Documentation

Implemented classes :

1- class matrix

```
{
public:
double arr[5][5];
int row;
int col;} ;
```

used operator overloading to assign and sum 2 matrices

2 - class particle

```
{
public:
    matrix position1; // 5 x 2
    matrix position2; // 2 x 5
    matrix best_pos1; // 5 x 2
   matrix best_pos2; // 2 x 5
    matrix velocity1; // 5 x 2
  matrix velocity2;  // 2 x 5
};
```

Note:

I didn't put all particle's elements in one matrix. I divided them into 2 matrices and the number of elements of the particle is the same and the same goes for the velocity matrices

important functions

- particle Optimize_Particles(int dimentions):

  it contains the entire algorithm. (I advice you to take a look at it first)

- void update_velocity(particle& P):
  updated velocity matrices using the formula

$$\mathbf{v}i \leftarrow \omega\, \mathbf{v}i + \phi p\, rp\, (\mathbf{p}i - \mathbf{x}i) + \phi g\, rg\, (\mathbf{g} - \mathbf{x}i)$$

  w = 1 , rp and rg are random numbers in [0,1), ϕp = ϕg = 2

- double cost_function(matrix X1, matrix X2):

  using the square mean square error

$$\mathbf{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}.$$

- voidgenerate_random_and_assign(particle&P,string choice,int dimension):
  this function generates random values and assign it to either the velocity matrices or the position matrices in a particle.

Random values generated according to normal distribution with mean = 0 , variance = 0.1