Exterior Penalty Optimization Method

Bu belge,exterior penalty yöntemi için MATLAB uygulamasının açıklamasını ve analizini içermektedir. Kod, kısıtlı optimizasyon problemlerini ceza fonksiyonları kullanarak kısıtsız bir şekle dönüştürmeyi ve çözmeyi amaçlamaktadır.

Hazırlayan: Muhammet Kaan ALSANCAK

Okul Numarası:245113009

Tarih: 1 Ocak 2025

Giriş

Bu kod, **Exterior Penalty Optimization** yöntemini uygulamak için tasarlanmıştır. Yöntem, kısıtlı bir optimizasyon problemini, ceza fonksiyonları kullanarak kısıtsız bir probleme dönüştürür ve iteratif olarak çözüm arar.

Aşağıda kodun parçalarını ve işleyişini detaylı bir şekilde açıklanmıştır:

1.Ana Fonksiyon: `exterior_penalty_optimization`

Bu fonksiyon, optimizasyon sürecini başlatır ve iterasyonlar boyunca ceza fonksiyonu yardımıyla çözüm arar.

Kullanıcı Girdisi

```
max_iterations = input('Lütfen iterasyon sayısını girin: ');
```

- Kullanıcıdan maksimum iterasyon sayısı alınır.
- Bu değer, algoritmanın kaç kez çalıştırılacağını belirler.

Başlangıç Parametreleri

```
x = [-0.5; 0.5]; % Başlangıç noktası [x1, x2]
r = 1; % Ceza katsayısı
r_increment = 10; % Ceza katsayısı artış oranı
tolerance = 1e-6; % Durdurma kriteri
```

- **x**: Başlangıç noktası $(x_1 = -0.5, x_2 = 0.5)$
- **r**: İlk ceza katsayısı (1).
- **r_increment**: Her iterasyonda ceza katsayısının artış oranı (10 kat).
- **tolerance**: Durdurma kriteri. Ceza fonksiyonu 10^{-6}dan küçükse algoritma durur.

Sonuçların Kaydedilmesi

```
results = struct('iteration', [], 'x1', [], 'x2', [], 'f', [], 'penalty', []);
```

- **`results`** yapısı, her iterasyondaki değerleri saklar:
- **iteration**: İterasyon sayısı.
- **x1, x2**: Çözüm değerleri.
- **f**: Hedef fonksiyon değeri.
- **penalty**: Ceza fonksiyonu değeri.

Optimizasyon Döngüsü

```
for iter = 1:max_iterations
    ...
end
```

Bu döngü, optimizasyon algoritmasını iteratif olarak çalıştırır.

Ceza Fonksiyonunun Optimizasyonu

```
options = optimoptions('fminunc', 'Display', 'iter', 'Algorithm', 'quasi-newton');
penalty_function = @(x) objective_with_penalty(x, r);
[x, fval] = fminunc(penalty_function, x, options);
```

- **`fminunc`**: Kısıtsız optimizasyon algoritması (Quasi-Newton yöntemi).
- **`penalty_function`**: Ceza fonksiyonu, rrr parametresi ile çağrılır.
- **`x` ve `fval`**: İterasyon sonunda bulunan çözüm ve hedef fonksiyon değeri.

Kısıtların Hesaplanması

```
g1 = max(0, -x(1) + 1);
g2 = max(0, -x(2));
```

Kısıtların ihlali kontrol edilir:

$$g_1(x) = \max(0, -x_1 + 1)$$
$$g_2(x) = \max(0 - x_2)$$

Sonuçların Kaydedilmesi

```
results(iter).iteration = iter;
results(iter).x1 = x(1);
results(iter).x2 = x(2);
results(iter).f = fval;
results(iter).penalty = r * (g1^2 + g2^2);
```

İterasyon sonuçları **`results`** yapısına eklenir.

Ceza Katsayısının Artışı

```
r = r * r_increment;
```

Ceza katsayısı her iterasyonda artırılır (r=r×10).

Durdurma Kriteri

```
if abs(results(iter).penalty) < tolerance
    fprintf('Optimizasyon durduruldu: Ceza fonksiyonu sıfıra yaklaştı.\n');
    break;
end
```

Ceza fonksiyonu tolerans seviyesine ulaştığında algoritma durur.

2.Ceza Fonksiyonu: `objective_with_penalty`

Bu fonksiyon, hedef fonksiyon ve ceza fonksiyonunu birleştirir.

```
function f = objective_with_penalty(x, r)
```

Orijinal Hedef Fonksiyonu

```
f_original = (1/3) * (x(1) + 1)^3 + x(2);
```

Hedef fonksiyon:

$$f_{\text{original}}(x) = \frac{1}{3}(x+1)^3 + x_2$$

Ceza Fonksiyonu

```
g1 = max(0, -x(1) + 1);
g2 = max(0, -x(2));
penalty = r * (g1^2 + g2^2);
```

Kısıtların ihlali durumunda ceza fonksiyonu:

$$\text{penalty} = r.\left(\max(0, g_1(2))^2 + \max(0, g_2(x))^2\right)$$

Ceza ve Hedef Fonksiyonunun Birleştirilmesi

```
f = f_original + penalty;
```

$$\text{Toplamfonksiyon}: f = f\text{original} + penalty$$

3. Görselleştirme: `visualize_results`

Bu fonksiyon, sonuçları grafiklerle ve tabloyla görselleştirir.

```
function visualize_results(results)
```

Orjinal Fonksiyon Çizimi

```
x = linspace(-1, 2, 100); % x1 değerleri aralığı
f_original = @(x1) (1/3) * (x1 + 1).^3;
plot(x, f_original(x), 'k--', 'LineWidth', 2, 'DisplayName', 'Orijinal Fonksiyon');
```

Kesik siyah çizgi ile orijinal fonksiyon gösterilir.

Ceza Fonksiyonlarının Çizimi

```
for i = 1:length(results)
    penalty_function = @(x1) objective_with_penalty([x1; results(i).x2], 10^(i-1));
    y = arrayfun(penalty_function, x);
    plot(x, y, 'DisplayName', sprintf('Iterasyon %d (r = %d)', results(i).iteration, 10^(i-1)))
end
```

Her iterasyonun ceza fonksiyonları farklı renklerle çizilir.

Optimum Noktanın İşaretlenmesi

```
x_optimum = results(end).x1; % Optimum x1 değeri
plot([x_optimum x_optimum], ylim, 'r-', 'LineWidth', 1.5, 'DisplayName', 'Barier');
plot(results(end).x1, results(end).f, 'ro', 'MarkerSize', 10, 'DisplayName', 'Optimum Nokta');
```

- Optimum nokta kırmızı daire ile işaretlenir.
- Optimum noktaya karşılık gelen dikey bir "barier" çizgisi eklenir.

Tablo Çıktısı

```
fprintf('\nIterasyon Sonuçları:\n');
fprintf('Iterasyon\tx1\t\t\tx2\t\tf(x1,x2)\t\tCeza\n');
for i = 1:length(results)
    fprintf('%d\t\t%.6f\t%.6f\t%.6f\t%.6f\n', results(i).iteration, results(i).x1, results(i).x
end
```

```
All functions in a script must be closed with an 'end'.
```

İterasyon sonuçları tablo halinde yazdırılır.

4 .Sonuç

Bu kod, **Exterior Penalty Optimization** yönteminin bir örneğidir. Hedef fonksiyon ve ceza fonksiyonu iterasyonlar boyunca optimize edilir. Sonuçlar grafikler ve tabloyla kullanıcıya sunulur. Kod, genel amaçlı kısıtlı optimizasyon problemleri için uygundur. Sonuçlar Engineering Optimization Theory and Practice kitabının tablo 7.9 ile tutarlıdır.

```
Iterasyon Sonuçları:
Iterasyon    x1           x2          f(x1,x2)         Ceza
1          0.236068    -0.500000    0.963107     0.833592
2          0.832159    -0.050000    2.306774     0.306705
3          0.980390    -0.005000    2.624948     0.040954
4          0.998004    -0.000500    2.662425     0.004234
5          0.999800    -0.000050    2.666242     0.000425
6          0.999980    -0.000005    2.666624     0.000043
7          0.999998    -0.000000    2.666662     0.000004
8          1.000000    -0.000000    2.666666     0.000000
```

Table 7.5 Results for Example 7.9

Value of r	x_1^*	x_2^*	$\phi_{\min}(r)$	$f_{\min}(r)$
0.001	−0.93775	−500.00000	−249.9962	−500.0000
0.01	−0.80975	−50.00000	−24.9650	−49.9977
0.1	−0.45969	−5.00000	−2.2344	−4.9474
1	0.23607	−0.50000	0.9631	0.1295
10	0.83216	−0.05000	2.3068	2.0001
100	0.98039	−0.00500	2.6249	2.5840
1,000	0.99800	−0.00050	2.6624	2.6582
10,000	0.99963	−0.00005	2.6655	2.6652
∞	1	0	$\frac{8}{3}$	$\frac{8}{3}$