کد برنامه:

برای اجرای برنامه ابتدا باید فایل Rundeopt را اجرا نمایید، این فایل خود تابع deopt است را فراخوانی می‌کند. اصل برنامه تفاضل تکاملی در تابع deopt است. در ادامه این دو فایل را توضیح خواهیم نمود:

فایل Rundeopt

حداقل مقدار کمینه اگر از این مقدار کمتر شود، اجرای برنامه متوقف می‌شود:

```
F_VTR = 0.00001;
```

تعداد ویژگی‌های مسئله مورد بررسی

```
I_D = 13;
```

حداقل و حداکثر مرز برای مقدار دهی جمعیت اولیه

```
FVr_minbound = -100*ones(1,I_D);
FVr_maxbound = +100*ones(1,I_D);
```

تعداد اعضای جمعیت

```
I_NP = 100;
```

حداکثر تعداد دفعات اجرای بهینه‌سازی

```
I_itermax = 50000;
```

طول گام

```
F_weight = 0.85;
```

مقدار احتمال crossover

```
F_CR = 1;
```

دستورات زیر مقادیر تعیین شده را به ساختار مسئله یا ورودی تابع تکامل تفاضلی را تعریف می‌کند:

```
S_struct.I_lentol   = I_lentol;
S_struct.FVr_x      = FVr_x;
S_struct.FVr_lim_up = FVr_lim_up;
S_struct.FVr_lim_lo = FVr_lim_lo;
```

```
S_struct.I_NP          = I_NP;
S_struct.F_weight      = F_weight;
S_struct.F_CR          = F_CR;
S_struct.I_D           = I_D;
S_struct.FVr_minbound = FVr_minbound;
S_struct.FVr_maxbound = FVr_maxbound;
S_struct.I_bnd_constr = I_bnd_constr;
S_struct.I_itermax     = I_itermax;
S_struct.F_VTR         = F_VTR;
S_struct.I_strategy    = I_strategy;
S_struct.I_refresh     = I_refresh;
S_struct.I_plotting    = I_plotting;
```

کنترل اجرای برنامه به تابع deopt توسط کد زیر:

```
[FVr_x,S_y,I_nf] = deopt('objfun',S_struct)
```

در ادامه بخش های اصلی تابع deopt را توضیح می‌دهیم:

مقدار دهی اولیه جمعیت

```
FM_pop = zeros(I_NP,I_D);
for k=1:I_NP
   FM_pop(k,:) = FVr_minbound +
rand(1,I_D).*(FVr_maxbound - FVr_minbound);
end
```

محاسبه بهترین جواب در هر دور و قراردادن آن در S_val

```
S_val(k)  = feval(fname,FM_pop(k,:),S_struct);
```

شرط ادامه تولید جوابها

```
while ((I_iter < I_itermax) & (S_bestval.FVr_oa(1) >
F_VTR))
```

تولید جواب جدید

```
FM_ui = FM_pm3 + F_weight*(FM_pm1 - FM_pm2);
FM_ui = FM_popold.*FM_mpo + FM_ui.*FM_mui;
FM_origin = FM_pm3;
```

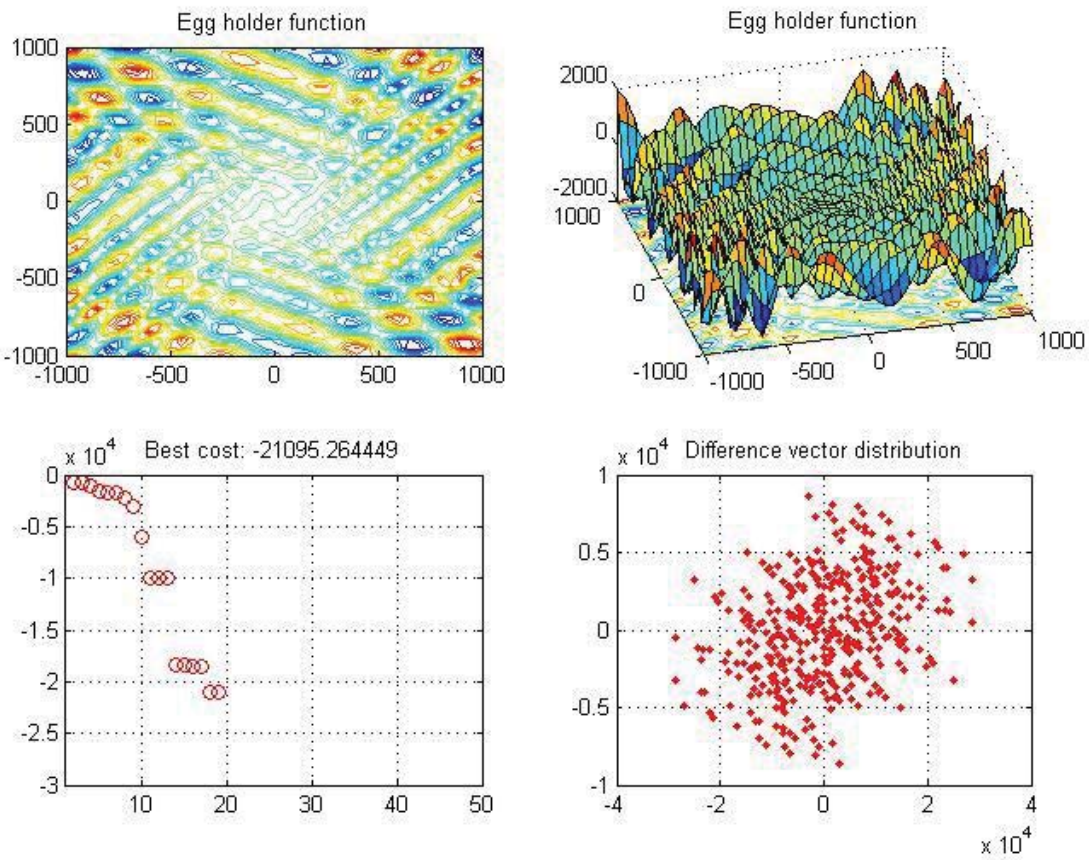
شرط جایگزینی جواب جدید به جای جواب قدیمی

```
S_tempval = feval(fname,FM_ui(k,:),S_struct);
if (left_win(S_tempval,S_val(k)) == 1)
         FM_pop(k,:) = FM_ui(k,:);
```

```
S_val(k)    = S_tempval;
```

خروجی مساله eggholder

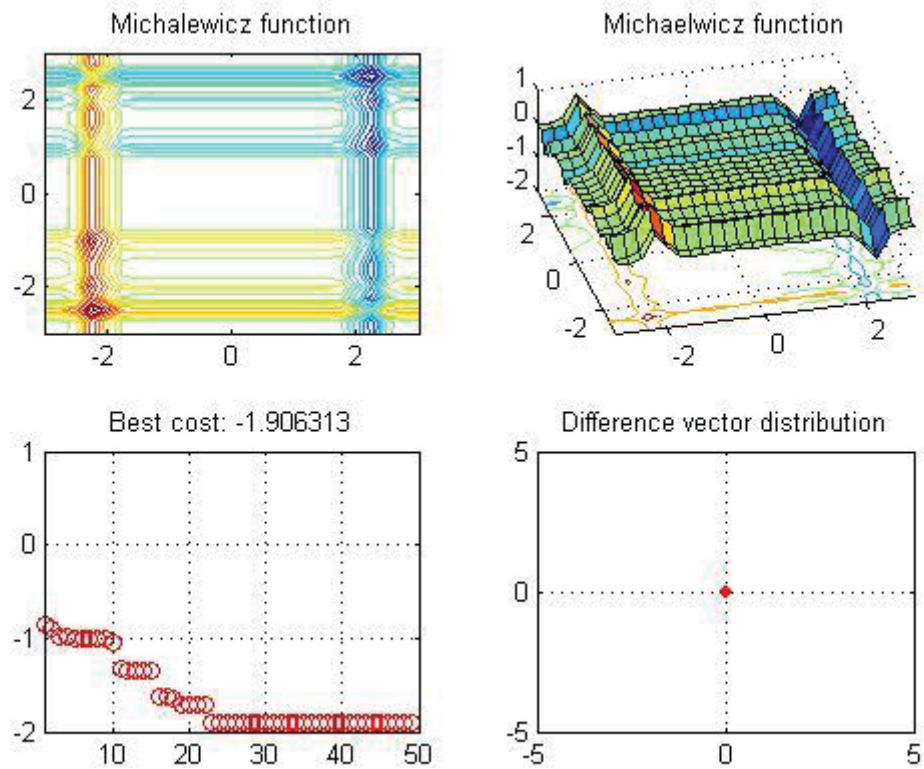
```
F_cost = -FVr_temp(1).*sin(sqrt(abs(FVr_temp(1)-
FVr_temp(2)-47)))-...

(FVr_temp(2)+47).*sin(sqrt(abs(0.5*FVr_temp(1)+FVr_temp
(2)+47)));
```



خروجی مساله Michalewicz

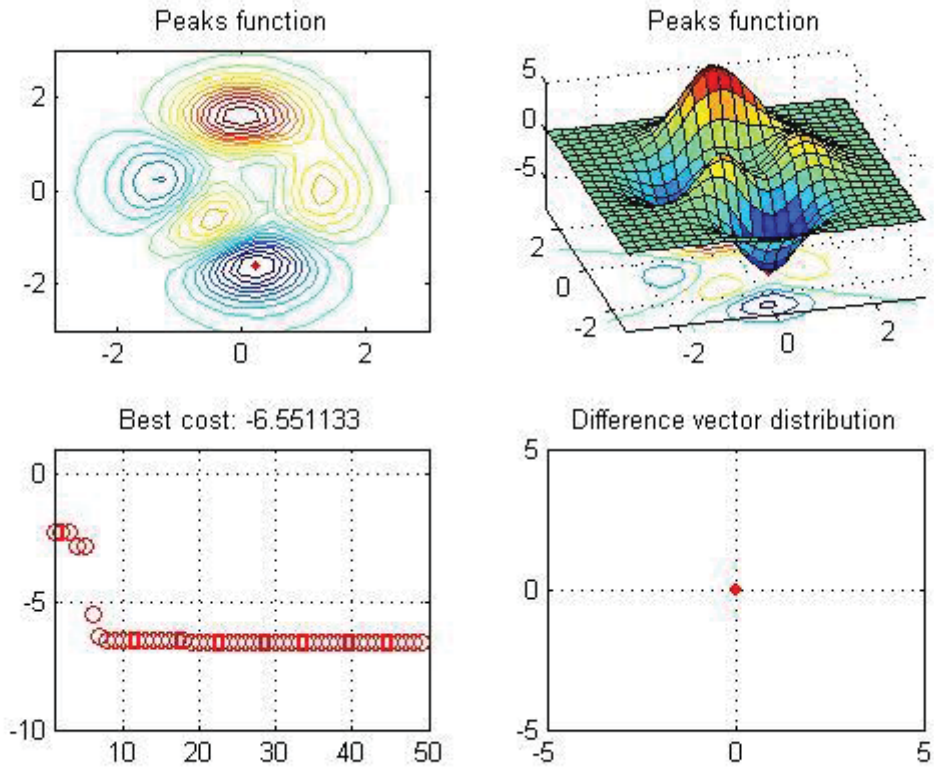
```
F_cost = -
(sin(FVr_temp(1)).*(sin((FVr_temp(1).*FVr_temp(1))/pi))
.^20 +...
```

```
sin(FVr_temp(2)).*(sin(4*(FVr_temp(2).*FVr_temp(2))/pi)
).^20);
```



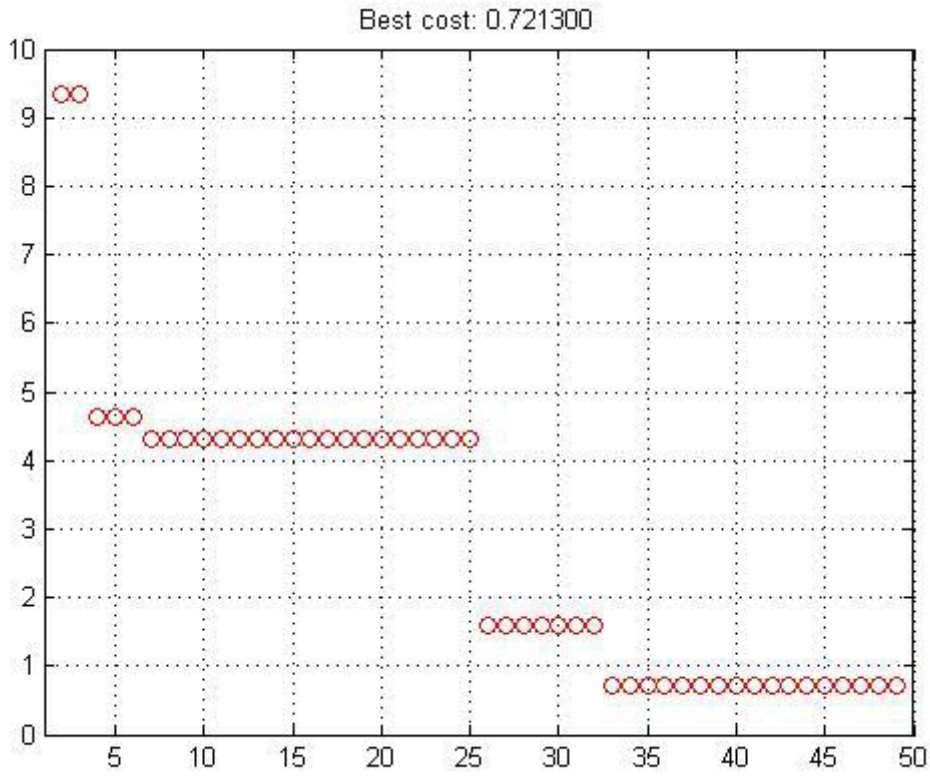
خروجی مساله Peak

```
F_cost = peaks(FVr_temp(1),FVr_temp(2));
```



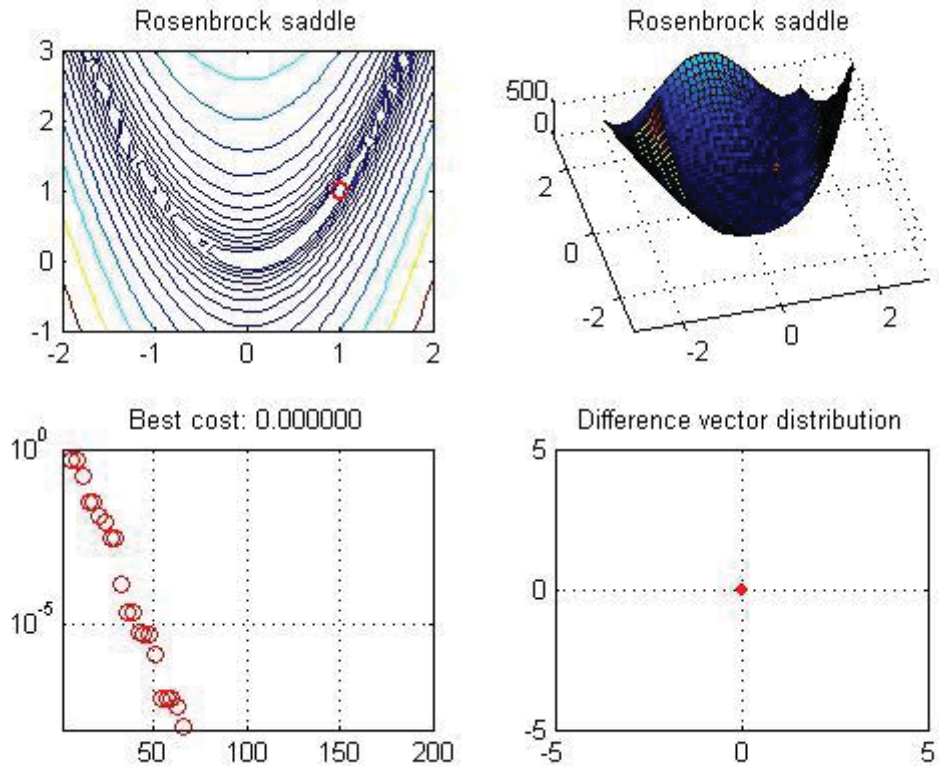
خروجی مسئله Rastrigin

```
F_cost = 10*S_struct.I_D;
for i=1:S_struct.I_D
   F_cost = F_cost + ((FVr_temp(i))^2-
10*cos(2*pi*FVr_temp(i)));
end
```



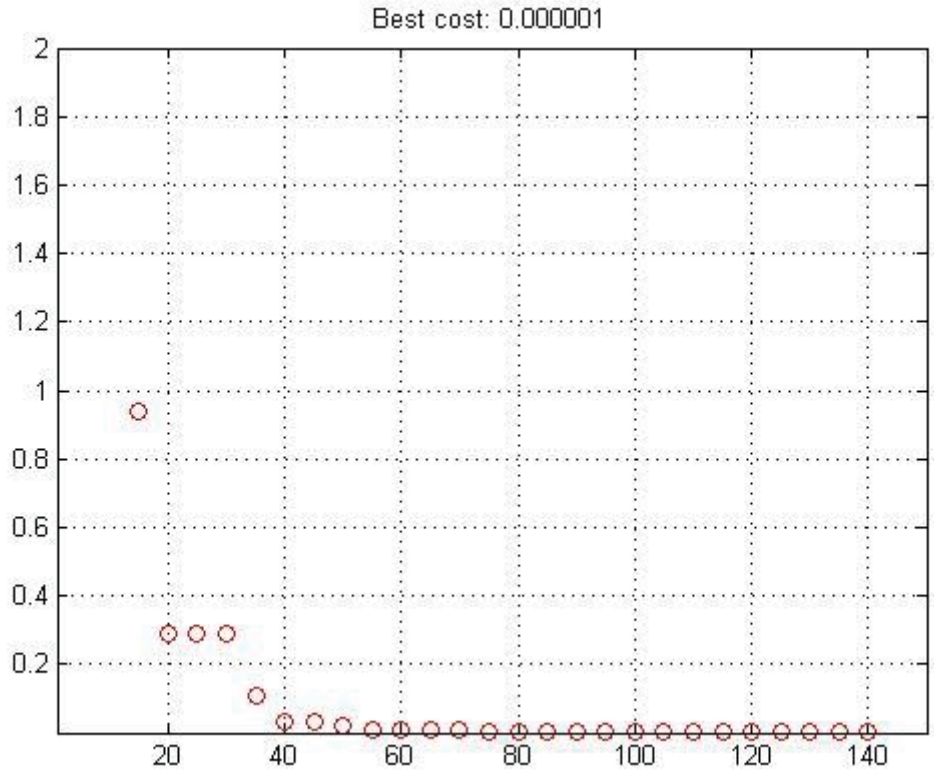
خروجی مسئله Rosenbrock

```
F_cost = 100*(FVr_temp(2)-FVr_temp(1)^2)^2+(1-
FVr_temp(1))^2;
```



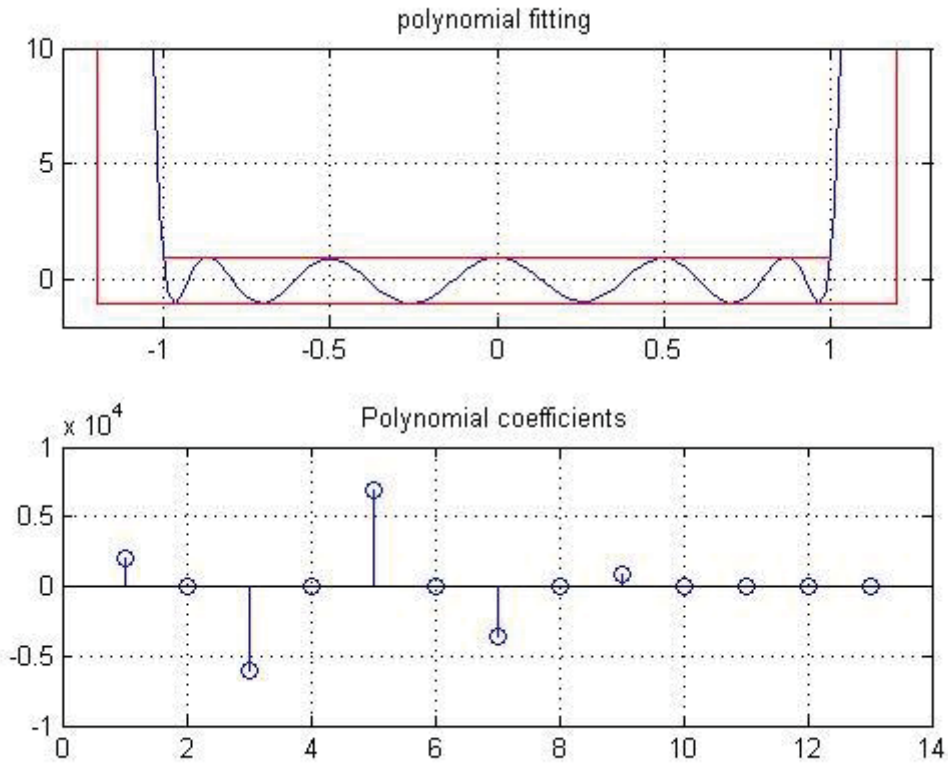
خروجی مسئله Zimmermann

```
F_h1 = 9 - FVr_temp(1) - FVr_temp(2);
F_h2 = (FVr_temp(1) - 3)^2 + (FVr_temp(2) - 2)^2 - 16;
F_h3 = FVr_temp(1)*FVr_temp(2) - 14;

F_pc1 = F_h1;
F_pc2 = 100*(1+F_h2)*step(F_h2);
F_pc3 = 100*(1+F_h3)*step(F_h3);
F_pc4 = 100*(1-FVr_temp(1))*step(-FVr_temp(1));
F_pc5 = 100*(1-FVr_temp(2))*step(-FVr_temp(2));

FVr_pc = [F_pc1 F_pc2 F_pc3 F_pc4 F_pc5];

F_cost = max(FVr_pc);
```



خروجی مسئله Chebychev

```
F_cost_tol  = sum(FVr_err_up(IVr_pos_up).^2)+
sum(FVr_err_lo(IVr_pos_lo).^2);
```


polynomial fitting
10
5
0
-1
-0.5
0
0.5
1
Polynomial coefficients
x 10^4
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14