فصل دوم

# الگوریتم اجتماع ذرات

## ۲-۱ مقدمه

الگوریتم بهینه‌سازی اجتماع ذرات[1] (PSO) که به الگوریتم اجتماع ذرات مشهور شده است یک روش جستجوی ابتکاری است که مکانیزم آن از رفتارهای گروهی جمعیت‌های زیستی الهام می‌گیرد. در مکانیزم PSO، در هر گام یک مجموعه از نقاط (جمعیت) به سمت مجموعه‌ی دیگری از نقاط، حرکت می‌کنند که احتمالاً در هر گام، بهبود کلی در نتایج حاصل می‌شود. در این فصل ابتدا به معرفی و تشریح الگوریتم PSO پرداخته می‌شود و در ادامه استفاده از این الگوریتم و روش‌های بهینه‌یافته‌ی آن را برای حل مسائل مهندسی بررسی می‌کنیم.

## ۲-۲ هوش جمعی

هوش جمعی[2] (SI) براساس رفتار اجتماعی سیستم‌های خود سامانده بنا شده است. سیستم‌های SI، معمولاً از جمعیتی از افراد تشکیل می‌شود که بطور محلی با یکدیگر و با محیط اطرافشان اثر متقابل دارند. این افراد از قوانین ساده‌ای پیروی می‌کنند و یک قانون متمرکز برای بیان رفتار آنها وجود ندارد. تاثیرگذاری افراد بصورت محلی بر روی هم، سبب ایجاد تعامل بین خودشان می‌شود که در نهایت، منجر به ایجاد تعامل سراسری بین همه‌ی افراد می‌شود. به عبارت دیگر در رفتار جمعی[3]، افراد برای رسیدن به یک هدف نهایی، با یکدیگر همکاری می‌کنند. بدیهی است که این روش نسبت به حالتی که افراد بصورت جداگانه عمل می‌کنند، موثرتر می‌باشد.

هوش جمعی از رفتار موجوداتی مانند دسته‌های ماهیان و دسته‌های پرندگان الگوبرداری می‌شود. در این نوع از اجتماعات، هر یک از موجودات ساختار نسبتاً ساده‌ای دارند، ولی رفتار جمعی آنها پیچیده است. به عبارتی دیگر یک رابطه‌ی بسیار پیچیده بین رفتار جمعی و رفتار فردی در یک اجتماع هوشمند وجود دارد. در این اجتماع رفتار جمعی فقط به رفتار فردی افراد آن اجتماع وابسته نیست، بلکه به چگونگی تعامل میان افراد نیز مرتبط است. تعامل بین افراد، تجربه‌ی آنها را در مورد محیط اطرافشان افزایش می‌دهد و موجب پیشرفت اجتماع می‌شود. ساختار اجتماعی سبب می‌شود تا افراد بتوانند به تبادل تجربه‌های شخصی بپردازند.

## ۲-۳ بهینه‌سازی اجتماع ذرات

الگوریتم PSO در سال 1995 توسط یک روان‌شناس اجتماعی به نام James Kennedy و یک مهندس

---

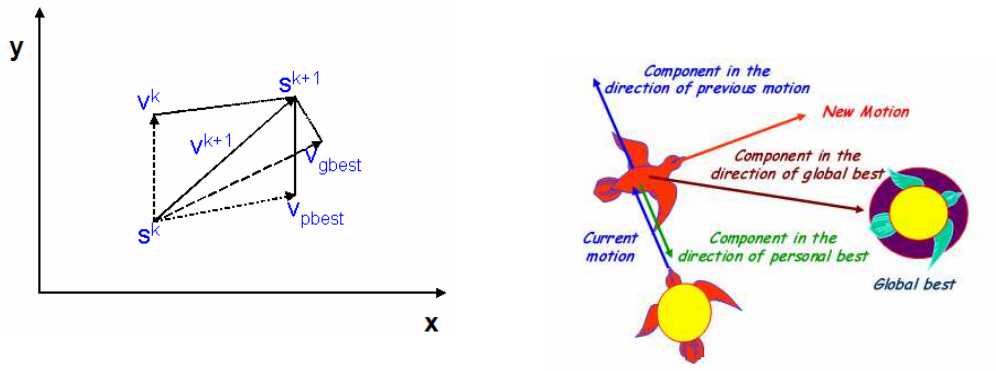
1 . Particle Swarm Optimization  2. Swarm Intelligence  3. Swarm behavior

الکترونیک به نام Russell Eberhart توسعه یافت. PSO یک تکنیک بهینه‌سازی تصادفی قوی است که بر اساس هوش گروه‌ها استوار می‌باشد. این تکنیک از رفتار دسته جمعی پرندگان یا ماهی‌ها الهام گرفته شده است (شکل ۲-۱).

فرض کنید که یک گروه از پرندگان بطور تصادفی در حال جستجوی غذا در یک ناحیه می‌باشند و تنها یک قطعه غذا در ناحیه‌ی مورد جستجو وجود دارد. هیچ یک از پرندگان اطلاع دقیقی از مکان غذا ندارند، اما می‌دانند که در هر مرحله چه فاصله‌ای از غذا دارند. بنابراین در اینجا مسأله‌ی مورد نظر، یافتن بهترین استراتژی برای پیداکردن غذا می‌باشد. یک روش موثر، دنبال کردن مسیر پرندگانی می‌باشد که نزدیکترین فاصله را به غذا دارند. PSO از این واقعیت، الهام گرفته و از آن برای حل مسائل بهینه‌سازی استفاده می‌کند.

در PSO هر راه‌حل واحد، معادل با یک پرنده در فضای جستجو می‌باشد و به عبارت دیگر هر راه‌حل واحد، ذره نامیده می‌شود. همچنین ذرات دارای سرعت‌هایی در فضای مسأله می‌باشند که با پیروی از ذرات بهینه‌ی کنونی بدست می‌آیند. الگوریتم PSO با یک گروه از ذرات (افراد) تصادفی که دارای مکان و سرعت صفر می‌باشند، آغاز می‌شود و سپس با اصلاح این ذرات با استفاده از روابط مختلف به جستجوی نقطه یا نقاط بهینه می‌پردازد. همانطور که در شکل ۲-۲ نشان داده شده است در هر مرحله، هر ذره با دنبال کردن دو مقدار بهینه به روزرسانی می‌شود. اولین مقدار بهینه، بهترین راه-حلی یا بهترین مقدار تابع برازش است که تاکنون به وسیله‌ی هر ذره بدست آمده است. این مقدار را

**شکل ۲-۱** رفتار دسته جمعی گروهی از پرندگان.




**شکل ۲-۲** مفهوم اصلاح یک نقطه جستجو توسط PSO.

بهترینِ شخص[1] می‌نامیم، که برای استفاده در مراحل بعدی ذخیره می‌شود. مقدار بهینه‌ی دوم، بهترین مقدار در میان همه ذرات می‌باشد که بهینه‌ی سراسری[2] نامیده می‌شود. اگر برای یک ذره بخشی از جمعیت را به عنوان همسایه در نظر بگیریم، به بهترین مقدار در آن همسایگی، بهترین مقدار محلی[3] گفته می‌شود. بعد از یافتن این دو مقدار بهینه، ذرات بر طبق این مقادیر بهینه، سرعت و مکانشان را به روزرسانی می‌کنند. هر ذره تلاش می‌کند تا مکانش را با استفاده از اطلاعاتی از قبیل: مکان‌های کنونی، سرعت‌های کنونی، فاصله‌ی میان مکان کنونی و بهترین شخص، فاصله‌ی میان مکان کنونی و بهینه‌ی سراسری اصلاح کند. اصلاح شدن سرعت و مکان ذرات را می‌توان با روابط ۲-۱ تا ۲-۳ بیان کرد.

$$v_i^{k+1} = w.v_i^k + c_1.rand_1(...).x(pbest_i - s_i^k) + c_2.rand_2(...).x(gbest - s_i^k) \qquad (۲-۱)$$

$$w = wMax - \frac{[(wMax - wMin)\times iter]}{Max\ \ iter} \qquad (۲-۲)$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \qquad (۲-۳)$$

$v_i^k$ سرعت ذره‌ی iام در kامین مرحله‌ی تکرار، rand یک عدد تصادفی و با توزیع یکنواخت در بازه‌ی (۱و۰)، $s_i^k$ مکان کنونی ذره iام در k امین تکرار، pbest برای iامین ذره و gbest مربوط به گروه می‌باشد. $C_1$ و $C_2$ فاکتورهای وزن دهی بوده و معمولاً برابر با عدد ثابتی انتخاب می‌شوند. w

1 . personal best (Pbest) 2. global best (gbest) 3. local best (lbest)

تابع وزن دهی می‌باشد و می‌تواند به صورت رابطه‌ی ۲-۲ تغییر کند که در آن، wMax وزن اوّلیه، wMin وزن نهایی، Max iter ماکزیمم تعداد تکرارها (مراحل) و iter تعداد تکرارها تا لحظه‌ی کنونی (تا این مرحله) می‌باشد. می‌توان فلوچارت الگوریتم PSO را به صورت شکل ۲-۳ نشان داد.

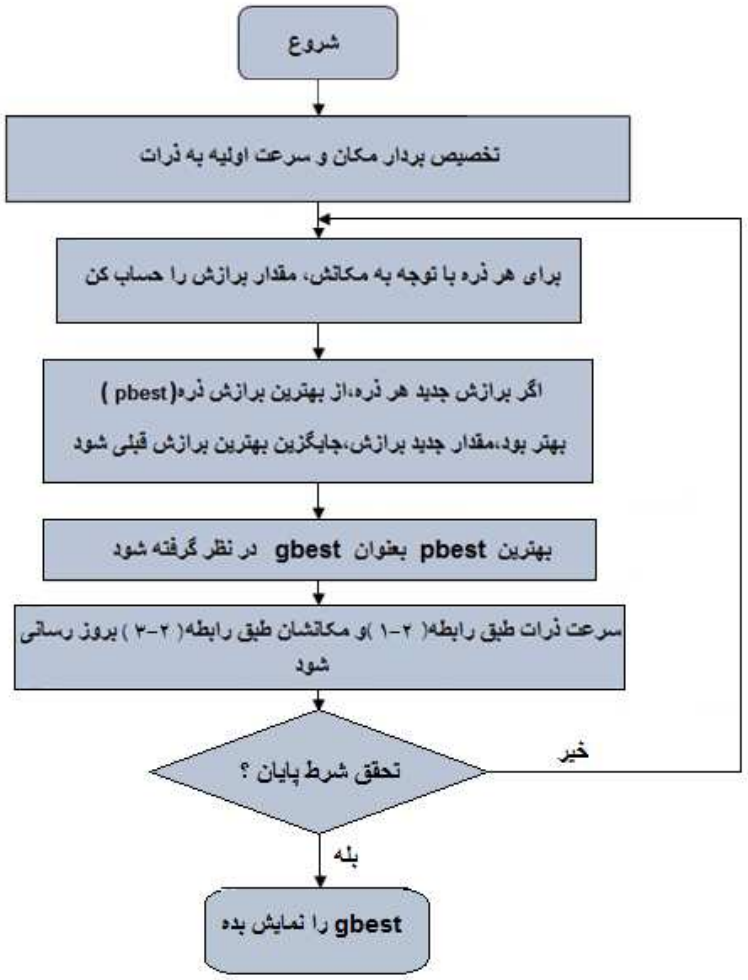
در الگوریتم PSO معمولاً هنگامی که w را به صورت خطی از یک مقدار نسبتاً بزرگ به یک مقدار کوچک کاهش می‌دهیم، الگوریتم PSO نسبت به حالتی که w مقداری ثابت است، عملکرد بهتری دارد. درحالتی که w بزرگ‌تر است، توانایی جستجوی سراسری بیشتر و در حالت w کوچک-تر، توانایی جستجوی محلی بیشتر است. معمولاً سرعت ذرات دارای یک مقدار ماکزیمم ($v_{max}$) می‌باشد. در رابطه‌ی بروزرسانی مکان (رابطه‌ی ۲-۳)، $v_i^{k+1}$ سرعت جدید ذره‌ی iام و $S_i^{k+1}$ مکان جدید ذره‌ی iام، در مرحله‌ی k+1 ام را بیان می‌کنند. w در طی اجرای الگوریتم مطابق شکل ۲-۴ کاهش می‌یابد.

دو نسخه از الگوریتم اجتماع ذرات معرفی شد; نسخه‌ی سراسری و محلی. نسخه‌ی سراسری سریع‌تر است ولی ممکن است در بعضی مسائل به مقادیر بهینه‌ی محلی همگرا شود. نسخه محلی مقدار اندکی کندتر می‌باشد ولی به آسانی در نقاط بهینه محلی به دام نمی افتد. می‌توان از نسخه سراسری برای بدست آوردن نتایج سریع و از نسخه محلی برای تصحیح نتایج جستجو استفاده کرد.

## ۲-۴ نحوه‌ی تنظیم پارامترهای الگوریتم اجتماع ذرات

همانند بسیاری از الگوریتم‌های بهینه‌سازی دیگر، بعضی از پارامترهای PSO باید توسط کاربر تنظیم شوند. یکی از آنها تعداد ذرات گروه می‌باشد. در الگوریتم بهینه‌سازی اجتماع ذرات، می‌بایستی به نحوه‌ی نمایش راه‌حل، تابع برازش و پارامترهای الگوریتم توجه داشت. یکی از مزایای الگوریتم اجتماع ذرات این است که در این الگوریتم می‌توان تعداد ذرات را یک عدد حقیقی در نظر گرفت. فرض می‌کنیم که بخواهیم تابع $f(x)=x_1^2+x_2^2+x_3^2$ را در محدوده‌های تعریف شده برای $x_1, x_2, x_3$ کمینه کنیم. می‌توان هر ذره را به صورت $(x_1, x_2, x_3)$ در نظر گرفت و تابع برازش را همان $f(x)$ انتخاب کرد. مشابه الگوریتم GA، در فرایند تکراری PSO، پارامترهای مختلفی باید تنظیم شود تا مقدار بهینه را بدست آوریم. مقادیر معمول برای برخی از پارامترهای مهم به شرح ذیل آورده شده است:

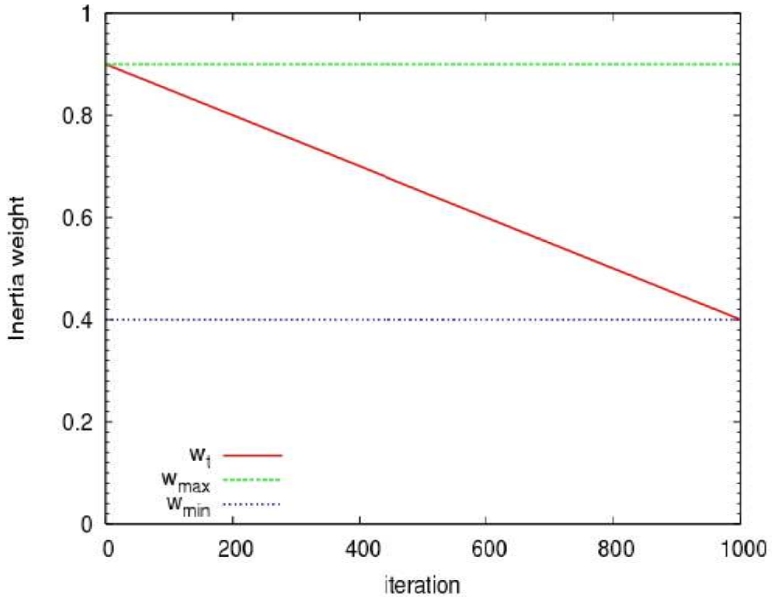
- **تعداد ذرات**: Kennedy و Eberhart متوجه شدند که PSO نسبت به الگوریتم‌های بهینه‌سازی دیگر به جمعیت‌های کوچکتر به خوبی پاسخ می‌دهد. در بیشتر پیاده‌سازی‌های PSO، از تعداد جمعیت اوّلیه برابر 20 استفاده شده است. تعداد معمول ذرات در محدوده‌ی ۱۰ الی ۴۰ می‌باشد. برای بعضی مسائل خاص تعداد ذرات به ۱۰۰ یا حتی ۲۰۰ افزایش می‌یابد.
- **محدوده‌ی ذرات**: این پارامتر توسط مسأله‌ای که باید بهینه‌سازی شود تعیین می‌گردد. می‌توان محدوده‌های مختلف مخصوصی برای ابعاد متفاوت ذرات در نظر گرفت. بطور مثال برای مسأله‌ی فوق




**شکل ۲-۳** فلوچارت الگوریتم PSO.

محدوده‌ی ذرات $(x_1, x_2, x_3)$ را می‌توان به صورت $[-10,10]$ انتخاب کرد.

- **ماکزیمم سرعت:** این پارامتر توسط تغییرات ماکزیمم یک ذره که در طی یک مرحله تکرار می‌شود، تعیین می‌گردد. مثلاً محدوده‌ی $v_{max}$ برای ذره‌ی $(x_1, x_2, x_3)$، که در آن $x_i$ ها متعلق به محدوده‌ی $[-10,10]$ می‌باشند را می‌توان، به صورت $v_{max} = 20$ تعیین کرد.
- **فاکتورهای یادگیری:** $c_1$ و $c_2$ معمولاً برابر ۲ در نظر گرفته می‌شوند. اگرچه تنظیمات دیگری نیز در مقالات مختلف بکار می‌رود، ولی در حالت کلی $c_1$ مساوی با $c_2$ و در محدوده‌ی $[0,4]$ قرار می‌گیرند.
- **شرط توقف:** الگوریتم زمانی خاتمه می‌یابد که به تعداد ماکزیمم تکرارها، اجرا شود و یا خطای




**شکل ۲-۴** کاهش یکنواخت وزن سکون در حین اجرای الگوریتم.

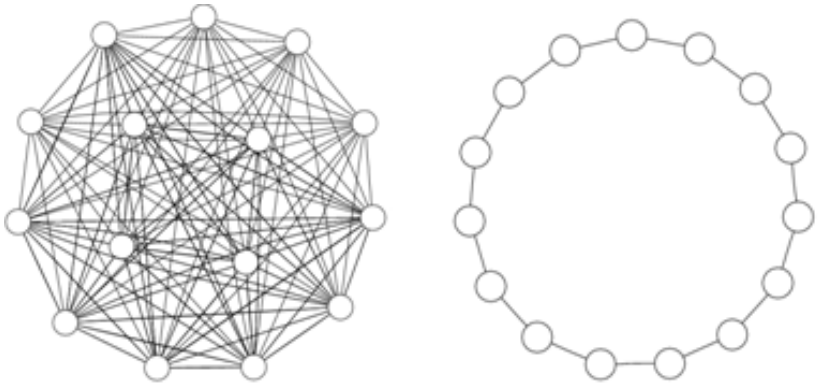
مینیمم بدست بیاید. برای نمونه برای یادگیری شبکه عصبی مصنوعی می‌توانیم مینیمم خطای مورد نیاز را یک الگوی کلاس‌بندی نشده در نظر بگیریم و تعداد ماکزیمم تکرارها به میزان ۲۰۰۰ تنظیم شود. این شرط توقف به مسأله‌ای که بهینه‌سازی می‌شود، وابسته می‌باشد.

## ۲-۵ جامعه سنجی پویا( ارتباط ذرات) در الگوریتم اجتماع ذرات

همسایگی یک ذره، مجموعه‌ای از ذرات موجود در گروه است که با یکدیگر ارتباط مستقیم دارند. شبکه‌ای که اتصال میان ذرات را برقرار می‌کند، با عنوان جامعه سنجی[1] یا توپولوژی شناخته می‌شود. با اعمال الگوریتم PSO بر روی مسائل واقعی، خودارزیابی‌های تابعی، بیشترین سهم از کل هزینه‌ی الگوریتم PSO را شامل می‌شوند. عملکرد PSO بطور کلی تحت تاثیر بُعد گروه و سنجش روابط بین افراد گروه است. در این بخش، سنجش روابط پویا مورد بررسی قرار می‌گیرد که از نظر عملکرد بسیار موثرتر از مسائل جامعه سنجی حلقه ای و ستاره‌ای خواهد بود (شکل۲-۵).

در ارتباط حلقه‌ای، هر ذره $p_i$ به $p_{i-1}$ و $p_{i+1}$ متصل است. این توپولوژی به یک جستجوی گسترده در فضای مسأله منجر می‌شود. وقتی یک ذره یک ناحیه محتمل را پیدا می‌کند، در ابتدا تنها نزدیک‌ترین همسایگی‌ها به آن ناحیه نزدیک می‌شوند و ذرات دیگر گروه در مورد آن ناحیه اطلاعی نخواهند داشت، مگر آنکه نزدیک‌ترین همسایگی آنها به آن ناحیه نزدیک شود. در ارتباط ستاره‌ای، هر

1. sociometry



**شکل ۲-۵** نمایشی از ارتباط ذرات به صورت حلقه ای و ستاره ای.

ذره به هر یک از ذرات دیگر متصل می‌باشد.

اگر یک ذره‌ی مورد بررسی، نقطه‌ی بهینه را در فضای مسأله پیدا کند، همه‌ی اعضای دیگر گروه فوراً به سمت آن سوق پیدا می‌کنند. در نتیجه گروه، همگرایی سریعی خواهد داشت. Kennedy وMendes [12] نیز توپولوژی‌هایی با قابلیت‌های اتصالات تصادفی و یک چرخ[1] ارائه داده‌اند که در آن یک ذره‌ی مرکزی به همه‌ی ذرات دیگر متصل است و اتصال دیگری وجود ندارد. همچنین از گروه‌های خوشه بندی شده نیز استفاده می‌شود که در این ارتباط، گروه به سه یا چهار زیر گروه تقسیم شده و ذرات در هر زیر گروه به هر یک از ذرات دیگر متصل می‌باشند. در این حالت اتصالات کمی میان زیر گروه‌ها وجود دارد. در همه‌ی موارد، روابط میان افراد گروه در زمان مقداردهی اولیه[2] مشخص می‌شود و در هنگام اجرای الگوریتم بدون تغییر است.

هنگامی که در حال جستجو در فضای یک مسأله بزرگ هستیم، عاقلانه است تا در ابتدا یک بررسی گسترده‌ای در فضا داشته باشیم و سپس تلاشمان را بر روی نواحی‌ای از فضا که محتمل‌تر به نظر می‌رسد متمرکز کنیم. این مسأله مفهوم جامعه‌سنجی پویا را در PSO بیان می‌کند. گروه با یک ارتباط افراد مقدار دهی اوّلیه می‌شود. هر ذره تنها به یکی از اعضای گروه‌های دیگر متصل است. با گذشت زمان خطوط اتصال بیشتر می‌شود. سرانجام اعضای شبکه بطور کامل با روش ستاره‌ای متصل می‌شوند. استراتژی در اینجا به گونه‌ای پیاده‌سازی می‌شود که گروه بطور کامل بعد از $\frac{4}{5}$ مجموعه اطلاعات ارزیابی‌های تابع اختصاص یافته، بطور کامل متصل شده و مورد استفاده قرار می‌گیرد. بعد از این زمان، اتصال جدیدی در فواصل منظم به هر ذره اضافه می‌شود. نکته‌ی قابل توجه در این روش پیاده‌سازی این است که اتصالات متقارن نیستند.

---

1. Wheel 2. Initialization

## ۲–۶ نمونه ای از شبه کد برای PSO پایه

در این قسمت شبه کدی برای الگوریتم PSO ارائه شده است. ایـن شـبه کـد کـاملاً عمـومی اسـت و می‌توان در کاربردهای مختلف از آن استفاده کرد.

```
Initialize the particle positions and their velocities //
 for I = 1 to number of particles n do
  for J = 1 to number of dimensions m do
   X[I][J] = lower limit + (upper limit - lower limit) * uniform random number
   V[I][J] = 0
  enddo
 enddo


 // Initialize the global and local fitness to the worst possible

 fitness_gbest = inf;
 for I = 1 to number of particles n do
  fitness_lbest[I] = inf
 enddo

 // Loop until convergence, in this example a finite number of iterations chosen

 for k = 1 to number of iterations t do

  // evaluate the fitness of each particle

  fitness_X = evaluate_fitness(X)

    //Update the local bests and their fitness

  for I = 1 to number of particles n do
   if (fitness_X[I] < fitness_lbest[I])

    fitness_lbest[I] = fitness_X[I]
    for J = 1 to number of dimensions m do
     X_lbest[I][J] = X[I][J]
    enddo
   endif
  enddo
   // Update the global best and its fitness

 [min_fitness, min_fitness_index] = min(fitness_X)

  if (min_fitness < fitness_gbest)

    fitness_gbest = min_fitness
    for J = 1 to number of dimensions m do
     X_gbest[J] = X(min_fitness_index,J)

    enddo
  endif

  // Update the particle velocity and position
```

```
 for I = 1 to number of particles n do
  for J = 1 to number of dimensions m do
   R1 = uniform random number
   R2 = uniform random number
   V[I][J] = w*V[I][J]
       +C1*R1*(X_lbest[I][J] - X[I][J])
       +C2*R2*(X_gbest[J] - X[I][J])
   X[I][J] = X[I][J] + V[I][J]
  enddo
 enddo

enddo
```

## ۲–۷ نمونه کد نرم افزار MATLAB برای مینیمم سازی توابع با استفاده از الگوریتم PSO

در این بخش با استفاده از الگوریتم PSO به مینیمم‌سازی تابع محک Griewank می‌پردازیم که رابطه‌ی آن به صورت رابطه‌ی ۲–۴ می‌باشد و در آن، n تعداد متغیرهای تابع است. این تابع دارای مینیمم سراسری با مقدار صفر، به ازای $x_1 = x_2 = ... = x_{30} = 0$ می‌باشد. با اجرای این نمونه کد در نرم افزار MATLAB می‌توان مشاهده کرد که الگوریتم PSO با تقریب خوبی به این جواب بهینه می‌رسد. اما نکته‌ای که باید به آن توجه کرد این است که چون در مراحل مختلف الگوریتم PSO از اعداد تصادفی استفاده شده است، با هر بار اجرای این برنامه، جوابی متفاوت با اجرای قبلی برنامه بدست می‌آید ولی در تمامی آنها جواب ارایه شده تا حد زیادی قابل قبول می‌باشد. در این نمونه برنامه، تعداد متغیرهای تابع ۱۰ در نظر گرفته شده است.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left[\frac{(x_i - 100)}{\sqrt{i}}\right] + 1 \qquad (۴–۲)$$

$$-600 \le x_i \le +600 \qquad (۵–۲)$$

کد نوشته شده در محیط نرم افزار MATLAB به این صورت است:

```
clc
clear all
close all
%************************ Initialize Population*****************************
%***************************************************************************
%constraints
low=-600;
high=600;
N=100;
n=10;
Vmax=40;
Wmin=0. 4;
Wmax=0. 9;
Iteration=80;
```