Note that the dimension of these vectors is the number of sigma points N_S.

The method to obtain μ_y is given as follows [7]:

$$\mu_y = E\left\{G(\vec{\mu_u} + \vec{\hat{u}})\right\} = w_0 G(\vec{\mu_u}) + \sum_{i=1}^{N_S} w_i G(\vec{\mu_u} + S_i) \tag{2}$$

where $E\{.\}$ is the expectation operator and $\vec{\hat{u}}$ is a vector of zero mean and unit variance random variables.

Once the mean is calculated, it is possible to obtain the variance σ_y^2 given by the equation 3.

$$\begin{aligned}\sigma_y^2 &= E\left\{(G(\vec{\mu_u} + \vec{\hat{u}}) - \vec{\mu_u})^2\right\} \\ &= \omega_0 (G(\vec{\mu_u} + \vec{\hat{u}}) - \vec{\mu_u})^2 + \sum_{i=1}^{N} \omega_i (G(\vec{\mu_u} + S_i) - \mu_y)^2\end{aligned} \tag{3}$$

The details of the derivation of the UT method can be found in [7].

IV. APPLICATION

This work addresses a 1D problem of a wind farm, considering the wind speed as the input random variable with a uniform probability density function, shown in Figure 1, and analyzes the voltage of the DC bus capacitor, when a fault of $0.5pu$ happens at $0.3s$. This model is an example of the Simscape Power Systems toolbox, which simulates a wind farm with 6 turbines of 1.5MW each [9]. A discrete solver with time step $5\mu s$ was used to capture switching effects. Moreover, each simulation requires a pre-processing, which consists of computing the initial states. The time to complete one simulation is $38.8s$. The wind farm schematics is shown in Figure 2, where the fault is implemented in the voltage source (1).

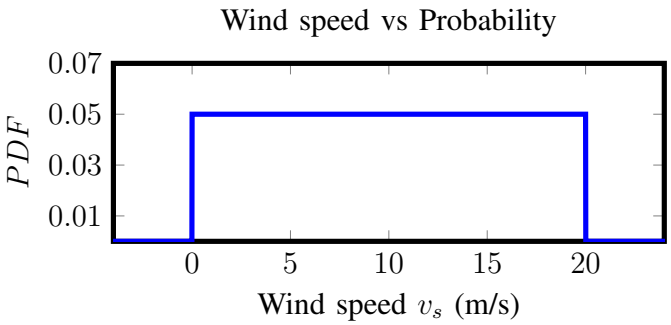


Fig. 1. Probability density function of the wind speed, uniform, average = 10m/s and $v_s = 5,7735m/s$

Method	Time
Monte Carlo	6 hours and 36 minutes
Unscented Transform	2 minutes and 46 seconds

TABLE I
TIME REQUIRED BY THE PRESENTED METHODS

V. RESULTS

This section presents the results of the UT method. A comparison was made with the Monte Carlo method, using 1000 samples. For the UT method, it was required only 5 simulations.

In Fig. 3, it can be seen the nominal voltage V_{DC} of the bus capacitor at $1150V$. At $t = 0.03s$, a $0.5pu$ fault occurs on the element (1) of the schematic in Fig. 2. The fault causes the V_{DC} to oscillate, with a peak value of about $1185V$ and a lowest value of about $1130V$. This result corresponds to the average behavior of V_{DC} when the wind speed varies from $0m/s$ to $20m/s$.

In Fig. 4, it is shown the probabilistic upper and lower bounds of V_{DC}, with a 0.1% confidence level. Note that the V_{DC} also oscillates, but with a peak of approximately $1200V$. Additionally, the undershoot reaches values of around $1100V$.

The simulation time for each method was also computed and Table I shows the results. The UT method presented a reduction in the simulation time of 160.93 times over the MC method.

VI. CONCLUSION

This work presented the Unscented Transform method for uncertainty quantification of wind energy systems.

The scheme was applied to a relevant model based on a real world wind farm and later compared with the MC method. The results presented a similar accuracy between

the MC and UT methods, with a time reduction factor of over 160. Additionally, this work shows the importance of considering the variability of the wind speed when specifying the components.

REFERENCES

[1] I. Dincer, "Renewable energy and sustainable development: a crucial review," *Renewable and Sustainable Energy Reviews*, vol. 4, no. 2, pp. 157–175, 2000.

[2] M. Ferber, C. Vollaire, L. Krhenbhl, J. L. Coulomb, and J. A. Vasconcelos, "Conducted emi of dc-dc converters with parametric uncertainties," *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 4, pp. 699–706, Aug 2013.

[3] M. Ferber, A. Korniienko, G. Scorletti, C. Vollaire, F. Morel, and L. Krhenbhl, "Systematic lft derivation of uncertain electrical circuits for the worst-case tolerance analysis," *IEEE Transactions on Electromagnetic Compatibility*, vol. 57, no. 5, pp. 937–946, Oct 2015.

[4] O. A. Oke and D. W. P. Thomas, "Probabilistic load flow in microgrid assessment and planning studies," in *2012 IEEE Electrical Power and Energy Conference*, Oct 2012, pp. 151–156.

[5] O. A. Oke, D. W. P. Thomas, and G. M. Asher, "A new probabilistic load flow method for systems with wind penetration," in *2011 IEEE Trondheim PowerTech*, June 2011, pp. 1–6.

[6] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar 2004.

[7] L. d. Menezes, A. Ajayi, C. Christopoulos, P. Sewell, and G. A. Borges, "Efficient computation of stochastic electromagnetic problems using unscented transforms," *IET Science, Measurement & Technology*, vol. 2, no. 2, p. 88, 2008.

[8] M. Ferber, C. Vollaire, L. Krhenbhl, and J. A. Vasconcelos, "Adaptive unscented transform for uncertainty quantification in emc large-scale systems," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 33, no. 3, pp. 914–926, 2014. [Online]. Available: http://dx.doi.org/10.1108/COMPEL-10-2012-0212

[9] Mathworks. (2016) Wind farm - DFIG detailed model. https://www.mathworks.com/help/physmod/sps/examples/wind-farm-dfig-detailed-model.html.

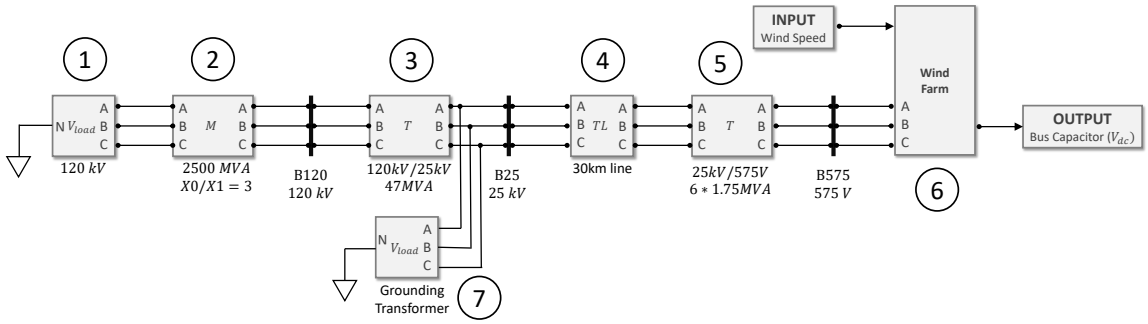


Fig. 2. Model of the wind farm consisting of: (1) 120-kV faulting system, (2) three phase impedance with mutual coupling between phases, (3) 120 kV grid, (4) 30 km transmission line, (5) 25 kV distribution system, (6) DFIG wind turbine , and (7) transformer providing a neutral to the three phase system.

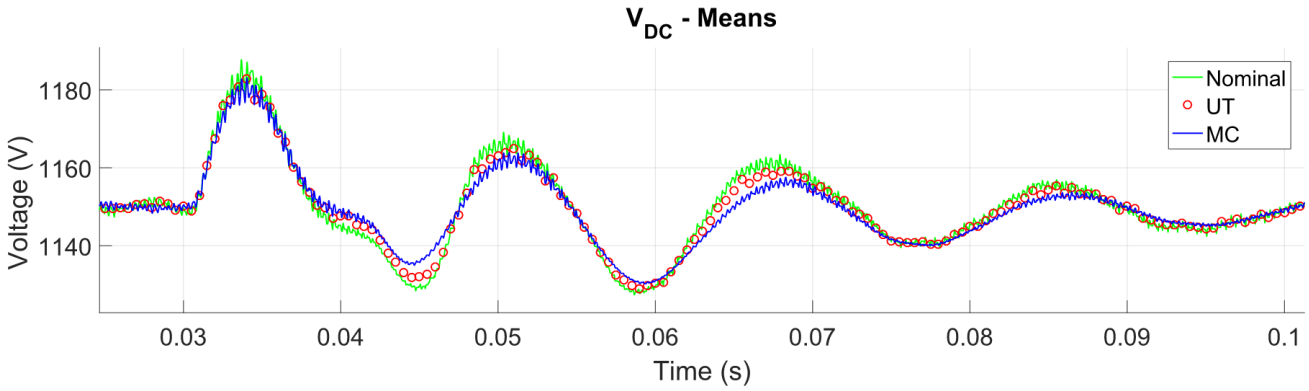


Fig. 3. Average voltage on the bus capacitor for both methods and nominal input.

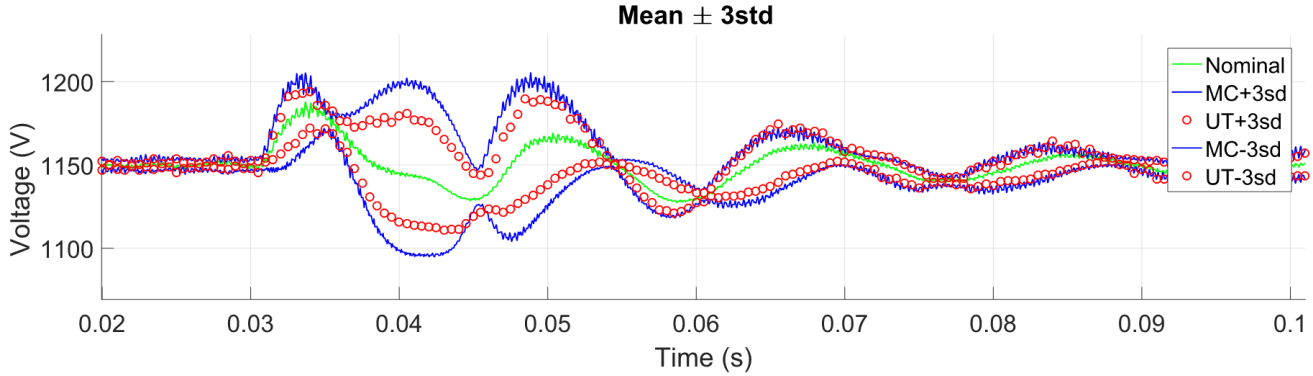


Fig. 4. Average voltage on the bus capacitor ± 3 SD for both methods.