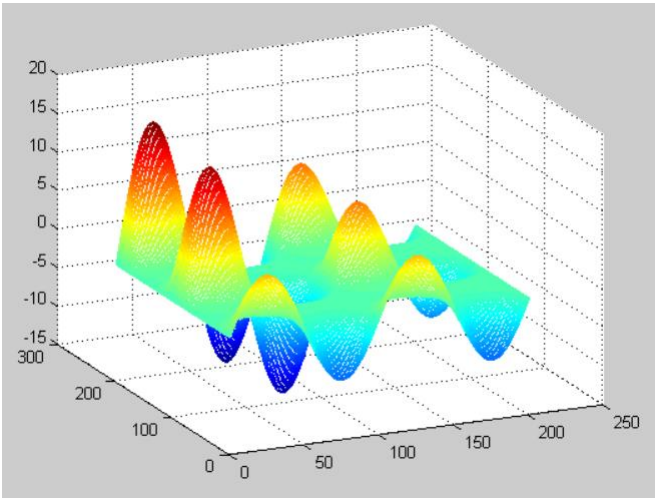
الگوریتمهای تکاملی (الگوریتم ژنتیک)

درس: کنترل هوشمند در فضای سایبرنتیک

مدرس: حمید محمودیان

بهینه سازی



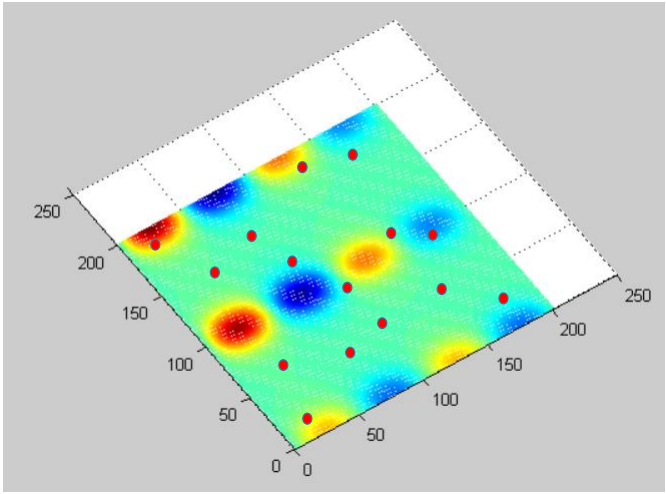
- تابع زیر را در نظر بگیرید. شکل سه بعدی این تابع در زیر رسم شده است.

$$z = e^{\sin(x)\sin(y)} e^{2\cos(2\pi x)} \sin(2\pi y)$$

سوال۱: تعیین محل های ماکزیمم و مینیمم این تابع تحلیلی چقدر ساده است؟

هدف از الگوریتمهای تکاملی مثل ژنتیک الگوریتم ارائه روشهای جستجو برای تعیین نقاط اکسترمم یک تابع است که معمولا این تابع دارای پیچیدگیهای زیادی است و یا تعداد متغیر های مستقل آن مثل (x ,y) زیاد هستند.

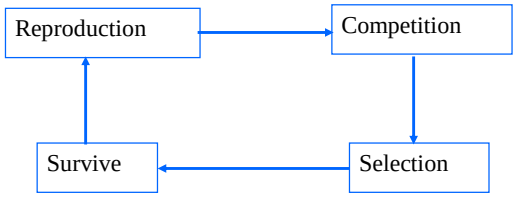
روش کلی الگوریتمهای تکاملی

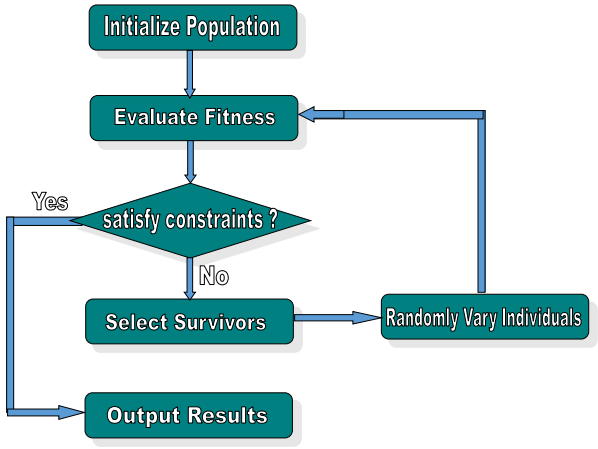


- این روشها در ابتدا معمولا یک مجموعه نقاط تصادفی را انتخاب کرده (به این نقاط جمعیت یا population) میگویند و سپس براساس روشهای جستجو که عمدتا تاثیر گرفته از طبیعت هستند به جستجوی نقاط اکسترمم (ماکزیمم یا مینیمم) تابع میپردازند.

الگوریتم ژنتیک

- الگوریتم ژنتیک براساس جستجو درتعداد مراحل از قبل تعیین شده (موسوم به نسل) تا رسیدن به پاسخی که شرایط را برآورده میکند انجام میشود.

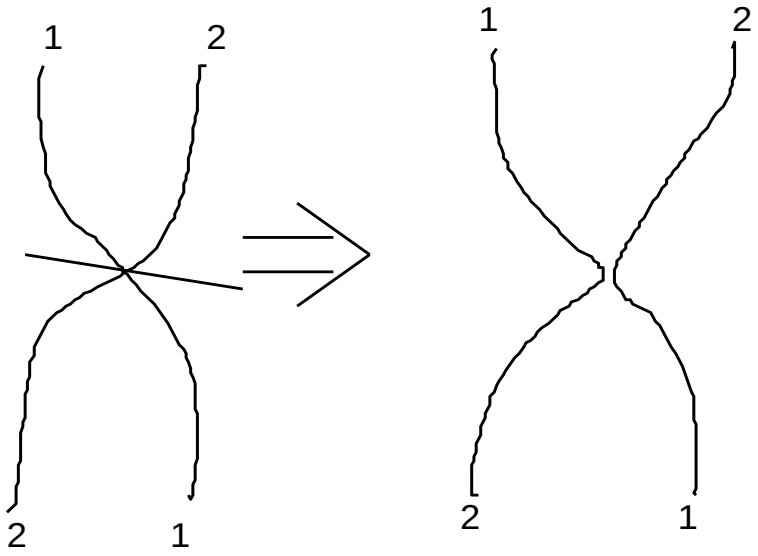




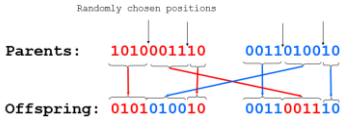
اصطلاحات مرسوم در الگوریتم ژنتیک

- ژن(Gene) : مشخصات یک عضو جامعه براساس یک مجموعه اطلاعاتی است که به آنها ژن میگوییم.
- کروموزوم (Chromosome) : ژنها در دنباله هایی قرار دارند که به آنها کروموزوم میگوییم.
- ژنوتایپ (Genotype) : به مجموعه کروموزومها ژنوتایپ میگوییم.
- فنوتایپ (Phenotype) : به همه اطلاعاتی که در ژنوتایپ ها بوده و برای ایجاد یک موجود زنده مورد نیاز است فنوتایپ میگوئیم.
- تولید مثل(Reproduction) : به فرایند ترکیب کروموزمهای ژنوتایپ و ایجاد نسل جدید تولید مثل میگوییم.
- تابع هزینه (Cost Function) : به تابعی که در آن معیار زنده ماندن هر موجود (نزدیک بودن به خواسته های ما) مشخص میشود تابع هزینه میگوییم.
- تقاطع (crossover): به جابجایی متقاطع قسمتی از ژنها در تولید مثل تقاطع میگوییم.
- جهش ژنی (Mutation): به تغییر ناگهانی توالی مولکولی DNA ها در یک ژن جهش ژنی میگوییم.

کروموزومها و پدیده تقاطع

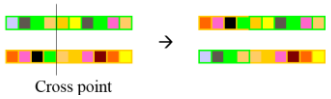


مفهوم تقاطع و جهش در الگوریتم ژنتیک



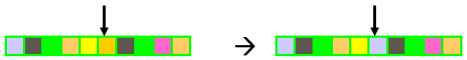
- چنانچه فرض کنیم که از یک زوج پدر و مادرمتناسب با نوع تقاطع ژنی (در الگوریتم ژنتیک هر ژن با یک رشته بیت ۰ و ۱ شناخته میشود مثل 001101011101) میتوان نسل بعدی (Next Generation) را به تعداد زیاد ایجاد نمود (تولید مثل)، آنگاه روشهای مختلف تقاطع بنی (Crossover) وجود دارد:

- Single point crossover



- Two point crossover (Multi point crossover)

- Mutation
  - Generating new offspring from single parent



0010101000010110 → 0010101010010110

مفهوم جهش در یک رشته بیت تغییر ناگهانی یک بیت از صفر به یک یا برعکس است که باعث ایجاد یک عضو از نسل بعدی میشود. در این حالت تولید مثل احتیاج به زوج والدین نداشته و بصورت تک والدینی انجام میشود

روند کلی الگوریتم ژنتیک

۱. ابتدا به تعداد کافی عضو جامعه (اعداد تصادفی) ایجاد میگردد (نسل اول)
۲. تابع هزینه برای هر عضو محاسبه شده و تعدادی (حداقل دو عضو به عنوان پدر و مادر) از بهترین اعضا که دارای کمترین هزینه هستند (شانس زیاد برای زنده ماندن دارند) نگهداشته شده و بقیه نابود میگردند.
۳. از اعضای باقیمانده (والدین) براساس تقاطع کروموزومی و جهش ژنی و یا هردو به تعداد کافی عضو ایجاد میگردد (نسل بعدی)
۴. تابع هزینه مجددا محاسبه میشود (وجود والدین در ایجاد نسل تضمین میکند که نسل دوم از نسل قبلی پاسخ بدتری نخواهد داد)
۵. شرایط اتمام الگوریتم بررسی میشود (رسیدن به پاسخ مناسب و یا محدودیت روی تعداد نسلها)
۶. اگر شرایط برآورده گردد پایان الگوریتم اعلام شده و نتایج بهینه سازی مشخص میگردد و در غیر اینصورت به قدم دوم بازگشت میشود.

مثال

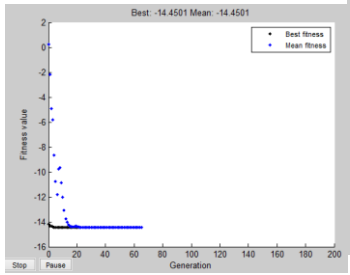
- تعیین نقطه مینیمم تابع $z = e^{\sin(x)\sin(y)}e^{2\cos(2\pi x)}\sin(2\pi y)$ در متلب:

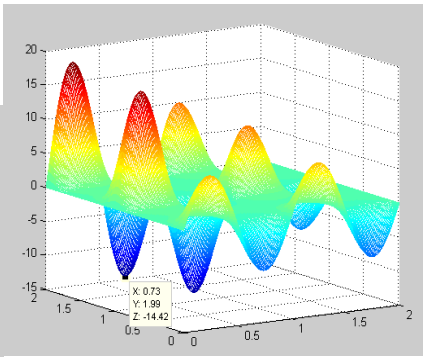
برنامه اصلی

```
clear all
lower_b=[0;0];
upper_b=[2;2];
options=gaoptimset('PopulationType','doubleVector','generations',200,'populationsize',1000,'PlotFcns',@gaplotbestf);
[val,J]=ga(@(param_vec)cost_fun(param_vec),2,[],[],[],[],lower_b,upper_b,[],options);
```

برنامه تابع هزینه

```
function [J,val]=cost_fun(param_vec)
x=param_vec(1);
y=param_vec(2);
J=exp(sin(x)*cos(y))*(sin(2*pi*y)*exp((((2*cos(2*pi*x))))));
val=[x;y];
```





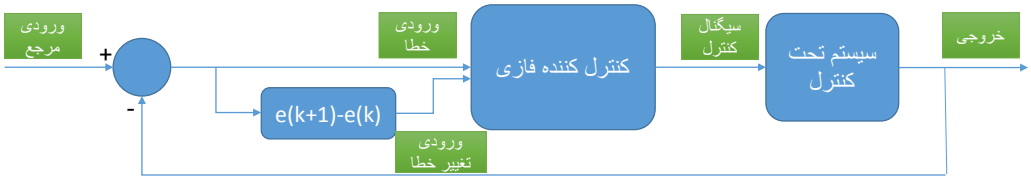
پاسخ به دست آمده

```
>> val

val =

   1.9961    0.7346
```

- مثال کنترلی (مثال جلسه پنجم): فرض کنید که کنترل کننده معرفی شده برای سیستم با معادلات حالت زیر طراحی شده باشد:

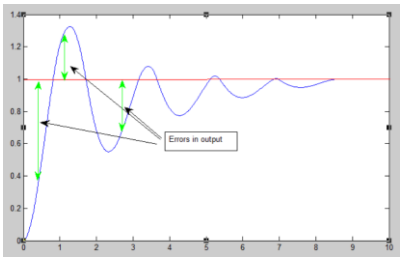
$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$
$$y = [1 \quad 0]x$$

فرض کنید که لازم است خروجی ورودی پله را تعقیب نماید:



- در مثال قبل پارامترهای قوانین فازی در Then Part به صورت تصادفی انتخاب گردید.
- R1: if e is Negative and de is negative then u1=2+3e+1.5de
- R2: if e is zero and de is zero then u2=0.05+0.2 e+ 0.05 de
- R3: if e is negative and de is positive then u3=-1+2e+3.1de

منظور از این پارامترها اعداد ۲، ۳، ۱/۵ در قانون اول و دیگر پارامتر ها در قوانین بعدی میباشد.



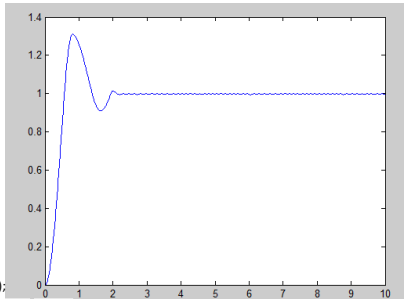
صورت مسئله: همینطور که مشاهده میشود برای مسئله فوق که تنها سه قاعده در نظر گرفته شده است، ۹ پارامتر وجود دارند که مقادیر آنها نقش مهمی در خروجی سیستم دارند. سوال: ایا میتوان با استفاده از ژنتیک الگوریتم این مقادیر را به نحوی به دست آورد که مجموع مربعات خطا مینیمم شود؟

طراحی مسئله

۱. تعداد پارامترهای آزاد مسئله مشخص شود (دراین مسئله ۹ پارامتر آزاد وجود دارد)
۲. حد بالا و پائین هر پارامتر مشخص شود (مثلا بین ۵- تا ۵+)
۳. تابع هزینه که قرار است مینیمم شود مشخص گردد (مثلا مجموع مربعات خطا در نقاط گسسته نمودار خروجی تا مقدار مرجع با رابطه $J = \sum_{i=1}^{n} e_i^2$)
۴. برنامه متلب به صورت فانکشن که ورودی آن پارامترهای آزاد (۹ پارامتر) و خروجی آن تابع هزینه مورد نظر باشد نوشته شود.
۵. برنامه اصلی متلب که در آن مشخصات الگوریتم ژنتیک مشخص شده و فانکشن تابع هزینه را صدا کند نوشته شود.
۶. برنامه اصلی اجرا گردد.

نتایج با بهینه سازی با ژنتیک الگوریتم

```
clear all
lower_b=-5*ones(9,1);
upper_b=5*ones(9,1);
options=gaoptimset('PopulationType','doubleVector','generations',100,'populationsize',100,'PlotFcns',@gaplotbestf);
[val,J]=ga(@(param_vec)cost_fun_tsk(param_vec),9,[],[],[],[],lower_b,upper_b,[],options);
%%%%%%%%%%%%%%%%%%%%%%%%%555
k=val;
save k k
i=0;
x(:,1)=[0;0];
y(1)=0;
dt=0.01;

for t=0:dt:10;
   i=i+1;
   y_reference=1;
   error(i)=y_reference-y(i);
   if i==1
      ch_error=0;
   else
      ch_error(i)=error(i)-error(i-1);
   end
   %%%% first rule
   mu1=trimf(error(i),[0 1 1])*trimf(ch_error(i),[-1  0  1]);
   %%%%%%% 2th rule
   mu2=trimf(error(i),[-1 0 1])*trimf(ch_error(i),[-1 -1 0]);
   %%%%%%%5553th rule
   mu3=trimf(error(i),[-1 -1 0])*trapmf(ch_error(i),[0 1 2 2]);

   %%%%%%%%%% controller output
   ud=(mu1*(k(1)+k(2)*error(i)+k(3)*ch_error(i))+mu2*(k(4)+k(5)*error(i)+k(6)*ch_error(i))+mu3*(k(7)+k(8)*error(i)+k(9)*ch_error(i)));
   if (mu1+mu2+mu3)==0
   u(i)=ud/(mu1+mu2+mu3+0.0001);
   else
      u(i)=ud/(mu1+mu2+mu3);
   end
   x(:,i+1)=([0 1;-2 -1]*x(:,i)+[0;1]*u(i))*dt+x(:,i);
   y(i+1)=[ 1 0]*x(:,i+1);
end
t=0:dt:10;
plot(t,y(1:1001))
```



مقادیر پارامتر های به دست آمده

k =

3.4341 4.2715 3.8155 -4.3344 1.0228 -1.6875 -0.9909 -4.5391 2.9706

```
function [J,val]=cost_fun_tsk(param_vec)
k=param_vec;
i=0;
x(:,1)=[0;0];
y(1)=0;
dt=0.01;

for t=0:dt:10;
   i=i+1;
   y_reference=1;
   error(i)=y_reference-y(i);
   if i==1
      ch_error=0;
   else
      ch_error(i)=error(i)-error(i-1);
   end
   %%%% first rule
   mu1=trimf(error(i),[0 1 1])*trimf(ch_error(i),[-1  0  1]);
   %%%%%%% 2th rule
   mu2=trimf(error(i),[-1 0 1])*trimf(ch_error(i),[-1 -1 0]);
   %%%%%%%5553th rule
   mu3=trimf(error(i),[-1 -1 0])*trapmf(ch_error(i),[0 1 2 2]);

   %%%%%%%%%% controller output
   ud=(mu1*(k(1)+k(2)*error(i)+k(3)*ch_error(i))+mu2*(k(4)+k(5)*error(i)+k(6)*
....ch_error(i))+mu3*(k(7)+k(8)*error(i)+k(9)*ch_error(i)));
   if (mu1+mu2+mu3)==0
   u(i)=ud/(mu1+mu2+mu3+0.0001);
   else
      u(i)=ud/(mu1+mu2+mu3);
   end
   x(:,i+1)=([0 1;-2 -1]*x(:,i)+[0;1]*u(i))*dt+x(:,i);
   y(i+1)=[ 1 0]*x(:,i+1);
end
J=sum(error.*error);

val=k;
```