عنوان پروژه:

الگوی سوئیچینگ بهینه برای مبدل PWM AC-AC با استفاده از الگوریتم کلونی زنبورها

مقدمه:

در این تحقیق روشی جهت مدولاسیون پهنای پالس مبدلهای AC-AC (PWM) با استفاده از الگوریتم بهینه سازی کلونی زنبورها (BCO)، ارائه می شود. زاویه‌ی تغییرِ (سوئیچینگ) بصورت بهینه با هدف به حداقل رساندن انحراف های هارمونیک توسط الگوریتم BCO بدست می آید. نتایج شبیه سازی نشان می دهد که BCO می‌تواند در زمینه‌های کیفیت توان و اصلاح فاکتورهای توان راه حل بهتری را نسبت به سایر روش‌ها، ارائه کند.

از آنجا که مبدل های AC-AC به یکی از اجزای اصلی تجهیزات کنترلی نظیر گرمایش، روشن سازی و کنترل‌های سرعت موتور، تبدیل شده اند، لذا نیازمند یافتن تکنیک‌هایی جهت بهبود بازده انرژی و تصحیح ضریب توان هستیم. یکی از متداول ترین تکنیک های حذف هارمونیک این است که شکل موج در دامنه‌ی فرکانس با استفاده از رویکردهای فوریر، آنالیز شود. علاوه بر آن، تکنیک نیوتن- رافسون برای حل کردن معادلات غیر خطی با محاسبات تکرار شونده، اتخاذ شده است.

بهینه سازی فرآیندی است که جهت بهبود پاسخ از آن استفاده می شود. بهینه سازی از یافتن بهترین پاسخ برای یک مساله صحبت می کند. لفظ بهترین به طور ضمنی بیان می کند که بیش از یک پاسخ برای مساله مورد نظر وجود دارد، که البته تمام پاسخ ها دارای ارزش یکسان نیستند. بهینه سازی شامل دو عنصر می نیمم و ماکزیمم می باشد. زمانی می گوییم عنصری می نیمم (یا ماکزیمم) است که مقادیر بدست آمده برای آن، یک سری شرایط را ارضاء کنند. فرآیند بهینه سازی، فرآیندی تکراری است که تا زمانی که شرایط خاتمه برقرار نشده باشد ادامه می یابد. شرایط خاتمه می توانند تعداد حلقه تکرار یا رسیدن به پاسخ دلخواه باشند. دو روش جستجوی مجزا جهت حل مسائل بهینه سازی وجود دارد که عبارتند از: جستجوی محلی[1] و جستجوی سراسری[2] . در روش جستجوی محلی، الگوریتم بهینه ترین پاسخ را از بین پاسخ های بدست آمده جستجو می کند، حال آنکه در روش جستجوی سراسری الگوریتم در کل فضای جستجو به دنبال پاسخ های بهینه می گردد. یکی از الگوریتم هایی که جزء دسته جستجوی سراسری می باشد، الگوریتم کلونی زنبورهاست. در این مقاله از الگوریتم زنبورها جهت یافتن زوایای بهینه با هدف کاهش حداکثریِ هارمونیک ها استفاده می کنیم.

بخش اول: الگوریتم بهینه سازی زنبور عسل

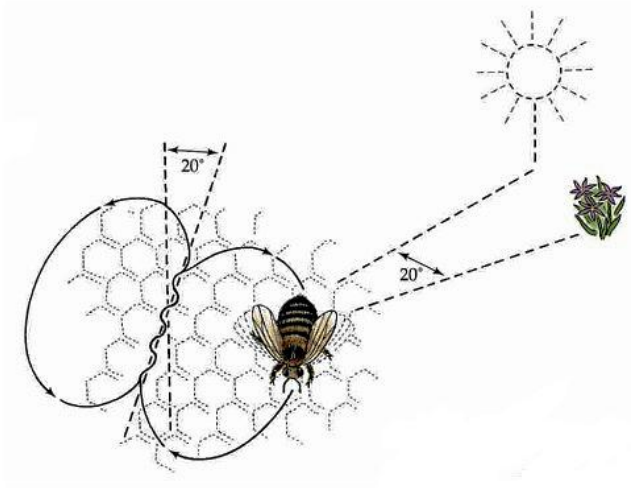
مطالعات بسیاری تا کنون در زمینه رفتارهای زنبورهای عسل صورت گرفته است، و لذا عضوی نسبتا جدید در خانواده الگوریتم های تکاملی به نام الگوریتم زنبورها (کلونی زنبورها)، که از بازسازی بخشی از رفتار و طبیعت زنبورهای عسل و اضافه کردن ویژگی هایی به آن بدست آمده است. این الگوریتم به طور کلی از دو رفتار زنبورها الهام گرفته است: نحوه جفت گیری و

---

[1] Local Search

[2] Global Search

نحوه کاوش غذا. یکی از مهمترین مکانیزم هایی که بر نحوه کاوش غذای زنبورهای عسل اثر می گذارد، رقص چرخشی[۳] آنها می باشد. زنبورهایی که در در امر جستجوی غذا موفق بوده اند (زنبورهای پیشاهنگ)، با انجام این رقص اطلاعات خود در زمینه میزان، جهت و فاصله سایت های غذایی را با دیگر زنبورهای کلونی به اشتراک می گذارند. مبنای نمایش جهت، زاویه نسبت به خورشید و مبنای نمایش فاصله، مدت زمان بخش چرخشی رقص می باشد. شکل (۱) نمایشی از رقص زنبورها را نشان می دهد.



شکل (۱). نحوه رقص زنبورها برای نشان دادن محل سایت غذایی

این روش، روشی بسیار موفق برای زنبورهای پیشاهنگ در هدایت زنبورهای کلونی می باشد. لذا زنبورها با سرعت و دقت بسیاری قادر به کشف سایت های غذایی و جمع آوری غذا خواهند بود. تا کنون از این الگوریتم در حل بسیاری از مسائل بهینه سازی استفاده شده است که برخی از آنها عبارتند از: آموزش شبکه های عصبی جهت تشخیص الگو، بهینه سازی سرویس اینترنت، بهینه سازی توابع پیوسته و مسائل داده کاوی. شبه کد الگوریتم زنبورها به صورت زیر می باشد.

۱. تعیین و مقداردهی اولیه اعضا (افراد) جمعیت
۲. بدست آوردن تابع هزینه برای هر فرد از جمعیت در موقعیت خود
۳. تا زمانی که شرایط توقف برقرار نشده است:
۴. انتخاب زنبورهای ممتاز (دارای بهترین تابع هزینه) بعنوان زنبورهای پیشاهنگ
۵. جستجوی سایت های غذایی توسط زنبورهای پیشاهنگ و انتخاب بهترین سایت ها
۶. هدایت زنبورهای دیگر به بهترین سایت های یافت شده
۷. بدست آوردن تابع هزینه برای افراد جمعیت و انتخاب بهترینِ آنها در هر سایت برای هدایت دیگران

---

[۳] Waggle Dance

۸. گماردن زنبورهای باقیمانده جهت جستجوی سایت های جدید

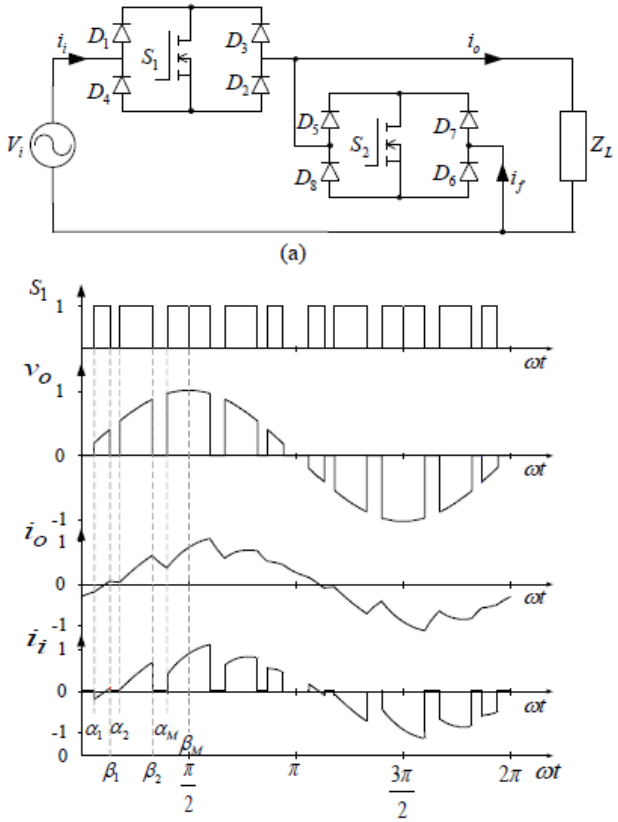
۹. پایان الگوریتم

بخش دوم: مبدل PWM AC-AC

شکل (۲) مدار یک مبدل PWM AC-AC را نشان می دهد. این مقاله از دو بخش تشکیل شده است. بخش اول با استفاده از نرم افزار matlab شبیه سازی شده و زوایای بهینه با استفادهاز الگوریتم زنبور عسل بدست می آید. در بخش دوم مدار در نرم افزار pspice شبیه سازی خواهد شده (که البته از حوزه کار این گزارش خارج می باشد). در چاپر PWM ولتاژ خروجی با استفاده از الگوی سوئیچینگ کنترل خواهد شد. بنابراین برای یک چهارم از یک دوره‌ی سینوسی، تعداد M پالس مورد نیاز خواهد بود. علاوه بر این، کلید S_1 در زاویه‌های تغییر مختلف α_1، α_2، ... α_M روشن و در زاویه‌های β_1، β_2، ... β_M خاموش می‌باشد. با استفاده از عبارت سری فوریر، ولتاژ خروجی می‌تواند به این صورت نوشته شود:

$$v_0(\omega t) = \sqrt{2}\, V_i \left(a_0 + \sum_{n=1}^{\infty}(A_n \sin(n\omega t) + B_n \cos(n\omega t))\right) \qquad (۱)$$

در اینجا، $n = 1, 2, 3, \ldots$ است.



شکل (۲). a. مدار شماتیک مبدل PWM AC-AC. b. شکل موج های ولتاژ و جریان خروجی و ورودی

با در نظر گرفتن جملات فرد رابطه (۱) خواهیم داشت:

$$v_0(\omega t) = \sqrt{2}\, V_i \sum_{n=1}^{\infty}(A_n \sin(n\omega t)) \quad (۲)$$

ضرایب اصلی A_n بصورت زیر بدست می آیند:

$$A_1 = \frac{1}{\pi_i} \sum_{k=1}^{M} \left((\beta_k - \alpha_k) - \frac{\sin 2\beta_k - \cos 2\alpha_k}{2}\right) \quad (۳)$$

$$A_n = \frac{1}{\pi} \sum_{k=1}^{M} \left(\frac{\sin((n-1)\beta_k) - \sin((n-1)\alpha_k)}{(n-1)} - \frac{\sin((n+1)\beta_k) - \sin((n+1)\alpha_k)}{(n+1)}\right) \quad (۴)$$

مساله الگوی سوئیچینگ بهینه برای PWM AC-AC بصورت تابع زیر نوشته می شود.

$$\underset{\alpha,\beta}{\text{Min}}\ F = [(A_1 - V_{0,ref})^2 + A_3^2 + A_5^2 + \dots + A_n^2]^{1/2} \quad (۵)$$

منوط به اینکه:

$$0 \le \alpha_1 \le \beta_1 \le \alpha_2 \le \beta_2 \le \dots \le \alpha_M \le \beta_M \quad (۶)$$

که در آن α_k و β_k از روابط زیر بدست می آیند.

$$\alpha_i = (\phi.(i-1)) + (\text{rand}(0,1).\phi) \quad (۷)$$

$$\beta i = \alpha i + (\text{rand}(0,1).((\phi.i - 1) - \alpha i)) \quad (۸)$$

بخش سوم: شبیه سازی و نتایج

جهت یافتن مقادیر بهینه پارامترهای الگوریتم زنبورها بصورت زیر تنظیم شده اند.

جدول (۱). نحوه رقص زنبورها برای نشان دادن محل سایت غذایی

اندازه‌ی جمعیت (N)	۲۰
تعداد نواحیِ انتخاب شده (S)	۱۴
تعداد بهترین نواحی (E)	۱۰
تعداد زنبورها پیرامون بهترین نواحی (NE)	۲۰
تعداد زنبورها پیرامون دیگر نواحی (NO)	۱۰
تعداد حلقه تکرار الگوریتم	۱۰۰

لذا خواهیم داشت:

```
function z=Cost(x)
global A1 A2 A3 Vo

alpha(1)=x(1);
alpha(2)=x(3);
alpha(3)=x(5);
beta(1)=x(2);
beta(2)=x(4);
beta(3)=x(6);
for k=1:3
    w1(k)=((beta(k)-alpha(k))-(sin(2*beta(k))-cos(2*alpha(k)))/2);
    w2(k)=(sin(beta(k))-sin(alpha(k)))-((sin(3*beta(k))-
sin(3*alpha(k)))/3);
    w3(k)=((sin(2*beta(k))-sin(2*alpha(k)))/2)-((sin(4*beta(k))-
sin(4*alpha(k)))/4);
end
A1=1/pi*(sum(w1));
A2=1/pi*(sum(w2));
A3=1/pi*(sum(w3));
F=sqrt(((A1-Vo)^2)+(A2^2)+(A3^3));
Fitness=1-F;
z=abs(Fitness);
end

function y=UniformBeeDance(x,r)
nVar=numel(x);
k=randi([1 nVar]);
y=x;
y(k)=x(k)+unifrnd(-r,r);
end


clc;
clear;
close all;
global A1 A2 A3 Vo
%% Problem Definition
CostFunction=@(x) Cost(x);
nVar=6;
VarSize=[1 nVar];
VarMin=1 ;
VarMax=90;
%% Bees Algorithm Parameters
MaxIt=100;              % Maximum Number of Iterations
nScoutBee=20;           % Number of Scout Bees
nSelectedSite=14;       % Number of Selected Sites
nEliteSite=10;          % Number of Selected Elite Sites
nSelectedSiteBee=20;    % Number of Recruited Bees for Selected Sites
nEliteSiteBee=10;       % Number of Recruited Bees for Elite Sites
r=0.1*(VarMax-VarMin);% Neighborhood Radius
rdamp=0.99;             % Neighborhood Radius Damp Rate
```

```
phi=30;
Vo=20;
%% Initialization
empty_bee.Position=[];
empty_bee.Cost=[];
bee=repmat(empty_bee,nScoutBee,1);
for j=1:nScoutBee
    for i=1:3
        alpha(i)=(phi*(i-1))+(unifrnd(0,1)*phi);
        beta(i)=alpha(i)+(unifrnd(0,1))*((phi*i-1)-alpha(i));
        bee(j).Position(i)=alpha(i);
        bee(j).Position(i+3)=beta(i);
    end
    bee(j).Cost=CostFunction(bee(j).Position);
end
[~, SortOrder]=sort([bee.Cost]);
bee=bee(SortOrder);
BestSol=bee(1);
BestCost=zeros(MaxIt,1);
%% Bees Algorithm Main Loop
for it=1:MaxIt
    for i=1:nEliteSite
        bestnewbee.Cost=inf;
        for j=1:nEliteSiteBee
            newbee.Position=UniformBeeDance(bee(i).Position,r);
            newbee.Cost=CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee=newbee;
            end
        end
        if bestnewbee.Cost<bee(i).Cost
            bee(i)=bestnewbee;
        end
    end
    for i=nEliteSite+1:nSelectedSite
        bestnewbee.Cost=inf;
        for j=1:nSelectedSiteBee
            newbee.Position=UniformBeeDance(bee(i).Position,r);
            newbee.Cost=CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee=newbee;
            end
        end
        if bestnewbee.Cost<bee(i).Cost
            bee(i)=bestnewbee;
        end
    end
    for j=nSelectedSite+1:nScoutBee
        for i=1:3
            alpha(i)=(phi*(i-1))+(unifrnd(0,1)*phi);
            beta(i)=alpha(i)+(unifrnd(0,1))*((phi*i-1)-alpha(i));
            bee(j).Position(i)=alpha(i);
            bee(j).Position(i+3)=beta(i);
```

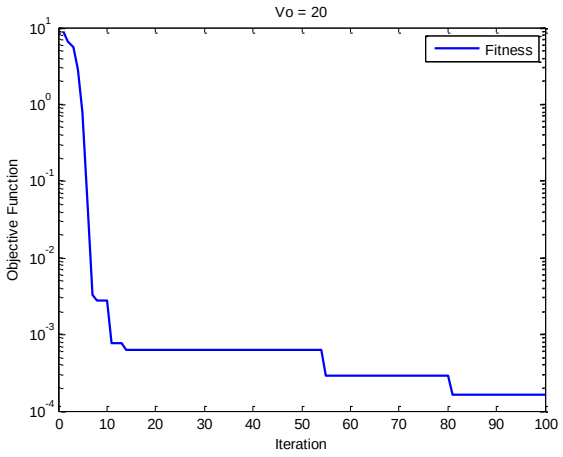
```
            end
            bee(j).Cost=CostFunction(bee(i).Position);
        end
        [~, SortOrder]=sort([bee.Cost]);
        bee=bee(SortOrder);
        BestSol=bee(1);
        BestCost(it)=BestSol.Cost;
        r=r*rdamp;
    end
    %% Results
    figure;
    semilogy(BestCost,'LineWidth',2);
    xlabel('Iteration');
    ylabel('Best Cost');
```

پس از اجرای شبیه سازی نتایج بصورت جدول زیر بدست آمده اند.

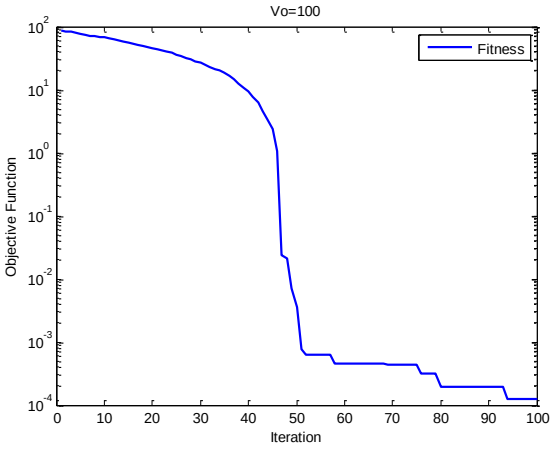
جدول (۲). زوایای بهینه بدست آمده از روش پیشنهادی با مقادیر مختلف خروجی

Output Voltage (Vo)	Optimal Switching Angle (Degree)						
	α_1	β_1	α_2	β_2	α_3	β_3	Fitness
20	26.7683	27.4161	58.9893	58.9927	86.9641	81.7961	8.1225e-06
30	25.1318	27.8199	57.5711	41.2003	86.2529	86.4729	1.1569e-04
40	24.9546	14.5043	57.4900	54.9356	83.4742	81.9226	9.3391e-05
50	23.6057	29.2080	56.8405	47.4105	82.9041	86.4759	5.5451e-05
60	21.2183	20.3566	54.7753	46.3652	81.4560	86.5878	2.4702e-04
70	21.0264	29.0356	53.9952	51.5518	78.9125	83.8345	3.7020e-04
80	20.0745	7.46561	52.1266	51.2668	76.6846	69.1856	7.4176e-05
90	18.9655	8.7884	50.9578	59.5833	76.5511	89.1584	3.4069e-04
100	16.6330	15.09645	49.5040	43.3459	73.6456	69.5836	1.8748e-04
110	15.5146	20.65984	49.3094	59.3599	73.2100	88.5668	1.1402e-04
120	15.0249	18.6849	46.2354	46.3268	71.0359	79.2684	2.3398e-04
130	14.5684	17.9852	45.2168	45.2203	70.2236	80.5659	1.5822e-03
140	13.6846	21.6916	43.8988	48.2688	70.0001	84.2260	5.6695e-02
150	12.9878	24.9950	41.0029	49.6548	69.3694	79.2841	2.3392e-01
160	11.0985	17.5468	39.9981	54.2236	67.2684	86.2359	1.8102
170	10.6002	18.0165	39.2994	51.2026	66.9982	88.2658	6.1853
180	7.3549	22.0348	37.2354	52.3321	65.2356	86.3219	20.1316
190	6.8486	15.6565	36.2468	51.0023	64.2977	85.2269	30.6293
200	4.7999	19.8485	33.5488	59.2351	62.0158	84.0031	42.1297

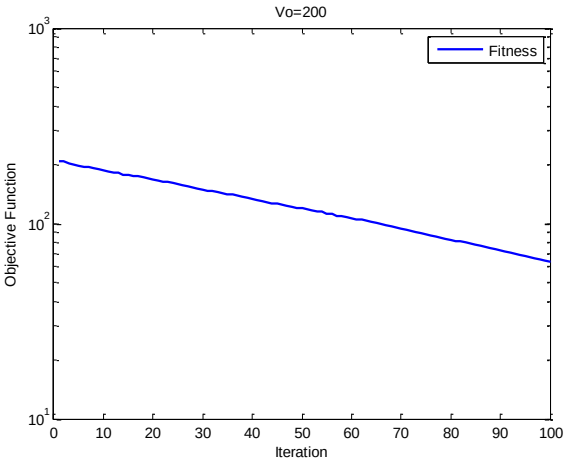
روند تغییر تابع هدف برای مقادیر خروجی ۲۰، ۱۰۰ و ۲۰۰ بصورت زیر می باشد.



شکل (۳). نحوه تغییرات تابع هدف برای Vo=20



شکل (۴). نحوه نحوه تغییرات تابع هدف برای Vo=100



شکل (۵). نحوه نحوه تغییرات تابع هدف برای Vo=200