UNIVERSIDADE FEDERAL DE SANTA CATARINA

Departamento de Engenharia Mecânica
Departamento de Engenharia Civil
Tutorial para utilização do software de otimização utilizando o EGO e o Kriging

Esse tutorial é um passo a passo sobre como utilizar o software de otimização *Efficient Global Optimization* (EGO) utilizando modelos substitutos com o Kriging. Grande parte do desenvolvimento desse otimizador foram baseados nos seguintes textos:

- FORRESTER, A.; SOBESTER, A. S.; KEANE, A. *Engineering design via surrogate modelling: a pratical guide*. Chichester, West Sussex, United Kingdom: John Wiley & Sons, 2008.
- JONES, D. R. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, v. 21, n. 4, p. 345-383, 2001.
- JONES, D. R.; SCHONLAU, M.; WELCH, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, v. 13, n. 4, p. 455-492, 1998.

Qualquer erro ou sugestão, favor entrar em contato com os criadores:

1. **Fábio Felipe dos Santos Nascentes**: Programa de Pós Graduação em Engenharia Mecânica, POSMEC - UFSC, fabiotrabmat@gmail.com;
2. **Dr. Rafael Holdorf Lopez**: Departamento de Engenharia Civil e Programa de Pós Graduação em Engenharia Civil, PPGEC - UFSC, rafaelholdorf@gmail.com.

O tutorial está dividido em duas partes. A primeira é referente aos ajustes dos parâmetros do otimizador e da função objetivo. Na segunda parte são fornecidas algumas dicas sobre a otimização.

PARTE 1

Na mesma pasta que contém esse tutorial, existe uma pasta chamada "EGO-Kriging". Dentro dessa pasta estão o arquivo e subpastas que controlam o otimizador. O arquivo `EGO_Kriging.m` é o responsável pelas definições dos parâmetros e do problema de otimização do usuário, este arquivo poderá ser editado a vontade pelo usuário. A subpasta "script_opt" possui todas as rotinas do otimizador em arquivos .p e não podem ser editadas pelo usuário. A subpasta "fobj" armazenará as funções objetivo do usuário em arquivos .m.

1 Ajuste dos Parâmetros

Primeiro recomenda-se editar os parâmetros para otimização. Para tanto abra o arquivo `EGO_Kriging.m`. Os parâmetros que deverão ser passados à rotina de otimização estão divididos em dois grupos:

Parâmetros Obrigatórios e Parâmetros Opcionais. Esses parâmetros deverão ser passados em forma de struct para a rotina de otimização, logo recomenda-se utilizar

```
NomeDaStruct.nomeDoParametro = ValorDoParametro.
```

Os nomes dos parâmetros deverão ser exatamente iguais aos que constam nas Tabelas 1.1 e 1.2. Qualquer divergência, incluindo maiúsculas e minúsculas, irá acarretar em erro de execução. Os parâmetros poderão ser definidos em qualquer ordem.

Tabela 1.1: Descrição dos parâmetros obrigatórios.

Parâmetros Obrigatórios	
Nome	**Descrição**
func	Nome da função objetivo (deve ser uma *string*)
maxFE	Número máximo de avaliações da função objetivo
k	Quantidade de dimensões do problema
LB	Vetor $1 \times k$ com os limites inferiores das variáveis
UB	Vetor $1 \times k$ com os limites superiores das variáveis
calculoVetorizado	Define se a função objetivo pode ser calculada em forma de vetores. Mais detalhes na Seção 2

Tabela 1.2: Descrição dos parâmetros opcionais.

Parâmetros Opcionais	
Nome	**Descrição**
n	Número de pontos pertencentes à amostra inicial. Caso não seja informado seu valor, a seguinte regra será usada por padrão: $n = 6.5k$ se $k \leq 5$ ou $n = 4.5k$ se $k > 5$, arredondado para cima
S	Amostra inicial com n pontos pertencentes a $[0, 1]^k$. Valor Padrão: Conjunto determinado por um Hipercubo Latino.
tolMinEI	Tolerância mínima permitida para o Expected Improvement. Valor padrão: 10^{-40}
nPop*	Número de indivíduos para o otimizador heurístico AG. Valor padrão: $20k$
graf	Faz uma saída gráfica nos casos de funções 1D e 2D. Somente dois valores possíveis: 0 - Não é realizada a saída gráfica (valor padrão) 1 - Realiza a saída gráfica na tela

* O otimizador AG do próprio Matlab é utilizado em dois processos diferentes e extremamente essenciais: a determinação dos parâmetros do modelo substituto e a determinação do Infill Point. Portanto seu ajuste influencia diretamente o desempenho do programa. Um valor baixo pode gerar uma otimização ruim dos parâmetros ou não otimizar corretamente a métrica do IP. Já um valor alto tende a obter melhores resultados, porém o tempo de processamento pode aumentar consideravelmente.

A chamada do otimizador é realizada ao se utilizar a função `otimEGO.m` (última linha do código em `EGO_Kriging.m`). Essa função deverá receber um único parâmetro de entrada, que será a struct

com os parâmetros do problema, e possui dois argumentos como saída: Resultado e fitKrg. A variável `Resultado` é uma struct com as seguintes variáveis:

Tabela 1.3: Descrição das variáveis de saída em `Resultado`.

Variáveis de Saída em `Resultado`	
Nome	**Descrição**
MelhorMinimizador	Melhor minimizador obtido
MelhorValorOtimo	Melhor valor mínimo obtido
PosMelhorMin	Iteração em que obteve-se o melhor valor mínimo
numFE	Número de avaliações da função objetivo
numIPAdd	Número de Infill Points que foram adicionados
numPontosIniciais	Número de pontos da amostra inicial
fobj	Contém os valores da função objetivo em todos os pontos
Histfmin	Histórico dos melhores mínimos
Histdmin	Histórico dos minimizadores
CriterioParada	Motivo do término do otimizador

A variável `fitKrg` é uma struct que contém todas as informações necessárias sobre o modelo substituto final obtido. Com essa variável pode se realizar a predição de novos pontos ou o cálculo do Expected Improvement. Mais detalhes serão dados na Parte 2 desse tutorial.

2 Função Objetivo

A função objetivo deverá ser especificada em um arquivo do tipo .m. A pasta fobj está criada para ajudar na organização do programa, porém não é necessário que o arquivo da função esteja nessa pasta, ele basta acompanhar o arquivo que faz a chamada do otimizador, nesse tutorial o arquivo `EGO_Kriging.m`.

Essa função deverá receber um único argumento e retornar somente um único argumento. Um modelo geral proposto é dado por

```
function f = NomeDaFuncao(d)
% Implemente o codigo
end
```

onde d é o argumento de entrada e f é o argumento de saída.

Caso o parâmetro `calculoVetorizado` possuir o valor 0 então a função precisa tratar um único vetor de variáveis de projeto, retornando apenas um único valor da função objetivo. Agora se caso for atribuído a `calculoVetorizado` o valor 1, então deverá ser implementado o cálculo vetorizado do argumento de entrada. Nesse contexto, d será uma matriz $n \times k$ onde n é um certo número de vetores de variáveis de projeto e k é o número de variáveis de projeto. O argumento de saída deverá ser portanto um vetor coluna f com n linhas, contendo o valor da função em cada um dos n vetores de variáveis de projeto. Nenhum outro valor poderá ser atribuído à variável `calculoVetorizado`.

Como exemplo suponha que queiramos otimizar a função $f(\mathbf{d}) = d_1^2+d_2^2+d_3^2$, onde $\mathbf{d} = \{d_1, d_2, d_3\}$ é

um vetor de 3 variáveis de projeto. Existem inúmeras formas de se implementar o cálculo dessa função para os dois casos vistos anteriormente, aqui propomos o seguinte: Para o caso `calculoVetorizado = 0` é indicado:

```
function f = NomeDaFuncao(d)
f = d(1)^2 + d(2)^2 + d(3)^2
end
```

porém para o caso `calculoVetorizado = 1` indicamos

```
function f = NomeDaFuncao(d)
f = d(:,1).^2 + d(:,2).^2 + d(:,3).^2
end
```

Para maiores detalhes consulte o manual do Matlab para tratamento de vetores e matrizes.

PARTE 2

Essa parte contém algumas dicas referentes aos parâmetros e também aos resultados que são esperados pelo otimizador.

- O otimizador foi proposto para funcionar bem com qualquer quantidade de dimensões na variável de projeto. Porém conforme k cresce o tempo computacional começa a crescer de forma exponencial. Portanto para problemas grandes é possível que o algoritmo demore horas, até dias, para poder realizar todas as rotinas. Portanto paciência;

- A quantidade de pontos iniciais n não tem um valor fixo, porém ela deve acompanhar a complexidade do problema. Uma população inicial muito grande inviabiliza o método pois o custo computacional do primeiro modelo já sairá muito caro, e cada iteração de IP se tornará extremamente lenta. Logo deixe para que sejam realizas muitas iterações de IP, já que o mesmo é baseado em uma métrica de melhora da superfície de resposta;

- A amostra inicial S, se não declarada, é obtida de um Hipercubo Latino contido no hiperespaço $[0, 1]^k$. Esse hipercubo é otimizado seguindo o processo proposto por Max D. Morris e Toby J. Mitchell em seu trabalho "*Exploratory designs for computational experiments*" de 1993;

- O número de IP adicionados é crucial para a obtenção do melhor mínimo. Não há um valor fixo desse número, porém ele está limitado pelo número máximo de vezes que a função objetivo poderá ser calculada. Quanto maior esse valor também cresce o tempo de processamento, assim normalmente a um acréscimo de tempo computacional a cada IP adicionado;

- Após o término da otimização, para que o usuário faça novas predições utilizando o Kriging e com as informações baseadas no último modelo substituto criado, basta que ele utilize a seguinte chamada

  `[predicao, erroPred] = predKrg(d, fitKrg),`

  onde `predicao` é o valor do Kriging e `erroPred` é a variância do erro na predição, ambos calculados no ponto $\mathbf{d}$. `fitKrg` é a variável de saída do otimizador, como explicado na Parte

1. Uma observação importante é que $\mathbf{d} \in [0,1]^k$, ou seja, para se realizar a predição deve-se converter o vetor de variáveis de projeto para o hipercubo $[0,1]^k$.

   Para calcular o Expected Improvement, pode-se utilizar a chamada

   ```
   EI = - expImp(d, fitKrg),
   ```

   onde o sinal de $-$ é necessário pois o valor do Expected Improvement é multiplicado por -1 para sua maximização pelo AG. Os argumentos de entrada são os mesmos utilizados na predição.

- Como todo o processo de otimização é baseado em números aleatórios, principalmente na população inicial e no AG, é bem provável que a cada simulação em que o algoritmo seja executado, o valor mínimo obtido seja diferente. Portanto é recomendado que o algoritmo seja executado um certo número de vezes e que o resultado final seja apresentado em forma estatística (média, percentis e desvio padrão). A seguir são exemplificados algumas execuções com o otimizador e os resultados em forma estatística.

3 Análises Numéricas

3.1 Função Branin Modificada

A função Branin Modificada é definida por

$$f(\mathbf{x}) = \left(x_2 - \frac{5,1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left[\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 1\right] + 5x_1$$

onde $\mathbf{x} \in S \subset \mathbb{R}^2$ com $\mathbf{lb} = \{-5, 0\}^T$ e $\mathbf{ub} = \{10, 15\}^T$, cujo mínimo global é $f_{\min} = -16.644022$ em $\mathbf{x}^* = \{-3.689285, 13.629987\}^T$.

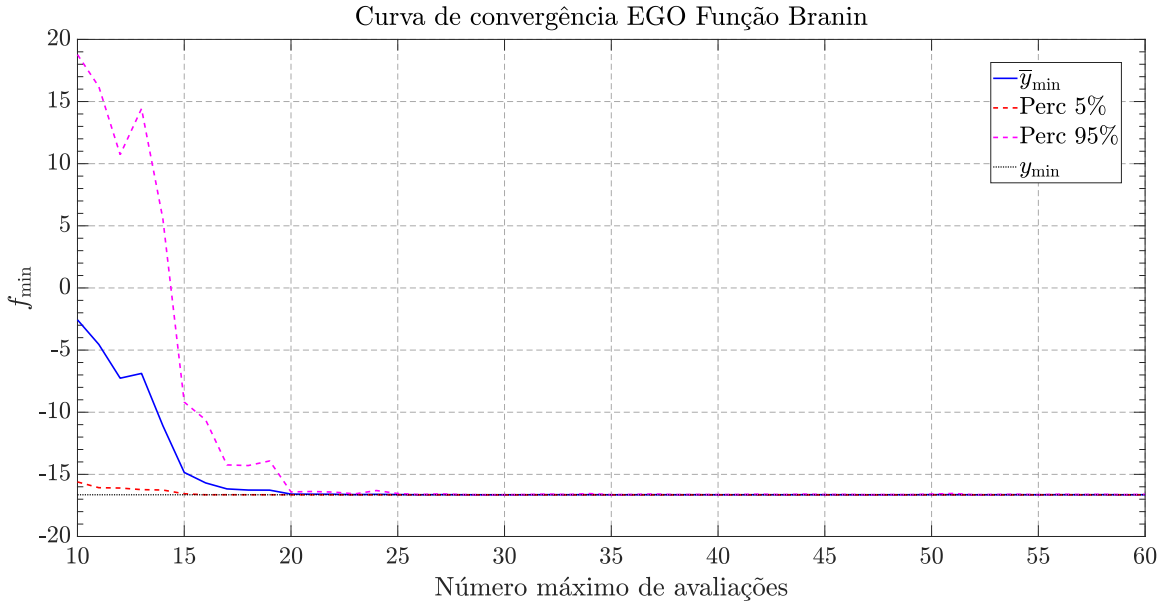

Figura 3.1: Valores estatísticos resultantes de 20 simulações do otimizador.

Para esse caso setamos o número de pontos iniciais como 10 e realizamos 20 simulações do otimizador variando-se de 10 a 60 avaliações da função, ou seja, de 0 a 50 Infill Points foram adicionados. O resultado está apresentado na Figura 3.1. Pode-se notar como a adição de IP é essencial para o sucesso do otimizador. Vemos que para um total de 20 avaliações da função (10 iniciais mais 10 IP) a média e os percentis já estão muito próximos do mínimo analítico da função, se mantendo assim até o máximo de 60 avaliações da função. Como mostrado pela curva que representa o percentil de 5%, em quase todas as simulações obtivemos um mínimo muito próximo do valor analítico, mesmo que para valores pequenos de avaliações da função.

3.2 Função Hartmann 6D

A função Hartmann em seis dimensões é dada por

$$f(\mathbf{x}) = -\sum_{i=1}^{4}\left\{\alpha_i \exp\left[-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right]\right\},$$

onde

$$\alpha = \{1.0, 1.2, 3.0, 3.2\}^T$$

$$\mathbf{A} = \begin{bmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}.$$

O espaço de busca será dado por $\mathbf{x} \in S \subset \mathbb{R}^6$ com $\mathbf{lb} = \{0, \cdots, 0\}^T$ e $\mathbf{ub} = \{1, \cdots, 1\}^T$. Seu mínimo é $f_{\min} = -3.32237$ em

$$\mathbf{x}^* = \{0.20196, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573\}^T.$$

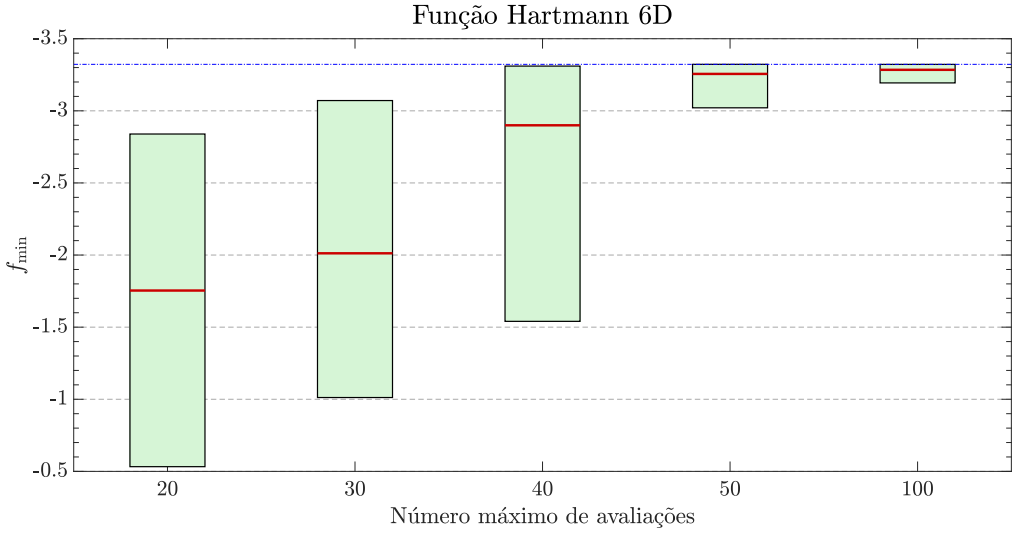

Figura 3.2: Média do valor mínimo da função Hartmann 6D em 100 simulações.

Novamente é explorado a influência do número de avaliações da função objetivo para a otimização. A amostra inicial possui 20 pontos, começando com 20 avaliações da função objetivo. Na Figura 3.2 temos representados as estatísticas das 20 simulações do otimizador. Cada retângulo representa a variabilidade dos resultados obtidos, onde o lado inferior representa um percentil de 5% (95% dos dados estão acima do lado inferior) e o lado superior representa o percentil de 95% dos dados. A linha vermelha no interior do retângulo representa o valor mínimo médio das simulações. A linha azul representa o valor mínimo analítico, ou o mínimo apresentado pela literatura atual. Podemos observar que para 40 avaliações já conseguimos obter um mínimo bem próximo do mínimo analítico, porém a melhor média do valor mínimo começa a estabilizar a partir de 100 avaliações.

3.3 Função Levy 10D

A função Levy em dez dimensões é dada por

$$f(\mathbf{x}) = \text{sen}^2(\pi w_1) + \sum_{i=1}^{9} \left\{ (w_i - 1)^2 \left[1 + 10\,\text{sen}^2(\pi w_i + 1) \right] \right\} + (w_{10} - 1)^2 \left[1 + \text{sen}^2(2\pi w_{10}) \right],$$

onde $w_i = 1 + \dfrac{x_i - 1}{4}$ para todo $i = 1, 2, \cdots, 10$. Seu mínimo é $f_{\text{min}} = 0.0$ e é obtido em $\mathbf{x}^* = \{1, 1, \cdots, 1\}^T$. O espaço de busca é dado por $\mathbf{x} \in S \subset \mathbb{R}^{10}$ com $\mathbf{lb} = \{-10, -10, \cdots, -10\}^T$ e $\mathbf{ub} = \{10, 10, \cdots, 10\}^T$.

Esse caso é um pouco mais complicado de se obter o mínimo devido aos vários mínimos locais que a função possui. Então o número de IP deverá ser o maior possível para uma boa aproximação. A Figura 3.3 apresenta os dados para as 10 simulações. A amostra inicial possui 30 pontos, começando a otimização com 30 avaliações da função objetivo

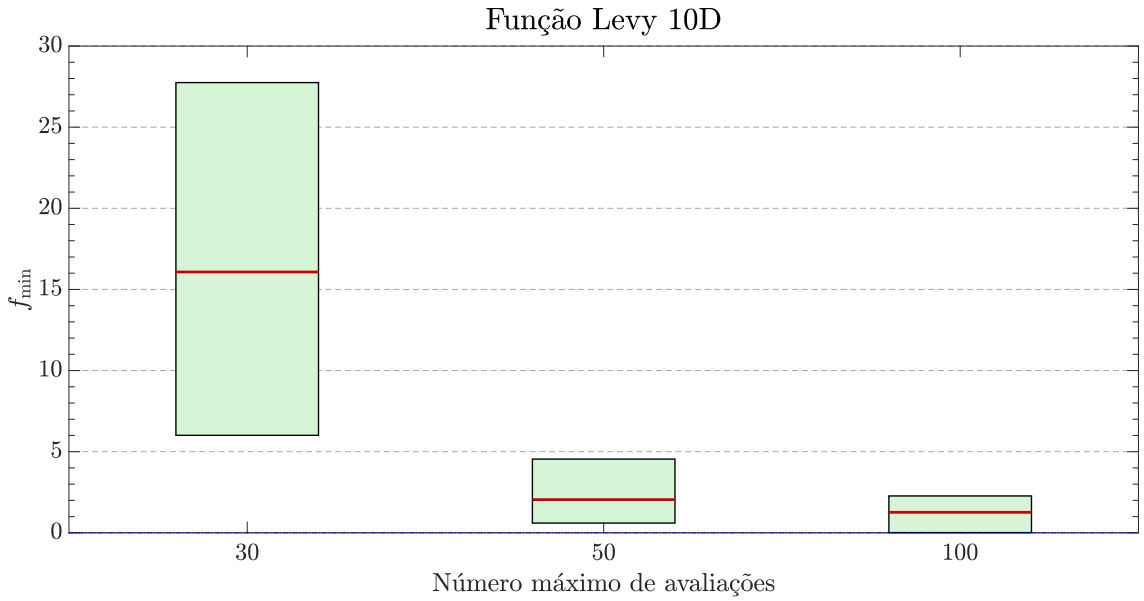

Figura 3.3: Média do valor mínimo da função Levy 10D em 100 simulações.

Vemos que para 30 avaliações não obtivemos sucesso em otimizar a função, porém para 50 avaliações o resultado melhora consideravelmente. Para 100 avaliações já obtemos o primeiro mínimo bem próximo do mínimo analítico com uma média de 1.2618369.