

UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

JOSÉ EDUARDO CORRÊA SANTANA E SILVA

**Desenvolvimento de *crash box* do tipo origami através de
metamodelos**

São Paulo
(2019)

JOSÉ EDUARDO CORRÊA SANTANA E SILVA

**Desenvolvimento de *crash box* do tipo origami através de
metamodelos**

Versão Revisada

Dissertação de mestrado apresentada à
Escola Politécnica da Universidade de São
Paulo para obtenção do título de Mestre
em Ciências.

Área de Concentração: Engenharia
Mecânica de Projeto e Fabricação

Orientador: Prof. Dr. Marcílio Alves – USP

São Paulo
(2019)

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Silva, José Eduardo Corrêa Santana e
Desenvolvimento de crash box do tipo origami através de metamodelos /
J. E. C. S. Silva -- versão corr. -- São Paulo, 2019.
123 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

1.Engenharia Automotiva 2.Otimização 3.Algoritmos genéticos 4.Método dos Elementos Finitos 5.Metamodelos I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecânica II.t.

Nome: SILVA, José Eduardo Corrêa Santana e

Título: Desenvolvimento de *crash box* do tipo origami através de metamodelos

Dissertação de mestrado apresentada à
Escola Politécnica da Universidade de São
Paulo para obtenção do título de Mestre
em Ciências.

Área de Concentração: Engenharia
Mecânica de Projeto e Fabricação

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

À minha família, por sua compreensão, carinho e apoio por todo o período de estudo e elaboração deste trabalho.

AGRADECIMENTOS

Ao Prof. Dr. Marcílio, que muito me ensinou com toda gentileza e disponibilidade.

À Volkswagen do Brasil, por seu apoio durante o desenvolvimento deste trabalho.

À Escola Politécnica da Universidade de São Paulo, pela oportunidade de realização do curso de Mestrado.

*Há um tempo para estar à frente,
um tempo para estar atrás,
um tempo para estar em movimento,
um tempo para descansar,
um tempo para ser vigoroso,
um tempo para estar exausto,
um tempo para estar à salvo,
um tempo para estar em perigo.*

*O Sábio vê as coisas como elas são,
sem tentar controlá-las.*

*Ele as deixa seguir seu próprio caminho
e habita no centro do círculo.*

Lao Tsé

RESUMO

Este trabalho inicia com uma contextualização histórica e motivação, seguida por revisão bibliográfica nos tópicos discutidos: segurança veicular, *crash box*, *crashworthiness*, absorvedores de energia, tubos de impacto, metamodelos, algoritmos genéticos, Planejamento de Experimentos (DoE – *Design of Experiments*), origami e engenharia, e métodos de otimização na engenharia.

Em seguida, o pesquisador propõe um experimento baseado em simulações, avaliando diversas *crash box* em forma de origami criadas a partir da variação de seus parâmetros dimensionais. Através de um algoritmo baseado em metamodelos, o autor realiza uma análise com o objetivo de maximizar a energia absorvida específica (*Specific Energy Absorption - SEA*) e a uniformidade de carga (*Load Uniformity - LU*). A fronteira de Pareto resultante dos dois objetivos é analisada de acordo a exemplos de critérios de decisão, e a configuração escolhida é então comparada a uma *crash box* da indústria. A configuração escolhida apresenta uma massa quatro vezes menor, e uma uniformidade de carga semelhante à *crash box* da indústria.

Conclui com novas proposições de trabalhos, envolvendo outros métodos de otimização disponíveis.

Palavras-Chave: Engenharia Automotiva. Otimização. Metamodelos. Algoritmo Genético. Método dos Elementos Finitos. Simulação. *Crash Box*. *Crashworthiness*. Origami.

ABSTRACT

This research begins with a historical background and motivation, followed by a bibliographic review on the discussed topics: vehicle safety, crash box, crashworthiness, energy absorbers, impact tubes, metamodels, Design of Experiments (DOE), origami and engineering, and optimization in engineering.

Next, the researcher proposes a simulation-based experiment, evaluating origami crash boxes created through the variation of several dimensional parameters. Through a metamodel-based algorithm, the author performs an analysis with the objective of maximizing the Specific Energy Absorption (SEA) and the load uniformity (LU). The resultant Pareto frontier of the two objectives is analyzed according to examples of decision criteria, and the chosen design is compared to a crash box from industry. The chosen design presents four times less mass, and a load uniformity similar to the crash box from industry.

The research concludes with propositions for new themes, involving other optimization methods available.

Keywords: Automotive Engineering. Optimization. Metamodels. Genetic Algorithm. Finite Element Method. Simulation. Crash Box. Crashworthiness. Origami.

LISTA DE ILUSTRAÇÕES

Figura 3-1. Número de sinistros pagos pelo DPVAT	20
Figura 3-2. Número de óbitos em acidentes de trânsito no Brasil	20
Figura 3-3. Programas NCAP ao redor do mundo	22
Figura 3-4. Modos de deformação final de tubos de parede fina que sofreram impacto axial dinâmico.	26
Figura 3-5. Modo de flambagem não compacto (esquerda) e modo simétrico (direita)	27
Figura 3-6. Forma básica de padrão de origami para tubos retangulares (a) e módulo de um tubo origami retangular (b).	30
Figura 3-7. Comparação entre (a) método de design convencional (análise) e (b) método de design do ponto ótimo (síntese)	31
Figura 4-1. Exemplo de quadrado latino para $k = 3, L = 3, N = 9$	37
Figura 4-2. Exemplos de fatoriais completos L^k	39
Figura 4-3. Exemplos de projetos fatoriais fracionados.....	42
Figura 4-4. Exemplos de projetos de compósito central.....	42
Figura 4-5. Exemplos de projetos de hipercubos latinos.....	48
Figura 4-6. Comparação entre diferente técnicas de DOE para preenchimento de espaço, com $k = 2, N = 1000$	49
Figura 4-7. Comparação entre diferente técnicas de DOE para preenchimento de espaço, com $k = 2, N = 1000$	49
Figura 4-8. Espaço preenchido com Sequência Sobol e ISF. Feito no software modeFRONTIER.....	50

Figura 4-9. Comparaçao entre diferente técnicas de DOE para preenchimento de espaço, com $k = 3, N = 1000$	50
Figura 4-10. Espaço preenchido com Sequência Sobol e ISF.....	51
Figura 4-11. Estratégia de otimização de design baseada em Metamodelos.....	55
Figura 4-12. Criação de metamodelo a partir de pontos amostrados	56
Figura 4-13. Esquema de uma Rede Neural Artificial com uma camada escondida	67
Figura 5-1. Modelo de simulação – condições de teste	70
Figura 5-2. Modelo de simulação – para-choques e impactor (esquerda) e corpo rígido conectado ao centro gravitacional do veículo (direita).....	70
Figura 5-3. Diagrama de processo de otimização	72
Figura 5-4. Diferentes tipos de inclinação dos tubos (<i>taper</i>).....	73
Figura 5-5. Padrão origami para tubos hexagonais (N = 6 lados).....	74
Figura 5-6. Lei dos materiais ajustada para o aço DP780 – Curvas de deformação obtidas através do modelo modificado Cowper-Symonds proposto em (ALVES, 2000)	76
Figura 5-7. Resposta carga-deflexão de um absorvedor de energia	79
Figura 5-8. Fluxo do processo utilizado no script de atualização da crash box.	82
Figura 5-9. Exemplo de <i>crash box</i> criada no script.....	83
Figura 5-10. Loop principal do algoritmo FAST	87
Figura 5-11. Ilustração de um cruzamento.	89
Figura 5-12. Ilustração de uma mutação.	89
Figura 5-13. O ponto A divide o espaço destes dois objetivos em duas áreas: o grupo de pontos dominados é representado pela área sombreada. A domina D, enquanto B, C e C' não são dominados.....	91

Figura 6-1. Matriz de dispersão das variáveis de entrada e saída	93
Figura 6-2. Gráfico de bolhas 4D. a escala de cores se refere ao tamanho total da crash box, e o diâmetro de cada círculo equivale ao raio do polígono básico da crash box.....	94
Figura 6-3. Gráfico de bolhas 4D, excluindo os pontos fora da curva. A fronteira de Pareto está demarcada em verde.....	96
Figura 6-4. Força versus Deslocamento das <i>Crash boxes</i> Origami da fronteira de Pareto	98
Figura 6-5. Crash boxes Origami da fronteira de Pareto.....	98
Figura 6-6. <i>Crash boxes</i> #438, #171 e modelo inspirado na indústria	99
Figura 6-7. Design #438 (superior) e design #171 (inferior) – antes do impacto (esquerda) e no ponto de deformação máxima (direita)	100
Figura 6-8. <i>Crash box</i> inspirada em modelo industrial – antes do impacto (esquerda) e na deformação máxima (direita).....	101
Figura 6-9. Comparação das curvas de Força por Deslocamento dos modelos de Pareto e do <i>crash box</i> inspirado em modelo industrial.....	101

SUMÁRIO

1. INTRODUÇÃO	15
2. OBJETIVOS	17
3. REVISÃO DA LITERATURA	18
3.1. ESTATÍSTICAS DE ACIDENTES DE TRÂNSITO.....	18
3.2. SEGURANÇA VEICULAR	21
3.3. MECANISMOS DE ABSORÇÃO DE ENERGIA.....	22
3.3.1. TUBO CIRCULAR	23
3.3.2. TUBOS QUADRADOS	26
3.4. ORIGAMI	27
3.4.1. HISTÓRIA DO ORIGAMI	27
3.4.2. ORIGAMI E ENGENHARIA	28
3.5. CRASH BOX ORIGAMI	29
3.6. MÉTODO DE SÍNTSE	30
4. MODELAGEM DAS SIMULAÇÕES – REVISÃO BIBLIOGRÁFICA.....	33
4.1. PLANEJAMENTO DE EXPERIMENTOS - DoE	33
4.1.1. Projeto de blocos completos aleatórios	34
4.1.2. Quadrado Latino	35
4.1.3. Fatorial Completo	37
4.1.4. Fatorial Fracionado	40
4.1.5. Compósito Central	42
4.1.6. Aleatório.....	43
4.1.7. Sequências de Halton, Faure e Sobol	44

4.1.8. Hipercubo Latino	47
4.1.9. Escolha do DOE.....	51
4.2. METAMODELOS OU SUPERFÍCIES DE RESPOSTA.....	53
4.2.1. Método dos Mínimos Quadrados	57
4.2.2. Superfície de Resposta Ótima.....	58
4.2.3. Sherpard e K-Nearest	59
4.2.4. Kriging	60
4.2.5. Polynomial SVD – Decomposição em valor singular polinomial	64
4.2.6. RBF – Função de Base Radial	64
4.2.7. Neural Network – Redes Neurais.....	66
5. OTIMIZAÇÃO NO modeFRONTIER	69
5.1. MODELO INICIAL – RCAR	69
5.2. OTIMIZAÇÃO POR METAMODELOS NO MODEFRONTIER	70
5.3. VARIÁVEIS DE ENTRADA	72
5.4. SIMULAÇÃO NUMÉRICA DO PROBLEMA.....	74
5.5. OTIMIZAÇÃO – OBJETIVOS.....	78
5.6. RESTRIÇÕES.....	80
5.7. PRÉ-PROCESSAMENTO.....	81
5.8. PÓS-PROCESSAMENTO.....	85
5.9. ALGORITMO DO modeFRONTIER – FAST	86
5.9.1. Algoritmo genético no modeFRONTIER	88
6. RESULTADOS	92
6.1. MATRIZ DE DISPERSÃO	92
6.2. PONTOS FORA DA CURVA E MODELOS INFATÍVEIS	93
6.3. FRONTEIRA DE PARETO.....	95
6.4. EXEMPLOS DE CRITÉRIOS PARA A DECISÃO FINAL.....	96

6.5. COMPARAÇÃO COM UMA <i>CRASH BOX</i> DA INDÚSTRIA	99
7. CONCLUSÃO E TRABALHOS FUTUROS	103
8. REFERÊNCIAS.....	104
ANEXO A	108
ANEXO B	109
ANEXO C	110
ANEXO D	114

1. INTRODUÇÃO

A concorrência do setor automotivo nos últimos anos tem aumentado a passos largos. Com a base de dados do Denatran (DENATRAN, 2016), vemos que a frota de veículos aumentou de 14,4 automóveis para cada 100 habitantes em 2001 (frota de aproximadamente 24,6 milhões de veículos, considerando apenas automóveis, caminhonetes, caminhonetes e utilitários) para 29,5 autos/100hab em Abril/2016 (60,6 milhões de veículos) e 31,6 autos/100hab em Julho/2018 (65,4 milhões de veículos).

Neste período também houve grande aumento do número de acidentes veiculares. Segundo a Associação Brasileira de Prevenção a Acidentes de Trânsito (VIAS SEGURAS, 2018), o número de mortes por acidentes cresceu de 35.105 em 2004 para 41.151 em 2017. Esses dados foram obtidos pelo seguro DPVAT, em (SEGURADORA LÍDER, 2017)

Mais marcas estão disponíveis no mercado. Uma frota maior combinada com a persistência de acidentes no trânsito traz a necessidade de desenvolvimentos mais eficientes, com estruturas mais robustas e seguras para seus ocupantes.

Aproveitando a grande evolução da capacidade computacional do século XXI, os fabricantes de veículos têm usado a virtualização de testes e protótipos em fases avançadas de seus projetos. Pode-se, por exemplo, avaliar o quanto o espaço do habitáculo de um veículo de passeio diminui durante um impacto – infligindo ou não danos em seus ocupantes.

No desenvolvimento de um veículo, muitas vezes os designers se inspiram em formas da natureza. O exemplo clássico é a forma externa – as curvas externas tentam se aproximar da forma de uma gota d'água em queda livre, que minimiza o atrito na superfície de contato com o ar.

O uso de modelos matemáticos em produtos não é novo. Reproduzindo formas da natureza, temos embalagens de iogurte que minimizam o material utilizado e maximizam o volume contido – como bolhas de sabão. Estruturas em forma de colmeia são utilizadas para aumentar a troca de calor em filtros de ar; cúpulas de igreja são construídas com a revolução da catenária, forma natural que uma corda ou corrente assumem quando presa pelas duas pontas e deixadas à ação da gravidade,

aumentando a resistência ao peso próprio e às intempéries. A relação de produtos com as formas naturais é clara na arte de dobradura de papéis – o origami.

A história dos origamis é inseparável da história do próprio papel. Usos antigos de dobradura de papel são vistos em guardanapos, tampas de garrafa, guarda-chuvas, caixas, sacolas, entre outros, como pode ser verificado em (HATORI, 2011; BRITT, 2012).

O fato da dobra induzir a flambagem com menor energia não passou despercebido. Uma experiência muito interessante foi proposta por (MA, 2011). Neste trabalho foram propostas novas formas de *crash box*, a partir da arte de dobraduras de folhas. Ma define e estuda cinco padrões para a composição de *crash boxes*. Estes serão os padrões iniciais utilizados neste trabalho.

Este trabalho visa utilizar técnicas de otimização modernas, aliadas ao poder computacional disponível nos dias de hoje, para descobrir como as variáveis dimensionais de uma *crash box* origami podem ser combinadas para produzir uma melhor absorção de energia por unidade de massa, aliada a uma uniformidade de carga que reduz a diferença entre a força de pico inicial e a força média durante o esmagamento da *crash box* origami.

2. OBJETIVOS

Este trabalho visa o desenvolvimento de testes virtuais em *crash box* do tipo Origami, com o objetivo maior de otimizar a peça com técnicas de metamodelos e DoE (*Design of Experiments*).

Aliando simulações não-lineares e dinâmicas do Método dos Elementos Finitos com técnicas de aprendizado de máquina, chega-se à geometria otimizada.

Assim, dentre os muitos valores que podem ser assumidos para seus parâmetros geométricos, modelos de *crash box* origami que maximizem a energia absorvida por unidade de massa (*Specific Energy Absorbed – SEA*) e a uniformidade de carga (*Load Uniformity – LU*). Consequentemente, também teremos um menor pico de força no início do impacto.

A segunda etapa deste projeto envolve a comparação das opções viáveis com uma *crash box* da indústria, para validação de resultados.

3. REVISÃO DA LITERATURA

3.1. ESTATÍSTICAS DE ACIDENTES DE TRÂNSITO

A segurança veicular é um assunto corrente e de interesse público. Todos os dias, a mídia veicula casos de crianças, jovens, adultos e idosos que são afetados direta ou indiretamente por acidentes de veículos automotores. Além de danos inestimáveis às vidas das vítimas e suas famílias, a sociedade arca com pesados custos. Os mais de 300 mil veículos envolvidos em acidentes de trânsito ocorridos nas rodovias federais brasileiras no ano de 2014 oneraram a sociedade em aproximadamente R\$ 12,3 bilhões, sendo que mais de um terço desses custos (34,7%) estavam associados aos veículos, com danos materiais e perda de cargas, além dos procedimentos de remoção dos veículos acidentados (IPEA, 2015). A tabela 1 resume todos os gastos relacionados aos acidentes nas rodovias federais em 2014.

Tabela 1. Custo de acidentes nas rodovias federais em 2014. (IPEA, 2015)

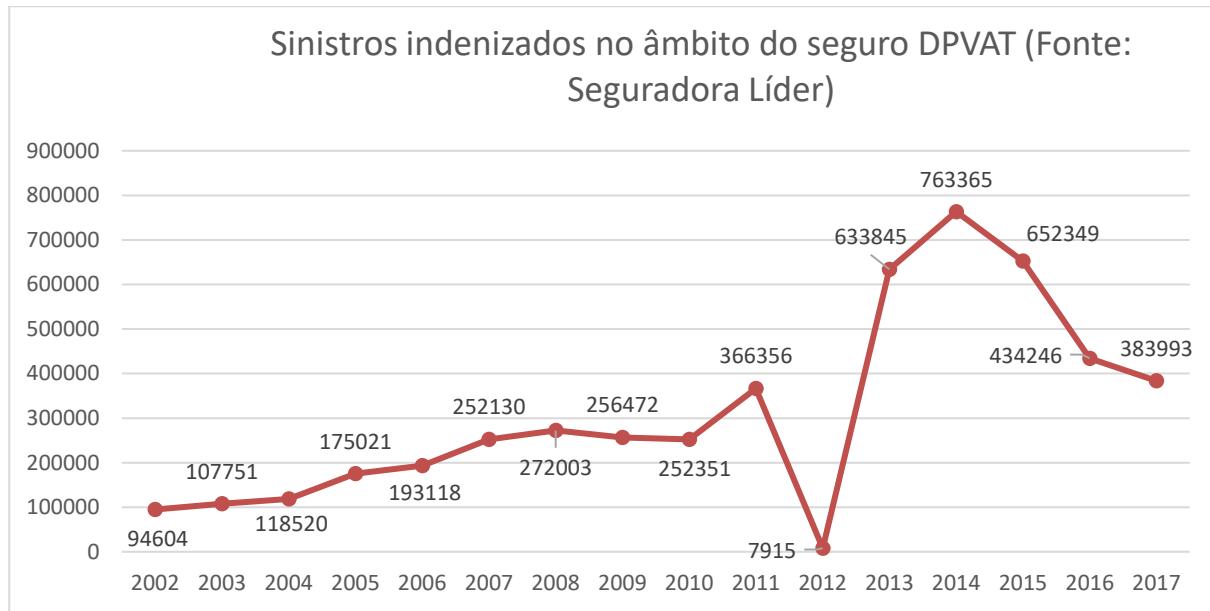
Custos	Descrição	Valor (R\$)	Valor (%)
Associados às pessoas	Despesas hospitalares; atendimento; tratamento de lesões; remoção de vítimas; e perda de produção.	7.958.883.201,04	64,72
Associados aos veículos	Remoção de veículos; danos aos veículos; e perda de carga.	4.268.587.302,76	34,71
Institucionais e danos a propriedades	Atendimento e processos e danos à propriedade pública e à privada.	70.850.037,27	0,58
Total		12.298.320.541	100,00

Em 2017, a Seguradora Líder declarou que o valor gasto em indenizações foi 41% da arrecadação bruta do DPVAT, totalizando R\$ 2,43 bilhões (SEGURADORA LÍDER, 2017).

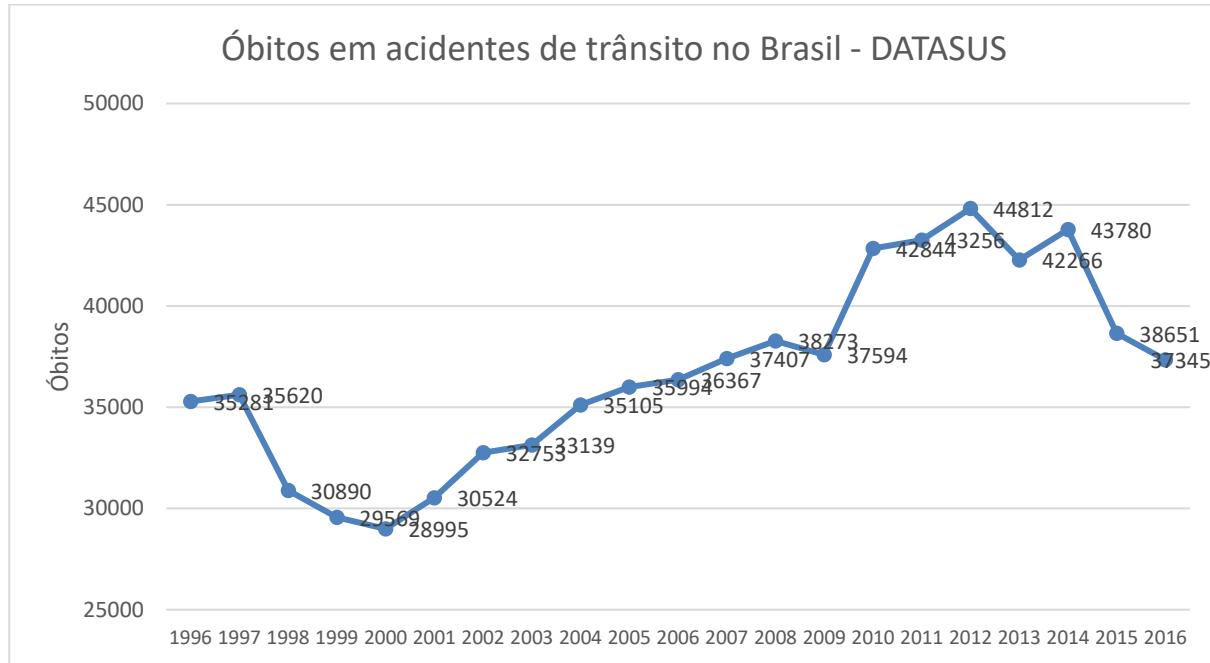
Só esses dados já seriam suficientes para incitar o desenvolvimento de novas soluções em segurança veicular, para uma frota crescente, mas a consciência dos perigos automotivos ainda não atingiu a todos. Em 2015, o Ministério da Saúde, o Ministério do Planejamento, Orçamento e Gestão e o Instituto Brasileiro de Geografia e Estatística publicaram a Pesquisa Nacional de Saúde 2013 (IBGE, 2015). Segundo essa pesquisa, 79,4% das pessoas de 18 anos ou mais de idade sempre usavam cinto de segurança no banco da frente quando andavam de automóvel, van ou táxi. Quanto

à utilização de cintos no banco de trás, o percentual é 50,2%. Este é um dado preocupante. O que podemos concluir é que uma grande parcela da população toma o automóvel como um objeto cuja segurança de utilização depende apenas da destreza do condutor, e descarta a função única do cinto de segurança – proteger os ocupantes.

No Brasil, o departamento nacional de trânsito – DENATRAN – recolhe anualmente de todos os proprietários de veículos o DPVAT. DPVAT é um seguro de cunho social criado em 1974 pela lei 6.194/74 para cobrir todos os cidadãos brasileiros vítimas de Danos Pessoais Causados por Veículos Automotores de Via Terrestre (ou por sua Carga a Pessoas Transportadas ou Não), e a seguradora Líder é a responsável pelos pagamentos dos seguros (Seguradora LÍDER, 2014). Segundo a seguradora, a frota brasileira aumentou de 53.315.094 veículos em 2008 para 94.325.914 veículos em 2017 – um aumento de 76,9%. Ainda apurou o pagamento de seguros de 272.003 sinistros em 2008 e 383.993 sinistros em 2017 (VIAS SEGURAS, 2017), conforme mostrado na Figura 3-1. Dados coletados pelo site Vias Seguras – a seguradora Líder apenas mantém em seu website os dados de 2011 a 2017. Lembrando que os sinistros não são pagos necessariamente no ano de ocorrência, comparamos os dados abaixo com os dados retirados do DATASUS (DATASUS, 2017).

Figura 3-1. Número de sinistros pagos pelo DPVAT

Fonte: (SEGURADORA LÍDER, 2014, 2016, 2017; VIAS SEGURAS, 2017)

Figura 3-2. Número de óbitos em acidentes de trânsito no Brasil

Fonte: (DATASUS, 2017)

Diante destes dados alarmantes, é cada dia mais necessário desenvolver sistemas de segurança veiculares passivos – que agem para a proteção do ocupante na hora das colisões.

Conhecido como capacidade *crashworthiness*, os veículos são concebidos de forma a proteger a integridade de seus ocupantes durante um impacto (WITTEMAN, 1999; BOIS et al., 2004; GHANNAM et al., 2011; LEE et al., 2013). Um destes sistemas é a própria estrutura do veículo, e sua capacidade de absorção de energia. A propriedade de flambagem controlada e progressiva de seus subcomponentes tem grande influência no progresso desta área (HSU; JONES, 2004; KOKKULA et al., 2006; KARAGIOZOVA; ALVES, 2008). Veremos no tópico 3.3 como funcionam alguns dos mecanismos de absorção de energia disponíveis na literatura.

3.2. SEGURANÇA VEICULAR

Em vista do panorama de fatalidades apresentado e da sempre crescente frota automotiva mundial, a maioria dos países criaram legislações específicas para a segurança veicular. No Brasil, estas legislações são criadas e geridas pelo CONTRAN, Conselho Nacional de Trânsito. Em 1941, o CONTRAN publicou sua primeira resolução, a qual se tratava de ultrapassagem entre ônibus. Até junho de 1997 seriam editadas mais 837 resoluções (ONSV, 2014).

A legislação brasileira prevê 29 itens de segurança obrigatórios nos veículos automotivos. Em 2014, o CONTRAN publicou nova resolução obrigando aos fabricantes a adoção de freios ABS e *airbags* frontais em todo veículo montado em território nacional. Em 2016, o uso obrigatório de faróis baixos nas rodovias, que já era previsto em resolução do CONTRAN desde 1998, ganhou força pela Lei 13.290/2016 – demonstrando a constante evolução das leis neste assunto.

Além disso, organizações voltadas para a proteção de ocupantes e pedestres foram fundadas. A Global NCAP, *Global New Car Assessment Program*, é a maior articuladora desses testes. Em seu site (www.globalncap.org) estão descritos seus objetivos: promover e conduzir pesquisas e programas de ensaios veiculares que atestam a segurança e características ambientais de veículos automotores e disseminar a comparação de seus desempenhos para o público; e promover o desenvolvimento de novos programas de avaliação independente, fornecendo suporte financeiro e assistência técnica, facilitando também a cooperação internacional entre os programas (GLOBAL NCAP, 2017). As avaliações sobre a segurança de veículos

destes programas são realizadas a partir de ensaios geralmente não exigidos pela legislação.

Desde sua fundação, a Global NCAP já articulou, junto à FIA (Federação Internacional de Automóveis) e a órgãos locais, nove NCAPs, mostrados na Figura 3-3. Os ensaios realizados por cada instituição são variados, mas semelhantes entre si. No Brasil, o órgão Latin NCAP baseia seus testes e avaliações no padrão europeu, pontuando 0 a 5 estrelas em três grandes áreas: proteção ao ocupante, proteção às crianças e sistemas de segurança primária. Além disso, premia os veículos que oferecem tecnologia de prevenção de acidentes e proteção ao pedestre.

Os ensaios realizados pelas NCAP da Europa e Américas podem ser vistos no quadro do ANEXO A .

Figura 3-3. Programas NCAP ao redor do mundo



Fonte: (GLOBAL NCAP, 2017)

3.3. MECANISMOS DE ABSORÇÃO DE ENERGIA

O conceito e desenvolvimento de sistemas de atenuação de impacto através da dissipação da energia cinética de maneira controlada recebeu muita atenção nos anos recentes, particularmente em relação à segurança veicular. O uso de componentes estruturais metálicos em tais sistemas, que dissipam a energia enquanto sofrem deformação plástica, foi revisada por Johnson e Reid (JOHNSON; REID, 1978) e citados por Norman Jones em (JONES; WIERZBICKI, 1983).

Uma revisão mais recente sobre o tema foi escrita por (LU; YU, 2003). Neste livro, os autores descrevem as deformações elasto-plásticas de tubos circulares (anéis), tubos de parede fina, de secções circular e retangular, tubos *top hat* e *double hat*, comparações entre deformações com tubos vazios e tubos preenchidos com espumas, estruturas celulares (como colmeias) e materiais compósitos.

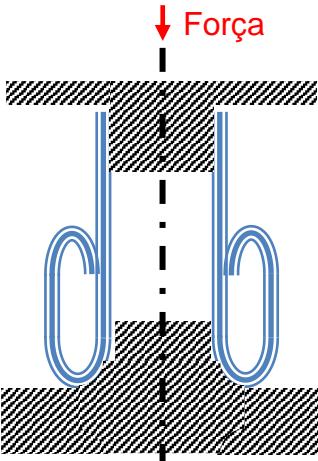
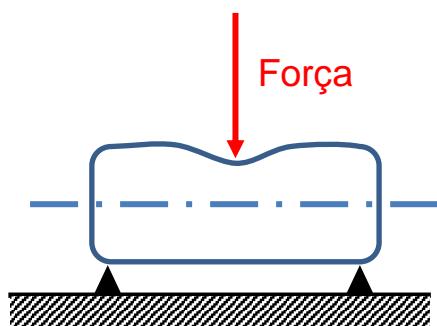
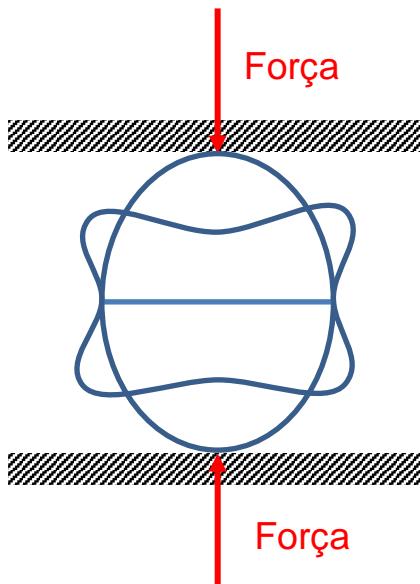
Para análise da efetividade de cada dispositivo, os autores geralmente levam em conta dois parâmetros: a absorção de energia específica (SEA), definida como a energia absorvida por unidade de massa, e a uniformidade de carregamento (LU), definida como a razão entre a força de pico sobre a força média de esmagamento. A força de pico é a maior força de reação resultante durante o processo de esmagamento e a força média de esmagamento é a energia absorvida total dividida pela distância final de esmagamento.

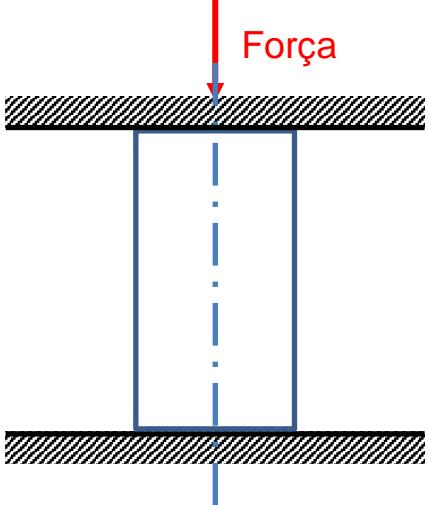
Neste capítulo apresentaremos alguns mecanismos de absorção de energia, e suas características mais relevantes para o estudo.

3.3.1. TUBO CIRCULAR

Por conta de sua alta utilização como elementos estruturais, tubos são considerados a forma mais comum e provavelmente a mais antiga a ser utilizada em absorvedores de energia. Em (ALGHAMDI, 2001) e (LU; YU, 2003), alguns modos de dissipar energia plástica em tubos metálicos através de vários modos de deformação são descritos, incluindo:

Tabela 2. Modos de deformação de tubos metálicos.

Modo	Esquema
Inversão	 A schematic diagram showing a blue U-shaped tube being inverted. A red arrow labeled "Força" (Force) points downwards from the top fixed support towards the bottom support. The tube is shown in its initial upright position and then again in its inverted state.
Compressão lateral em carregamento concentrado (indentação)	 A schematic diagram showing a rectangular tube being compressed laterally. A red arrow labeled "Força" (Force) points downwards from a central point on the top surface towards the bottom support. Two black arrows at the base indicate the direction of lateral compression.
Compressão lateral com carregamento uniforme (planificação);	 A schematic diagram showing a circular tube being compressed laterally with uniform loading. A red arrow labeled "Força" (Force) points downwards from the top surface towards the bottom support. The tube is shown in its initial circular shape and then again in a flattened, elliptical shape.

Modo	Esquema
Compressão axial	

Os modos de compressão lateral e indentação lateral podem induzir a uma curva de força x deslocamento suave, mas o SEA é relativamente baixo pois o curso de impacto é pequeno. Inversão de tubos pode gerar um SEA alto, mas requer matrizes especiais para que os modos de deformação sejam induzidos. Compressão axial tem os méritos de um curso de impacto relativamente longo, alto SEA, e simplicidade mecânica, sendo, portanto, vastamente aplicado na prática. O principal mecanismo de dissipação energética deste tipo de dispositivo é a deformação plástica do material. (MA, 2011)

Alghamdi, 2001, ressalta que o tubo circular fornece uma força de operação razoavelmente constante, o que é, em algumas aplicações, a característica principal do absorvedor de energia. Além disso, Alghamdi destaca a capacidade de absorção energética e comprimento do choque por unidade de massa. Por exemplo, ao comparar a compressão lateral com a compressão axial, o modo de flambagem axial tem uma capacidade de absorção de energia específica que é aproximadamente 10 vezes maior que quando o mesmo tubo é comprimido lateralmente por duas chapas planas. (ALGHAMDI, 2001)

No mesmo artigo, o autor cita classificações dos modos de flambagem axial dos tubos cilíndricos sob carregamento quase-estático. Elas são combinações entre os modos concertina (axissimétrico) ou diamante (não-axissimétrico), como mostrado na Figura 3-4.

Figura 3-4. Modos de deformação final de tubos de parede fina que sofreram impacto axial dinâmico.



Fonte: (ALVES, 2018)

3.3.2. TUBOS QUADRADOS

Como os tubos circulares, tubos quadrados também exibem vários modos de falha dependendo da razão entre a largura do tubo e a sua espessura, b / t . Tubos muito finos, por exemplo com a relação $b / t = 100$, usualmente falham no modo não compacto, Figura 3-5 (esquerda). Esse modo é indesejável da perspectiva de absorção energética pois poderia resultar na flambagem de Euler¹, e isto reduziria consideravelmente a capacidade de absorção de energia. Tubos de espessura moderada normalmente falham no modo simétrico, Figura 3-5 (direita). (MA, 2011)

¹ Fenômeno no qual tubos relativamente longos tendem a flambar globalmente.

Figura 3-5. Modo de flambagem não compacto (esquerda) e modo simétrico (direita)



Fonte: (SILVA; DRIEMEIER; ALVES, 2019)

Tubos circulares e quadrados são os dispositivos tubulares de absorção de energia mais usados devidos à sua disponibilidade e baixo custo. Muitos trabalhos foram dedicados à melhoria da performance dos tubos circulares e quadrados. O desafio principal no design é como eliminar efetivamente a alta força inicial de deformação, mantendo ou até mesmo melhorando sua capacidade de absorção energética. (MA, 2011)

Um meio simples, mas eficaz de remover a alta força de deformação inicial é introduzir imperfeições geométricas no tubo, como corrugações estampadas, dentes, e padrões estampados nas laterais das superfícies de tubos. No caso deste trabalho, os padrões estampados seguem a técnica de dobraduras de folhas, origami.

3.4. ORIGAMI

3.4.1. HISTÓRIA DO ORIGAMI

A história do origami, também conhecido como arte da dobra de papel, não pode ser contada sem trazer à tona a história do próprio papel. A manufatura de papel pode ser rastreada ao ano 105 d.C., quando Ts'ai Lun, um oficial ligado à corte imperial da China, criou uma folha de papel usando folhas de amora e fibras de entrecasca junto a redes de pesca, trapos velhos e sobras de linho. Em sua lenta

trajetória rumo ao ocidente, a arte de fazer papel alcançou Samarkand, na Ásia Central, em 751; e em 793 o primeiro papel foi feito em Bagdá nos tempos de Hārūn ar-Rashīd, na era dourada da cultura Islâmica que trouxe a manufatura de papel para as fronteiras da Europa. (BRITT, 2012)

No século XIV já existiam diversas fábricas de papel na Europa, particularmente na Espanha, Itália, França e Alemanha. A invenção das prensas de papel na década de 1450 aumentou em muito a demanda por papel, e favoreceu sua utilização em outras áreas além da escrita. (BRITT, 2012)

A história acima, no entanto, já foi contestada. Em (HATORI, 2011), o autor relata a descoberta de uma peça em Fan Ma Tan in 1986, que estima-se ter sido feita no meio do século II a.C. Além disso, outros tipos de papéis feitos de casca de árvore, chamados de *amate* na América Central, *kapa* no Havaí e *tapa* no Sudeste Asiático datam até o ano 5000 a.C.

Escrever uma história comprehensiva da arte de dobradura de papel é quase impossível, sendo que informações sobre esta arte prévias ao século XV são basicamente inexistentes. Apesar de muitos acreditarem que o origami se originou no Japão na era Heian (794-1185), não há indícios da dobradura de papel como tradição neste período. (HATORI, 2011)

Os primeiros usos de origami no Japão parecem ter sidos cerimoniais. O *Noshi*, forma de invólucros para *noshi-awabi*, ou carne seca de um tipo de molusco chamado haliote, encontrado na maioria dos mares temperados. Também eram usados como embrulhos de garrafas de *sake*, e outros presentes. (HATORI, 2011)

Em contraste, a origem do origami ocidental é de certificados de batismo, dobrados unindo os quatro vértices de um quadrado ao seu centro, e repetindo a dobra no quadrado menor. Esta prática parece ter sido popular na Europa Central nos séculos XVII e XVIII.

3.4.2. ORIGAMI E ENGENHARIA

O origami ganhou espaço na engenharia na década de 1970, quando engenheiros perceberam que painéis solares poderiam ser compactados com um padrão de dobras em um pequeno volume para transporte nas *shuttles*, e facilmente desdobrá-los no espaço. Por um tempo, o espaço foi o único local que uniu a

engenharia ao origami. Estudos prosseguiram no meio acadêmico, e alguns outros projetos surgiram usando o conceito de placas rígidas ligadas por dobradiças e juntas esféricas para a economia de volume. (MERALI, 2011)

Um nome proeminente nesta área é Zhong You. You começou nesta área em 1990, na Universidade de Shanghai, e depois mudou-se para a Inglaterra, para estudar na Universidade de Cambridge. Lá ele conheceu seu colega Simon Guest, que usava o truque dos painéis solares para armazenar quilômetros de tubos que seriam desdobrados em voos espaciais. You percebeu que dobradiças e peças deslizantes poderiam se grudar ou emperrar, especialmente quando as estruturas fossem abertas e fechadas muitas vezes. Dobraduras feitas com origami rígido seriam muito mais confiáveis. (MERALI, 2011)

Outra figura icônica no meio acadêmico sobre Origami é Robert Lang. Lang é um físico que trabalhava com lasers na NASA, e deixou seu emprego na indústria aeroespacial para desenvolver trabalhos com origami – matematização, designs e até programas de computador para concepção de novos padrões de dobradura.

Robert Lang organiza uma conferência chamada *International Meeting of Origami Science, Mathematics and Education*. Em 2011, You levou o trabalho de seu orientado, Jiayao Ma, para a quinta edição da conferência de Lang, a *crash box* origami.

3.5. CRASH BOX ORIGAMI

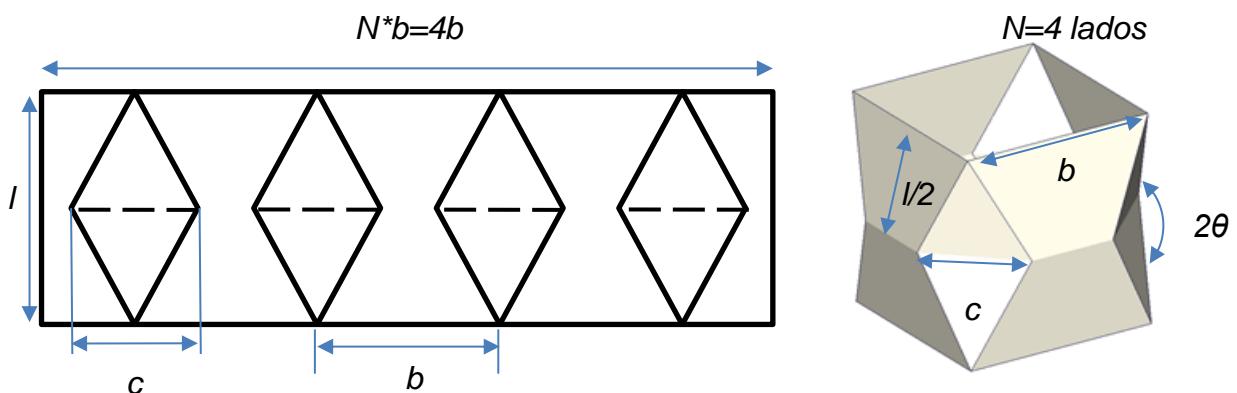
Em sua tese, Jiayao Ma revisou a literatura de tubos utilizados como absorvedores de energia. A principal dificuldade neste tipo de design é como eliminar a alta força inicial de flambagem sem prejudicar a capacidade de absorção energética do dispositivo. E uma solução simples para este problema é a introdução de imperfeições geométricas no tubo.

As razões pelas quais Ma escolheu a forma de origami para a *crash box* são que, primeiro, os padrões de dobra atendem a esta aplicação em particular, já que o material com o qual lidamos é de espessura fina. Em segundo lugar, a maioria das formas origami é desenvolvível. Portanto, padrões origami podem ser manufaturados na superfície de tubos sem muitas distorções. Por último, ferramentas de

modelamento matemático de origami estão disponíveis, o que facilita a exploração de designs de padrões variados. (MA, 2011)

A investigação de Ma abrange algumas formas de origami, variações da forma básica apresentada na Figura 3-6. A partir disso, o autor realiza uma análise paramétrica de 4 diferentes grupos-padrões: tubo quadrado, tubo retangular, tubos de seções poligonais diversas e tubos cônicos.

Figura 3-6. Forma básica de padrão de origami para tubos retangulares (a) e módulo de um tubo origami retangular (b).



Fonte: Figura baseada em (MA, 2011)

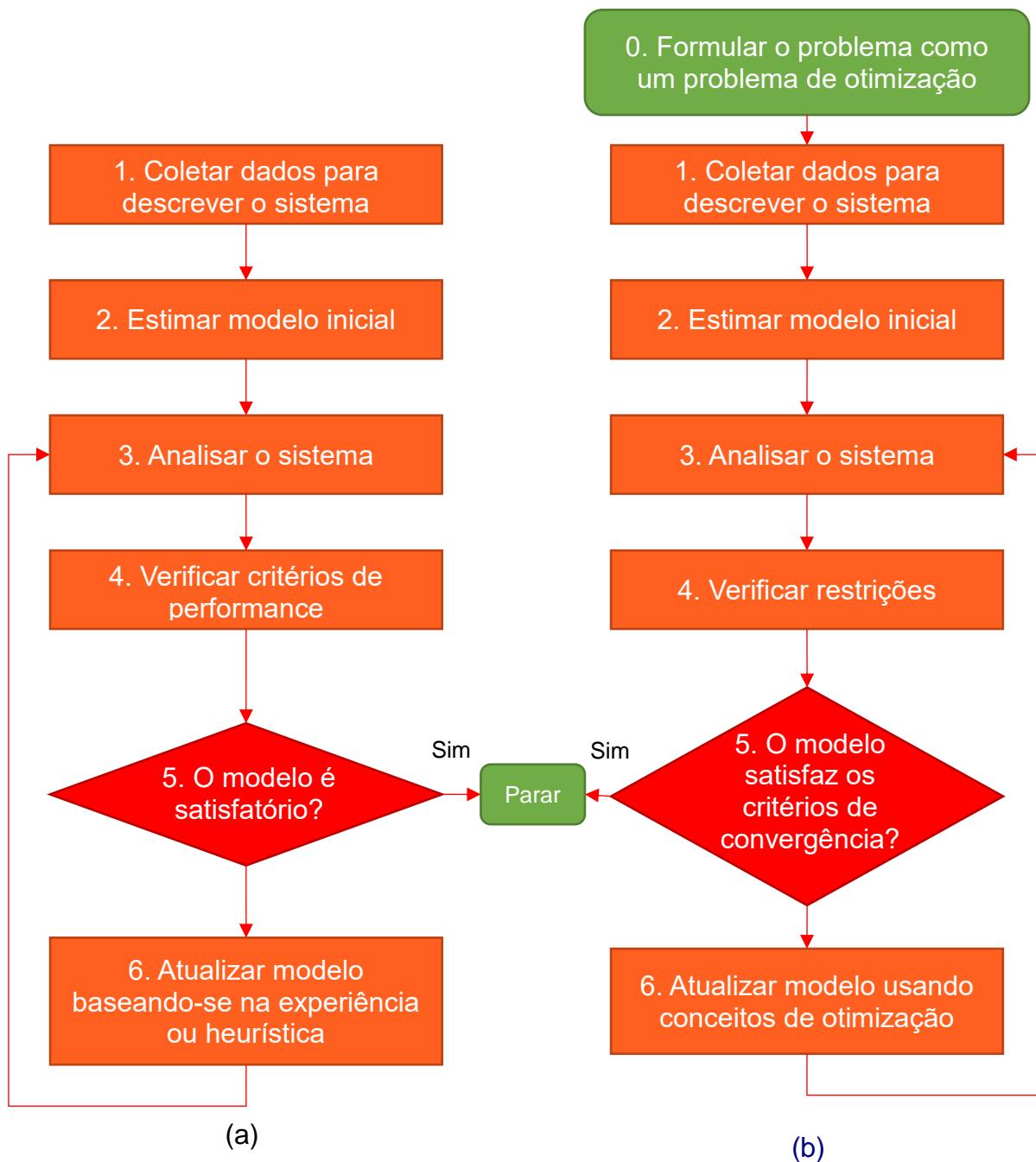
3.6. MÉTODO DE SÍNTESE

O método clássico de desenvolvimento de produtos em engenharia é a abordagem de análise. Nessa abordagem, o produto é elaborado de forma linear. A função do produto é definida, através da coleta de dados que descrevem o sistema. Em seguida, o desenho inicial do produto é criado, para então o sistema ser analisado. Caso o sistema atinja os critérios de performance, e o design seja satisfatório, o desenvolvimento é concluído.

Existe, no entanto, um outro método. O método de síntese, também conhecido como método do design ótimo, tenta levar o produto não somente a um design satisfatório, mas ao design que melhor cumpre a função, considerando as restrições do problema. Em (ARORA, 2016), o autor define o fluxograma dos dois procedimentos, reproduzido na Figura 3-7. Ainda ressalta que ambos os métodos são iterativos, como indicado no laço entre os blocos 6 e 3. Ambos os métodos têm blocos

que requerem cálculos similares e outros que requerem cálculos distintos. Os traços principais dos dois processos são:

Figura 3-7. Comparação entre (a) método de design convencional (análise) e (b) método de design do ponto ótimo (síntese)



Fonte: (ARORA, 2016)

0. O método da síntese (ou design ótimo) possui o bloco 0, onde o problema é formulado como um problema de otimização. Define-se a função objetivo que mensura o mérito de diferentes modelos.
1. Ambos os métodos requerem dados iniciais para descrever o sistema no bloco 1.
2. Ambos os métodos requerem que um design inicial seja estimado no bloco 2.
3. Ambos os métodos requerem a análise do sistema no bloco 3.
4. No bloco 4, a abordagem analítica verifica se os critérios de performance foram respeitados, enquanto no método do design ótimo verifica-se se todas as restrições do problema formulado no bloco 0 são satisfeitas.
5. No bloco 5, critérios de parada são verificados nos dois métodos, e a iteração é interrompida caso os critérios sejam satisfeitos.
6. No bloco 6, o método de análise convencional atualiza o design baseado na experiência e intuição do engenheiro ou projetista e outras informações advindas de testes e projetos anteriores; o método de design ótimo utiliza conceitos de otimização e procedimentos para atualizar o design atual.

Com a competitividade do mercado automotivo, há a necessidade de executar projetos em tempos menores, com o maior número possível de simulações na fase de pré-projeto, e a menor quantidade possível de protótipos. Portanto, o método de design ótimo traz uma adequação maior aos novos tempos que o método de análise.

O propósito deste trabalho é aplicar o método de síntese, figurado aqui pela implementação de metamodelos e heurística, para o desenvolvimento de um novo conceito de *crash box*.

4. MODELAGEM DAS SIMULAÇÕES – REVISÃO BIBLIOGRÁFICA

4.1. PLANEJAMENTO DE EXPERIMENTOS - DoE

Planejamento de experimentos, do inglês *Design of Experiments*, não é uma técnica de otimização em si. DoE é o método de escolha das amostras no campo de todos os experimentos possíveis, de modo a extrair o máximo de informações utilizando a menor quantidade de recursos possíveis, ou seja, um baixo número de amostras. (CAVAZZUTI, 2013)

Neste trabalho, o DoE servirá de base para alimentar nosso metamodelo – uma superfície de resposta, RSM – *Response Surface Modelling*. Segundo (CAVAZZUTI, 2013), chamamos de RSM toda técnica aplicada com o intuito de interpolar ou aproximar a informação proveniente de um DOE. Diferentes métodos de interpolação ou aproximação (linear, não-linear, polinomial, estocástico, ...) resultam em diferentes técnicas RSM. A ideia é criar uma superfície n -hiperdimensional no espaço $(n + 1)$ -dimensional, criado por n variáveis mais a função objetivo.

O resultado da função objetivo em cada experimento determinado pela DoE será um ponto para a interpolação da superfície de resposta. Nessa superfície é possível aplicar técnicas de otimização propriamente ditas, com resultado relativamente rápido, já que será baseado na avaliação analítica da interpolação dos resultados dos experimentos.

As técnicas de DOE e RSM trarão enorme economia de tempo e recursos computacionais – característica essencial, já que cada simulação de veículo completo pode levar dias para ser concluída, dependendo do poder de processamento disponível.

Neste capítulo, iremos abordar algumas técnicas de DOE, focando nas técnicas apropriadas para construção de superfície de resposta, segundo a Tabela 3.

Tabela 3. Tabela sinóptica de métodos DOE (CAVAZZUTI, 2013)

Método	Número de experimentos	Apropriado para
RCDB	$N(L_i) = \prod_{i=1}^k L_i$	Foco em um fator primário utilizando técnicas de blocos
Quadrados Latinos	$N(L) = L^2$	Foco em um fator primário com menor custo
Fatorial Completo	$N(L, k) = L^k$	Computar todos os efeitos principal e de interação, e construir superfícies de resposta
Fatorial Fracionado	$N(L, k, p) = L^{k-p}$	Estimar os efeitos principal e de interação
Compósito Central	$N(k) = 2^k + 2k + 1$	Construir superfícies de resposta
Box-Behnken	$N(k)$ das tabelas	Construir superfícies de resposta quadráticas
Plackett-Burman	$N(k) = k + 4 - \text{mod}\left(\frac{k}{4}\right)$	Estimar os efeitos principais
Taguchi	$N(k_{in}, k_{out}, L) = N_{in}N_{out}, N_{in}(k_{in}, L), N_{out}(k_{out}, L)$ das tabelas	Endereçar a influência das variáveis de ruído
Aleatório	Escolhido pelo projetista	Construir superfícies de resposta
Halton, Faure, Sobol	Escolhido pelo projetista	Construir superfícies de resposta
Hipercubo Latino	Escolhido pelo projetista	Construir superfícies de resposta
Design ótimo	Escolhido pelo projetista	Construir superfícies de resposta

4.1.1. Projeto de blocos completos aleatórios

Do inglês *Randomized Complete Block Design*, RCDB é uma técnica de DoE baseada em blocos. Em um experimento sempre existem diversos fatores que podem afetar o resultado final. Alguns fatores não podem ser controlados, então devem ser escolhidos aleatoriamente durante a realização do experimento para que na média, a sua influência seja desprezível. Outros fatores são controláveis. RCDB é útil quando estamos interessados em focar em um fator particular cuja influência na variável de resposta é supostamente mais relevante. Nós nos referimos a este parâmetro como *fator primário*, fator de projeto, fator de controle ou fator de tratamento. Os outros fatores são desprezados, e entram na estatística como erros ou resíduos. Como estamos interessados em focar nossa atenção no fator primário, é interessante usar a técnica dos blocos nos outros fatores, isto é, mantendo constantes os valores dos fatores de erro, um lote de experimentos é realizado onde o fator primário assume todos os seus possíveis valores. Para completar o projeto de blocos aleatórios tal lote

de experimentos é realizado para toda combinação possível dos fatores de erro. (CAVAZZUTI, 2013)

Vamos assumir que em um experimento existem k fatores controláveis X_1, \dots, X_k e um deles, X_k , é de suma importância. O número de níveis de cada fator é L_1, L_2, \dots, L_k . Se n é o número de repetições de cada experimento, o número total de experimentos necessários para completar um RCDB é $N = L_1 * L_2 * \dots * L_k * n$. No que se segue consideraremos $n = 1$.

Vamos assumir $k = 2$, $L_1 = 3$, $L_2 = 4$, X_1 fator de erro, X_2 fator primário, então $N=12$. Sendo os três níveis de X_1 : A, B, e C, e os quatro níveis de X_2 α, β, γ e δ . O conjunto de experimentos para completar o projeto de experimentos RCBD está na Tabela 4. (CAVAZZUTI, 2013)

Tabela 4. Exemplo de RCBD para $k = 2$, $L_1 = 3$, $L_2 = 4$, $N = 12$, X_1 fator de erro, X_2 fator primário.
(CAVAZZUTI, 2013)

The diagram illustrates the mapping from a 3x4 experimental design table to a 4x4 treatment combination table. On the left, a table shows 12 experiments (N=12) across 3 blocks (Bloco 1, Bloco 2, Bloco 3) with 4 treatments each. The treatments are represented by combinations of levels for factors X_1 and X_2 . An arrow points from this table to a second table on the right, which lists all 12 treatment combinations in a 4x4 grid.

Número do Experimento		Nível do Fator	
		X_1	X_2
Bloco 1	1	A	α
	2	A	β
	3	A	γ
	4	A	δ
Bloco 2	5	B	α
	6	B	β
	7	B	γ
	8	B	δ
Bloco 3	9	C	α
	10	C	β
	11	C	γ
	12	C	δ

X_1	X_2			
	$A\alpha$	$A\beta$	$A\gamma$	$A\delta$
$B\alpha$	$B\beta$	$B\gamma$	$B\delta$	
$C\alpha$	$C\beta$	$C\gamma$	$C\delta$	

4.1.2. Quadrado Latino

O Quadrado Latino é baseado na mesma ideia do RCBD. No entanto, o RCBD tem um tamanho de amostra que aumenta muito com a quantidade de fatores. O Quadrado Latino objetiva realizar apenas um único experimento por bloco, reduzindo

o número de amostras requeridas sem comprometer muito a relevância do fator primário. (CAVAZZUTI, 2013)

Um Quadrado Latino apresenta algumas condições necessárias para ser aplicável: $k = 3$, X_1 e X_2 como fatores secundários, X_3 como fator primário, $L_1 = L_2 = L_3 = L$. O tamanho da amostra para este método é $N = L^2$. (CAVAZZUTI, 2013)

Em um esquema representativo, os primeiros fatores secundários são divididos em uma tabela com L linhas e L colunas. Em cada célula, uma letra maiúscula é escrita de forma que cada linha e cada coluna recebam as primeiras L letras do alfabeto latino uma única vez. O número da linha e o número da coluna indicam os níveis dos fatores secundários, e as letras maiúsculas representam o nível do fator primário. (CAVAZZUTI, 2013)

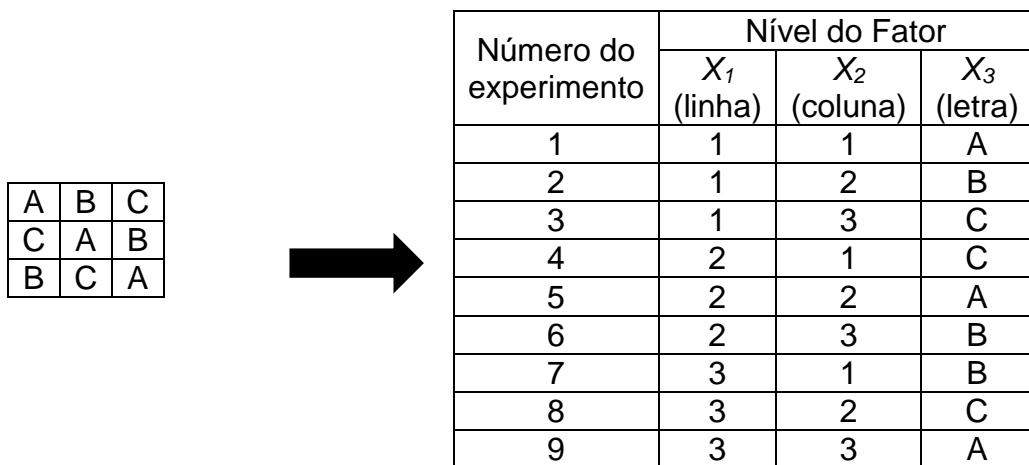
Na verdade, a ideia do Quadrado Latino é aplicável para qualquer $k > 3$, e os primeiros 3 designs são conhecidos por diferentes nomes:

- Se $k = 3$: Quadrado Latino,
- Se $k = 4$: Quadrado Greco-Latino,
- Se $k = 5$: Quadrado Hiper-Greco-Latino,

Nos projetos com $k > 3$, os fatores secundários adicionais são inclusos na tabela como letras gregas e outros símbolos. Isto é feito em respeito à regra que em cada linha e em cada coluna os níveis de fatores não devem ser repetidos, e à regra adicional que cada fator deve seguir um padrão diferente de letras/números na tabela. A regra adicional permite que as influências de duas variáveis não sejam completamente comprometidas entre si. Para cumprir esta regra, não é possível que um Quadrado Hiper -Greco-Latino tenha $L = 3$, já que apenas dois padrões de letras são possíveis em uma tabela 3x3. Se $k = 5$, L deve ser ≥ 4 .

A vantagem do Quadrado Latino é a característica deste método de manter separados diversos fatores secundários em um custo relativamente baixo em termos de tamanho da amostra. Por outro lado, como os fatores nunca são alterados um por vez de amostra em amostra, seus efeitos são parcialmente confundidos. Um exemplo deste método pode ser visto na Tabela 5 (CAVAZZUTI, 2013).

Tabela 5. Exemplo de Quadrado Latino para $k = 3$, $L = 3$, $N = 9$ (CAVAZZUTI, 2013)



The diagram illustrates the conversion of a 3x3 Latin square design into a 3D cube representation. On the left, a 3x3 grid shows the initial design:

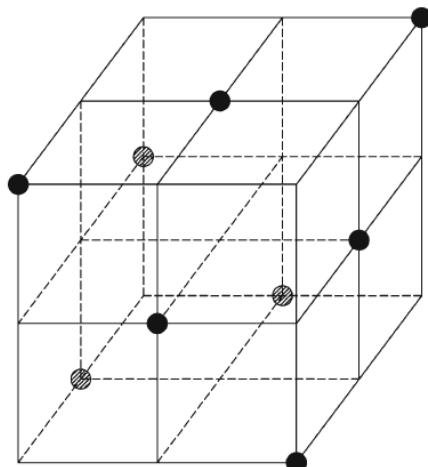
A	B	C
C	A	B
B	C	A

An arrow points from this grid to a 3D cube on the right. The cube has vertices labeled with the letters A, B, and C. The vertices are arranged such that each face of the cube contains one of the three letters. The vertices are labeled as follows: front face (bottom-left-front): A, front face (top-right-back): C, top face (bottom-right-back): B, top face (top-left-front): A, back face (bottom-left-front): B, back face (top-right-back): C.

Número do experimento

	Nível do Fator		
	X_1 (linha)	X_2 (coluna)	X_3 (letra)
1	1	1	A
2	1	2	B
3	1	3	C
4	2	1	C
5	2	2	A
6	2	3	B
7	3	1	B
8	3	2	C
9	3	3	A

Figura 4-1. Exemplo de quadrado latino para $k = 3$, $L = 3$, $N = 9$



Fonte: (CAVAZZUTI, 2013)

4.1.3. Fatorial Completo

A técnica do Fatorial Completo é intuitiva. Em sua forma mais simples, existem k fatores e $L = 2$ níveis por fator. As amostras são dadas por toda combinação possível entre os valores de cada fator. Portanto, o tamanho da amostra é $N = 2^k$. (CAVAZZUTI, 2013)

Diferentemente de outras técnicas DoE, este método e os seguintes não mais distinguem entre fatores primários e secundários a priori. Os dois níveis são chamados altos ("h", do inglês *high*) e baixo ("l", do inglês *low*) ou "+1" e "-1". Começando de

qualquer amostra dentro do esquema do fatorial completo, as amostras nas quais os fatores são alterados um por vez ainda são parte do espaço amostral. Esta propriedade permite que o efeito de cada fator sobre a variável de saída não seja confundido com os outros fatores. Por vezes, pôde-se encontrar na literatura fatoriais completos que também têm o ponto central no espaço de projeto adicionado às amostras. O ponto central é a amostra na qual todos os parâmetros possuem um valor que é a média entre seus níveis baixos e altos e em tabelas dos fatoriais completos 2^k podem ser designados como “m” (valor médio) ou “0”. (CAVAZZUTI, 2013)

Tabela 6. Exemplo de fatorial completo (CAVAZZUTI, 2013)

Número do Experimento	Nível do fator			Variável de resposta	Interações de 2 e 3 fatores			
	X_1	X_2	X_3		$X_1 \cdot X_2$	$X_1 \cdot X_3$	$X_2 \cdot X_3$	$X_1 \cdot X_2 \cdot X_3$
1	-1 (l)	-1 (l)	-1 (l)	$y_{l,l,l}$	+1	+1	+1	-1
2	-1 (l)	-1 (l)	+1 (h)	$y_{l,l,h}$	+1	-1	-1	+1
3	-1 (l)	+1 (h)	-1 (l)	$y_{l,h,l}$	-1	+1	-1	+1
4	-1 (l)	+1 (h)	+1 (h)	$y_{l,h,h}$	-1	-1	+1	-1
5	+1 (h)	-1 (l)	-1 (l)	$y_{h,l,l}$	-1	-1	+1	+1
6	+1 (h)	-1 (l)	+1 (h)	$y_{h,l,h}$	-1	+1	-1	-1
7	+1 (h)	+1 (h)	-1 (l)	$y_{h,h,l}$	+1	-1	-1	-1
8	+1 (h)	+1 (h)	+1 (h)	$y_{h,h,h}$	+1	+1	+1	+1

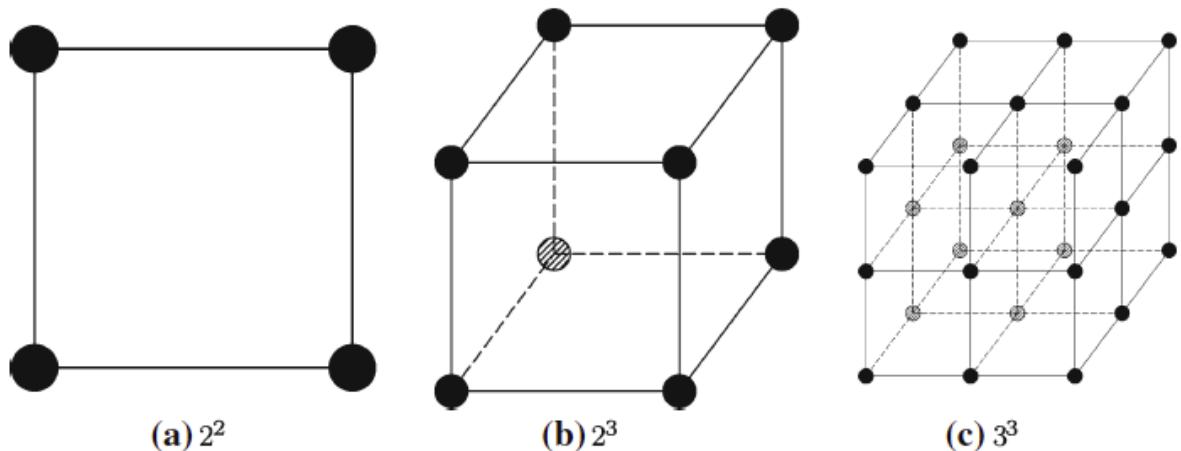
Consideremos um fatorial completo com três fatores e dois níveis por fator. O fatorial completo é um método de projetos de experimentos *ortogonal*. O termo ortogonal deriva do fato que o produto escalar entre as colunas de dois fatores quaisquer é zero. (CAVAZZUTI, 2013)

É definida como *interação principal* M de uma variável X a diferença entre a média das variáveis de resposta das amostras do nível alto e a média das variáveis de resposta das amostras do nível baixo. Na Equação 1 está expressa a interação principal M da variável X_1 . Expressões similares podem ser derivadas para as variáveis X_2 e X_3 . O efeito de interação de dois ou mais fatores é definido, similarmente, como a diferença entre a média de respostas no nível alto e nível baixo na coluna de interação. (CAVAZZUTI, 2013)

$$M_{X_1} = \frac{y_{h,l,l} + y_{h,l,h} + y_{h,h,l} + y_{h,h,h}}{4} - \frac{y_{l,l,l} + y_{l,l,h} + y_{l,h,l} + y_{l,h,h}}{4} \quad (1)$$

$$M_{X_1, X_2} = \frac{y_{l,l,l} + y_{l,l,h} + y_{h,h,l} + y_{h,h,h}}{4} - \frac{y_{h,l,l} + y_{h,l,h} + y_{l,h,l} + y_{l,h,h}}{4} \quad (2)$$

Figura 4-2. Exemplos de fatoriais completos L^k



Fonte: (CAVAZZUTI, 2013)

A vantagem do fatorial completo é que um uso muito eficiente dos dados é realizado, e não se confundem os efeitos dos parâmetros, tornando possível avaliar os efeitos principal e de interação claramente. Por outro lado, o tamanho da amostra cresce exponencialmente com o número de parâmetros e com número de níveis. A família de L^k projetos, ou seja, os projetos de fatorial completos onde o número de níveis é o mesmo para cada fator, é particularmente favorável para interpolação por superfícies de resposta polinomiais, dado que um projeto 2^k pode ser interpolado com uma forma bilinear completa, um projeto 3^k com uma forma biquadrática completa, um projeto 4^k uma forma bicúbica completa, e daí por diante. A Figura 4-2 mostra representação gráficas de alguns projetos fatoriais completos. (CAVAZZUTI, 2013)

4.1.4. Fatorial Fracionado

Conforme o número de parâmetros aumenta, um projeto de fatorial completo pode tornar-se muito oneroso para ser finalizado. A ideia de um projeto *Fatorial Fracionado* é realizar apenas experimentos de um subgrupo do projeto de fatorial completo. Ao fazê-lo, ainda é possível extrair uma boa quantidade de informações sobre os efeitos principais e algumas informações sobre os efeitos de interação. O tamanho da amostra do Fatorial Fracionado pode ser metade, ou um quarto, e assim por diante, do fatorial completo. As amostras devem ser devidamente escolhidas, em particular devem ser *balanceadas* e ortogonais. Por balanceados queremos dizer que o espaço amostral deve ser criado de tal forma que cada fator tenha o mesmo número de amostras para cada nível. (CAVAZZUTI, 2013)

Vamos considerar um fatorial fracionado que seja metade de um fatorial completo de 2^k . Será conhecido como fatorial fracionado 2^{k-1} . Assumindo $k = 3$, para construir a lista de amostras, começamos com um fatorial completo 2^{k-1} , e os níveis para os parâmetros adicionais são escolhidos como a interação entre alguns dos outros parâmetros. No nosso caso, podemos adicionar o produto $X_1 \cdot X_2$ ou $-X_1 \cdot X_2$. (CAVAZZUTI, 2013)

Tabela 7. Exemplo de fatorial fracionado 2^{3-1} (CAVAZZUTI, 2013)

Número do experimento	Nível do fator			
	$X_1(A)$	$X_2(B)$	$X_3 = X_1 \cdot X_2 (C)$	$I = X_1 \cdot X_2 \cdot X_3$
1	-1	-1	+1	+1
2	-1	+1	-1	+1
3	+1	-1	-1	+1
4	+1	+1	+1	+1

O projeto de fatorial fracionado na Tabela 7 é identificado pela *palavra geradora* $+ABC$, pois a multiplicação elemento-a-elemento da primeira coluna (A), segunda coluna (B) e terceira coluna (C) é igual à coluna de identidade (I). O efeito principal e o efeito de interação são calculados como no parágrafo anterior. No entanto, o preço a se pagar em tal projeto experimental é que o efeito principal do fator X_3 (C) não pode ser distinguido do fator de interação $X_1 \cdot X_2$ (AB). Em termos técnicos nós dizemos que o termo X_3 foi confundido com $X_1 \cdot X_2$. Contudo, este não é o único termo confundido: multiplicando as colunas apropriadamente, é fácil perceber que se

$C = AB$, nós temos $AC = A \cdot AB = B$ e $BC = B \cdot AB = A$, ou seja, todo efeito principal é confundido com um efeito de interação de dois-fatores. (CAVAZZUTI, 2013)

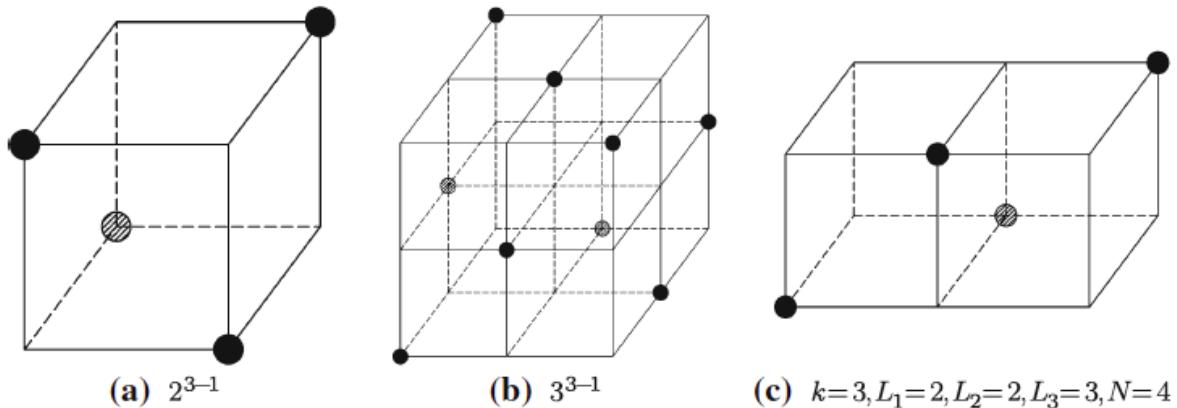
O projeto 2^{3-1} com palavra geradora $I = +ABC$ (ou $I = -ABC$) é um projeto de resolução III . Para denotar a resolução do projeto um numeral romano subscrito é usado ($2_{III}^{(3-1)}$). Um projeto é de resolução R se quaisquer efeitos q -fatores não são atrelados com outro efeito com menos de $R - q$ fatores. Isso significa:

- Em um projeto de resolução III os efeitos principais são atrelados com pelo menos os efeitos de 2-fatores;
- Em um projeto de resolução IV os efeitos principais são atrelados com pelo menos os efeitos de 3-fatores e os efeitos de 2-fatores são atrelados uns aos outros;
- Em um projeto de resolução V os efeitos principais são atrelados pelo menos com os efeitos de 4-fatores, e os efeitos de 2-fatores são atrelados pelo menos com os efeitos de 3-fatores.

Em geral, a definição de um projeto 2^{k-p} requerem p “palavras”. Considerando todos os principais atrelamentos, teremos $2^p - 1$ palavras. A resolução é igual ao menor número de letras em quaisquer das $2^p - 1$ palavras. Essas $2^p - 1$ palavras são encontradas multiplicando as p palavras principais umas às outras em toda combinação possível. A resolução diz o quanto o projeto é confundido. Quanto maior a resolução do método, maior a expectativa para resultados melhores. É necessário considerar que a resolução depende da escolha das palavras definidoras, portanto as palavras devem ser escolhidas criteriosamente para alcançar a maior resolução possível. (CAVAZZUTI, 2013)

A Figura 4-3 mostra os exemplos de alguns fatoriais fracionados, inclusive com níveis não homogêneos (Figura 4-3c). (CAVAZZUTI, 2013)

Figura 4-3. Exemplos de projetos fatoriais fracionados.

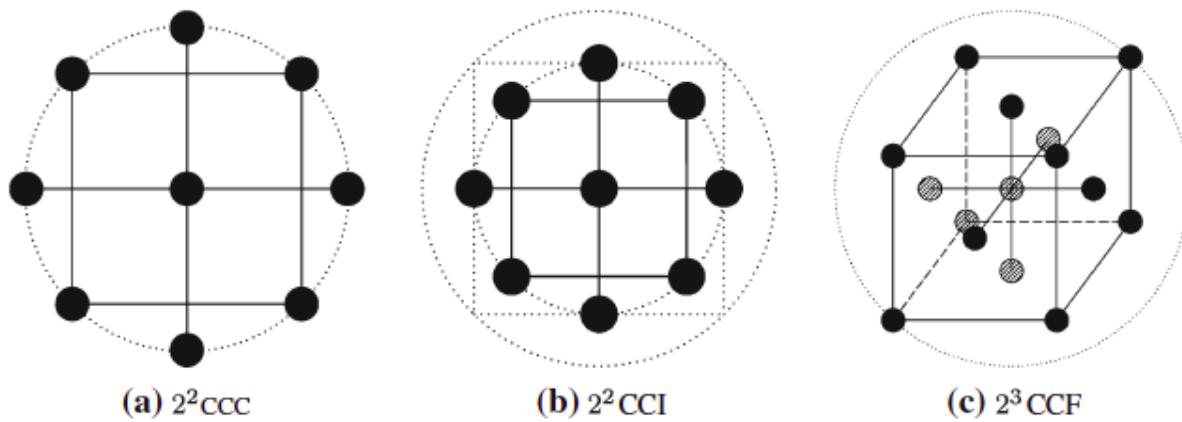


Fonte: (CAVAZZUTI, 2013)

4.1.5. Compósito Central

Um projeto de Compósito Central nada mais é que um factorial completo 2^k no qual o ponto central e os *pontos estrelas* são adicionados. Ou seja, os pontos nos quais todos os parâmetros menos um são ajustados ao nível médio “m”. Exemplos de compósitos centrais podem ser vistos na . (CAVAZZUTI, 2013)

Figura 4-4. Exemplos de projetos de compósito central.



Fonte: (CAVAZZUTI, 2013)

O valor dos pontos estrelas é definido em relação à distância do ponto central. Sendo a distância entre o ponto central e cada amostra do factorial completo

normalizada a 1, a distância dos pontos estrelas ao ponto central pode ser escolhida de diversas formas, de acordo com (CAVAZZUTI, 2013):

- Se definida como 1, todas as amostras são alocadas em uma hiperesfera centralizada no ponto central (Compósito Central Circunscrito, CCC).
- Se definida como $\frac{\sqrt{k}}{k}$, o valor do parâmetro permanece no mesmo nível do fatorial completo 2^k (Compósito de Face Central, CCF, do inglês *central composite faced*).
- Se deseja-se uma amostragem como o CCC, mas os limites especificados para os níveis não podem ser violados, o CCC pode ter sua escala reduzida para que todas as amostras tenham distância ao ponto central igual à $\frac{\sqrt{k}}{k}$ (Compósito Central Inscrito, CCI).
- Se a distância é definida para qualquer outro valor, menor que $\frac{\sqrt{k}}{k}$ (pontos estrela dentro do espaço de projeto), menor que 1 (pontos estrela dentro da hiperesfera) ou maior que 1 (pontos estrela fora da hiperesfera), falamos de Compósito Central em Escala (CCS, do inglês *central composite scaled*).

4.1.6. Aleatório

Além das técnicas previamente discutidas, originadas do campo estatístico, temos outra família de métodos que são dadas pelas técnicas DOE de preenchimento de espaço. Estas se apoiam em diferentes métodos para preencher uniformemente o espaço de projeto. Por esta razão, estas técnicas não se baseiam no conceito de níveis, não requerem parâmetros discretizados, e o tamanho de amostra é escolhido pelo projetista independentemente do número de parâmetros do problema. Geralmente, estas técnicas são uma boa escolha para criar superfícies de resposta. Isso se deve ao fato de que, para um dado N , áreas vazias, que estão longe de qualquer amostra e nas quais a interpolação pode ser inexata, não são propensas a acontecer. No entanto, como técnicas de preenchimento não são baseadas em níveis,

não é possível avaliar os efeitos principais nos parâmetros e seus efeitos de interação de forma tão simples como no caso dos projetos fatoriais. (CAVAZZUTI, 2013)

A mais óbvia das técnicas é a aleatória, na qual o espaço de projeto é preenchido uniformemente com amostras criadas aleatoriamente. Apesar disso, o DOE aleatório não é particularmente eficiente, pois a aleatoriedade não garante de forma alguma que as amostras não ficarão agrupadas, de forma que elas não atinjam o objetivo de preencher uniformemente o espaço de projeto. (CAVAZZUTI, 2013)

4.1.7. Sequências de Halton, Faure e Sobol

No lugar de criar um espaço de projeto totalmente aleatório, muitas técnicas se baseiam em geradores de números *pseudoaleatórios*. A qualidade dos números aleatórios é averiguada através de testes especiais. Geradores de números pseudo-aleatórios são séries matemáticas que geram grupos de números que são capazes de passar nos testes de aleatoriedade. Um número pseudoaleatório é essencialmente uma função $\Phi: [0,1] \rightarrow [0,1]$, que é aplicada iterativamente com o objetivo de encontrar uma série de γ_k valores

$$\gamma_k = \Phi(\gamma_{k-1}), \text{ para } k = 1, 2, \dots \quad (3)$$

começando de um dado γ_0 . A dificuldade é escolher Φ para obter uma distribuição uniforme de γ_k . Algumas das técnicas de preenchimento de espaço mais populares utilizam uma sequência quasi-aleatória monodimensional de baixa-discrepância chamada *Van Der Corput* (CAVAZZUTI, 2013).

Na sequência Van Der Corput, uma base $b \geq 2$ é fornecida e números inteiros sucessivos n são expressados na forma

$$n = \sum_{j=1}^T a_j b^{j-1} \quad (4)$$

onde a_j são os coeficientes da expansão. A função

$$\varphi_b: \mathbb{N}_0 \rightarrow [0, 1) \quad (5)$$

$$\varphi_b(\mathbf{n}) = \sum_{j=1}^T \frac{a_j}{b^j} \quad (6)$$

produz os números da sequência. (CAVAZZUTI, 2013)

Vamos considerar $b = 2$ e $n = 4$: 4 tem uma expansão binária 100, os coeficientes da expansão são $a_1 = 0, a_2 = 0, a_3 = 1$. O quarto número da sequência é $\varphi(4) = \frac{0}{2} + \frac{0}{4} + \frac{1}{8} = \frac{1}{8}$. Os números da sequência Van der Corput de base dois são:

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \dots$$

A ideia básica das técnicas de preenchimento de espaço multidimensional baseadas na sequência Van der Corput é subdividir o espaço de projeto em subvolumes e colocar uma amostra em cada um deles antes de prosseguir para uma malha mais fina. (CAVAZZUTI, 2013)

A Sequência de Halton usa a sequência de base-dois de Van Der Corput para a primeira dimensão, base três para a segunda dimensão, base cinco para a terceira dimensão, e assim por diante, utilizando os números primos como base. O maior desafio é evitar agrupamento multidimensional. De fato, a sequência de Halton mostra fortes correlações entre as dimensões nos espaços de muitas dimensões. Outras sequências tentam evitar este problema. (CAVAZZUTI, 2013)

A sequência de Faure e Sobol usam apenas uma base para todas as dimensões e uma permutação diferente dos elementos dos vetores para cada dimensão. (CAVAZZUTI, 2013)

A base de uma sequência de Faure é o menor número primo ≥ 2 que é maior ou igual ao número de dimensões do problema. Para reordenar a sequência, uma equação recursiva é aplicada aos coeficientes a_j . Passando da dimensão $d - 1$ para a dimensão d a equação de reordenamento é

$$a_i^{(d)}(\mathbf{n}) = \sum_{j=i}^T \frac{(j-1)!}{(i-1)!(j-i)!} a_j^{(d-1)} \text{mod}(\mathbf{b}) \quad (7)$$

A sequência Sobol usa base dois para todas as dimensões, ou seja, não é uma sequência de pontos no hipercubo $(0,1)^D$, mas sim uma coleção de D sequências de números em $(0,1)$, e a tarefa de reordenar é muito mais complexa que a adotada

pela sequência Faure. Sequência Sobol é a mais resistente à degradação em altas dimensões. (CAVAZZUTI, 2013; SAVINE, 2018)

Sobol gera sequências de inteiros y_i^d entre 0 e $2^{32}-1$, então o i -ésimo número no eixo d é

$$x_i^d = \frac{y_i^d}{2^{32}} \in (\mathbf{0}, \mathbf{1}) \quad (8)$$

Os inteiros y_i^d em dado eixo d são produzidos por recursão: $y_0^d = 0$ (o ponto 0 não é válido em Sobol; o primeiro ponto válido é o ponto número 1) e

$$y_{i+1}^d = y_i^d \oplus DN_{J_i}^d \quad (9)$$

Onde \oplus denota o operador de bit xor ('ou' exclusivo), J_i é o bit 0 mais à direita na expansão binária de i , e $\{DN_{J_i}^d, 0 \leq j \leq 31\}$ são os 32 *números direcionais* para a sequência número d .

O bit mais à direita de todos os números pares é 0; portanto, a cada dois pontos, a recursão Sobol consiste em aplicar o operador xor entre o primeiro direcional DN_0^d com sua variável de estado y^d . A operação xor é associativa e possui a seguinte propriedade

$$x \oplus x = \mathbf{0} \quad (10)$$

Então o direcional DN_0^d se alterna com a variável de estado y^d a cada dois pontos. Pela mesma razão, DN_1^d entra a cada quatro pontos, DN_2^d a cada oito pontos, e, mais genericamente, DN_k^d ($0 \leq k \leq 31$) aparece na sequência a cada 2^{k+1} pontos. A sequência de Sobol é ilustrada abaixo:

Tabela 8. Sequência de Sobol (SAVINE, 2018)

i	DN0	DN1	DN2	DN3	y
0	0	0	0	0	0
1	1	0	0	0	DN0
2	1	1	0	0	DN0 + DN1
3	0	1	0	0	DN1
4	0	1	1	0	DN1 + DN2
5	1	1	1	0	DN0 + DN1 + DN2
6	1	0	1	0	DN0 + DN2
7	0	0	1	0	DN2

8	0	0	1	1	DN2 + DN3
9	1	0	1	1	DN0 + DN2 + DN3
10	1	1	1	1	DN0 + DN1 + DN2 + DN3
11	0	1	1	1	DN1 + DN2 + DN3
12	0	1	0	1	DN1 + DN3
13	1	1	0	1	DN0 + DN1 + DN3
14	1	0	0	1	DN0 + DN3
15	0	0	0	1	DN3

Mais genericamente, para $i < 2^k$, o estado y_i^d é uma combinação dos primeiros k números direcionais:

$$y_i^d = \sum_{j=0}^{k-1} a_j^i D N_j^d \quad (11)$$

Onde os pesos a_j^i são ou zero ou 1: a_0 entra a cada 2 pontos, a_1 a cada 4 pontos, a_2 a cada 8 pontos, a_j a cada 2^{j+1} pontos. Consequentemente, os 2^k primeiros pontos na sequência y_i^d , $0 \leq i \leq 2^k$ abrem todas as 2^k combinações possíveis dos k primeiros direcionais, cada combinação sendo representada exatamente uma vez.

A consequência é que o escalar Sobol x_i^d em $(0,1)$ é:

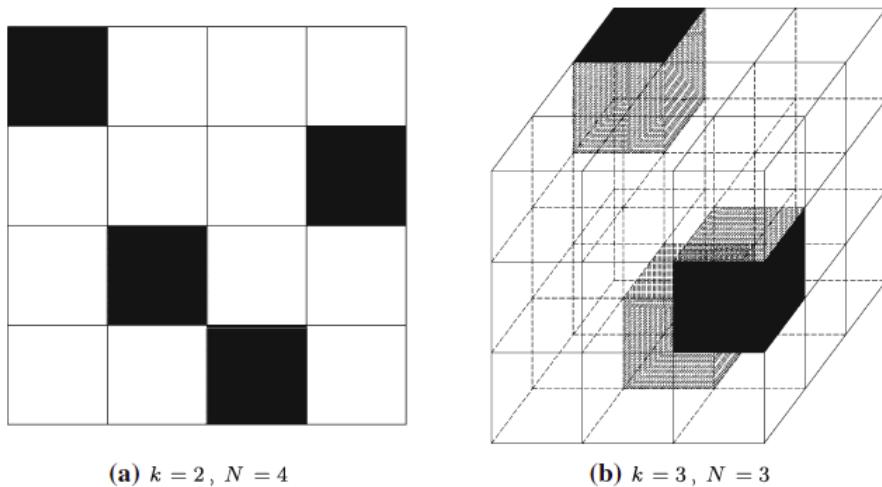
$$x_i^d = \frac{y_i^d}{2^{32}} = \sum_{j=0}^{k-1} a_j \left(\frac{D N_j^d}{2^{32}} \right) = \sum_{j=0}^{k-1} a_j D_j^d \quad (12)$$

Onde $D_j^d \equiv \frac{D N_j^d}{2^{32}}$ são os números direcionais normalizados.

4.1.8. Hipercubo Latino

Como nas técnicas aleatórias anteriores, o Hipercubo Latino visa preencher uniformemente o espaço. Dentro da grade multidimensional, N sub-volumes são individualizados com a mesma lógica do quadrado latino – apenas um sub-volume é escolhido para cada linha e coluna. Dentro de cada sub-volume, uma amostra é escolhida aleatoriamente. Na Figura 4-5, vemos duas representações gráficas típicas para projetos de hipercubos latinos.

Figura 4-5. Exemplos de projetos de hipercubos latinos



Fonte: (CAVAZZUTI, 2013)

É importante escolher os sub-volumes de forma a evitar correlações espúrias entre as dimensões ou, o que é quase equivalente, espalhar as amostras por todo o espaço de projeto. Por exemplo, um grupo de amostras na diagonal do espaço de projeto satisfaz os requisitos de um hipercubo latino, apesar de trazer uma correlação forte entre as dimensões e deixaria a maior parte do espaço de projeto inexplorada. Existem técnicas para reduzir as correlações nos hipercubos latinos. (CAVAZZUTI, 2013)

Cavazzuti fez uma comparação entre as distribuições de amostras no espaço de projeto para as técnicas de hipercubo latino, com e sem redução de correlações, e as técnicas aleatório, Sobol e hipercubo latino para duas dimensões, 1000 sub-volumes (Figura 4-6). O mesmo experimento foi feito no software modeFRONTIER, e a comparação está na Figura 4-7. Uma nova comparação foi feita para três dimensões, na Figura 4-9.

Podemos notar que, no experimento feito no modeFRONTIER, existe uma certa padronização do espaço na sequência Sobol (Figura 4-7) – deixando vários “buracos” no espaço de design. Para a correção desses espaços, uma outra técnica pode ser utilizada – o *Incremental Space Filling* (ISF).

Pela descrição em (ESTECO, 2016), o ISF é um algoritmo feito para aumentar o número de pontos no espaço de design. ISF adiciona novos modelos sequencialmente, maximizando a distância mínima a partir dos modelos existentes. No geral, novos pontos são adicionados de forma a preencher uniformemente o

espaço de design. O algoritmo ISF feito com a sequência Sobol da Figura 4-7b está na Figura 4-8, e o preenchimento ISF para a sequência Sobol em três dimensões está na Figura 4-10.

Figura 4-6. Comparação entre diferente técnicas de DOE para preenchimento de espaço, com $k = 2, N = 1000$.

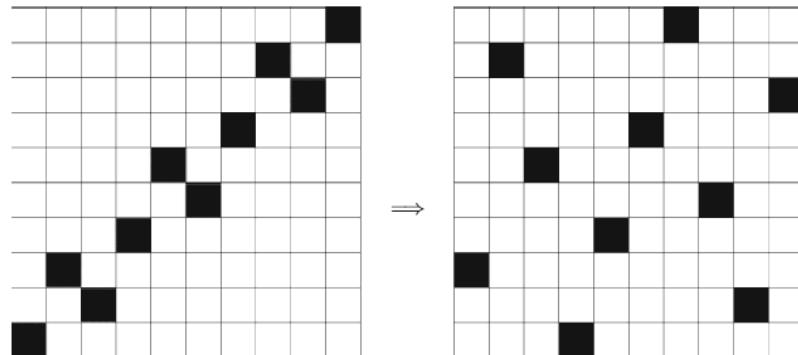
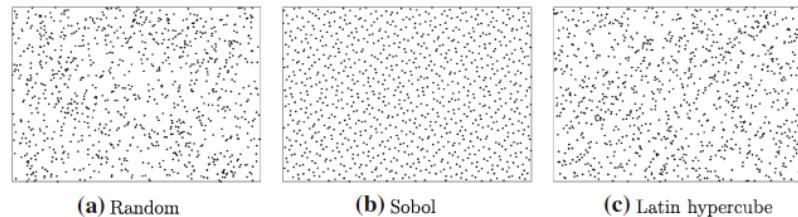
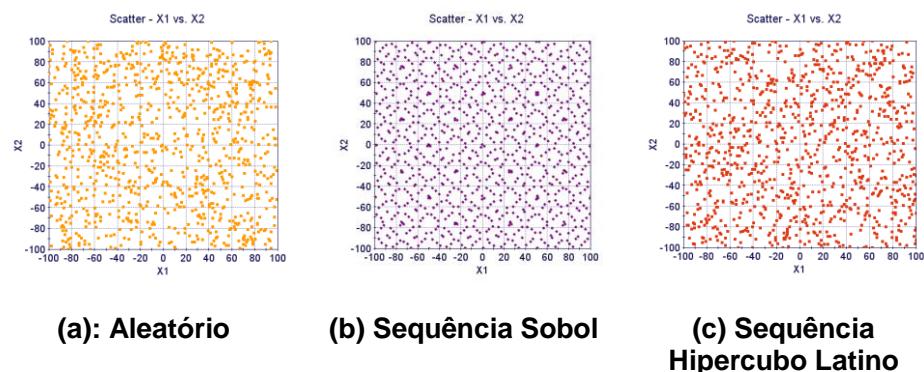


Fig. 2.9 Example of correlation reduction in a latin hypercube DOE with $k = 2, N = 10$



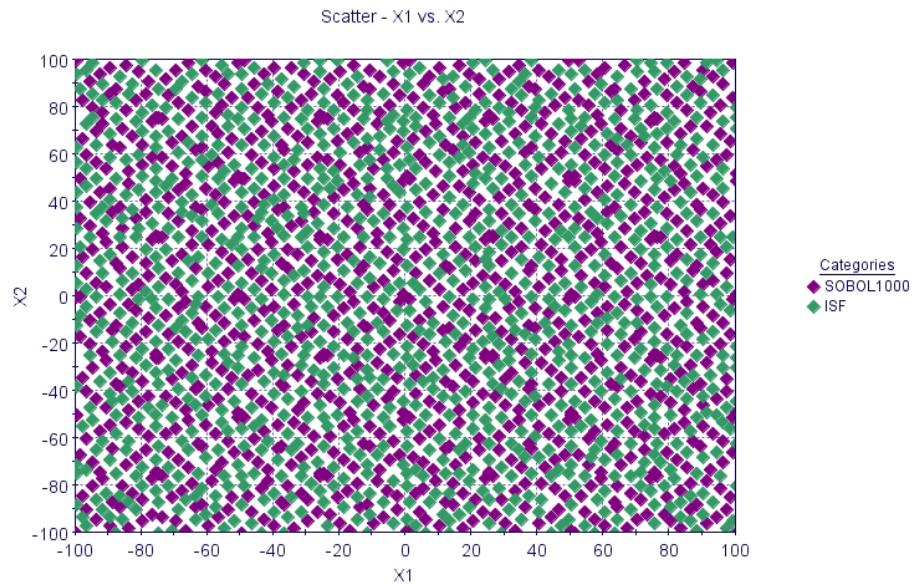
Fonte: (CAVAZZUTI, 2013)

Figura 4-7. Comparação entre diferente técnicas de DOE para preenchimento de espaço, com $k = 2, N = 1000$.



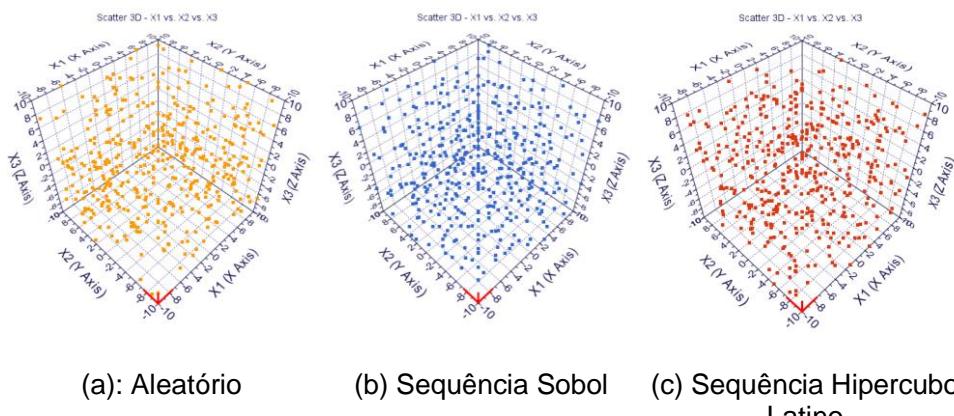
Fonte: Próprio autor. Feito no software modeFRONTIER.

Figura 4-8. Espaço preenchido com Sequência Sobol e ISF. Feito no software modeFRONTIER.



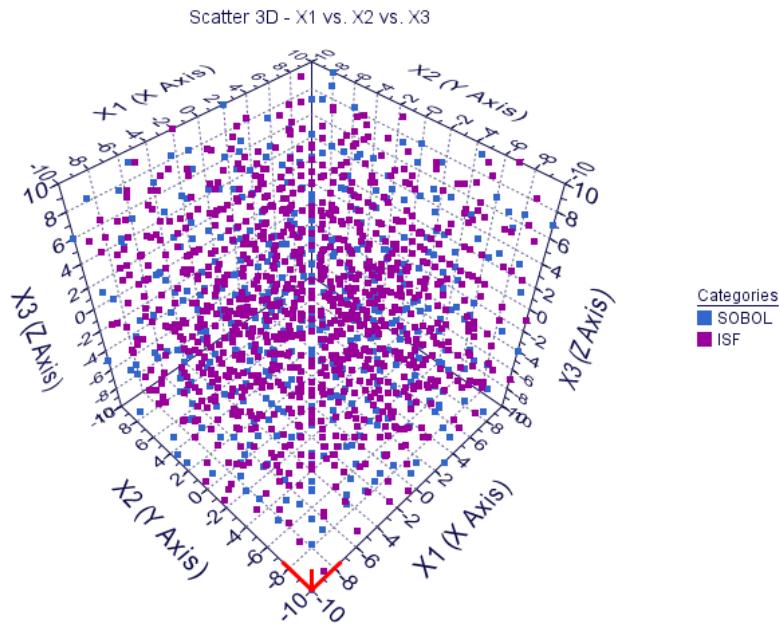
Fonte: Próprio autor. Feito no software modeFRONTIER.

Figura 4-9. Comparação entre diferentes técnicas de DOE para preenchimento de espaço, com $k = 3, N = 1000$.



Fonte: Próprio autor. Feito no software modeFRONTIER.

Figura 4-10. Espaço preenchido com Sequência Sobol e ISF.



Fonte: Próprio autor. Feito no software modeFRONTIER.

4.1.9. Escolha do DOE

Diversas técnicas de projetos de experimentos estão disponíveis para o projetista. No entanto, a eficiência do DOE está atrelada a fatores importantes:

- Quantidade de experimentos que podem ser realizados;
- Número de parâmetros de cada experimento;
- Número de níveis L para cada parâmetro;
- O objetivo do DOE.

ESTECO (2016) lista algumas aplicações para diversos tipos de DOE:

- DOEs para análises estatísticas são usados para extrair as informações qualitativas mais relevantes de um banco de dados de experimentos limitado. Estas técnicas são adequadas para estudar o

efeito de variáveis nas respostas dos sistemas, assim como os efeitos das interações entre os fatores. Para este propósito, as correlações entre as variáveis devem ser as menores possíveis. Algoritmos recomendados em (ESTECO, 2016): DOEs fatoriais, Hipercubos latinos, quadrado latino.

- DOEs para treino de superfícies de resposta são usados para gerar grupos de dados adequados que permitam a criação de superfícies de respostas confiáveis quando os recursos computacionais disponíveis são insuficientes para uma otimização clássica. A uniformidade do projeto deve ser tão alta quanto possível, e quanto mais amostras, maior a confiabilidade da superfície de resposta. Algoritmos recomendados por (ESTECO, 2016): DOE de preenchimento de espaço (Hipercubo Latino, Quadrado Latino)
- DOEs para otimização são usados para criar uma população inicial adequada para algoritmos de otimização. Cada algoritmo de otimização requer um DOE de tamanho distinto. Otimizadores robustos, como os algoritmos genéticos, geralmente são pouco influenciados pela qualidade do DOE. Algoritmos recomendados por (ESTECO, 2016): DOE de preenchimento de espaço (Hipercubo Latino, Aleatório, Sequência Sobol, Designs ótimos).
- DOEs para robustez e análise de confiabilidade são usados para avaliar os efeitos da variabilidade aleatória de certos fatores sobre as respostas. Algoritmos recomendados por (ESTECO, 2016): Hipercubo Latino com Monte Carlo, Vetores Ortogonais de Taguchi.

ESTECO (2016) também cita que uma fórmula empírica genérica para escolher o tamanho correto do DOE é duas vezes o número de parâmetros vezes o número de objetivos. No caso do presente trabalho, com 8 parâmetros e 2 objetivos a serem variados, o tamanho correto seria no mínimo 32 modelos como população inicial. O autor optou por não executar as simulações de todas as combinações possíveis (fatorial completo). Para uma análise inicial, o método de DOE escolhido será o Hipercubo latino (ULH), pois ele tem a vantagem de cada modelo poder analisar a influência de todos os fatores, sem repetição. Existem combinações de fatores que

violam as restrições naturais da construção da *crash box* origami, e tais combinações aparecem quando formulamos o DoE via ULH. Para atingir uma amostragem inicial maior do que a recomendada por (ESTECO, 2016), também foram incluídos experimentos escolhidos via Sobol e *Incremental Space Filling* (ISF). Neste projeto, o tamanho inicial da amostra foi de 64 modelos: 16 por SOBOL, 34 por Hipercubo Latino e 14 por ISF. Um pequeno exemplo da escolha do DOE e da aplicação do algoritmo FAST está disponível no ANEXO C.

4.2. METAMODELOS OU SUPERFÍCIES DE RESPOSTA

Em um processo de otimização, podem existir dezenas de variáveis de projeto, possibilitando milhares ou até milhões de configurações distintas do design ótimo. Realizar simulações para cada configuração, com horas de simulação para cada resultado, torna o procedimento inviável.

Para viabilização de um processo de otimização de simulações ou experimentos que consomem muito tempo, podem ser utilizados metamodelos. O metamodelo é “modelo de um modelo” ou “modelo substituto”, ou seja, uma aproximação usada para substituir os modelos computacionalmente custosos durante a análise, design e processo de otimização. Em geral, os metamodelos são funções matemáticas ou algoritmos que relacionam as características de sistemas em termos de design ou parâmetros de ajuste. (MOHANTY, 2015)

Cavazzuti (2013) considera as superfícies de resposta como estritamente relacionadas ao DOE. A ideia principal é utilizar os resultados de um DOE para criar uma função que aproxima a variável de resposta no espaço de design. Esta aproximação é chamada de superfície de resposta ou metamodelo e pode ser criada para qualquer parâmetro de saída. A razão para criar uma superfície de resposta é que, apesar de ser apenas uma aproximação, ela pode ser utilizada para estimar o conjunto de parâmetros de entrada que levam a uma resposta otimizada.

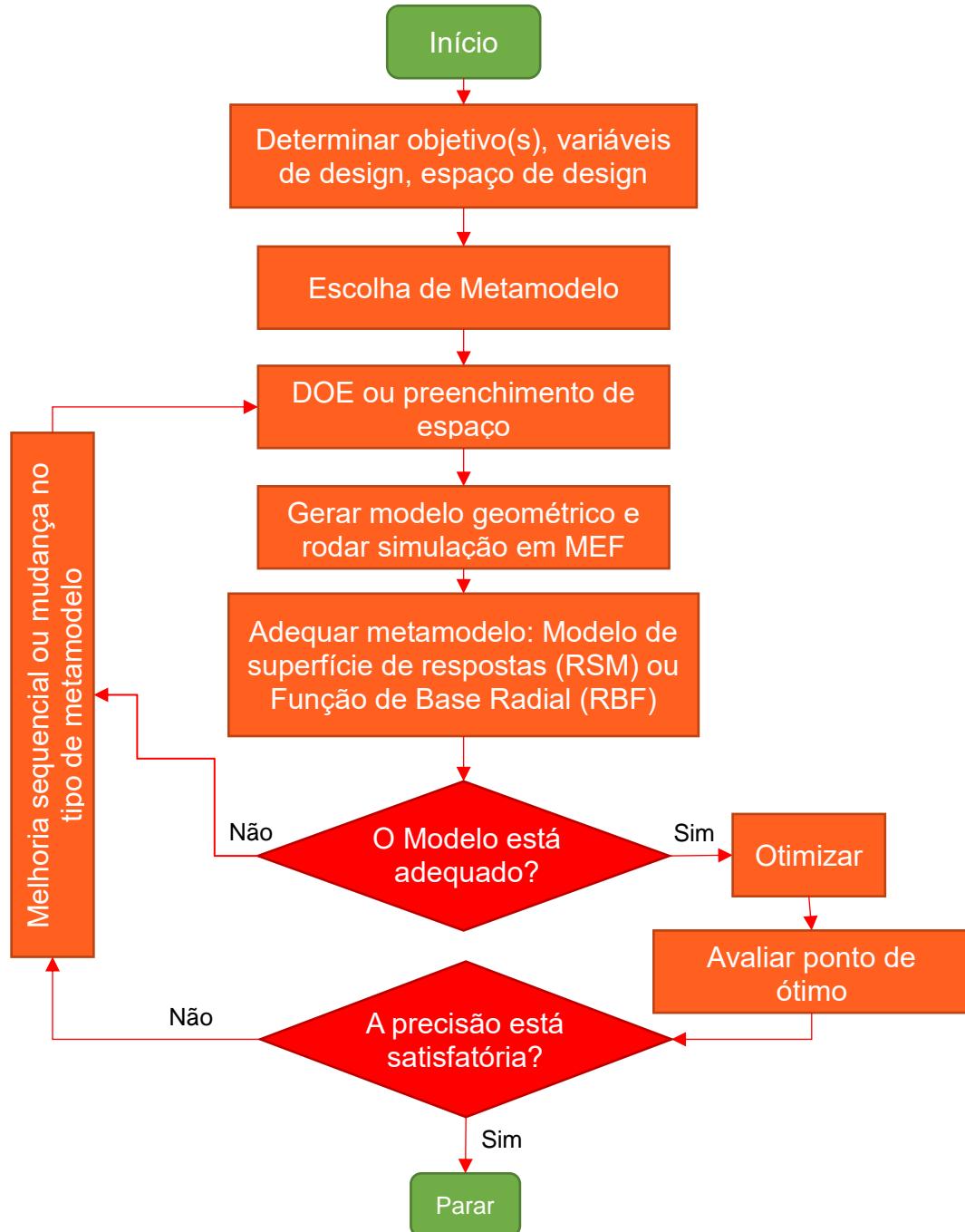
A superfície de resposta é uma função analítica, portanto uma otimização baseada nela é bem rápida e não requer a realização de experimentos ou simulações adicionais. Portanto, o uso de metamodelos pode ser bem vantajoso, e pode ser aplicado até mesmo quando pouco se sabe sobre o problema, apesar de ser

necessário manter em mente que se a exploração do espaço de design (feita pelo DOE ou pelo modelo de superfície de resposta adotado) é pobre, e a variável de resposta é particularmente irregular, o resultado de uma otimização auxiliada por metamodelos pode estar longe do ponto verdadeiro, por conta da má estimativa dos coeficientes do modelo ou pela escolha de um metamodelo impróprio. (CAVAZZUTI, 2013)

Anderson Lima, em 2016, utilizou os conceitos de metamodelos na criação de um veículo urbano de transporte que atendeu aos requisitos de impacto frontal e lateral definidos pela EuroNCAP. Lima utilizou simulações numéricas de elementos finitos para gerar um metamodelo, e a partir deste obteve uma combinação ótima para suas variáveis de projeto.

O processo de criação do metamodelo deste trabalho consistirá em diversas simulações do modelo de elementos finitos, com valores e combinações distintas das variáveis de projeto. Cada simulação dessas gera uma resposta das funções objetivo, SEA e LU. O metamodelo será criado sobre os valores da função objetivo, através de técnicas de regressão, como na Figura 4-12, uma função de base radial. Finalmente, sobre o metamodelo será feita uma otimização matemática e, uma vez encontrado o ponto ótimo, as variáveis de projeto para este ponto serão resgatadas e teremos nossa forma ótima de *crash box origami*. Este processo está ilustrado na Figura 4-11. Mais detalhes sobre o algoritmo que coordena a criação e otimização dos metamodelos estão descritos no tópico 5.9.

Figura 4-11. Estratégia de otimização de design baseada em Metamodelos.



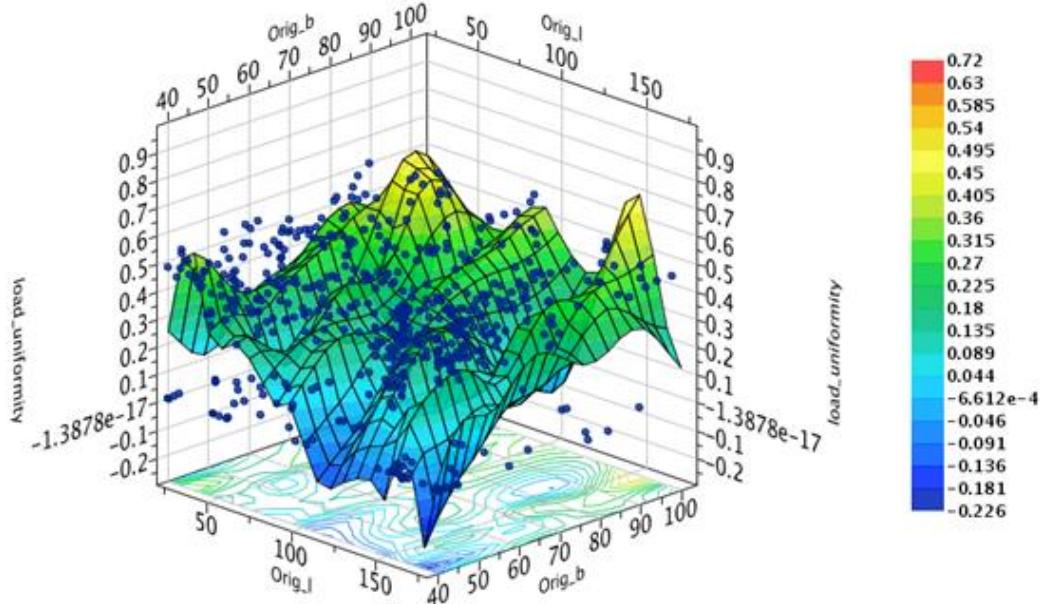
Fonte: (PARK; DANG, 2010)

Figura 4-12. Criação de metamodelo a partir de pontos amostrados

Nuvem de pontos:
Respostas exatas das
simulações

Superfície: metamodelo
com uma formulação matemática
Mais simples

Espessura = 1.8mm; Modulos = 3; N (lados) = 4; c = 50%*b; No tampering



Fonte: Próprio autor

Serão considerados neste trabalho os requisitos de projeto de *crash box*, entre eles:

- Peso e momento de inércia do veículo;
- Método construtivo;
- Geometria Inicial;
- Material do tubo;
- Dimensões máximas e mínimas.

Serão analisadas as seguintes variáveis:

- Espessura do tubo;
- Altura do módulo origami (variável l – ver Figura 3-6);

- Tamanho do lado do polígono de base do origami (variável b – ver Figura 3-6);
- Tamanho da dobra (variável c – ver Figura 3-6)
- Quantidade de módulos origami;
- Número de lados de cada módulo;
- Alteração de inclinações e proporções do tubo;

Como função objetivo, será maximizada a energia cinética absorvida (energia interna da *crash box*), minimizando o pico da força de contato entre o impactor e o veículo.

Um modelo-exemplo foi montado para o experimento-base do trabalho. Neste exemplo, temos uma *crash box* origami com os seguintes valores para as dimensões identificadas na Figura 3-6: $b = 75\text{mm}$; $c = 70\% \cdot b$; $l = 60\text{ mm}$.

Para impactor, foi utilizada uma simples placa rígida, de massa 1400 kg, com deslocamento axial paralelo ao eixo central da *crash box*. A partir da simulação deste modelo, foi obtida tanto a força de pico inicial quanto a energia total absorvida. Maiores detalhes sobre o modelo estão descritos no tópico 5.1.

4.2.1. Método dos Mínimos Quadrados

A método dos mínimos quadrados (*LSM*, do inglês *Least squares method*) é utilizada para resolver sistemas sobre determinados e pode ser interpretada como um método para adequação de dados. O método foi desenvolvido por Gauss por volta de 1795 e publicado muitos anos depois. Consiste no ajuste dos coeficientes de uma função modelo (a superfície de resposta) de forma que melhor se adeque ao grupo de dados (os resultados de um DOE). (CAVAZZUTI, 2013)

Uma função modelo é uma função $f(x, \beta)$, onde $\beta = [\beta_1, \dots, \beta_m]^T$ é o vetor dos m coeficientes a serem ajustados e $x = [x_1, \dots, x_k]^T$ é o vetor de k parâmetros de entrada. O grupo de dados consiste em pares (x_i, y_i) , $i = 1, \dots, N$, onde x_i é o vetor dos parâmetros de entrada do i -ésimo experimento, cuja variável de resposta é y_i . O melhor ajuste que o método dos mínimos quadrados define é a escolha dos coeficientes β_j , $j = 1, \dots, m$ que fornecem a menor soma S dos resíduos quadrados da

distância dos valores do grupo de dados aos valores preditos pela função modelo nos pontos x_i no espaço de projetos. (CAVAZZUTI, 2013)

$$\begin{aligned} S &= \sum_{i=1}^N \epsilon_i^2 \\ \epsilon_i &= y_i - f(x_i, \beta), i = 1, \dots, N \end{aligned} \quad (13)$$

O mínimo desta soma de quadrados é encontrado ao igualar o gradiente de S em relação a β a zero, ou seja

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_{i=1}^N \epsilon_i \frac{\partial \epsilon_i}{\partial \beta_j} = -2 \sum_{i=1}^N [y_i - f(x_i, \beta)] \frac{\partial f(x_i, \beta)}{\partial \beta_j} = 0, \quad j = 1, \dots, m \quad (14)$$

4.2.2. Superfície de Resposta Ótima

A Superfície de resposta ótima (*O-RSM*, do inglês *Optimal RSM*) é uma generalização do método dos mínimos quadrados. Dados os resultados de um experimento (x_i, y_i) , $i = 1, \dots, N$, vamos assumir que queremos construir uma superfície de resposta dos mínimos quadrados com m coeficientes β_j , $j = 1, \dots, m$, e m funções de base $X_j(x)$, $j = 1, \dots, m$, de forma que a soma dos erros quadrados seja minimizada nos pontos x_i , $i = 1, \dots, N$, na relação

$$y = \hat{f}(x, \beta, X'(x)) + \epsilon(x) = \sum_{j=1}^m \beta_j X_j(x) + \epsilon(x) \quad (15)$$

Designa-se com $X'(x)$ o vetor $[X_1(x), \dots, X_m(x)]^T$. No O-RSM, não se assume uma função modelo em particular, mas sim funções ótimas de bases assim como seus coeficientes serão determinados. As funções ótimas de base são escolhidas de um grupo $X(x) = [X_1(x), \dots, X_p(x)]^T$, $p > m$, onde os termos $X_j(x)$, $j = 1, \dots, p$ podem ser qualquer função de x . (CAVAZZUTI, 2013)

O-RSM é um procedimento iterativo no qual, na iteração l as funções bases $X'^{(l)}(x)$ são escolhidas aleatoriamente de $X(x)$ e a superfície de resposta dos mínimos quadrados é computada:

$$\hat{\mathbf{y}}^{(l)} = \hat{\mathbf{f}}\left(\mathbf{x}, \boldsymbol{\beta}^{(l)}, X'^{(l)}(\mathbf{x})\right) = \sum_{j=1}^m \boldsymbol{\beta}_j^{(l)} X_j^{(l)}(\mathbf{x}) \quad (16)$$

Para cada termo em $X(x)$ o parâmetro de performance $r_i, i = 1, \dots, p$ é definido e inicialmente é zero. Depois de cada iteração o parâmetro de performance das funções base envolvidas na interação é recalculado como $r_i = r_i + \delta^{(l)}$, onde $\delta^{(l)}$ é uma medida da performance da superfície de resposta na iteração l . Por exemplo, esta medida pode ser qualquer parâmetro de regressão. Com um vasto número de iterações, uma estimativa heurística das melhores funções base é dada pelos elementos em $X(x)$ cujos parâmetros de performance divididos pelo número de vezes que esta função base foi escolhida durante as iterações é máximo. A O-RSM é então dada pela função de mínimos quadrados

$$\hat{\mathbf{y}} = \hat{\mathbf{f}}\left(\mathbf{x}, \boldsymbol{\beta}, X'^{best}(\mathbf{x})\right) = \sum_{j=1}^m \boldsymbol{\beta}_j X_j^{best}(\mathbf{x}) \quad (17)$$

onde X'^{best} é o vetor das melhores funções base. (CAVAZZUTI, 2013)

4.2.3. Shepard e K-Nearest

Shepard e *K-nearest* (ou *Kriging nearest*) RSM são métodos de interpolação que não são computacionalmente intensos, e, portanto, se adequam bem a grandes grupos de dados. Por outro lado, não fornecem muitas informações para pequenos grupos de dados. (CAVAZZUTI, 2013)

Consideremos os resultados de um DOE $(x_i, y_i), i = 1, \dots, N$, sendo x_i o vetor de k elementos. De acordo com o método Shepard, o valor da superfície de resposta em qualquer ponto x dado por uma média ponderada dos resultados experimentais é

$$\hat{\mathbf{f}}(\mathbf{x}) = \sum_{i=1}^N \lambda_i(\mathbf{x}) \mathbf{f}(\mathbf{x}_i) = \sum_{i=1}^N \lambda_i(\mathbf{x}) \mathbf{y}_i \quad (18)$$

onde os pesos λ_i são inversamente proporcionais à distância euclidiana d_i normalizada ao p -ésimo exponente entre x e x_i

$$\widehat{\lambda}_i = \frac{\frac{1}{c + d_i^p}}{\sum_{j=1}^N \frac{1}{c + d_j^p}} \quad (19)$$

onde

$$d_i = \sqrt{\sum_{j=1}^k (x_j - x_{i,j})^2} \quad (20)$$

p é geralmente escolhido no intervalo [1,3] e c é uma constante pequena cujo propósito é evitar divisões por zero quando x coincide com algum dos x_i .

A diferença entre Shepard e K-nearest é que a segunda não está computando a superfície de resposta como uma média ponderada de todos os resultados, mas apenas dos q resultados mais próximos aos x pontos experimentais, onde q é escolhido pelo projetista. Se q não é muito pequeno, as duas superfícies de resposta não diferem muito, mas para grandes grupos de dados o esforço computacional para gerar uma resposta via K-nearest é menor. (CAVAZZUTI, 2013)

4.2.4. Kriging

Kriging é a ferramenta principal para realizar previsões em geoestatística. A técnica foi originalmente desenvolvida pelo engenheiro de minas Sul Africano D.G. Krige em 1951, que em sua tese de mestrado aplicou esta técnica para estimar o verdadeiro grau de minério de amostras. (LIMA, 2016)

Kriging foi introduzido como um metamodelo para *modelos de simulação determinística* ou “modelos de computador” em 1989. Modelos de simulação têm combinações de entradas em k -dimensões onde k é um número inteiro positivo dado,

enquanto geoestatísticas consideram apenas duas ou três dimensões. (KLEIJNEN, 2015)

Tipicamente, modelos Kriging são mais adequados para dados obtidos em áreas experimentais abrangentes do que em áreas usadas em metamodelos de baixa ordem polinomiais; isto é, modelos Kriging são globais ao invés de locais. Modelos Kriging são usados para previsões. Os objetivos finais são a análise de sensibilidade e análise de risco e otimização. (KLEIJNEN, 2015)

Kriging é uma metodologia *Bayesiana* adequada para respostas altamente não-lineares e é computacionalmente intensa. Pode ser um método de interpolação ou aproximação, dependendo se o parâmetro de ruído, chamado *nugget*, é definido ou não como zero. (CAVAZZUTI, 2013)

Kriging pertence à família dos algoritmos de mínimos quadrados lineares. Como no caso do método Shepard, a estimativa da variável de resposta no ponto x é dada por uma combinação linear dos resultados de um DOE

$$\hat{f}(x) = \sum_{i=1}^N \lambda_i(x) f(x_i) = \sum_{i=1}^N \lambda_i(x) y_i \quad (21)$$

A diferença entre os dois métodos é o modo como os pesos λ_i são escolhidos. No Kriging os pesos são a solução de um sistema de equações lineares obtido assumindo que $f(x)$ é um “caminho-amostra” de um processo aleatório cujo erro de predição deve ser minimizado. O método procura pelo *melhor estimador imparcial linear* (BLUE, do inglês *best linear unbiased estimator*) baseado em um modelo estocástico da dependência espacial quantificada ou pelo *semivariograma*

$$\gamma(x, y) = \frac{1}{2} \text{var}(f(x) - f(y)) = \frac{1}{2} E[(f(x) - \mu - f(y) + \nu)^2] \quad (22)$$

ou pelo valor esperado

$$\mu = E[f(x)] = \sum_{i=1}^N \frac{f(x_i)}{N} \quad (23)$$

que é a média das respostas de cada experimento/simulação, e a *função de covariância*

$$c(x, y) = \text{cov}(f(x), f(y)) = E[(f(x) - \mu)(f(y) - \nu)] \quad (24)$$

onde ν é o valor esperado de $f(y)$. Ainda segundo (CAVAZZUTI, 2013), das equações acima e das definições da função de covariância e semivariograma, a equação seguinte é válida para quaisquer dois pontos x e y no espaço de projetos

$$\gamma(x, y) = \frac{1}{2} \text{var}(f(x)) + \frac{1}{2} \text{var}(f(y)) - c(x, y) \quad (25)$$

Na verdade, a Equação 23 é válida para o Kriging ordinário, a mais comum das técnicas Kriging, e a técnica que será utilizada neste trabalho. Tipos diferentes de Kriging existem de acordo com o modo de computar μ :

- Kriging simples assume uma constante $\mu(x) = 0$
- Kriging ordinário assume uma constante desconhecida $\mu(x) = \mu$
- Kriging universal assume uma variável $\mu(x) = \sum_{j=1}^k \beta_j x_{i,j}$
- Kriging IRF-k assume $\mu(x)$ como um polinômio desconhecido
- Kriging indicador e Kriging de múltiplos indicadores, que fazem uso de funções indicadoras
- Kriging disjuntivo que é uma generalização não-linear de Kriging
- Kriging lognormal que interpola dados por meio de logaritmos

Os pesos $\lambda_i, i = 1, \dots, N$ são escolhidos de forma que a variância, também chamada de variância Kriging ou erro Kriging (CAVAZZUTI, 2013):

$$\begin{aligned} \hat{\sigma}^2 &= \text{var}(\hat{f}(x) - f(x)) = \text{var}(\hat{f}(x)) - \text{var}(f(x)) - 2\text{cov}(\hat{f}(x), f(x)) \\ &= \text{var}\left(\sum_{i=1}^N \lambda_i(x) f(x_i)\right) + \text{var}(f(x)) - 2\text{cov}\left(\sum_{i=1}^N \lambda_i(x) f(x_i), f(x)\right) \end{aligned} \quad (26)$$

$$\begin{aligned}
&= \sum_{i=1}^N \sum_{j=1}^N \lambda_i(x) \lambda_j(x) \text{cov}\left(f(x_i), f(x_j)\right) + \text{var}(f(x)) \\
&\quad - 2 \sum_{i=1}^N \lambda_i(x) \text{cov}(f(x_i), f(x)) \\
&= \sum_{i=1}^N \sum_{j=1}^N \lambda_i(x) \lambda_j(x) c(x_i, x_j) + \text{var}(f(x)) - 2 \sum_{i=1}^N \lambda_i(x) c(x_i, x)
\end{aligned}$$

seja minimizada sob a condição imparcial:

$$\begin{aligned}
E[\hat{f}(x) - f(x)] &= \sum_{i=1}^N \lambda_i(x) E[f(x_i)] - E[f(x)] \\
&= \sum_{i=1}^N \lambda_i(x) \mu(x_i) - \mu(x) = \mathbf{0}
\end{aligned} \tag{27}$$

que no caso do Kriging ordinário fica:

$$\sum_{i=1}^N \lambda_i(x) = \mathbf{1} \tag{28}$$

Unindo às outras equações, temos

$$c(x_i, x_j) \lambda(x) = c(x_i, x) \Rightarrow \lambda(x) = c^{-1}(x_i, x_j) c(x_i, x) \tag{29}$$

onde

$$\lambda(x) = \begin{pmatrix} \lambda_1(x) \\ \lambda_2(x) \\ \vdots \\ \lambda_N(x) \end{pmatrix}, c(x_i, x_j) = \begin{pmatrix} c(x_1, x_1) & \cdots & c(x_1, x_N) \\ \vdots & \ddots & \vdots \\ c(x_N, x_1) & \cdots & c(x_N, x_N) \end{pmatrix}, \tag{30}$$

$$\mathbf{c}(\mathbf{x}_i, \mathbf{x}) = \begin{pmatrix} c(\mathbf{x}_1, \mathbf{x}) \\ c(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ c(\mathbf{x}_N, \mathbf{x}) \end{pmatrix} \quad (31)$$

Nas equações acima, $\lambda(x)$ deve ser encontrado, enquanto $c(x_i, x_j)$ e $c(x_i, x)$ são desconhecidos e devem ser estimados através de um modelo de *semivariograma* (CAVAZZUTI, 2013).

Neste trabalho, iremos nos ater ao Kriging ordinário, portanto não nos aprofundaremos mais no método de Kriging por semivariogramas.

4.2.5. Polynomial SVD – Decomposição em valor singular polinomial

A decomposição de valor singular polinomial (SVD – do inglês *Single Value Decomposition*) produz aproximações polinomiais de respostas ao minimizar as previsões de erros quadráticos no conjunto de dados. Não gera metamodelos precisos das respostas, mas sim uma estimativa confiável das principais tendências, sejam elas lineares, quadráticas ou outras, que são muito úteis para ter uma ideia geral do comportamento das variáveis de saída, detectando e isolando regiões de possível interesse. (MONTRONE; TURCO; RIGONI, 2014)

A formulação simples do modelo e a efetividade do algoritmo SVD dão velocidade ao processo de treinamento, quando comparado à outras técnicas de metamodelagem.

Por padrão as variáveis de entrada e saída são normalizadas no intervalo [0,1]. Deste modo, os dados são bem proporcionados, sendo perfeitamente comparáveis no que tange à extensão dos intervalos. Essa transformação previne problemas numéricos que geralmente aparecem durante os treinos de superfícies de resposta na presença de variáveis em diferentes escalas. (MONTRONE; TURCO; RIGONI, 2014)

4.2.6. RBF – Função de Base Radial

As Funções de Base Radial (RBF – do inglês *Radial Basis Function*) são ferramentas poderosas para interpolação de dados dispersos multivariados. Como a

RBF gera superfícies de resposta de interpolação, as superfícies sempre abrangem os pontos do conjunto de treinamento. (MONTRONE; TURCO; RIGONI, 2014)

Dado um conjunto de treinamento de n pontos amostrados de uma função $f(x): \mathbb{R}^d \rightarrow \mathbb{R}$,

$$f(x_i) = f_i, \quad i = 1, \dots, n \quad (32)$$

Uma interpolação por RBF tem então a forma

$$s(x) = \sum_{j=1}^n c_j \phi(\|x - x_j\|/\delta) \quad (33)$$

Onde $\|\cdot\|$ é a norma Euclidiana no espaço d -dimensional, e δ é um parâmetro multiplicador fixo. A *função radial* (ou *kernel*) $\phi(r) : [0, +\infty) \rightarrow \mathbb{R}$ é uma função fixa adequada escolhida de uma lista dada. A interpolação RBF s é simplesmente uma combinação linear de funções esféricas simétricas idênticas, centradas nos n diferentes locais dos pontos de treinamento. (MONTRONE; TURCO; RIGONI, 2014)

O coeficiente c_j representa os parâmetros livres do modelo RBF. Os valores são obtidos ao impor as *equações de interpolação*:

$$s(x_i) = f(x_i) = f_i, \quad \forall i = 1, \dots, n \quad (34)$$

Esse grupo de equações são solucionados através do algoritmo SVD, que é uma excelente ferramenta para solucionar sistemas de equações lineares, mesmo quando lidando com situações numéricas complicadas. (MONTRONE; TURCO; RIGONI, 2014)

No modeFRONTIER estão disponíveis cinco diferentes funções radiais: Gaussianas (G), Splines poli harmônicas de Duchon (PS), Multi Quádricas de Hardy (MQ), Multi Quádricas Inversas (IMQ), e C^2 Suportados Compactamente de Wendland (W2) (RIGONI, 2007). Esta lista representa um conjunto completo e amplamente usado de funções radiais em estado da arte encontradas na literatura. O padrão é MQ,

por seu bom desempenho em diversos problemas. (MONTRONE; TURCO; RIGONI, 2014)

4.2.7. Neural Network – Redes Neurais

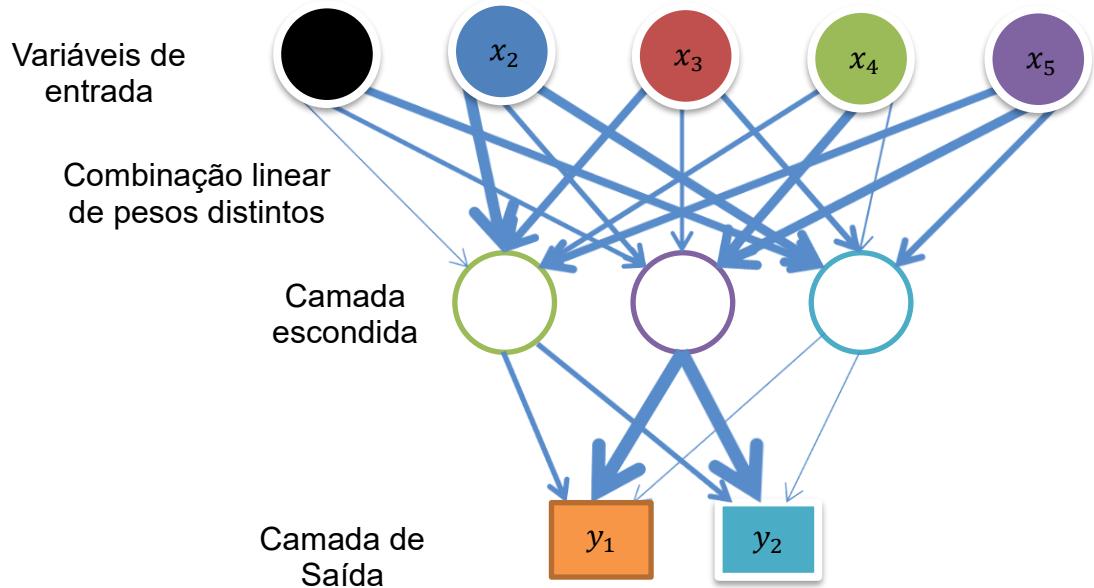
Para propósito de aproximação de funções, as Redes Neurais compõem uma ferramenta poderosa e muito eficiente. Analogamente ao funcionamento do cérebro, as Redes Neurais Feedforward são organizadas em camadas sucessivas de neurônios. O fluxo de dados é em um só sentido, ou seja, vai apenas da primeira camada de entrada para a última camada de saída, sem retorno, e é elaborado de forma incremental por sucessivas camadas intermediárias escondidas (MONTRONE; TURCO; RIGONI, 2014). Na Figura 4-13 temos a esquematização de uma rede neural com cinco variáveis de entrada e duas saídas. Os pesos são representados pela espessura de cada ligação entre as camadas.

O modelo de cada neurônio individual é relativamente simples:

$$\mathbf{u} = \sum_{i=1}^n w_i x_i + b \quad y = f(\mathbf{u}) = f\left(\sum_{i=1}^n w_i x_i + b\right) \quad (35)$$

A entrada de rede u é uma combinação linear dos valores de entrada x_i : os pesos w_i e a constante b (ou patamar de disparo) representam os parâmetros livres do modelo. A entrada de rede é transformada através de uma função de transformação f (ou função de ativação) – que geralmente é não-linear – retornando a saída do neurônio y . O comportamento e a complexidade da rede são definidos pelo modo que seus neurônios são conectados: então em geral o modelo $f(x)$ é uma função recursiva complexa que é usualmente tratada como uma caixa preta, sem explicitar uma expressão analítica (MONTRONE; TURCO; RIGONI, 2014).

Figura 4-13. Esquema de uma Rede Neural Artificial com uma camada escondida



Fonte: Próprio Autor.

É demonstrado em (IRIE; MIYAKE, 1988) que Redes Neurais Artificiais com uma camada escondida individual não-linear e uma camada de saída linear são suficientes para representar qualquer função arbitrária (desde que suficientemente regular). A condição necessária é um número suficientemente alto de neurônios na camada escondida.

Redes Neurais aprendem por exemplos: dado um conjunto de dados de treinamento como variáveis de entrada e metas para as saídas, os pesos da rede e constantes são ajustados de forma a minimizar os erros em suas previsões sobre os dados de treinamento. O melhor algoritmo conhecido é a propagação reversa: uma abordagem bem eficiente feita pelo algoritmo Levenberg-Marquardt, como mostrado por (HAGAN; MENHAJ, 1994).

O software modeFRONTIER utiliza as Redes Neurais Feedforward clássicas com uma camada escondida. Esta camada possui uma função de transferência sigmoid, enquanto a camada de saída tem uma função de transferência linear. O algoritmo de propagação reversa é o escolhido, em particular o algoritmo Levenberg-Marquardt, com um número fixo de iterações. O conjunto de dados de treinamento é normalizado internamente, tanto nas variáveis de entrada quanto nas de saída, a fim de explorar o intervalo natural da definição das funções de transferência.

Neste contexto apenas um parâmetro livre deve ser definido: o número de neurônios na camada escondida. (MONTRONE; TURCO; RIGONI, 2014).

5. OTIMIZAÇÃO NO modeFRONTIER

5.1. MODELO INICIAL – RCAR

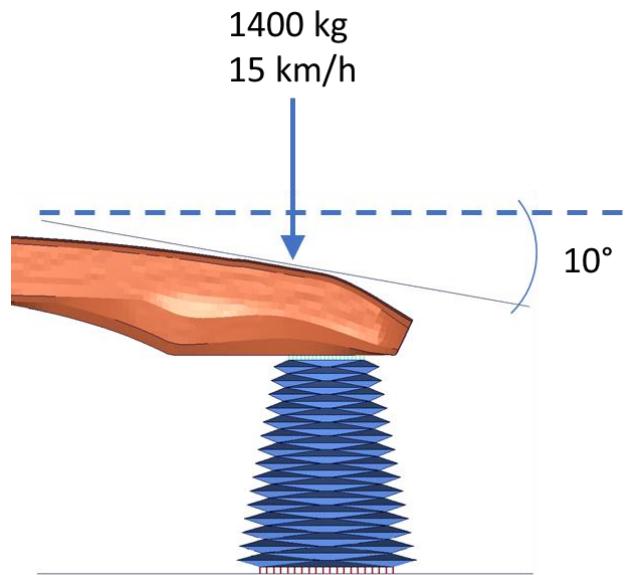
Ma (2011) avaliou o desempenho de tubos com simulações baseadas um teste de impacto axial, verificando a quantidade de energia absorvida e a força média de esmagamento.

Este trabalho se propõe a otimizar um absorvedor de energia em forma de tubo com geometria origami, a *crash box* origami, através de técnicas de Projetos de Experimentos (DOE) e modelos substitutos (metamodelos). A otimização será desenvolvida de acordo ao teste estrutural de baixa velocidade da RCAR (RCAR, 2018).

Uma barreira de 1400 kg é lançada contra o sistema de absorção de impactos do veículo de passeio (para-choques), a um ângulo de 10°, em uma velocidade de 15 km/h (Figura 5-1). O modelo do veículo é um corpo rígido simplificado, afixado a uma massa concentrada localizada no centro de gravidade de um veículo real da indústria, com massa e propriedades inerciais provenientes deste veículo (1000 kg, 1504 x 1645 x 3689 mm).

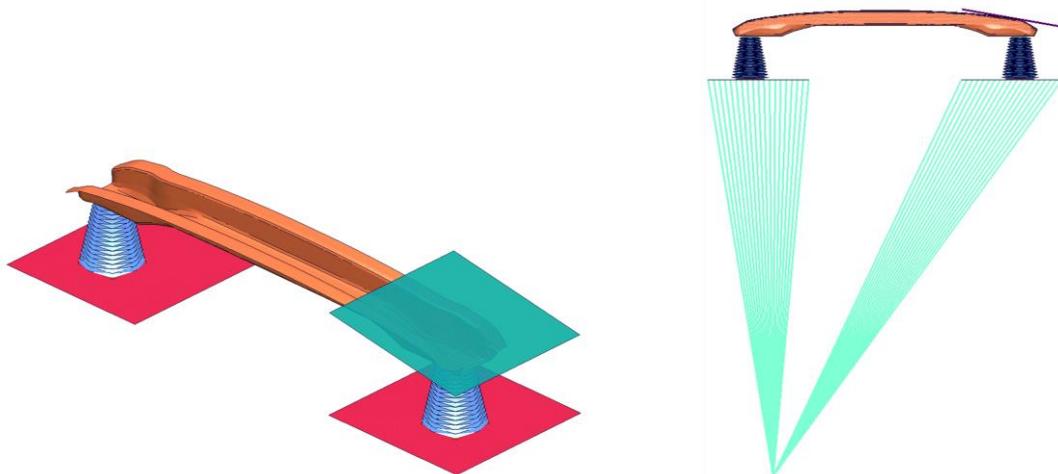
O para-choques é formado por duas *crash box* origami, conectadas ao modelo de veículo através de duas placas rígidas e a uma lâmina de para-choques adaptada, conforme Figura 5-2. Os elementos de conexão utilizados foram os conectores PAMCRASH TIED. O sistema de para-choques auxilia a reduzir a instabilidade das *crash box* durante o teste de impacto, através da conexão entre as duas *crash box*, enquanto a geometria simplificada do veículo reduz o custo computacional da simulação.

Figura 5-1. Modelo de simulação – condições de teste



Fonte: Próprio Autor.

Figura 5-2. Modelo de simulação – para-choques e impactor (esquerda) e corpo rígido conectado ao centro gravitacional do veículo (direita)



Fonte: Próprio Autor.

5.2. OTIMIZAÇÃO POR METAMODELOS NO MODEFRONTIER

Uma otimização para simulações numéricas acontece quando, após um número finito de repetições das simulações, atinge-se um conjunto de parâmetros que apresenta a melhor performance virtual para uma função objetivo. O diagrama de

processo para uma otimização é definido por Cavazzuti (2013), de acordo com a Figura 5-3.

As configurações básicas para o loop de otimização, que foi coordenado pelo software modeFRONTIER, são:

Identificar o problema: Maximizar a energia absorvida pela *crash box*, enquanto minimiza a força de pico no impacto.

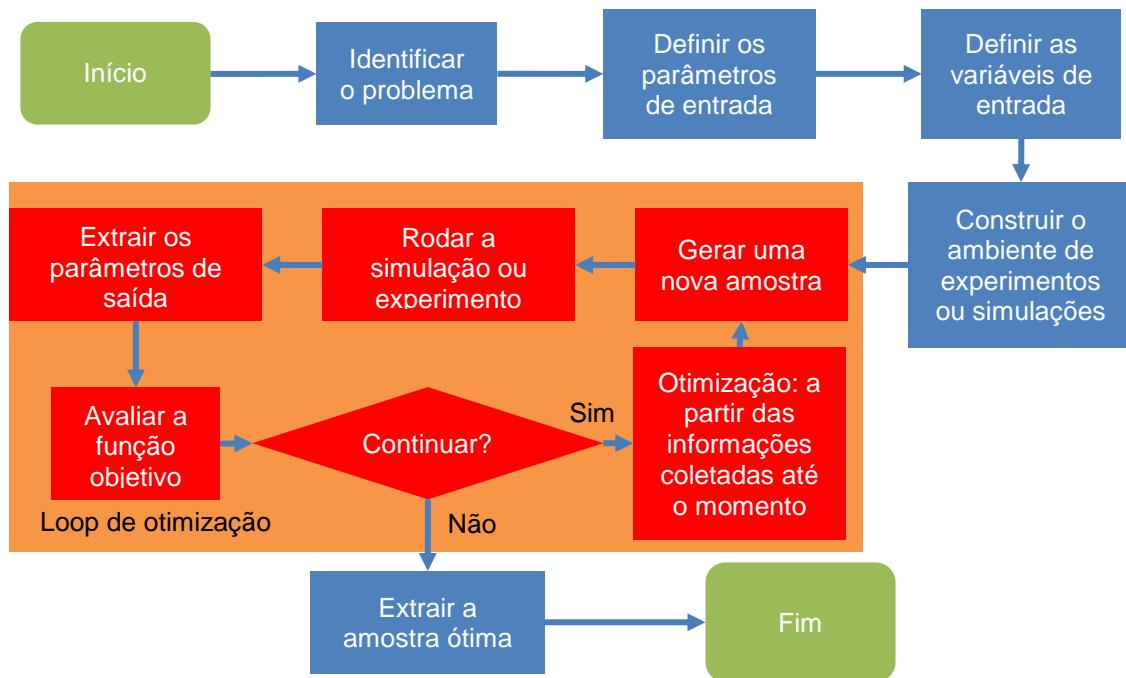
Definir os parâmetros de entrada: Geometria parametrizada dos tubos, de acordo com as variações do módulo origami básico na Figura 3-6; condições de contorno e carga inicial. A maioria dos parâmetros de entrada são definidos no protocolo de teste de baixa velocidade da RCAR.

Definir as variáveis de entrada: Todos os parâmetros de entrada podem ser utilizados como variáveis. No entanto, como Cavazzuti ressalta, é importante ter em mente que a complexidade do problema de otimização cresce exponencialmente com o número de variáveis. Portanto, o número de variáveis de entrada deveria ser mantido o menor possível, e um estudo preliminar para estimar quais são as variáveis mais importantes pode ser valioso. Neste caso, as variáveis de entrada podem ser um subgrupo dos parâmetros de entrada.

Construção do ambiente de simulação: o ambiente de simulação é:

- Pré-processador: O modelo parametrizado de elementos finitos foi construído e é reconstruído no software BetaCAE Ansa 18.1.0.
- Solver: O solver para todas as simulações é o ESI Virtual Pam-Safe (PAMCrash) 2014.05.
- Pós-processador: As simulações foram pós-processadas no software GNS Animator 4 2.1.3.
- Software de Otimização: As simulações foram guiadas através do software ESTECO modeFRONTIER 2016 – o algoritmo FAST, a ser explicado nas próximas seções.

Figura 5-3. Diagrama de processo de otimização



Fonte: (CAVAZZUTI, 2013)

5.3. VARIÁVEIS DE ENTRADA

As variáveis de entrada para a otimização da *crash box origami* são (ver Figura 3-6):

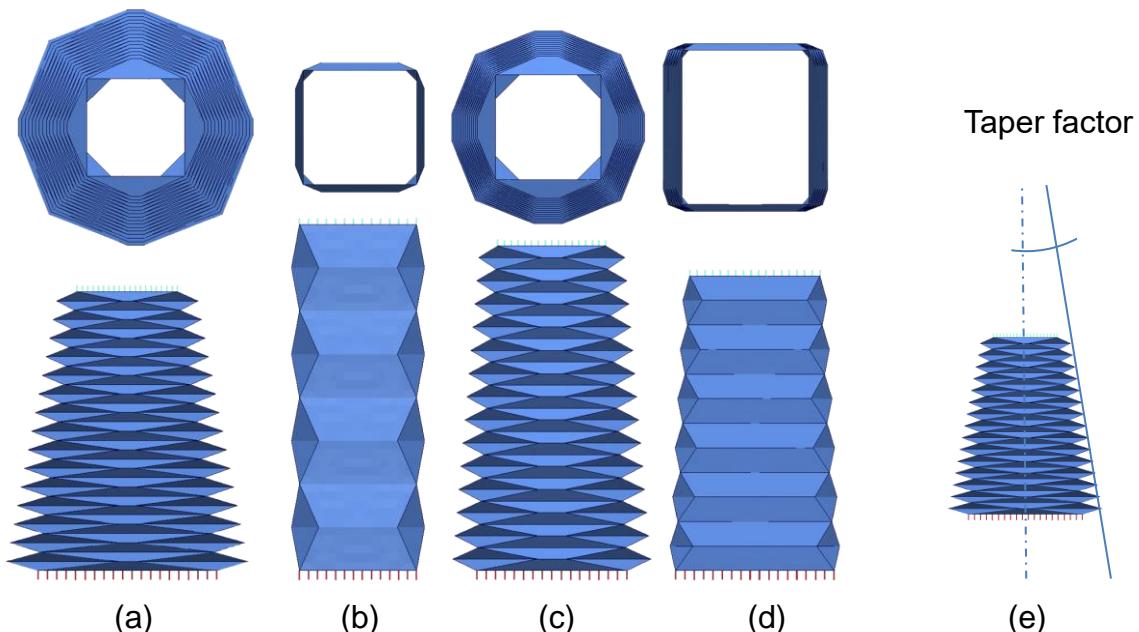
- l : Altura do módulo, antes da dobragem;
- b : largura de cada lado do módulo;
- c : largura da dobragem;
- t : espessura do tubo;
- N : número de lados do tubo;
- *Taper*: tipo de inclinação do tubo. São quatro tipos de alteração:
 - Sem inclinação (Figura 5-4b);
 - Apenas de topo, simétrica nos dois eixos do plano (Figura 5-4a);
 - Apenas de topo, com alteração de apenas um eixo (topo retangular) (Figura 5-4c);

- No topo e na base, com alteração de apenas um eixo (forma retangular) (Figura 5-4d).
- *Taper Factor*: fator que guia a inclinação do tubo (Figura 5-4e).

O ângulo 2θ depende da relação entre as variáveis. Ele é calculado pela seguinte equação:

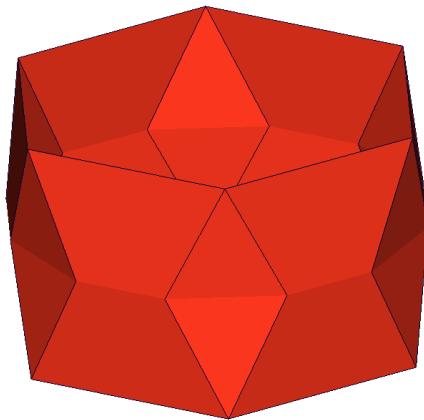
$$\cos \theta = \tan \left(\frac{\pi}{2N} \right) \frac{c}{l} \quad (36)$$

Figura 5-4. Diferentes tipos de inclinação dos tubos (taper)



Fonte: Próprio Autor.

Figura 5-5. Padrão origami para tubos hexagonais ($N = 6$ lados)



Fonte: Próprio Autor.

5.4. SIMULAÇÃO NUMÉRICA DO PROBLEMA

Conforme já mencionado, o ambiente de simulação foi coordenado pelo software modeFRONTIER. O processo básico do modeFRONTIER é composto pelo fluxo de processos e pelo fluxo de dados. O fluxo de processos determina a sequência cronológica das simulações usadas na avaliação de modelos. O fluxo de dados descreve quais dados devem ser considerados em cada passo do fluxo de processo. O fluxo de otimização é executado toda vez que uma avaliação de modelo é requisitada (ESTECO, 2016).

Como explicado anteriormente, o modelo de simulação é um veículo simplificado, com um para-choques feito de duas *crash box* origami e uma lâmina de para-choques (Figura 5-2). O material da *crash box* é considerado elasto-plástico com dano isotrópico (material do tipo 105 da biblioteca de materiais do PAMCRASH). O material é para elementos do tipo casca. As características do material necessárias para a simulação foram obtidas das propriedades experimentais do material DP-780, um aço automotivo de duas fases, como descrito na Tabela 9. A Tabela 9 mostra as propriedades elásticas E (Módulo de Young), ρ (densidade), ν (Poisson) e A_s (fator de correção transversa de cisalhamento). Tabela 10 mostra os parâmetros

elastoplásticos definidos por Calle, Oshiro and Alves (2017), de acordo com a tensão de escoamento descrita por Alves (2000) e definida abaixo,

$$\sigma = [k + R_0 \varepsilon_p + R_\infty (1 - e^{-b\varepsilon_p})] \left[1 + \left(\frac{\dot{\varepsilon}}{D} \right)^{1/p} \right] \quad (37)$$

onde k é a tensão de escoamento inicial; R_0 , R_∞ e b são as variáveis do modelo de material descrito em (VOCE, 1955), ε_p é a deformação plástica equivalente, $\dot{\varepsilon}$ é a taxa de deformação e D e p são parâmetros do modelo Cowper-Symonds. A Figura 5-6 mostra as curvas de tensão-deformação de acordo com as taxas de deformação.

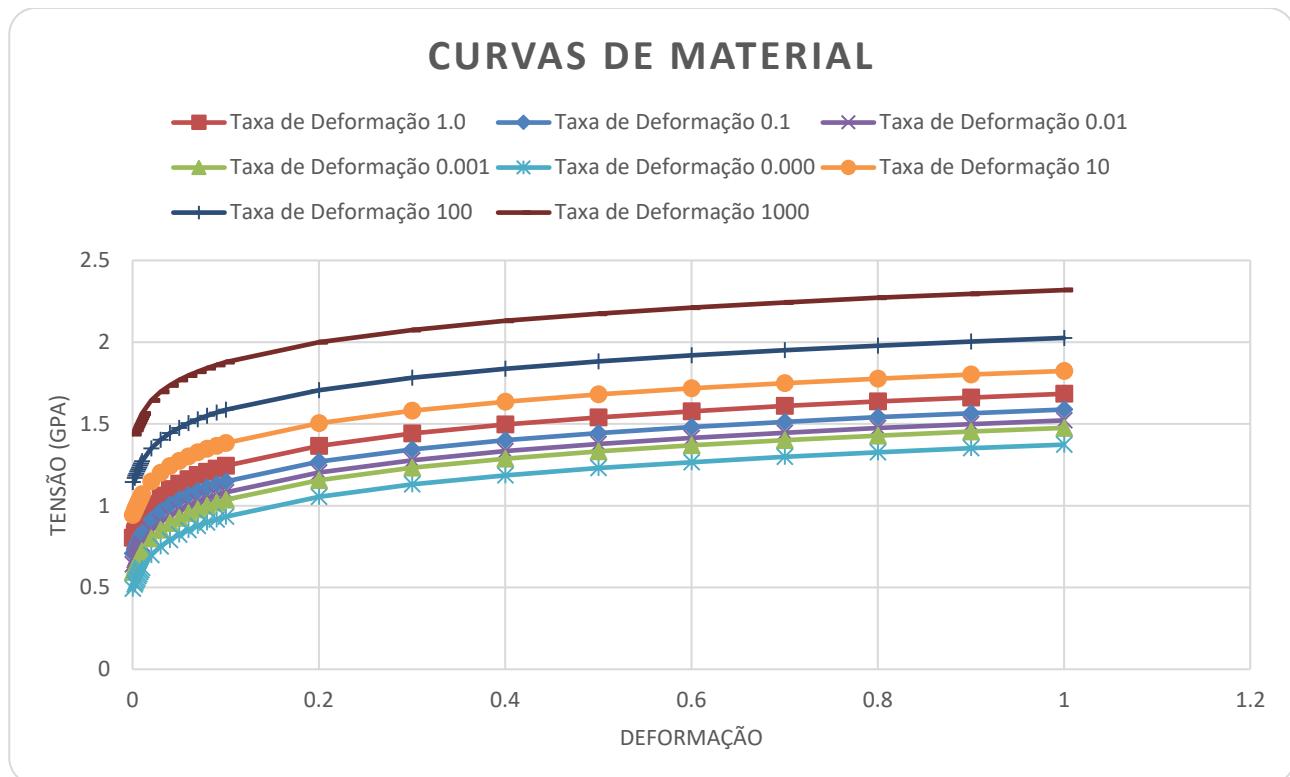
Tabela 9. Propriedades do Aço

$E(\text{GPa})$	$\rho (\text{kg/mm}^3)$	ν
210	7.85e-06	0.3

Tabela 10. Parâmetros ajustados a partir da lei dos materiais em (CALLE; OSHIRO; ALVES, 2017)

$k (\text{MPa})$	$R_0 (\text{MPa})$	$R_\infty (\text{MPa})$	b	$D (s^{-1})$	q
492.63	360	550.44	15	19.13	6.21

Figura 5-6. Lei dos materiais ajustada para o aço DP780 – Curvas de deformação obtidas através do modelo modificado Cowper-Symonds proposto em (ALVES, 2000)



Fonte: Próprio Autor.

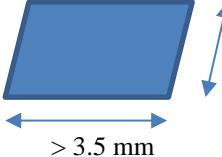
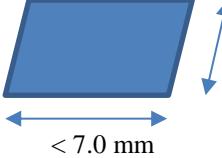
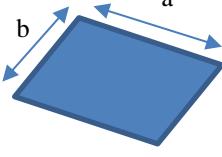
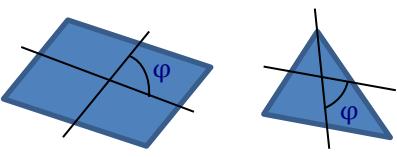
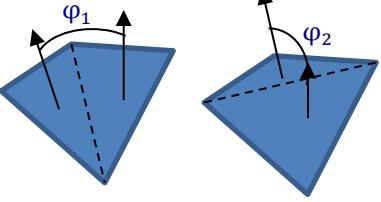
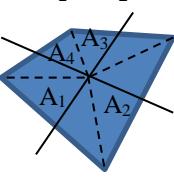
Foram utilizados elementos do tipo casca, que atendem às qualidades de malha listadas na Tabela 11.

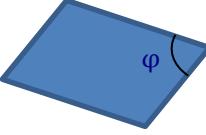
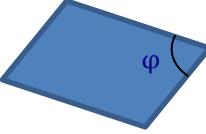
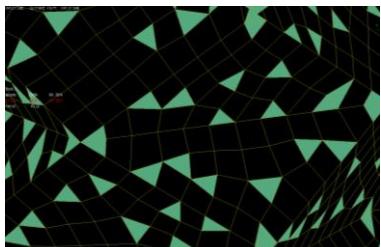
Tabela 11. Propriedades do modelo

Média de elementos por modelo de <i>Crash box</i>	Média de nós por modelo de <i>Crash box</i>	Peso do veículo	Dimensões do veículo
4900	4820	1000 kg	1504 x 1645 x 3689 mm

Tabela 12. Critérios de qualidade de malha usados no pré-processador. Definições extraídas de (BETACAE, 2016)

Critério	Tipo de Cálculo	Valor
Tamanho-alvo do elemento		5 mm

Critério	Tipo de Cálculo	Valor
Tamanho mínimo de elemento		3.5 mm
Tamanho Máximo de elemento		7.0 mm
Tipo de elementos		Quads e Trias
Aspect ratio	<p>NASTRAN $\text{aspect ratio} = \max\left(\frac{a}{b}\right)$</p> 	3
Skewness	<p>PATRAN $\text{Skew angle} = 90^\circ - \varphi < 45^\circ$</p> 	45°
Warping	<p>IDEAS $\text{Warping angle} = \max(\varphi_1, \varphi_2) < 10^\circ$</p> 	10°
Taper	<p>IDEAS $\text{Taper} = \frac{4 \cdot \min(A_1, A_2, A_3, A_4)}{A_1 + A_2 + A_3 + A_4} > 0.35$</p> 	0.35

Critério	Tipo de Cálculo	Valor
Mínimo ângulo de quads	IDEAS Angle = $F_{min} < \varphi < F_{max}$	 45°
Máximo ângulo de quads	IDEAS Angle = $F_{min} < \varphi < F_{max}$	 135°
Mínimo ângulo de trias	IDEAS Angle = $F_{min} < \varphi < F_{max}$	 30°
Máximo ângulo de trias	IDEAS Angle = $F_{min} < \varphi < F_{max}$	 120°
Porcentagem de trias		 20%

5.5. OTIMIZAÇÃO – OBJETIVOS

De acordo com Ma (2011), apesar da ampla pesquisa feita para desenvolver absorvedores energéticos eficientes, ainda é um desafio conceber um sistema com baixa força de flambagem inicial, alto SEA, modo de falha estável, que se encaixe bem na estrutura principal, e que possa ser manufaturado a um custo razoável (ver também Lu & Yu, 2003). Com foco neste desafio, o autor definiu duas funções-objetivo da otimização como:

$$\text{maximizar } SEA = \frac{(E_{cb}^{def})_{\max}}{m_{cb}} \quad (38)$$

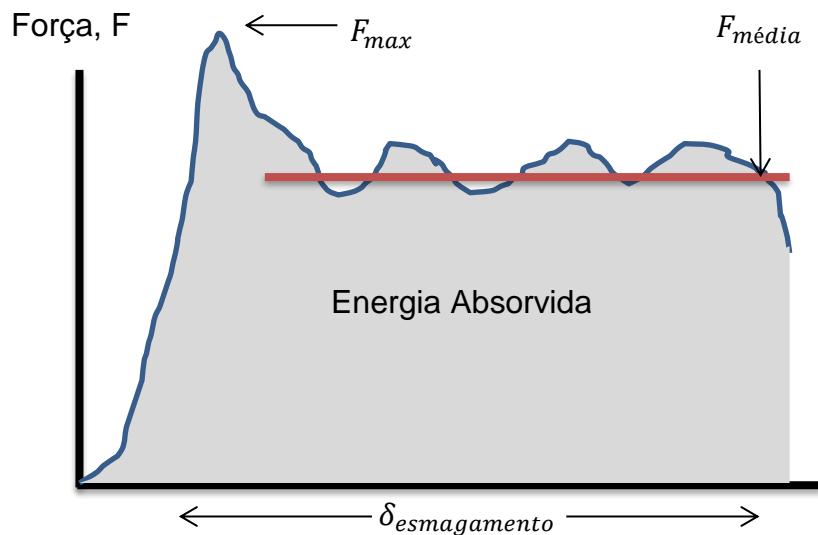
$$\text{maximizar } LU = \frac{F_{média}}{F_{\max}} \quad (39)$$

Onde F_{\max} é a força de pico entre o impactor e o para-choques durante a simulação; $\delta_{esmagamento}$ é a distância máxima de esmagamento da *crash box*; E_{cb}^{def} é a energia interna das *crash box*, que nada mais é que a energia absorvida pelas *crash box* durante o impacto; m_{cb} é a massa das *crash box*; $F_{média}$ é a força média de esmagamento durante o impacto, definida como:

$$F_{média} = \frac{\int_0^{\delta_{esmagamento}} F(x) dx}{\delta_{esmagamento}} \quad (40)$$

A uniformidade de carga utilizada neste trabalho é o inverso da uniformidade de carga definida por Ma (2011). Esta foi uma decisão tomada para manter a LU no domínio $[0, 1]$, evitando problemas numéricos durante a otimização.

Figura 5-7. Resposta carga-deflexão de um absorvedor de energia



Fonte: Próprio Autor.

5.6. RESTRIÇÕES

Como esperado, nem todas as combinações entre variáveis de entrada são possíveis. Portanto, um passo necessário é definir os domínios para cada variável, e também as restrições do problema. As restrições estão listadas na Tabela 13:

Tabela 13. Restrições do problema

Restrição	Descrição	Valor mínimo	Valor máximo
b	Lado da <i>crash box</i>	40 mm	100 mm
l	Altura do módulo antes da dobragem	20 mm	180 mm
$h = \frac{l}{2} \sin(\theta)$	Altura do módulo depois da dobragem	> 0	< l
c	Tamanho da dobragem (vale) Espessura do tubo – dimensões comuns na indústria automotiva	$10\% * b$	$90\% * b$
t	Número de lados. Ma (2011) cita que quando tubos poligonais têm mais de 12 lados, comportam-se como tubos cilíndricos.	1.50 mm	2.10 mm
N		3	13
$\theta = \arccos \left(\tan \left(\frac{\pi}{2N} \right) \left(\frac{c}{l} \right) \right)$	Ângulo da dobragem	$> 0^\circ$	$< 90^\circ$
N^*h	Tamanho da <i>crash box</i> , baseado no tamanho usual para veículos de passeio	140 mm	200 mm
Módulos	Quantidade de módulos. 20 módulos para uma <i>crash box</i> de 140 mm resultam em 7 mm de altura por módulo.	$\max \left(1, \lfloor \frac{200}{h} \rfloor \right)$	$\min \left(20, \lceil \frac{200}{h} \rceil \right)$
$\frac{b}{\sin \left(\frac{\pi}{N} \right)}$	Raio da circunferência circunscrita aos vértices da base da <i>crash box</i>	40 mm	140 mm
Fator	Fator de alteração do módulo	30%	100%

Há um ponto crucial a ser considerado quando variamos o número de módulos. A altura do módulo, l , e o número de módulos não são mutuamente independentes. Não é permitida a adição de um segundo módulo na mesma *crash box*, de mesmo formato, caso a altura somada dos dois módulos viole a restrição de tamanho total da *crash box*. A expressão abaixo foi utilizada para selecionar aleatoriamente uma quantidade de módulos, baseada na quantidade permitida:

$$h = l \sin \left[\arccos \left(\tan \left(\frac{\pi}{2N} \right) \frac{c}{l} \right) \right] \quad (41)$$

$$Modulos_{max} = floor \left(\frac{200}{h} \right) \quad (42)$$

$$Modulos_{min} = ceil \left(\frac{140}{h} \right) \quad (43)$$

$$Modulos = round \left[rand() \cdot (Modulos_{max} - Modulos_{min}) + Modulos_{min} \right] \quad (44)$$

Onde:

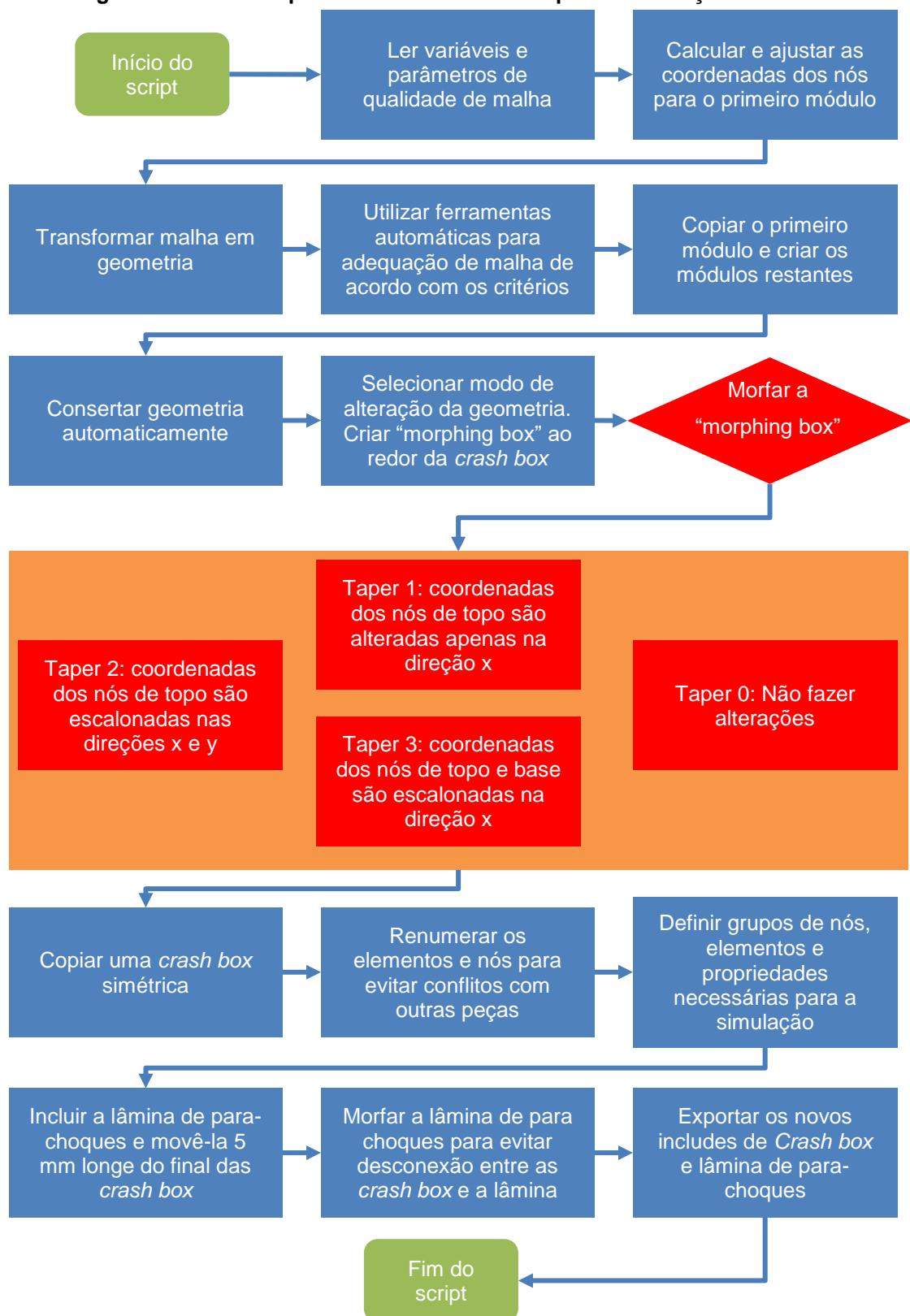
- h é a altura do módulo após a dobraria,
- $rand()$ é uma função que seleciona um número pseudoaleatório entre 0 (inclusive) e 1 (excluído),
- $floor$ é uma função que arredonda para baixo o argumento de entrada,
- $ceil$ é uma função que arredonda para cima o argumento de entrada,
- $round$ é uma função que arredonda um número para o número inteiro mais próximo.

No fluxo de trabalho do modeFRONTIER, todas as restrições foram colocadas em suas devidas variáveis, e as restrições globais foram colocadas no nó de pré-processamento.

5.7. PRÉ-PROCESSAMENTO

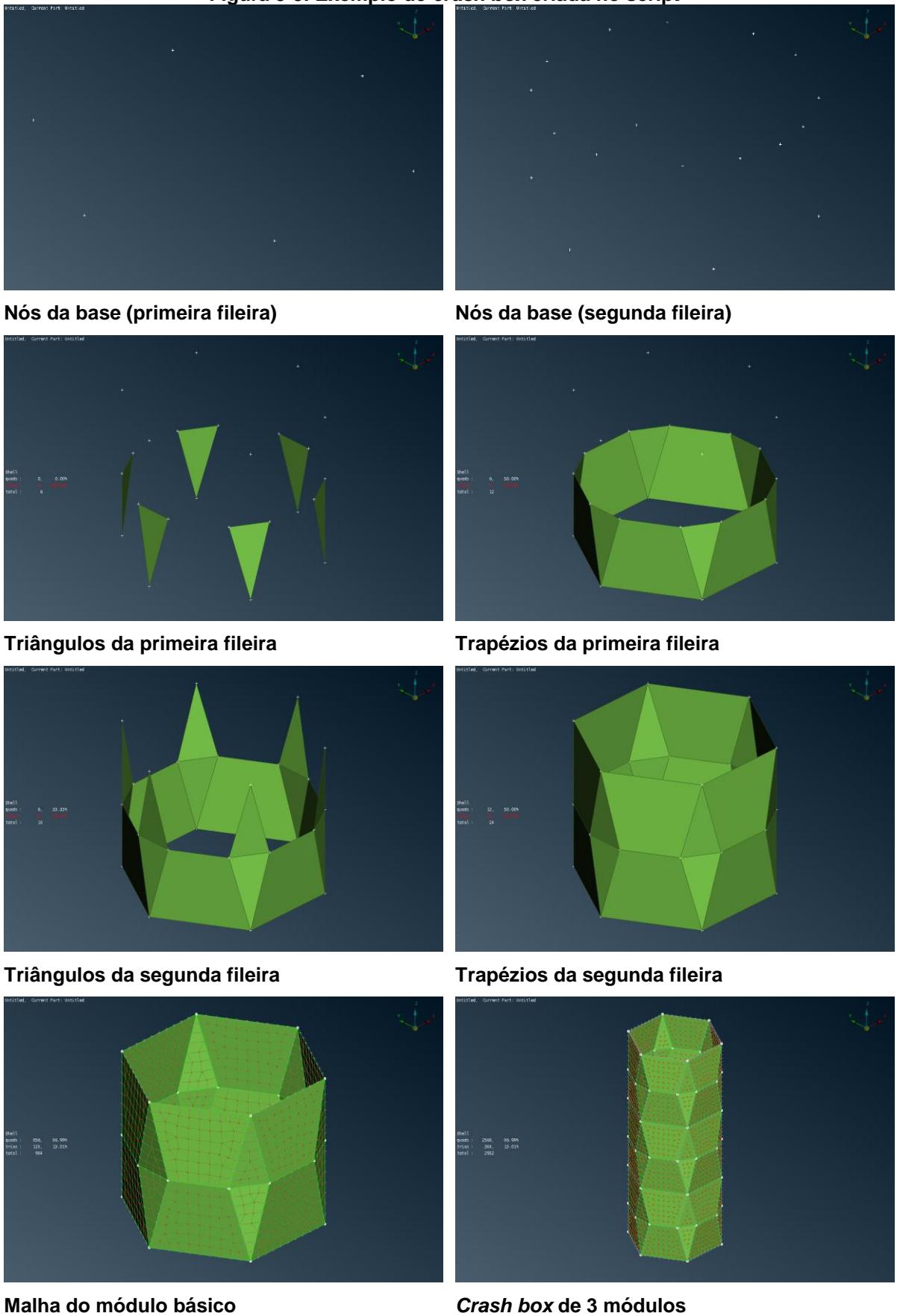
O software BETACAE Ansa 18.1.0 é configurável para rodar em segundo plano. Os scripts são escritos em Python, e utilizam a biblioteca implementada no próprio software. O algoritmo escrito para a criação de cada modelo de crash box está representado na Figura 5-8. O programa lê as novas variáveis, gera uma nova malha e geometria do tubo e o conecta às outras partes do modelo. O script completo está disponível no ANEXO C, e um exemplo em figuras está na Figura 5-9.

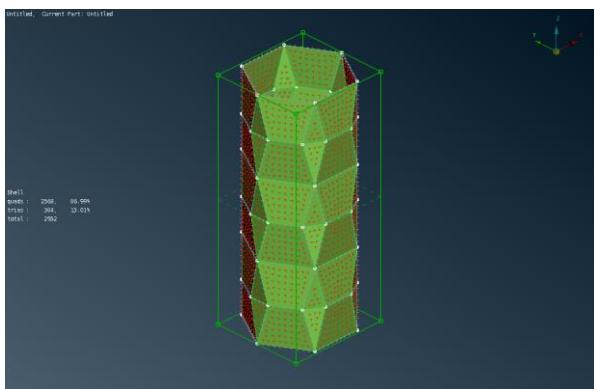
Figura 5-8. Fluxo do processo utilizado no script de atualização da crash box.



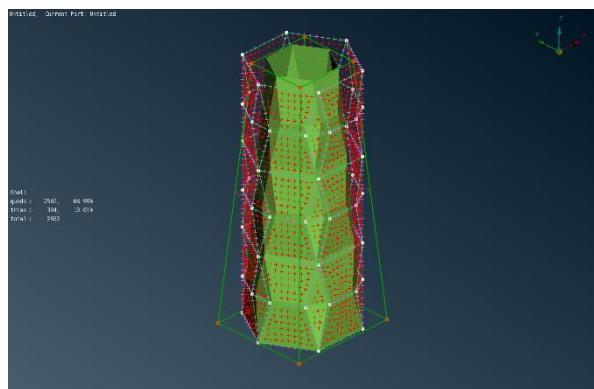
Fonte: Próprio Autor.

Figura 5-9. Exemplo de *crash box* criada no script

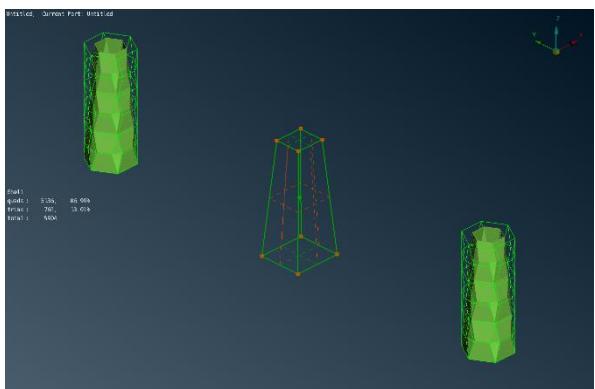




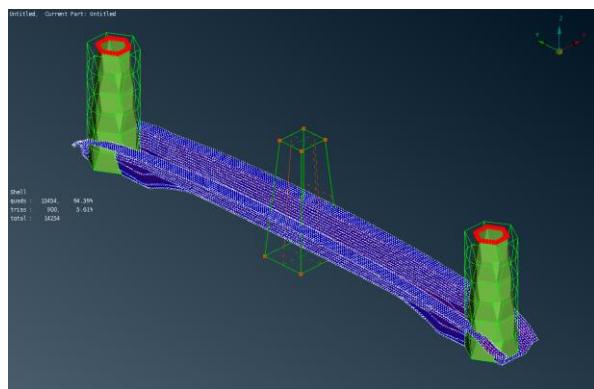
Morphing box na crash box



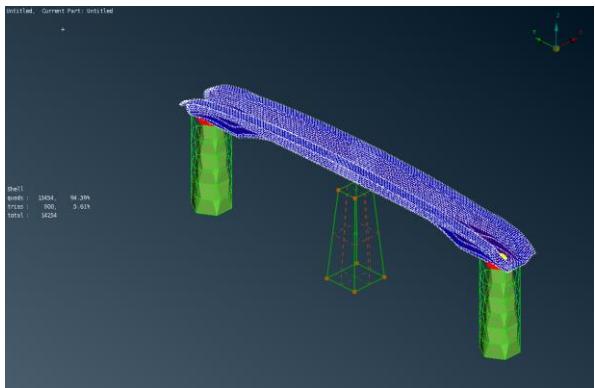
Crash box alterada segundo fator de inclinação



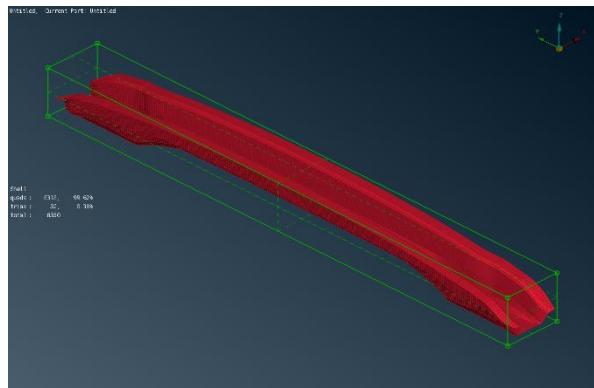
Cópia da crash box com simetria ao plano XZ



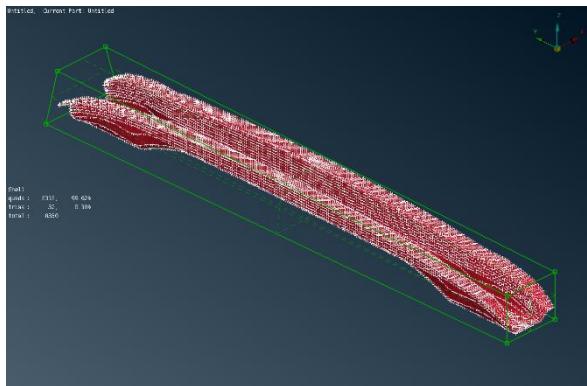
Inserção da lâmina de para-choques básica



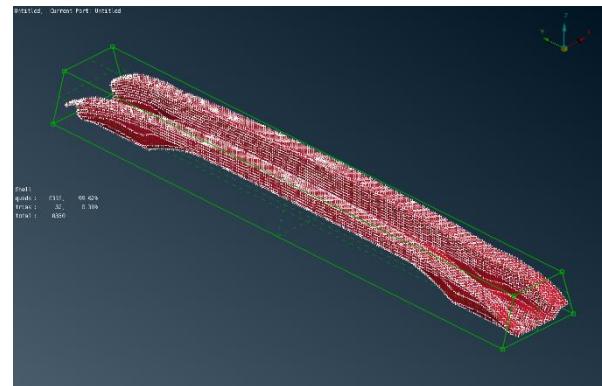
Movimentação da lâmina para a altura certa



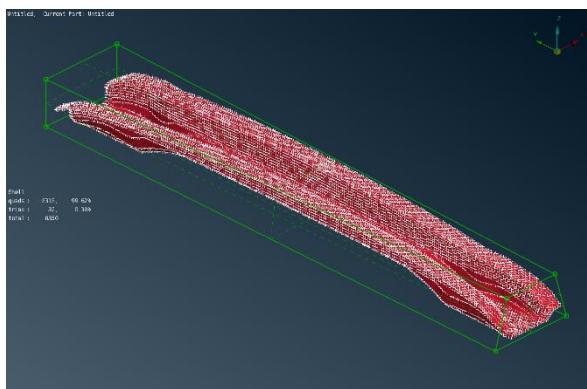
Morphing box na lâmina do para-choques



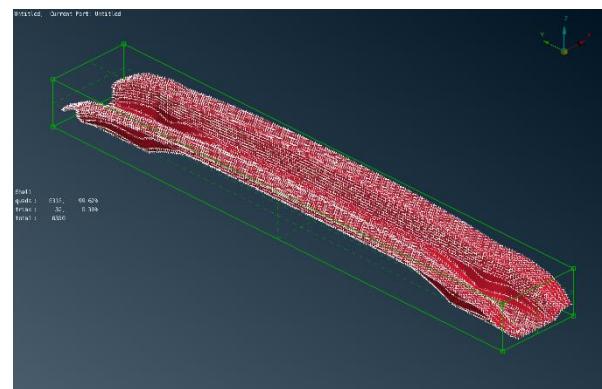
Alteração dos dois primeiros vértices da *morphing box*



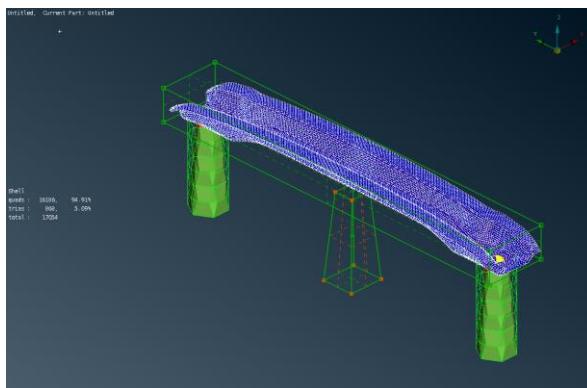
Alteração de mais dois vértices da *morphing box*



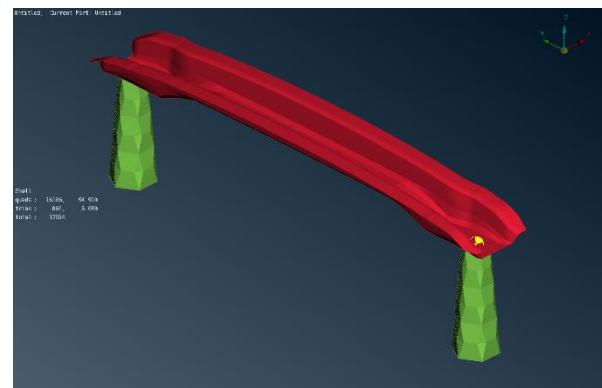
Alteração de mais dois vértices da *morphing box*



Alteração dos últimos dois vértices da *morphing box*



Sistema de para-choques pronto para exportar



Sistema de pára-choques exportado

Fonte: próprio autor

5.8. PÓS-PROCESSAMENTO

O software de pós-processamento Animator 4 2.1.3 é utilizado para coletar os seguintes dados de saída:

- Força de contato entre o impactor e o para-choques ao longo do tempo de simulação;
- Deslocamento do impactor ao longo do tempo de simulação;
- Massa da *crash box*;
- Energia interna da *crash box* ao longo do tempo de simulação.

A partir destas informações, tanto a SEA (equação 38) quanto a LU (equação 39) puderam ser calculadas:

- F_{max} – Pico da força de contato
- $F(x)$, $\delta_{esmagamento}$, $F_{média}$ – advindas do deslocamento do impactor e da força de contato
- E_{cb}^{def} – Advinda da energia interna da *crash box*.

5.9. ALGORITMO DO modeFRONTIER – FAST

Montrone, Turco and Rigoni (2014) declaram que a abordagem via superfícies de resposta (RSM) ou metamodelos encontra um conjunto satisfatório de soluções ótimas dentro de um número reduzido de avaliações, acelerando o processo de otimização. Um conjunto de dados, ou dados para treinamento são utilizados para ajustar o modelo, isto é, para aprender os parâmetros do sistema. Além do mais, um conjunto de validação é necessário para avaliar se o modelo pode resolver corretamente novos exemplos.

O algoritmo FAST disponível no software modeFRONTIER utiliza a abordagem por metamodelos. O loop principal do FAST está na Figura 5-10.

Figura 5-10. Loop principal do algoritmo FAST



Fonte: (MONTRONE; TURCO; RIGONI, 2014)

Inicialmente, um grupo inicial de dados é gerado, com variáveis de entrada baseadas em DOE e dados de saída obtidos por simulações numéricas do problema via PAMCRASH. Então, quatro modelos de superfície de resposta (ou metamodelos) para cada função objetivo são treinados com os dados iniciais, através de quatro métodos: SVD polinomial, função de base radial, Kriging e Redes neurais. Uma descrição completa destes algoritmos foi feita no capítulo de revisão bibliográfica.

A performance dos metamodelos é computada através do erro quadrático normalizado nos pontos de validação e os melhores metamodelos são selecionados para os próximos passos. Em seguida, o algoritmo ISF é usado para enriquecer a exploração do espaço de design em torno dos pontos na fronteira de Pareto. Os melhores m pontos, em termos do ranking de Pareto e distâncias de aglomeração são selecionados para o processo de validação.

O próximo passo é rodar o algoritmo de otimização virtual, isto é, no algoritmo os metamodelos selecionados são usados para obter uma resposta próxima à resposta do sistema. Durante o passo de otimização virtual, um DOE de tamanho m , construído a partir de uma amostra aleatória de 50% dos pontos da fronteira de Pareto mais recente e 50% de pontos gerados por um DOE aleatório, é usado como população inicial da otimização. Novamente, os melhores m pontos em termos de ranking de Pareto e distâncias de aglomeração do conjunto de dados completo avaliado durante a otimização virtual são selecionados para o processo de validação. (MONTRONE; TURCO; RIGONI, 2014).

Durante o processo de validação, m pontos, de $2m$ previamente definidos, são aleatoriamente selecionados e validados através de análises numéricas por elementos finitos. O desempenho do metamodelo é avaliado e estes m pontos são também usados para enriquecer o banco de dados a ser usado na próxima iteração.

5.9.1. Algoritmo genético no modeFRONTIER

No passo de otimização do metamodelo, o projetista tem diversas técnicas à sua disposição. Neste trabalho, a técnica utilizada foi o algoritmo MOGA-II – algoritmo genético multiobjetivo. Algoritmos genéticos são um método de otimização que realizam uma busca no espaço de design, armazenando e atualizando as iterações com pontos discretos de uma potencial solução. Cada adição é comparada com todos os valores das funções objetivo, para determinar se os novos pontos são dominados ou não. Se os pontos não são dominados, então o grupo é armazenado como uma nova potencial solução – mesmo que o ponto ainda não seja ótimo de Pareto. (POLES, 2003; ARORA, 2016)

Como os algoritmos genéticos não necessitam dos gradientes das funções objetivo para serem executados, eles podem ser efetivos independentemente da natureza das funções do problema. O uso de números aleatórios e as informações das iterações anteriores são combinados para avaliar e melhorar uma população de pontos (um grupo de soluções em potencial), ao invés de um ponto individual por vez.

A ideia básica de um algoritmo genético é gerar um novo grupo de modelos (população) a partir do grupo atual, de forma que a média de resultados da população melhore. O processo continua até atingir um critério de parada ou o número de

iterações exceder o limite especificado. Três operadores genéticos são usados para completar esta tarefa: reprodução, cruzamento e mutação. (ARORA, 2016)

Reprodução ou seleção é um operador no qual um modelo antigo é copiado na nova população, de acordo com o desempenho do modelo. *Cruzamento* é quando dois modelos da nova população trocam algumas características entre si. *Mutação* é o terceiro passo que protege o processo de uma perda prematura de material genético valioso durante a reprodução e o cruzamento. (ARORA, 2016)

Exemplificando em uma *string* binária, podemos ilustrar o cruzamento de acordo com a Figura 5-11, e a mutação de acordo com a Figura 5-12.

Figura 5-11. Ilustração de um cruzamento.

Cromossomo 1	11011 001
Cromossomo 2	11011 110
Descendência 1	11011 110
Descendência 2	11011 001

Fonte: Adaptado de (OBITKO, 1998)

Figura 5-12. Ilustração de uma mutação.

Descendência Original 1	110 1 1110
Descendência Original 2	110110 0 1
Descendência Mutada 1	110 0 1110
Descendência Mutada 2	110110 1 1

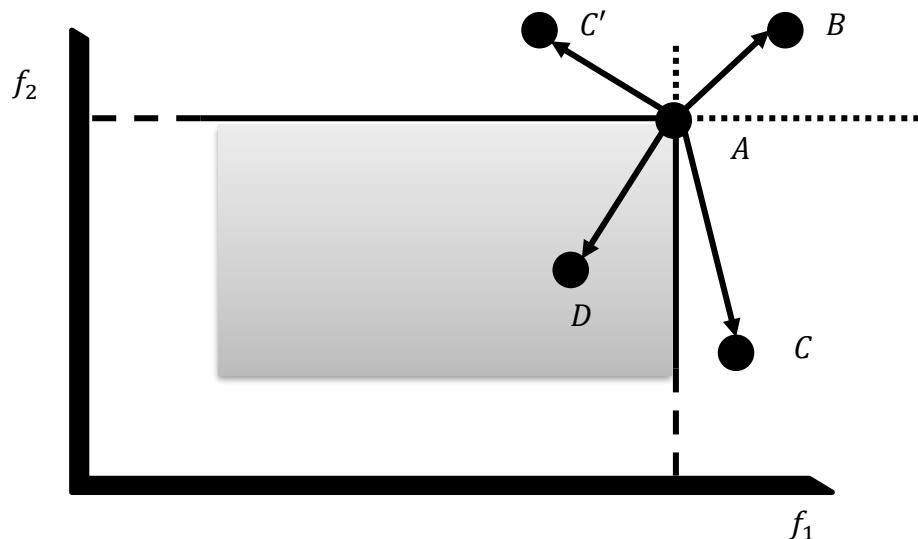
Fonte: Adaptado de (OBITKO, 1998)

Existem alguns métodos de seleção de uma geração para a próxima. No modeFRONTIER, o método aplicado é o Elitismo. Nesse método, antes da execução dos passos de cruzamento e mutação, os melhores modelos são copiados para a próxima geração. O elitismo no algoritmo MOGAI é aplicado como segue (traduzido de (POLES, 2003)):

1. O algoritmo MOGAIl começa com a população inicial P de tamanho N e grupo de elite $E = \emptyset$.
2. Para cada geração, computa $P' = P \cup E$
3. Se a cardinalidade de P' é maior que a cardinalidade de P , reduz P' removendo aleatoriamente os pontos em excesso
4. Computa a evolução de P' para P'' aplicando operadores MOGA (mutação e cruzamento)
5. Calcula a adequação para a população P''
6. Copia todos os pontos fora do domínio atual de P'' para E
7. Atualiza E removendo pontos duplicados ou dominados
8. Redimensiona o grupo de elite E se é maior que o tamanho de geração N removendo aleatoriamente pontos em excesso
9. Retorna ao passo 2 considerando P'' como novo P

POLES (2003) exemplifica o que é um ponto dominado através de um exemplo analítico simples. Considerando duas funções f_1 e f_2 , e um ponto inicial A, temos algumas possibilidades de pontos: B, C, C' e D. Ver Figura 5-13. Enquanto o ponto D é dominado por A, os pontos B, C, C' não são. Se A é um ponto de uma geração anterior e a geração atual contém o ponto B, então a nova posição é bastante favorável: não apenas B é não-dominado por A, mas também B domina A. Este tipo de evolução é sempre desejada, e esta transição certamente deve ser preservada. Se a evolução leva A para C (ou C'), o novo ponto também não é dominado por A e deve ser preservado, a fim de constituir uma nova Fronteira de Pareto.

Figura 5-13. O ponto A divide o espaço destes dois objetivos em duas áreas: o grupo de pontos dominados é representado pela área sombreada. A domina D, enquanto B, C e C' não são dominados.



Fonte: Adaptado de (OBITKO, 1998)

6. RESULTADOS

Os resultados de uma otimização multiobjetivo raramente são únicos. A melhoria de um aspecto geralmente compromete o outro. O desafio está na determinação do ponto ótimo de Pareto, isto é, qualquer melhoria incremental não pode ser atingida em qualquer dos objetivos sem prejudicar ao menos um dos outros (ESTECO, 2016).

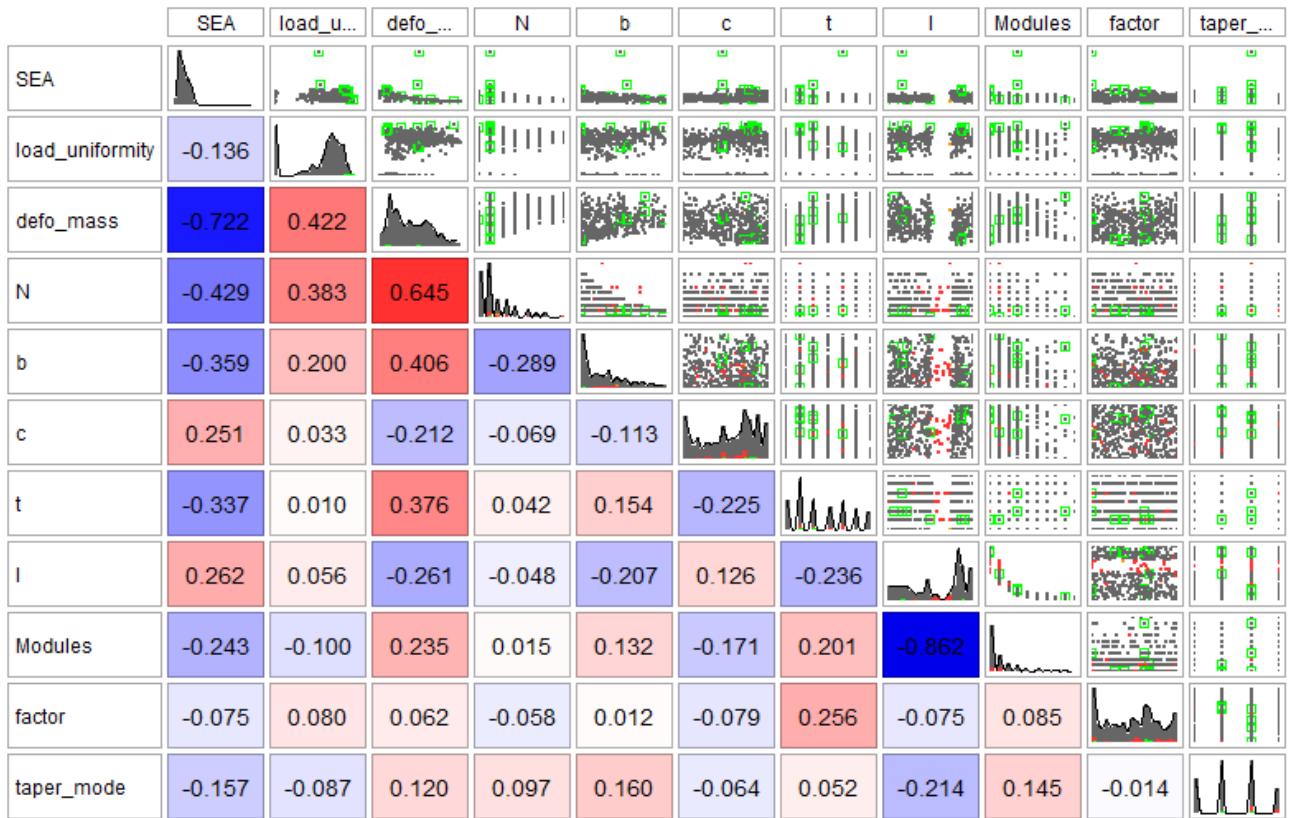
O conjunto dos melhores pontos em um espaço de design é conhecido como conjunto ótimo de Pareto, e constitui a fronteira de Pareto em um gráfico, como o mostrado na Figura 6-2. Ele representa a dispersão de pontos do processo de otimização, e os pontos ótimos de Pareto estão marcados em verde. Além do mais, a matriz de dispersão da Figura 6-1, como dito em (ESTECO, 2016), é uma ferramenta extremamente útil para verificar se há alguma relação linear entre as variáveis. A Matriz de Dispersão é desenhada junto à linha de regressão e auxilia a visualização das soluções ótimas e o entendimento de como elas são distribuídas no espaço.

6.1. MATRIZ DE DISPERSÃO

Observando a Figura 6-1, é possível concluir que a variável t tem alta correlação com a quantidade de módulos. Como o número de módulos é limitado pelo tamanho total da *crash box*, isso já é esperado. O SEA é altamente correlacionado ao número de lados, N . Como o SEA é calculado pela taxa entre a energia absorvida e a massa da *crash box*, e esta massa (no gráfico, *defo mass*) tem uma correlação positiva com o número de lados, é esperada uma correlação negativa entre a SEA e as variáveis N , b , t , e a quantidade de módulos – veja Figura 6-1.

A correlação positiva entre a massa da *crash box* e a uniformidade de carga LU também é previsível. Quanto mais pesada a *crash box*, mais provável que ela tenha uma diferença pequena entre a força de pico e a força média de esmagamento.

Figura 6-1. Matriz de dispersão das variáveis de entrada e saída

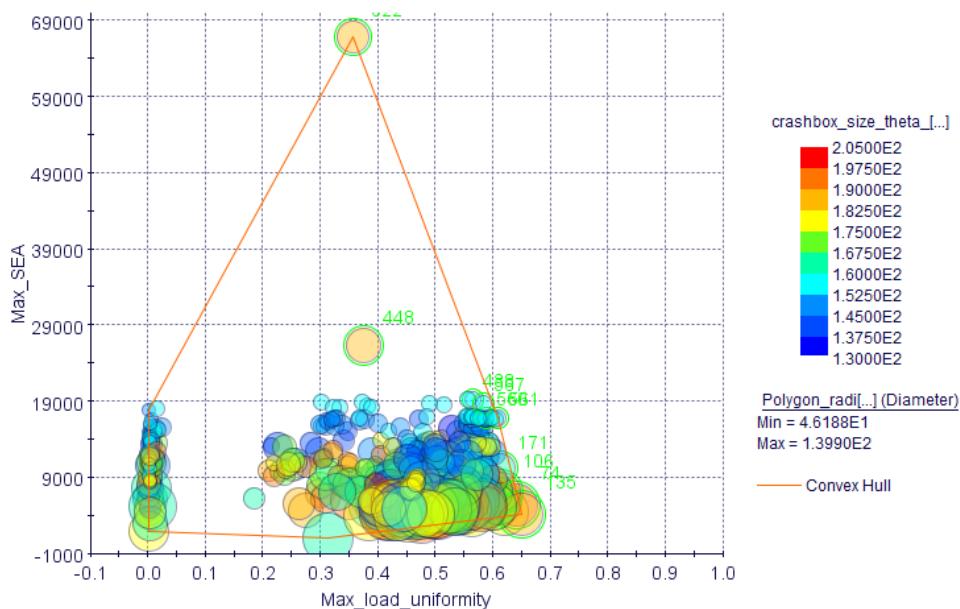


Fonte: Próprio Autor. Produzido no software modeFRONTIER.

6.2. PONTOS FORA DA CURVA E MODELOS INFATÍVEIS

Na Figura 6-2, vemos os objetivos LU e SEA para as configurações otimizadas finais. Apesar do software de otimização ter removido diversos modelos infatíveis durante a otimização, que violavam as restrições, ainda existem alguns pontos fora da curva que devem ser apontados aqui. Valores de LU próximos a zero significam que ou a força média de esmagamento foi extremamente baixa durante o impacto, ou a força de pico foi extremamente alta. Estas situações ocorrem quando a crash box colapsa lateralmente, ao invés de colapsar uniformemente, ou quando a crash box é rígida por demais e não houve deformação plástica observável.

Figura 6-2. Gráfico de bolhas 4D. a escala de cores se refere ao tamanho total da crash box, e o diâmetro de cada círculo equivale ao raio do polígono básico da crash box.



Fonte: Próprio Autor. Produzido no software modeFRONTIER.

O algoritmo realizou algumas escolhas de parâmetros que geraram SEA não realísticos. Existem dois possíveis limites para o SEA, baseando-se na energia cinética de entrada do sistema e na combinação de variáveis geométricas que resultam em um intervalo possível para a massa. Esses limites estão apresentados na tabela 14:

Tabela 14. Limites de SEA

Limite	Energia Cinética Inicial (J)	N	t (mm)	b (mm)	Altura (mm)	Densidade (kg/mm³)	Massa das duas crash box (kg)	SEA (J/kg)
Máximo SEA	12152.8	10	2.10	43.26	140	7.85e-6	0.39564	30717
Mínimo SEA	12152.8	3	1.50	40	200	7.85e-6	2.85272	4260

Os modelos que apresentaram valores de SEA fora dos limites apresentados na tabela acima são descartados. Eles violam a restrição de tamanho da crash box ou a restrição de raio máximo do polígono de base.

6.3. FRONTEIRA DE PARETO

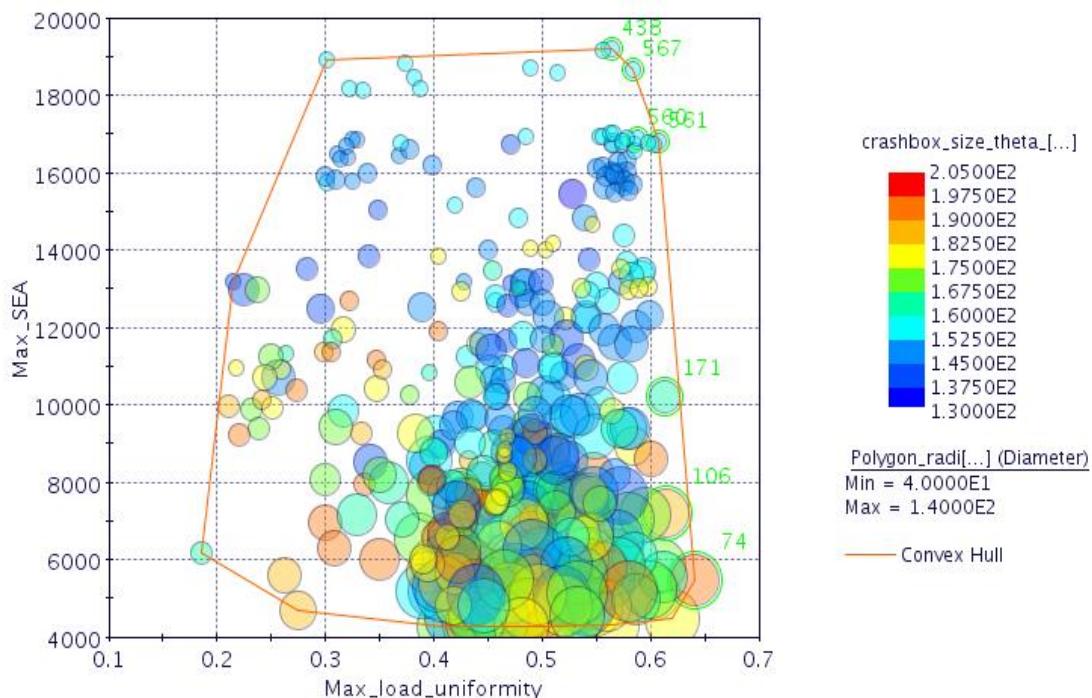
A Figura 6-3 ilustra a fronteira de Pareto, em verde, cujos parâmetros estão listados na Tabela 15:

Tabela 15. Parâmetros dos modelos da fronteira de Pareto

Id	t (mm)	Módulos	N	b (mm)	c (mm)	l (mm)	fator	Modo de alteração	Diâmetro do polígono (mm)	Comprimento da <i>crash box</i> (mm)
74	1.70	4	4	84.90	62.83	55.60	0.73	2	120.07	196.54
106	1.60	2	3	97.60	46.85	98.60	0.77	1	112.70	189.64
171	1.60	1	4	60.80	48.64	163.5 0	0.77	1	85.98	162.25
438	1.60	1	4	40.00	26.80	159.6 0	0.57	2	56.57	159.21
560	1.60	1	4	40.00	30.80	159.3 0	0.73	1	56.57	158.79
561	1.60	1	4	40.00	30.80	159.3 0	0.74	1	56.57	158.79
567	1.60	1	4	40.00	28.40	159.6 0	0.49	2	56.57	159.17

Ao examinar a tabela acima, constata-se que a maioria das *crash box* ótimas de Pareto possuem 4 lados, com alteração de aproximadamente 75%, em modelos retangulares ou quadrados. No entanto, há um amplo intervalo dos parâmetros Diâmetro do polígono e comprimento da *crash box*. Tal intervalo concede uma liberdade de escolha maior ao projetista, com um modelo final muito flexível. O projetista poderá analisar outros requisitos que não haviam sido considerados na otimização. Exemplos de aspectos não previamente considerados são dados no capítulo a seguir.

Figura 6-3. Gráfico de bolhas 4D, excluindo os pontos fora da curva. A fronteira de Pareto está demarcada em verde.



Fonte: Próprio Autor. Produzido no software modeFRONTIER

6.4. EXEMPLOS DE CRITÉRIOS PARA A DECISÃO FINAL

As variáveis de saída dos pontos ótimos de Pareto estão dispostas na Tabela 16. Todos os candidatos da fronteira de Pareto podem ser vistos na Figura 6-5. Os seguintes critérios podem ser discutidos para decidir qual é o modelo mais adequado da fronteira de Pareto,

- Maior SEA – modelo #438. Este modelo apresenta a menor uniformidade de carga dentro da fronteira de Pareto, 0.56. Se o projetista estiver disposto a sacrificar um pouco de SEA para obter maior uniformidade de carga, o próximo candidato com maior SEA é o modelo #567.
- Maior uniformidade de carga – O modelo escolhido seria então o #74, com LU = 0.64. Também é este o modelo que apresenta o menor SEA dentro de toda a fronteira de Pareto. Para ter um SEA maior, mas ainda priorizando a uniformidade de carga, existem três candidatos com LU = 0.61: #561 (candidato com SEA de 16841.88 J/kg), #171 (candidato com menor

deslocamento do impactor e SEA de 10226.57 J/kg) e #106 (candidato com SEA 7207.17 J/kg).

- Menor massa de *crash boxes* – um critério comum na indústria automotiva, o modelo da fronteira de Pareto com menor massa é o #438, com 0.542 kg.
- Menor deslocamento total do impactor – este critério pode ser utilizado a fim de reduzir danos no *frontend* do veículo. O modelo da fronteira de Pareto que apresenta menor deslocamento é o #171, conforme pode ser averiguado na Figura 6-4.

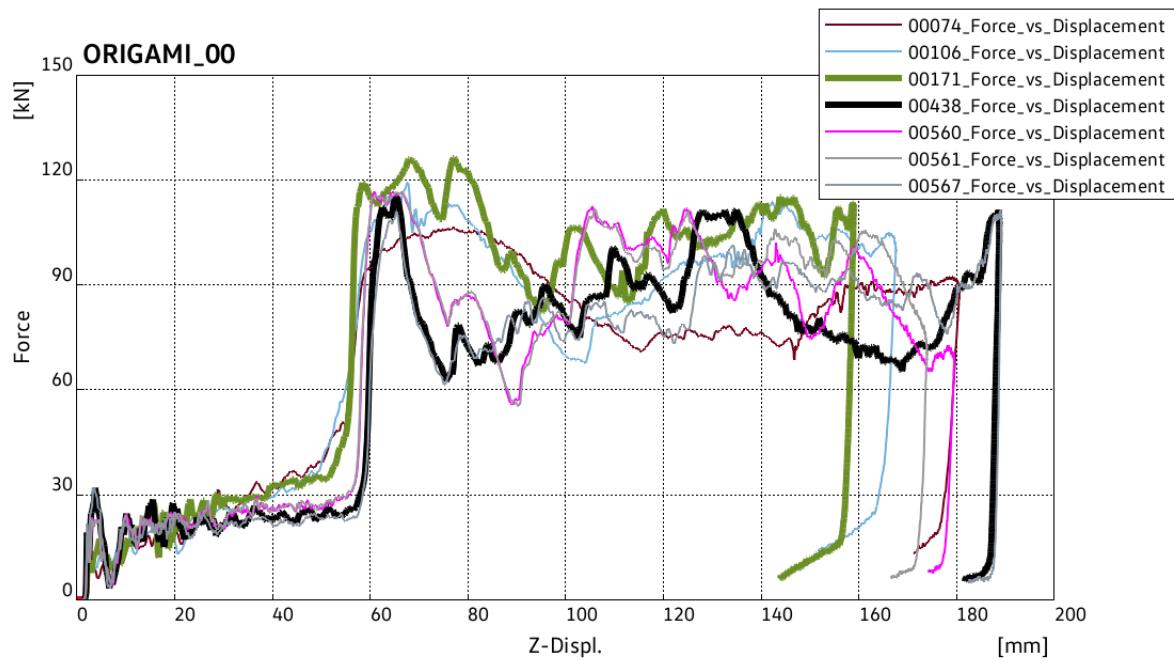
Dois destes critérios apontam para o mesmo modelo, #438. Como mencionado na Tabela 15, esta *crash box* tem espessura de 1.60 mm, 1 módulo, 4 lados (N), 40 mm de lado (b), 26.80 mm de vale (variável c), 159.60 mm de altura antes da dobragem (l), alteração do tipo 2 (quadrada – alteração nos dois eixos no topo da *crash box*), com um fator de 0.57. Depois da dobragem, o tamanho total da *crash box* foi 159.21 mm.

Outros dois critérios apontam para o modelo #171. O modelo #171 apresenta alta uniformidade de carga, e o menor deslocamento do impactor entre os modelos da fronteira de Pareto. Esta *crash box* tem espessura de 1.60 mm, 1 módulo, 4 lados, 60.80 mm de lado, 48.64 mm de vale, 163.5 mm de altura antes da dobragem, alteração do tipo 1 (retangular – alteração em apenas um eixo no topo da *crash box*) com um fator de 0.57. Depois da dobragem, o tamanho total da *crash box* foi de 162.25 mm.

Tabela 16. Variáveis de saída na fronteira de Pareto

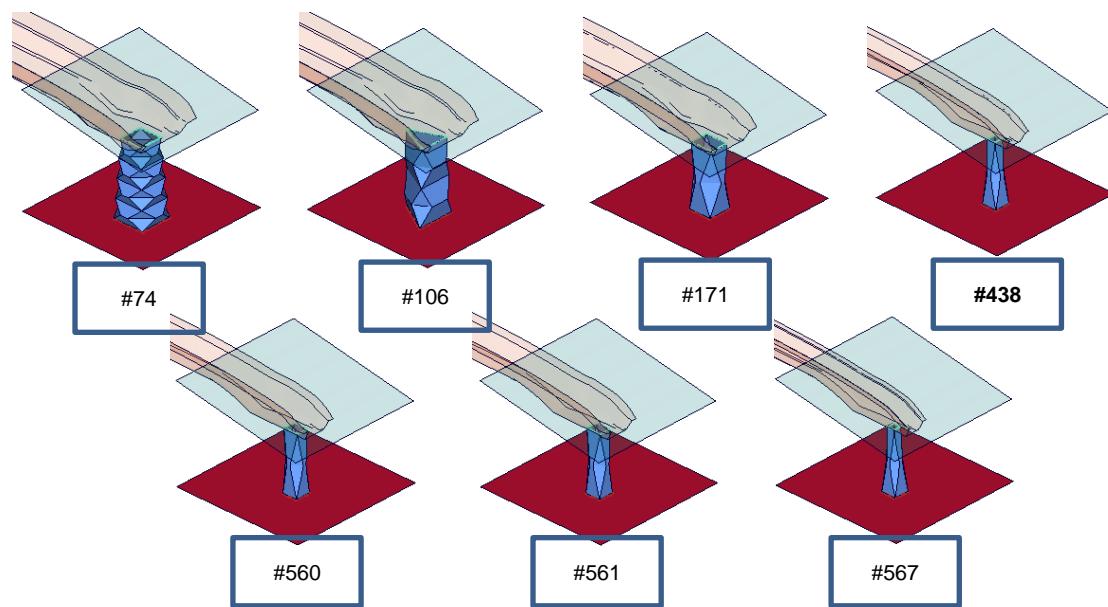
Id	Categoría	F_{max} (kN)	$F_{média}$ (kN)	LU	E_{def}^{cb} (J)	Massa <i>crash box</i> (kg)	SEA (J/kg)
74	REAL_I1	106.34	68.12	0.64	9362.52	1.713	5465.57
106	VEXP_I1	119.23	73.15	0.61	9837.79	1.365	7207.17
171	VOPT_I2	126.26	77.26	0.61	9650.81	0.944	10226.57
438	VOPT_I6	115.36	65.05	0.56	10416.90	0.542	19215.83
560	VOPT_I8	116.61	68.54	0.59	10192.80	0.603	16895.08
561	VOPT_I8	116.45	70.71	0.61	10182.60	0.605	16841.88
567	VOPT_I8	111.44	64.98	0.58	10504.30	0.562	18690.93

Figura 6-4. Força versus Deslocamento das Crash boxes Origami da fronteira de Pareto



Fonte: Próprio Autor.

Figura 6-5. Crash boxes Origami da fronteira de Pareto



Fonte: Próprio Autor.

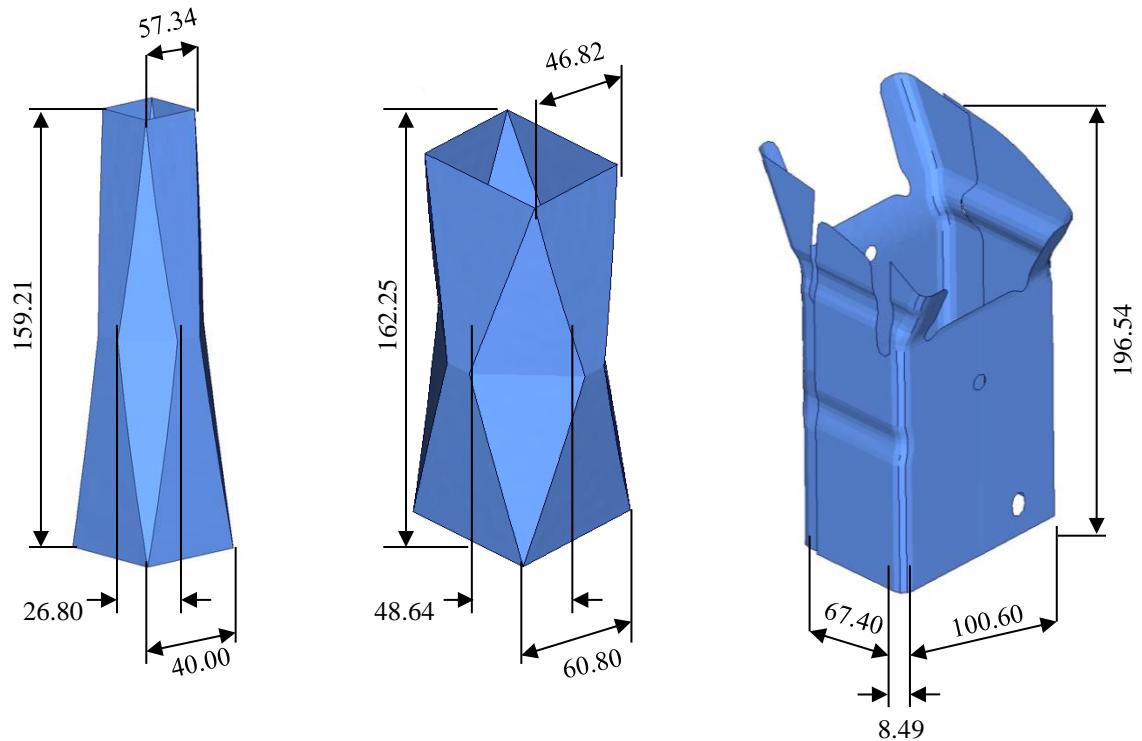
6.5. COMPARAÇÃO COM UMA CRASH BOX DA INDÚSTRIA

Uma *crash box* com geometria definida a partir de um veículo de passeio foi modelada e submetida às mesmas condições iniciais e modelo de material. As dimensões geométricas análogas estão listadas na Tabela 17. A Figura 6-7 mostra os modelos #438 e #171 antes e depois do impacto, e a Figura 6-8 mostra a *crash box* inspirada em um modelo comercial antes e depois do impacto.

Tabela 17. Parâmetros da *crash box* inspirada em modelo da indústria e dos modelos #438 e #171

Id	t (mm)	Módulos	N	b (mm)	c (mm)	l (mm)	fator	Modo de alteração	Diâmetro do polígono (mm)	Comprimento da <i>crash box</i> (mm)
Ind.	1.70	1	4	100.6	8.49	-	0.67	1	113.9	196.54
438	1.60	1	4	40.00	26.80	159.6	0.57	2	56.57	159.21
171	1.60	1	4	60.80	48.64	163.5	0.77	1	85.98	162.25

Figura 6-6. Crash boxes #438, #171 e modelo inspirado na indústria



Fonte: Próprio Autor

Tabela 18. Variáveis de saída

Id	Categoría	F_{max} (kN)	$F_{média}$ (kN)	LU	E_{def}^{cb} (J)	Massa crash box (kg)	SEA (J/kg)
-	INDUSTRY	159.1	92.75	0.58	10213.30	2.170	4707.674
438	VOPT_I6	115.36	65.05	0.56	10416.90	0.542	19215.83
171	VOPT_I2	126.26	77.26	0.61	9650.81	0.944	10226.57

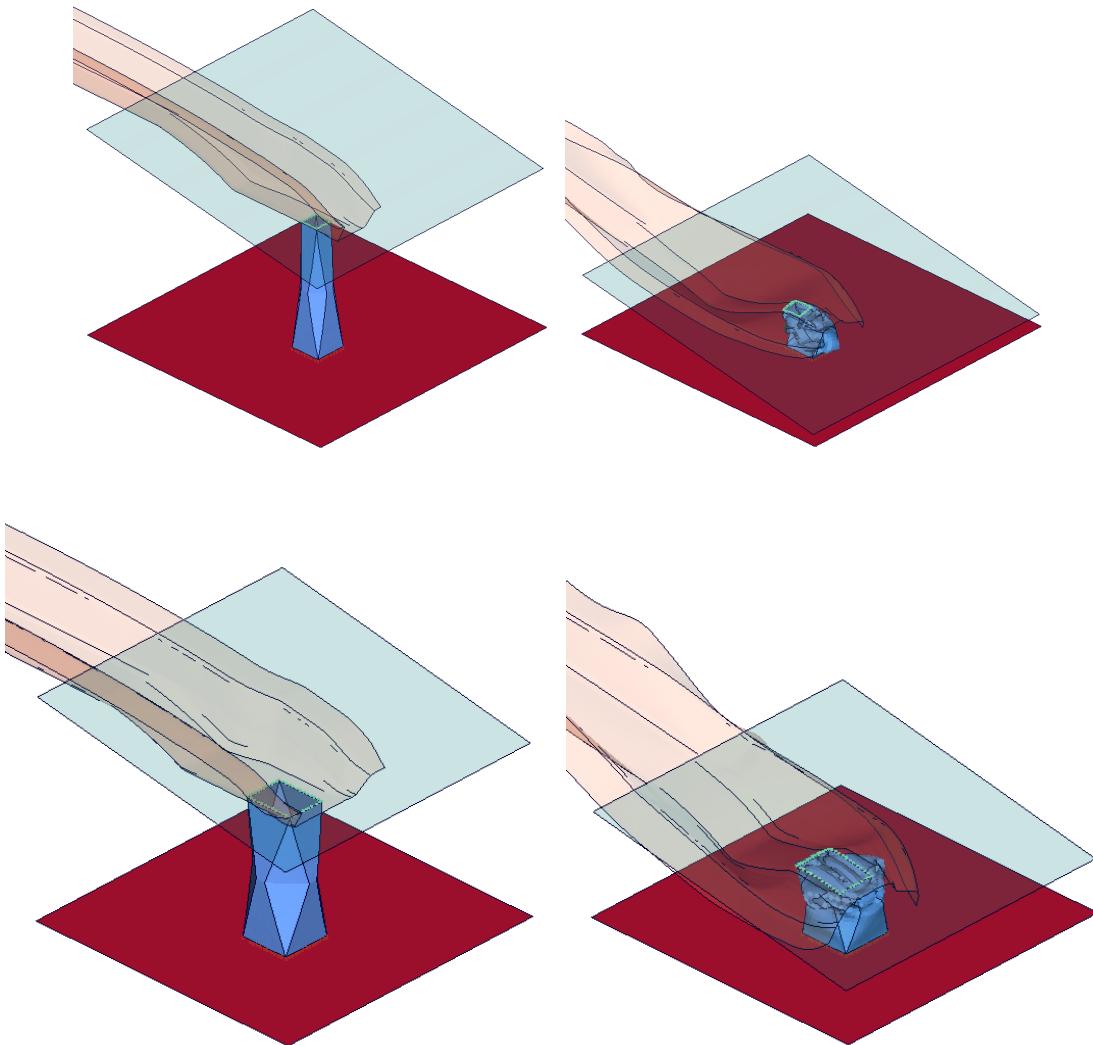
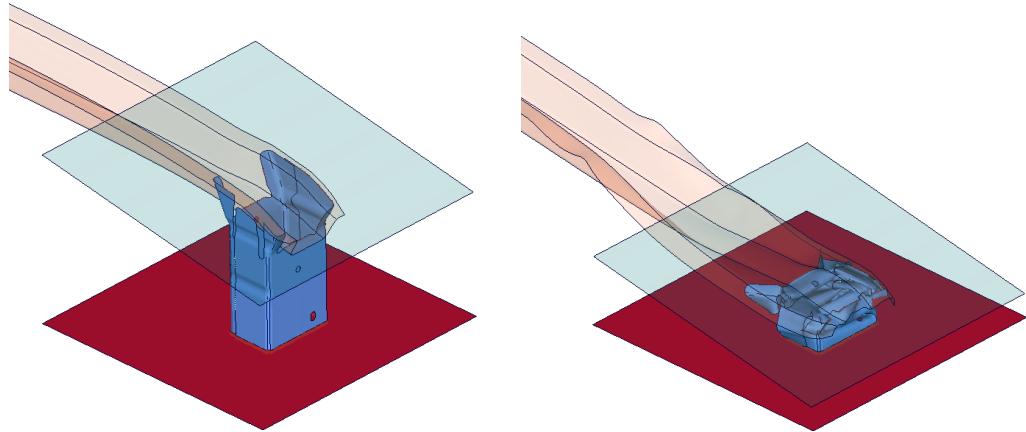
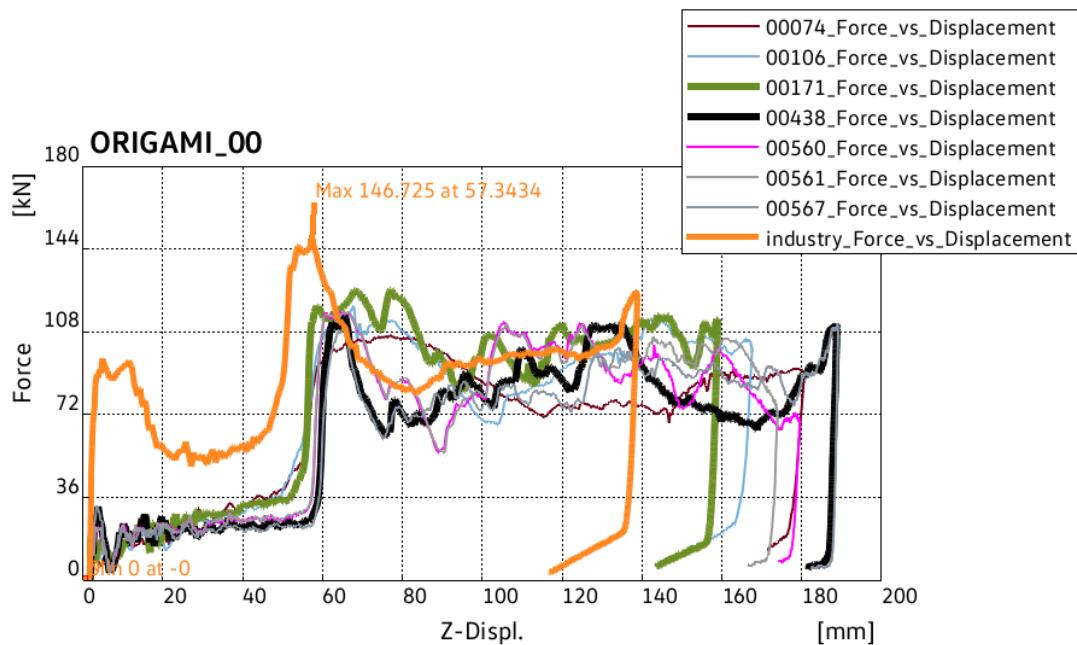
Figura 6-7. Design #438 (superior) e design #171 (inferior) – antes do impacto (esquerda) e no ponto de deformação máxima (direita)**Fonte: Próprio Autor**

Figura 6-8. Crash box inspirada em modelo industrial – antes do impacto (esquerda) e na deformação máxima (direita)



Fonte: Próprio Autor

Figura 6-9. Comparação das curvas de Força por Deslocamento dos modelos de Pareto e do crash box inspirado em modelo industrial.



Fonte: Próprio Autor

De acordo com os parâmetros de desempenho na Tabela 18, a solução ótima #438 absorve tanta energia quanto a *crash box* da indústria, com uniformidade de carga similar e aproximadamente 4 vezes menos massa.

Já a solução ótima #171 absorve um pouco menos de energia, porém com uniformidade de carga maior e 2 vezes menos massa. Este modelo apresenta um deslocamento de barreira de 158.9 mm, em comparação aos 138.8 mm de deformação da *crash box* da indústria.

A Figura 6-9 mostra as curvas da força de contato do impactor contra o deslocamento do mesmo impactor para todos os designs de Pareto e da indústria. É fácil notar que o pico da força de contato do impactor na *crash box* da indústria (curva laranja) é maior que os picos da força de contato do impactor no modelos origami. Aliado a um curso total de esmagamento maior, os modelos #438 e #171 apresentam uma força média menor que a *crash box* da indústria.

Podemos ver que a real força de resistência do Origami #438 se inicia por volta de 60 mm após o início do esmagamento, e o modelo #171 oferece resistência por volta de 56 mm após o início do esmagamento. Isso acontece devido à diferença de designs: a *crash box* da indústria foi feita de modo a apoiar todo o corpo da lâmina do para-choques, na região da *crash box* (ver Figura 6-6). A *crash box* origami apoia apenas a base da lâmina. Portanto, durante o esmagamento da lâmina, a *crash box* origami ainda não está trabalhando.

Considerando o momento onde pode-se atribuir a força de resistência primária às *crash box* origami (60 mm para #438 e 56mm para #171), temos por volta de 128 mm de curso para o modelo #438, e 102 mm para o modelo #171, enquanto a *crash box* da indústria tem por volta de 129 mm.

O aproveitamento da *crash box* origami #438 em relação ao seu esmagamento é de 128 mm em 159 mm de comprimento inicial; o aproveitamento da *crash box* origami #171 em relação ao seu esmagamento é de 102 mm em 162 mm do comprimento inicial, e a *crash box* da indústria é de 129 mm em 196 mm.

7. CONCLUSÃO E TRABALHOS FUTUROS

Está claro neste trabalho que a otimização das *crash box* origami possui baixa convergência, como pode ser confirmado pela Figura 6-3 e pela Tabela 16. A fronteira de Pareto é formada por modelos em diferentes estágios do processo de otimização, assim como diferentes iterações, como pode ser visto na coluna “categoria”. O software modeFRONTIER nomeia as categorias como “Algoritmo”+“Estágio”+“Iteração”. A parte do algoritmo foi suprimida, já que todos os modelos da fronteira de Pareto foram gerados pelo algoritmo FAST-MOGA2. O “estágio” pode ser tanto real (validado com uma simulação em PAMCrash) quanto virtual (gerado por metamodelos), e nos modelos virtuais ele pode ser criado durante o passo de exploração ou durante a otimização do metamodelo. Observa-se aqui que modelos das iterações 1 a 8 compuseram a fronteira de Pareto.

No entanto, mesmo com uma convergência pobre, o projetista pode ter uma ideia do problema, e coletar informações suficientes para decidir qual modelo é a melhor solução, baseado em seu critério de decisão. A *crash box* #438, escolhida caso o projetista priorize o critério de menor massa, possui quatro vezes menos massa que uma *crash box* da indústria – portanto apresentando um SEA quatro vezes maior que a mesma *crash box*.

Caso o critério do projetista seja baixo deslocamento do impactor e otimização da uniformidade de carga, o modelo #171 levará vantagem sobre o #438. Ele apresenta uma redução de massa em relação ao *crash box* inspirado na indústria, e 26 mm a menos de intrusão do impactor.

Se outro modelo fosse escolhido por diferentes critérios de decisão, como por exemplo o modelo #106, de base triangular, soluções para a manufatura seriam uma preocupação principal. (MA, 2011) sugeriu um método de estampagem por pressão de água para a construção das *crash box*.

Este trabalho conclui portanto que a metodologia apresentada proporciona a análise e otimização de muitas combinações diferentes entre as possíveis para uma peça – no caso deste trabalho, a *crash box* origami. O método DOE + Metamodelos não se restringe apenas à peças automotivas, podendo ser expandido para quaisquer experimentos cujas variáveis possam ser mensuradas.

8. REFERÊNCIAS

- ALGHAMDI, A. A. A. **Collapsible impact energy absorbers: an overview**. *Thin-Walled Structures*, v. 39, n. 2, p. 189–213, 2001. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0263823100000483>>.
- ALVES, M. **Material constitutive law for large strains and strain rates**. *Journal of Engineering Mechanics*, v. 126, n. 2, p. 215–218, 2000.
- ALVES, M. **Impact engineering: fundamentals, experiments and non-linear finite elements**. [s.l: s.n.]
- ARORA, J. **Introduction to Optimum Design**. [s.l.] Elsevier Science, 2016.
- BETACAE. **ANSA User's Guide v.17.x.x**. Thessaloniki, Greece: BetaCAE Systems SA, 2016.
- BOIS, P. Du et al. **Vehicle Crashworthiness and Occupant Protection**. 2004.
- BRITT, K. W. **Papermaking**. Londres: Encyclopædia Britannica Inc., 2012. Disponível em: <<https://www.britannica.com/topic/papermaking>>.
- CALLE, M. A. G.; OSHIRO, R. E.; ALVES, M. **Ship collision and grounding: scaled experiments and numerical analysis**. *International journal of impact engineering*, v. 103, p. 195–210, 2017.
- CAVAZZUTI, M. **Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics**. [s.l.] Springer-Verlag Berlin Heidelberg, 2013.
- DENATRAN. **Site do Denatran - Departamento Nacional de Trânsito**. 31 de Maio. Disponível em: <<http://www.denatran.gov.br>>.
- GLOBAL NCAP. **Global NCAP**. 30.Jun.2017. Disponível em: <<http://www.globalncap.org>>.
- HAGAN, M. T.; MENHAJ, M. B. **Training feedforward networks with the Marquardt algorithm**. *IEEE Transactions on Neural Networks*, v. 5, n. 6, p. 989–993, 1994.

HATORI, K. **History of Origami in the East and the West before Interfusion.** In: WANG-IVERSON, P.; LANG, R. J.; YIM, M. (Ed.). Origami^5. [s.l.] CRC Press - Taylor and Francis Group, 2011. 5p. 3–11.

HSU, S. S.; JONES, N. **Dynamic axial crushing of aluminium alloy 6063 - T6 circular tubes.** Latin American Journal of Solids and Structures, v. 1, n. 3, p. 277–296, 2004.

IRIE, B.; MIYAKE, S. **Capabilities of Three-layered Percept rons.** In: IEEE 1988 International Conference on Neural Networks, San Diego, CA, USA. IEEE 1988 International Conference on Neural Networks, San Diego, CA, USA: IEEE, 1988.

JOHNSON, W.; REID, S. R. **Metallic energy dissipation systems.** Applied Mechanics Reviews, v. 31, n. 277, 1978.

JONES, N.; WIERZBICKI, T. **Structural Crashworthiness.** Londres: Butterworths, 1983.

KARAGIOZOVA, D.; ALVES, M. **Dynamic Elastic-Plastic Buckling of Structural Elements: A Review.** Applied Mechanics Reviews, v. 61, n. 4, p. 40803–40826, 8 jul. 2008. Disponível em: <<http://dx.doi.org/10.1115/1.2939481>>.

KLEIJNEN, J. P. C. **Design and Analysis of Simulation Experiments.** Segunda Ed ed. Suíça: Springer International Publishing, 2015. v. 230

KOKKULA, S. et al. **Behaviour of an automotive bumper beam-longitudinal system at 40% offset impact: An experimental and numerical study.** Latin American Journal of Solids and Structures, v. 3, n. 1, p. 59–73, 2006.

LEE, S. J. et al. **Design flow for the crash box in a vehicle to maximize energy absorption.** Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, v. 227, n. 2, p. 179–200, 2013.

LIMA, A. de. **Desenvolvimento de Um Veículo Urbano Seguro Utilizando Otimização Baseada em Metamodelos.** 2016. 376 f. Tese (Doutorado em Engenharia Mecânica) - Escola Politécnica da USP, Universidade de São Paulo, São Paulo, 2016.

LU, G.; YU, T. X. **Energy Absorption of Structures and Materials.** [s.l.] Elsevier, 2003.

MA, J. **Thin-walled Tubes with Pre-folded Origami Patterns as Energy Absorption Devices.** 2011. Doctor of, 194 p. f. , Doctor of University of Oxford, 2011.

MERALI, Z. “**Origami Engineer” Flexes to Create Stronger, More Agile Materials.** Science, v. 332, p. 1376–1377, 2011.

MOHANTY, S. P. **Nanoelectronic Mixed-Signal System Design.** [s.l.] McGraw-Hill Education, 2015.

MONTRONE, T.; TURCO, A.; RIGONI, E. **Technical Report 2014-001 FAST Optimizers : General Description.** 2014.

OBITKO, M. **Introdução aos Algoritmos Genéticos.** Disponível em: <<http://www.obitko.com/tutorials/genetic-algorithms/portuguese/operators.php>>. Acesso em: 20 jan. 2019.

PARK, H.-S.; DANG, X.-P. **Structural optimization based on CAD–CAE integration and metamodeling techniques.** Computer-Aided Design, v. 42, n. 10, p. 889–902, 2010. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0010448510001107>>.

POLES, S. **Technical Report 2003-006 An improved Multi-Objective Genetic Algorithm.** 2003.

RIGONI, E. **Technical Report 2007-001 Radial Basis Functions Response Surfaces.** p. 18, 2007.

SAVINE, A. **Modern computational Finance. AAD and Parallel Simulations with professional implementation in C++.** Hoboken, Nova Jersey: John Wiley & Sons, Inc., 2018.

SILVA, J. E. C. S.; DRIEMEIER, L.; ALVES, M. **STUDY OF AN ORIGAMI CRASHBOX THROUGH METAMODELS.** In: MECSOL 2019 – 7th International Symposium on Solid Mechanics, São Carlos. MECSOL 2019 – 7th International Symposium on Solid Mechanics, São Carlos: 2019.

VOCE, E. **A practical strain hardening function.** Metallurgia, v. 51, p. 219–226, 1955.

WITTEMAN, W. J. **Improved Vehicle Crashworthiness Design by Control of the Energy Absorption for Different Collision Situations.** [s.l: s.n.]

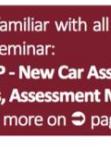
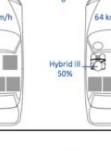
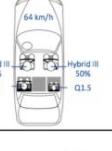
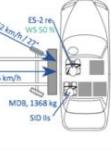
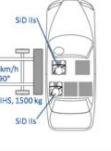
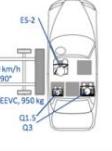
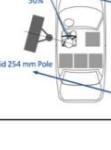
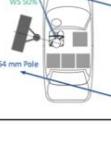
ANEXO A

SAFETY WISSEN  **UPDATE**

carhs.

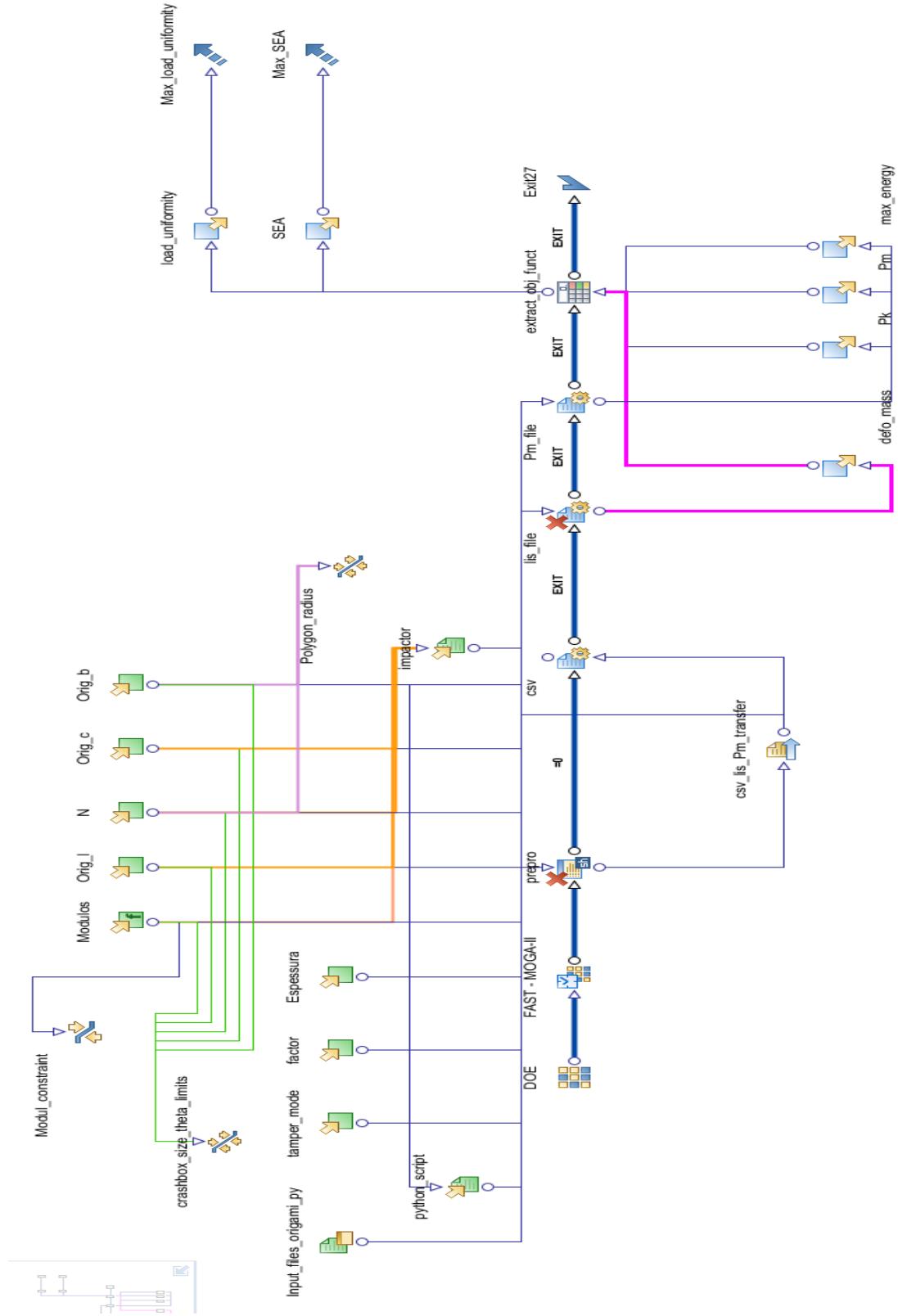
NCAP-Tests in Europe and America

2017 2018

	Euro NCAP	U.S. NCAP	IIHS	Latin NCAP
Full Width				<p>Get familiar with all NCAP tests in just 2 days with our seminar: NCAP - New Car Assessment Programs: Tests, Assessment Methods, Ratings learn more on  page 118</p>
ODB / SOB				
MDB	 ■ Far Side Occupant Protection			
Pole				 (prerequisite for 5 star rating)
Rollover		■ SSF	■ Roof Crush	
Pedestrian	<ul style="list-style-type: none"> ■ Flex PLI ■ Upper Legform ■ Headforms ■ AEB VRU Pedestrian ■ AEB VRU Cyclist 	<ul style="list-style-type: none"> ■ Flex PLI ■ Upper Legform ■ Headforms ■ AEB Pedestrian ■ Rear Automatic Braking 		■ Award
Child Safety	<ul style="list-style-type: none"> ■ Frontal ODB ■ Side MDB ■ CRS- Installation ■ Vehicle based assessment 		<ul style="list-style-type: none"> ■ LATCH (Lower Anchors and Tethers for Children) 	<ul style="list-style-type: none"> ■ Frontal ODB ■ Side MDB ■ CRS- Installation ■ Vehicle based assessment
Whiplash	<ul style="list-style-type: none"> ■ static front / rear ■ dynamic (3 pulses) ■ AEB City 		<ul style="list-style-type: none"> ■ static ■ dynamic (1 pulse) 	SafetyWissen by carhs.
Other	<ul style="list-style-type: none"> ■ Assistance systems: SBR, SAS, AEB, LDW, LKA ... 	<ul style="list-style-type: none"> ■ FCW, LDW, Rear View Cameras, AEB, DBS, BSD 	<ul style="list-style-type: none"> ■ AEB, FCW ■ Headlights 	<ul style="list-style-type: none"> ■ SBR, ABS (prerequisite for ≥ 3 star rating) ■ ESC (prereq. f. ≥ 4 star)

ANEXO B

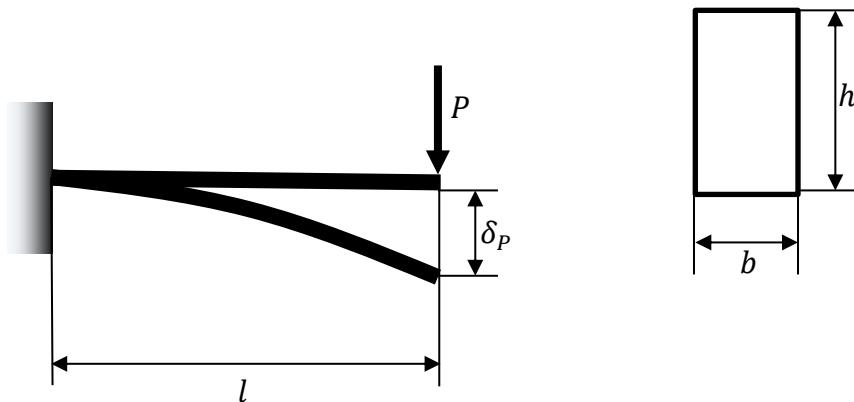
Workflow no modeFRONTIER



ANEXO C

Exemplo de Escolha de DOE

Neste exemplo, temos uma viga engastada de lados b e h, comprimento l, com uma carga concentrada P perpendicular à extremidade solta:



Usando o modeFRONTIER, faremos alguns experimentos de DOEs para as variáveis de entrada l , b e h . A otimização será minimizar o deslocamento máximo da viga, com a menor massa possível.

P	E	ρ	b	h	I
1 kN	210 GPa	7.85e-6 kg/mm ³	10 ≤ b ≤ 100; 5 níveis	10 ≤ h ≤ 100 5 níveis	50 ≤ l ≤ 600; 5 níveis

O cálculo do deslocamento da viga se dá pela seguinte equação:

$$\delta_b = \frac{Pl^3}{3EI}, \text{ com } I = \frac{bh^3}{12}$$

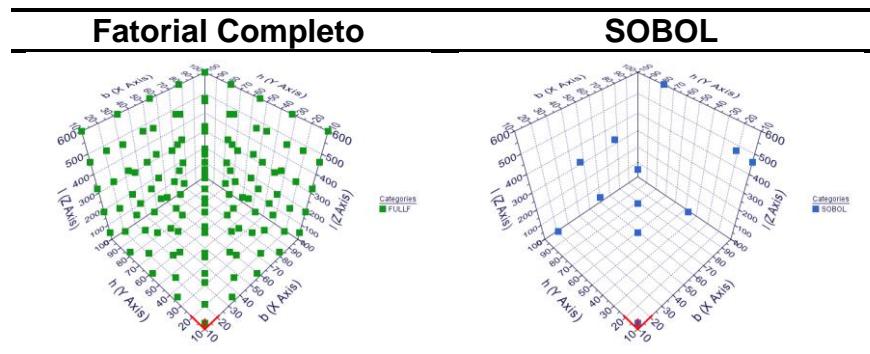
Com 5 níveis, em três variáveis de entrada, temos um total de 125 combinações distintas – Fatorial completo.

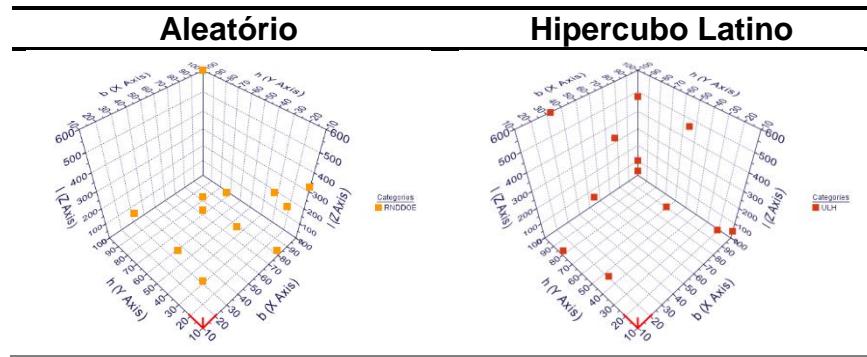
Segundo (ESTECO, 2016), o número indicado para um DOE é de duas vezes o número de variáveis vezes o número de objetivos, isto é, $2^*3^*2 = 12$.

Abaixo, temos as escolhas de 12 designs, através de 3 técnicas diferentes: Sobol, Aleatório (RND) e Hipercubo Latino (ULH):

ID	Categoria	b	h	I
00000	SOBOL	10.00	10.00	50.00
00001	SOBOL	32.50	77.50	462.50

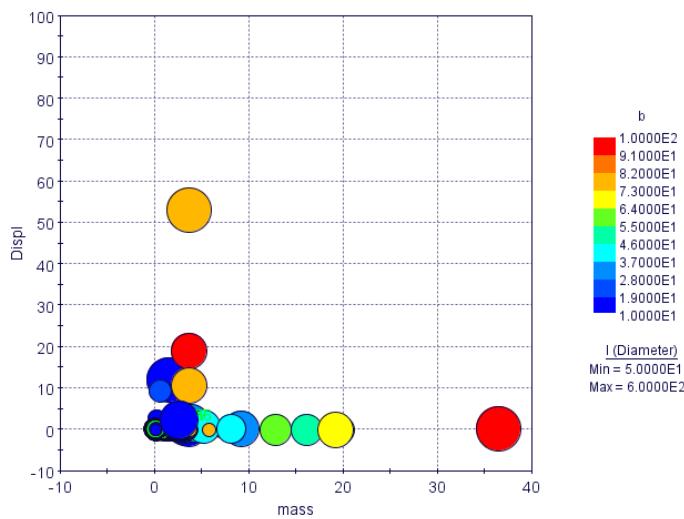
ID	Categoría	b	h	I
00002	SOBOL	77.50	32.50	187.50
00003	SOBOL	100.00	100.00	50.00
00004	SOBOL	32.50	32.50	462.50
00005	SOBOL	10.00	77.50	187.50
00006	SOBOL	77.50	10.00	600.00
00007	SOBOL	77.50	100.00	325.00
00008	SOBOL	10.00	32.50	600.00
00009	SOBOL	55.00	55.00	50.00
00010	SOBOL	100.00	10.00	462.50
00011	SOBOL	100.00	77.50	600.00
00012	RNDDOE	77.50	55.00	187.50
00013	RNDDOE	32.50	100.00	50.00
00014	RNDDOE	100.00	100.00	600.00
00015	RNDDOE	100.00	32.50	187.50
00016	RNDDOE	32.50	55.00	50.00
00017	RNDDOE	77.50	77.50	50.00
00018	RNDDOE	32.50	10.00	462.50
00019	RNDDOE	100.00	10.00	325.00
00020	RNDDOE	77.50	10.00	325.00
00021	RNDDOE	55.00	55.00	187.50
00022	RNDDOE	77.50	10.00	50.00
00023	RNDDOE	10.00	10.00	325.00
00024	ULH	100.00	10.00	50.00
00025	ULH	55.00	32.50	325.00
00026	ULH	32.50	32.50	600.00
00027	ULH	77.50	77.50	600.00
00028	ULH	55.00	100.00	50.00
00029	ULH	77.50	10.00	187.50
00030	ULH	10.00	32.50	187.50
00031	ULH	32.50	100.00	600.00
00032	ULH	100.00	55.00	462.50
00033	ULH	77.50	100.00	325.00
00034	ULH	10.00	77.50	50.00
00035	ULH	55.00	55.00	462.50



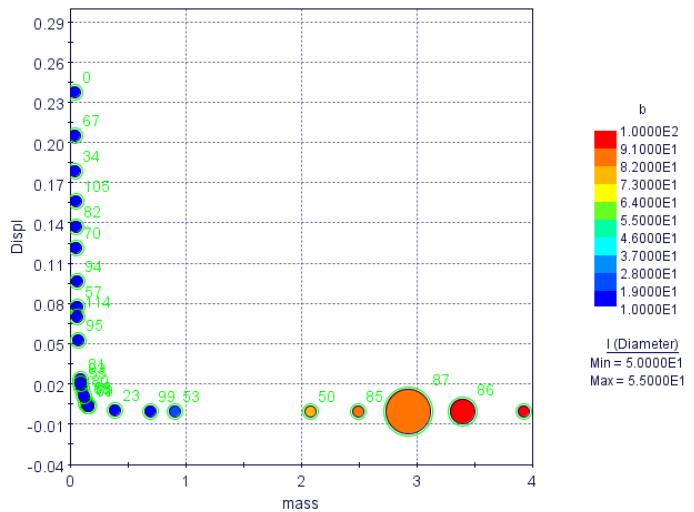


Podemos averiguar pelas figuras acima que o Sobol e o Hipercubo Latino evitam a aglomeração que pode acontecer no DOE aleatório, e apresentam uma melhor exploração do espaço de projeto. Também averiguamos que as técnicas podem ser utilizadas em conjunto e que podem se complementar.

Utilizando o Sobol como população inicial e o algoritmo FAST para minimizar o deslocamento máximo da viga e a massa, e aumentando o número de valores permitidos para b, h e l (de 5 níveis para todos os valores entre os limites, com passo de 0.5 mm), temos o seguinte resultado:



Neste gráfico de bolhas, podemos ver uma clara fronteira de Pareto formando-se ao redor do ponto ótimo. Abaixo, temos apenas os modelos selecionados pela fronteira de Pareto:



Os modelos que possuem a menor massa e o menor deslocamento são os quatro no “vértice” da fronteira:

ID	Categoria	b (mm)	h (mm)	I (mm)	Massa (kg)	δ_b (mm)
00066	FAST	10.00	33.50	50.00	0.131	0.0063
00068	FAST	10.00	34.50	50.00	0.135	0.0058
00069	FAST	10.00	39.00	50.00	0.153	0.0040
00071	FAST	10.00	38.50	50.00	0.151	0.0042

ANEXO D

ALGORITMO EM PYTHON PARA O PRÉ-PROCESSAMENTO

```

1. # This function creates a N-sided-
  polygonal crashbox based on nodes coordinates.
2. # One can set up the number of sides, the quantity of moduls and several
3. # dimensional parameters. Then the model is remeshed.
4. # The function exports an include.
5. #
6. # WORKS ON ANSA 18.X
7. #
8. # History:
9. # 2018-07-06: Added deck variable
10. # 2018-04-25: 2 Defos Symmetry on XZ plane
11. # 2018-04-23: CMS_Morph implemented
12. # 2018-04-10: Implemented CMS crossbeam treatment
13. # 2018-03-15: Implemented reconstruct based on quality elements check
14. # 2018-03-14: Implemented moduls, remesh
15. # 2018-02-12: Created
16. # PYTHON script
17. import os
18. importansa
19. import sys
20. import re
21. import math
22. import csv
23. from math import *
24. fromansa import *
25.
26. # Defining the Deck
27. deck = constants.PAMCRASH
28.
29.
30. def crashbox_polygon(N,Modul,b,c,l,thickness,tamper_mode=0,factor=0.8):
31.
32.     """This function imports a pamcrash include for an Origami Crashbox modu
  le,
33.     replicates it and remeshes it.
34.     N: Number of sides
35.     Modul: number of moduls
36.     b: Side length
37.     c: Crest length (should be smaller than b)
38.     l: Modul height before folding
39.     tamper_mode:    1 for tampering only in X-direction;
40.                  2 for tampering in X and Y directions;
41.                  3 for tampering both base and top.
42.     factor: Percentage of b, used to tamper
43.     """
44.
45.     # New Ansa file
46.     session.New("discard")
47.     ansacritdir = "/user2/jecss/biblioteca/ansa/"
48.
49.     mesh.ReadQualityCriteria(ansacritdir + "origami_05mm_b.ansa_qual")
50.     mesh.ReadMeshParams(ansacritdir + "origami_5mm.ansa_mpar")
51.     base.SetEntityVisibilityValues(deck, {"all":"on"})
52.
53.
54.     # number of polygon sides (square tube has four sides)
55.     Orig_N = int(N)
56.     # number of moduls
57.     modulos = int(Modul)
58.     # Side length

```

```

59.     Orig_b = b
60.     # Crest Length
61.     Orig_c = c*b
62.     # module height before folding
63.     Orig_l = l
64.     # Thickness
65.     t = thickness
66.     #Several plane angles
67.     gamma = 360/Orig_N
68.     gamma_rad=2*pi/Orig_N
69.     beta = 2*atan(
70.         sin(gamma_rad/2) /
71.         (Orig_c / (Orig_b - Orig_c) + cos(gamma_rad / 2)))
72.     )
73.     alfa = gamma_rad-beta
74.
75.     #Crest Angle
76.     theta=acos(Orig_c / Orig_l * tan(pi / 2 / Orig_N))
77.
78.     #Row Height
79.     Orig_h=Orig_l/2*sin(theta)
80.
81.     #variables start
82.     Nodes_1 = []
83.     Nodes_2 = []
84.     Nodes_3 = []
85.     include = base.CreateEntity(
86.         deck,
87.         'INCLUDE',
88.         {'Name': 'origami_v0.3_20180314_recons.inc'})
89.     base.SetCurrentEntity(include)
90.
91.
92.     #Origami Start
93.     first_point = [
94.         -Orig_b/2,
95.         -Orig_b/(2*tan(gamma_rad/2)),
96.         0]
97.     N = 1001
98.     first_node = {
99.         'N':N,
100.        'X':first_point[0],
101.        'Y':first_point[1],
102.        'Z':first_point[2]
103.    }
104.    Node = base.CreateEntity(deck, 'NODE',first_node)
105.    Nodes_1.append(Node)
106.    vals_property = {"IDPRT": 8000100,
107.        "Name": "Defo Element",
108.        "IMAT": 8103600,
109.        "TCONT": t,
110.        "h": t,
111.        "NINT": 5}
112.    property = base.CreateEntity(deck, "PART_SHELL", vals_property)
113.    # Loop of first row nodes
114.    for point in range(1,Orig_N,1):
115.        next_point = {
116.            "X": first_point[0],
117.            "Y": first_point[1],
118.            "Z": first_point[2],
119.            "N": N+1
120.        }
121.        Node = base.CreateEntity(deck, "NODE",next_point)
122.        base.GeoRotate()

```

```

123.         "MOVE",
124.         0,
125.         "SAME PART",
126.         "EXPAND",
127.         0, 0, 0, 0, 0, 1,
128.         point*gamma,
129.         Node)
130.     Nodes_1.append(Node)
131.     N=N+1
132.
133.
134.     # Loop of second row nodes
135.     N = 2001
136.     i = 0
137.     second_point = [
138.         -(Orig_b-Orig_c)/2,
139.         -(Orig_b-Orig_c)/(2*tan(beta/2)),
140.         Orig_h]
141.     for point in range(1,2*Orig_N+1,1):
142.         next_point = {
143.             "X": second_point[0],
144.             "Y": second_point[1],
145.             "Z": second_point[2],
146.             "N": N
147.         }
148.         Node = base.CreateEntity(deck,"NODE",next_point)
149.
150.         if int(i) % 2 == 0:
151.             rotate=degrees(alfa)*i/2+degrees(beta)*i/2
152.         else:
153.             rotate=degrees(alfa)*(i-1)/2+degrees(beta)*(i+1)/2
154.
155.         base.GeoRotate(
156.             "MOVE",
157.             0,
158.             "SAME PART",
159.             "EXPAND",
160.             0, 0, 0, 0, 0, 1,
161.             rotate,
162.             Node)
163.         Nodes_2.append(Node)
164.         N=N+1
165.         i = i+1
166.
167.
168.     # Loop of third row nodes
169.     coord_Nodes_3 = []
170.     for i in range(0,Orig_N,1):
171.         coord_Nodes_3=[
172.             base.GetEntityCardValues(deck, Nodes_1[i], "X"),
173.             base.GetEntityCardValues(deck, Nodes_1[i], "Y"),
174.             base.GetEntityCardValues(deck, Nodes_1[i], "Z"),
175.             base.GetEntityCardValues(deck, Nodes_1[i], "N")
176.         ]
177.         next_point = {
178.             "X":coord_Nodes_3[0].pop('X'),
179.             "Y": coord_Nodes_3[1].pop('Y'),
180.             "Z": coord_Nodes_3[2].pop('Z')+2*Orig_h,
181.             "N": coord_Nodes_3[3].pop('N')+2000
182.         }
183.         Node = base.CreateEntity(deck,"NODE",next_point)
184.         Nodes_3.append(Node)
185.     #
186.
187.
188.     # trias first row. Element orientation is outwards

```

```

189.     M = 1001
190.     shell = {
191.         "M": M,
192.         "N1": Nodes_1[0],
193.         "N2": Nodes_2[0],
194.         "N3": Nodes_2[2*Orig_N-1],
195.         "type": "TRIA",
196.         "IPART": 8000100}
197.     tria = base.CreateEntity(deck, "SHELL", shell)
198.     for i in range(1, Orig_N, 1):
199.         M = M+2
200.         shell = {
201.             "M": M,
202.             "N1": Nodes_1[i],
203.             "N2": Nodes_2[2*i],
204.             "N3": Nodes_2[2*i-1],
205.             "type": "TRIA",
206.             "IPART": 8000100}
207.         tria = base.CreateEntity(deck, "SHELL", shell)
208.
209.
210.     # quads first row. Element orientation is outwards
211.     M = 1000
212.     for i in range(0, Orig_N, 1):
213.         M = M+2
214.         if i >= len(Nodes_1)-1:
215.             shell = {"M": M,
216.                       "N1": Nodes_1[i],
217.                       "N2": Nodes_1[0],
218.                       "N3": Nodes_2[2*i+1],
219.                       "N4": Nodes_2[2*i],
220.                       "type": "QUAD",
221.                       "IPART": 8000100}
222.         else:
223.             shell = {"M": M, "N1": Nodes_1[i], "N2": Nodes_1[i+1], "N3": Nodes_2[2*i+1], "N4": Nodes_2[2*i], "type": "QUAD", "IPART": 8000100}
224.             quad = base.CreateEntity(deck, "SHELL", shell)
225.
226.
227.     # trias second row. Element orientation is outwards
228.     M = 2001
229.     shell = {"M": M, "N1": Nodes_2[2*Orig_N-1], "N2": Nodes_2[0], "N3": Nodes_3[0], "type": "TRIA", "IPART": 8000100}
230.     tria = base.CreateEntity(deck, "SHELL", shell)
231.     for i in range(1, Orig_N, 1):
232.         M = M+2
233.         shell = {"M": M, "N1": Nodes_2[2*i-1], "N2": Nodes_2[2*i], "N3": Nodes_3[i], "type": "TRIA", "IPART": 8000100}
234.         tria = base.CreateEntity(deck, "SHELL", shell)
235.
236.     # quads second row. Element orientation is outwards
237.     M = 2000
238.     for i in range(0, Orig_N, 1):
239.         M = M+2
240.         if i >= len(Nodes_1)-1:
241.             shell = {"M": M, "N2": Nodes_3[i], "N1": Nodes_3[0], "N4": Nodes_2[2*i+1], "N3": Nodes_2[2*i], "type": "QUAD", "IPART": 8000100}
242.         else:
243.             shell = {"M": M, "N2": Nodes_3[i], "N1": Nodes_3[i+1], "N4": Nodes_2[2*i+1], "N3": Nodes_2[2*i], "type": "QUAD", "IPART": 8000100}
244.             quad = base.CreateEntity(deck, "SHELL", shell)
245.
246.     containers = list()
247.     containers.append(ansa.base.GetEntity(deck, "PART_SHELL", 8000100))

```

```

248.
249.     # Create new mesh. Old one is deleted
250.     shells_moduls = base.CollectEntities(deck, containers, "SHELL", recursive
251.     =True)
252.     mesh.FEMToSurf(shells_moduls, True, False)
253.     base.SetCurvesResolution(1, '')
254.     mesh.ApplyNewLengthToMacros("1")
255.     mesh.ApplyNewLengthToMacros("5")
256.     mesh.CreateBestMesh()
257.     utils.DeckInfo('deck_report.csv', 'WHOLE DB', 'TEXT')
258.
259.     filename = "deck_report.csv"
260.     pattern = re.compile("Total Shell Elements OFF")
261.     # in_file = open(filename,"rt")
262.     # in_file.close()
263.     fim = False
264.     contador = 0
265.     while fim == False:
266.         OFF = [0,0]
267.         with open(filename, "rt") as in_file:
268.             for line in in_file:
269.                 if pattern.search(line) != None:
270.                     OFF = line.split(", ")
271.                     break
272.                 if contador <= 4:
273.                     contador = contador + 1
274.                     if int(OFF[1]) > 0:
275.                         mesh.Reconstruct()
276.                         utils.DeckInfo('deck_report.csv', 'WHOLE DB', 'TEXT')
277.                     else:
278.                         fim = True
279.                         break
280.                 else:
281.                     fim = True
282.                     break
283.         faces_array = base.CollectEntities(constants.NASTRAN, None, "FACE", False
284.
285.
286.     # Create Moduls
287.     for i in range(1,modulos):
288.         new_faces = ansa.base.GeoTranslate("COPY",
289.             0,           #Offset will take place
290.             "SAME PART",          #No Group Options
291.             "NONE",            #Option for Sets is set to NONE
292.             0,
293.             0,
294.             i*2*Orig_h,
295.             faces_array,
296.             keep_connectivity=True, #Node Connectivity will be kept
297.             draw_results=True)
298.
299.     base.Topo()
300.
301.     ansa.mesh.AutoPaste(visible = True,
302.             project_on_geometry = False,
303.             project_2nd_order_nodes = False,
304.             move_to="average pos",
305.             preserve_id ="max",
306.             distance=0.2)
307.
308.
309.
310.     mesh.FixQuality()
311.

```

```

312.     # renumber nodes and elements
313.     general_element_rule = base.CreateNumberingRule(
314.         deck,
315.         "TOOL",
316.         0,
317.         "SHELL",
318.         "PER_GROUP",
319.         18000000,
320.         18999999,
321.         "new_elements_rule"
322.     )
323.     general_node_rule = base.CreateNumberingRule(
324.         deck,
325.         "TOOL",
326.         0,
327.         "NODE",
328.         "PER_GROUP",
329.         18000000,
330.         18999999,
331.         "new_grids_rule")
332.
333.     base.Renumber(general_element_rule)
334.     base.Renumber(general_node_rule)
335.
336.     base.SetViewAngles(f_key="F10")
337.
338.     # Tampering
339.     if tamper_mode == 0:
340.         print("Origami Finished")
341.         factor = 1.0
342.         tamper(tamper_mode,factor)
343.     elif tamper_mode == 1 or tamper_mode == 2 or tamper_mode == 3 :
344.         tamper(tamper_mode,factor)
345.         print("Origami Finished")
346.     else:
347.         print("Please use tamper_mode = 1, 2 or 3")
348.
349.     move_array = base.CollectEntities(deck, None, "PART_SHELL", recursive=True)
350.     base.GeoTranslate("MOVE",
351.         0,                      #Offset will take place
352.         "SAME PART",           #No Group Options
353.         "NONE",                #Option for Sets is set to NONE
354.         0,
355.         -482.8631458992081,
356.         0,
357.         move_array,
358.         keep_connectivity=True, #Node Connectivity will be kept
359.         draw_results=True)
360.     base.SetViewAngles(f_key="F10")
361.
362.     base.GeoSymmetry("COPY",
363.         0,                      #Offset will take place
364.         "SAME PART",           #No Group Options
365.         "EXPAND",               #Option for Sets is set to NONE
366.         containers,
367.         keep_connectivity=True, #Node Connectivity will be kept
368.         draw_results=True)
369.     base.Renumber(general_element_rule)
370.     base.Renumber(general_node_rule)
371.     base.SetViewAngles(f_key="F10")
372.
373.
374.     # Grouping Nodes for TIED

```

```

375.     set = ansa.base.CreateEntity(deck, "GROUP", {"Name" : 'Defo_to_Car'})
376.     set2 = ansa.base.CreateEntity(deck, "GROUP", {"Name" : 'TIED_S_CMS_vo'})
377.     shells_recons = ansa.base.CollectEntities(deck, containers, "SHELL", recursive=True)
378.     Bound_nodes = ansa.base.CollectEntities(deck, shells_recons, "NODE", recursive=True)
379.     for node in Bound_nodes:
380.         z_coord = ansa.base.GetEntityCardValues(deck, node, "Z")
381.         # print(z_coord['Z'])
382.         if abs(z_coord['Z'] - 0.0) <= 0.02:
383.             ansa.base.AddToSet(set, node, False)
384.         elif abs(z_coord['Z'] - modulos*2*Orig_h) <= 0.02:
385.             ansa.base.AddToSet(set2, node, False)
386.
387.
388.     # base.AddToInclude(include, m)
389.     # Insert and Move CMS crossbeam
390.
391.     include_cms = base.InputPamCrash(
392.         filename="CMSQT.inc",
393.         merge_parts="off",
394.         merge_sets_by_name="on",
395.         new_include="on"
396.     )
397.     CMSQT = list()
398.     CMSQT.append(base.GetEntity(deck, "PART_SHELL", 1800800))
399.     CMSQT.append(base.GetEntity(deck, "NODE", 18100000))
400.     CMSQT.append(base.GetEntity(deck, "SENPT", 70008001))
401.     base.GeoTranslate("MOVE",
402.         0,                      #Offset will take place
403.         "SAME PART",            #No Group Options
404.         "NONE",                 #Option for Sets is set to NONE
405.         0,
406.         0,
407.         modulos*2*Orig_h+5,
408.         CMSQT,
409.         keep_connectivity=True, #Node Connectivity will be kept
410.         draw_results=True)
411.
412.
413.     # Output include
414.
415.     include_file = get_include("origami_v0.3_20180314_recons.inc")
416.     base.OutputPamCrash(include=include_file)
417.     CMS_morph(factor)
418.     include_cms = base.GetEntity(deck, "INCLUDE", 2)
419.     vals = {'Name': "CMSQT_recons.inc"}
420.     base.SetEntityCardValues(deck, include_cms, vals)
421.     base.OutputPamCrash(include=include_cms)
422.
423.
424. # Tamper functions
425. def tamper(tamper_mode=2, factor=0.8):
426.
427.     morph.MorphOrtho(db_or_visible='Visible', min_flag=False)
428.     if tamper_mode == 0:
429.         return 0
430.     mopnts = base.CollectEntities(deck, None, "MORPHPOINT")
431.     mopnts_scale = []
432.     mopnts_scale1 = []
433.     mopnts_scale2 = []
434.     mopnts_scale3 = []
435.     vals = ('ID', )
436.     coord_y = []
437.     centre1 = [0,0,0]

```

```

438.     for mopnt in mopnts:
439.         z_coord = base.GetEntityCardValues(deck, mopnt, "Z")
440.         mopnt_id = base.GetEntityCardValues(deck, mopnt, vals)
441.         if tamper_mode == 1:
442.             coord_y_aux = base.GetEntityCardValues(deck, mopnt, ('Y', ))
443.             coord_y.append(coord_y_aux.get('Y'))
444.             if z_coord['Z'] > 0 and coord_y_aux['Y'] < 0:
445.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

446.                 mopnts_scale.append(mopnt_id)
447.                 centre=[0,coord_y_aux['Y'], z_coord['Z']]
448.             elif z_coord['Z'] > 0 and coord_y_aux['Y'] > 0:
449.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

450.                 mopnts_scale1.append(mopnt_id)
451.                 centre1=[0,coord_y_aux['Y'], z_coord['Z']]
452.
453.
454.
455.             elif tamper_mode == 2:
456.                 coord_y = 0
457.                 if z_coord['Z'] > 0:
458.                     mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

459.                     mopnts_scale.append(mopnt_id)
460.                     centre=[0,coord_y, z_coord['Z']]
461.
462.
463.
464.
465.         else:
466.             coord_y_aux = base.GetEntityCardValues(deck, mopnt, ('Y', ))
467.             coord_y.append(coord_y_aux.get('Y'))
468.             if z_coord['Z'] > 0 and coord_y_aux['Y'] < 0:
469.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

470.                 mopnts_scale.append(mopnt_id)
471.                 centre=[0,coord_y_aux['Y'], z_coord['Z']]
472.             elif z_coord['Z'] > 0 and coord_y_aux['Y'] > 0:
473.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

474.                 mopnts_scale1.append(mopnt_id)
475.                 centre1=[0,coord_y_aux['Y'], z_coord['Z']]
476.             elif z_coord['Z'] <= 0 and coord_y_aux['Y'] > 0:
477.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

478.                 mopnts_scale2.append(mopnt_id)
479.                 centre2=[0,coord_y_aux['Y'], z_coord['Z']]
480.             elif z_coord['Z'] <= 0 and coord_y_aux['Y'] <= 0:
481.                 mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])

482.                 mopnts_scale3.append(mopnt_id)
483.                 centre3=[0,coord_y_aux['Y'], z_coord['Z']]
484.
485.
486.     param_id = morph.MorphParamCreateScale("scale_points",mopnts_scale,centre
487. [0],centre[1],centre[2])    ###Create the Scale parameter
488.     ###Create the Scale parameter
489.     scale_param = base.GetEntity(0,"PARAMETERS",param_id)      ### Collect the
490.     created scale parameter
491.     morph.MorphParam(scale_param,factor)      ###Morph the parameter

```

```

492.         param_id1 = morph.MorphParamCreateScale("scale_points",mopnts_scale1,
493.             centre1[0],centre1[1],centre1[2])
494.         scale_param1 = base.GetEntity(0,"PARAMETERS",param_id1)
495.         morph.MorphParam(scale_param1,factor)      ###Morph the parameter
496.
497.     if tamper_mode == 3:
498.         param_id2 = morph.MorphParamCreateScale("scale_points",mopnts_scale2,
499.             centre2[0],centre2[1],centre2[2])
500.         scale_param2 = base.GetEntity(0,"PARAMETERS",param_id2)
501.         morph.MorphParam(scale_param2,factor)      ###Morph the parameter
502.         param_id3 = morph.MorphParamCreateScale("scale_points",mopnts_scale3,
503.             centre3[0],centre3[1],centre3[2])
504.         scale_param3 = base.GetEntity(0,"PARAMETERS",param_id3)
505.         morph.MorphParam(scale_param3,factor)      ###Morph the parameter
506.
507.     def CMS_morph(factor):
508.         ent = base.GetEntity(deck, 'PART_SHELL', 1800800)
509.         status = base.Or(ent)
510.         (x,y,z)=base.Cog(ent)
511.         morph.MorphOrtho(db_or_visible='Visible', min_flag=False)
512.         mopnts = base.CollectEntities(deck,None,"MORPHPOINT",filter_visible=True)
513.
514.         mopnts_scale = []
515.         mopnts_scale1 = []
516.         mopnts_scale2 = []
517.         mopnts_scale3 = []
518.         vals = ('ID', )
519.         coord_y = []
520.         x_i = [0,0,0,0]
521.         x_measure_defo = []
522.         centre = [0,0,0,0]
523.         for mopnt in mopnts:
524.             y_coord = base.GetEntityCardValues(deck, mopnt, "Y")
525.             z_coord = base.GetEntityCardValues(deck, mopnt, "Z")
526.             mopnt_id = base.GetEntityCardValues(deck, mopnt, vals)
527.             mopnt_id = base.GetEntity(deck, "MORPHPOINT", mopnt_id['ID'])
528.             if y_coord['Y'] > 0 and z_coord['Z'] < z:
529.                 mopnts_scale.append(mopnt_id)
530.                 centre[0]=[0,y_coord['Y'], z_coord['Z']]
531.             elif y_coord['Y'] < 0 and z_coord['Z'] < z:
532.                 mopnts_scale1.append(mopnt_id)
533.                 centre[1]=[0,y_coord['Y'], z_coord['Z']]
534.             elif y_coord['Y'] > 0 and z_coord['Z'] > z:
535.                 mopnts_scale2.append(mopnt_id)
536.                 centre[2]=[0,y_coord['Y'], z_coord['Z']]
537.             elif y_coord['Y'] < 0 and z_coord['Z'] > z:
538.                 mopnts_scale3.append(mopnt_id)
539.                 centre[3]=[0,y_coord['Y'], z_coord['Z']]
540.         mopnts_all=[mopnts_scale,mopnts_scale1,mopnts_scale2,mopnts_scale3]
541.
542.         # CMS Factor calculation
543.         for i in range(0,2):
544.             base.Invert()
545.             mopnts = base.CollectEntities(deck,None,"MORPHPOINT",filter_visible=T
546.             rue)
547.             for mopnt in mopnts:
548.                 x_coord = base.GetEntityCardValues(deck, mopnt, "X")
549.                 if x_coord['X'] <= x_i[0]:
550.                     x_i[0] = x_coord['X']
551.                 if x_coord['X'] >= x_i[1]:
552.                     x_i[1] = x_coord['X']
553.             x_measure_defo.append(x_i[1]-x_i[0])
554.         cms_factor = x_measure_defo[0]/x_measure_defo[1]/44.17*126.45

```

```
553.  
554.     # Morph  
555.     for i in range(0,4):  
556.         param_id = morph.MorphParamCreateScale("scale_points",mopnts_all[i],c  
entre[i][0],centre[i][1],centre[i][2])    ###Create the Scale parameter  
557.         scale_param = base.GetEntity(0,"PARAMETERS",param_id)      ### Collect  
the created scale parameter  
558.         morph.MorphParam(scale_param,cms_factor*factor)      ###Morph the param  
eter  
559.     mesh.Reconstruct()  
560.     base.All()  
561.     base.SetViewAngles(f_key="F10")  
562.  
563.  
564. def get_include(name):  
565.     ents = base.NameToEnts(name)  
566.     for ent in ents:  
567.         if ent._ansaType(deck) == 'INCLUDE':  
568.             return ent  
569.     return 0
```