Appendix A
Implementation in Matlab® of Differential Evolution with Particle Collision (DEwPC)

A.1 Main Function: DEwPC

```
function [Errorfinal, Eval]=DEwPC(n,z,Cscal,Ccross,
          a,b,... MaxIter,MaxIterc)
% Main function: Algorithm DEwPC
% Author: L\'{\i}dice Camps Echevarr\'{\i}a
% May-2017

% % Input / Entrada:
% n: dimension of the search space (number of
     variables)
% z: number of solutions in the population
% Cscal: Scaling Factor
% Ccross: Crossover Factor
% a,b: minimal and maximum allowed values for the
     variables
%       (It is consideredthe same cotes for each
    variable)
% MaxIter: maximum number of iterations of the main
%          loop of the algorithm
% MaxIterc: maximum number of iterations  to be
%          performed for the operator Local

% % Output /Salida
% ErrorFinal: final error achieved by the algorithm
% Eval: number of final function evaluations performed
%       by the algorithm
```


L. Camps Echevarría et al., *Fault Diagnosis Inverse Problems: Solution with Metaheuristics*, Studies in Computational Intelligence 763,
https://doi.org/10.1007/978-3-319-89978-7

```
% Choosing he function to minimize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    strfitnessfct ='f2';
    funcMin=1;

%%%%%%%%%%%%%%%%%%
% Stop Criteria
%%%%%%%%%%%%%%%%%%


   ErrorTarget=0.00000001;

 EvalMax=10000*n;

%%% DEwPC parameters and inicialization %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Initial Population matrix (X0) (1xk+1) (ACO)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      X0= a + (b-a).*rand(n,z);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %  Computing the remaining initial conditions for
         PSO
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   [Xbest, Errorfinal]=Initio(X0, z, funcMin,
     strfitnessfct);

     Xactual=X0;

     XbestTotal=zeros(n,MaxIter);

     ErrorfinalTotal=zeros(1,MaxIter);

     Eval=z;

 %%%%%%%%%%%%%%%%%%%%%
 % Main Loop of DEwPC
 %%%%%%%%%%%%%%%%%%%%%

 for i=1:MaxIter

      % Applying Operator Mutation
```

```
    Xtemp=Mutation(Xactual, z, Cscal, Xbest);

  % Applying Operator Crossover
       Xtemp=Crossover(Xactual, Xtemp, n, z, Ccross);

       for j=1:z
          FunObj(j)=feval(strfitnessfct, Xactual(:,j));
       end

      % Applying Operator Selection of DEwPC
       for j=1:z

          if j<=sqrt(z)  % Apply Selection from DE

             [Xtemp(:,j), Eval, FunObj(j)]=
                         Selection(Xtemp(:,j),...
                         Xactual(:,j),strfitnessfct,...
                         NumEval, FunObj(j), j);

          else   % Apply   Absorption-Scattering
              from PCA


         [Xtemp(:,j), Eval, FunObj(j)]=
             AbsScatt(Xtemp(:,j),...
             Xactual(:,j), Fbest, MaxIterc,a,b,...
             strfitnessfct, n, j, FunObj(j), NumEval);

          end

       end

       Xactual=Xtemp;

       [Xbest, Errorfinal]=Update(Xactual, funcMin,
          FunObj, z);

  Eval=NumEval+(i*z);

       if Eval>=NumEvalMax

          break

       end
```

```
        if Errorfinal<= ErrorTarget

            break

        end

    end
```

A.1.1 Function Initio

```
%%Function Initio
%%This function computes the best initial solution and
%%its value
%Function Name: Initio
%This function returns:
%   Xbest: the best solution of the initial population
%   Errorfinal: the error of the function value at the
%               best solution Xbest

function [Xbest, Errorfinal]=Initio(X0, z, funcMin,...
         strfitnessfct);

F=zeros(z,1);

for i=1:z
    F(i)=feval(strfitnessfct, Xactual(:,i));
end

[Fbest, I]=min(F);

Xbest=Xactual(:,I);

Errorfinal=Fbest-funcMin;
```

A.1.2 Function Mutation

```
%%Function Mutation
%%This function applies Mutation operator to each
     solution
```

```
%%
%Function Name: Mutation
%This function returns:
%   Xtemp: the solutions after the Mutation
%

function Xtemp=Mutation(Xactual, z, Cscal, Xbest)


q= floor(1 + (z).*rand(4,z));

Xtemp=Xactual;

for i=1:z

    Xtemp(:,i)=Xbest+Cscal*(Xactual(:,q(1,i))-...
               Xactual(:,q(2,i))+Xactual(:,q(3,i))-...
               Xactual(:,q(4,i)));

end
```

A.1.3 Function Crossover

```
%%Function Crossover
%%This function applies Crossover operator to each
     solution
%%
%Function Name: Crossover
%This function returns:
%   Xtemp: the solutions after the Crossover
%

function Xtemp=Crossover(Xactual, Xtemp, n, z, Ccross)

q= rand(n,z);

for i=1:z

    for j=1:n

        if q(j,i) > Ccross
```

```
            Xtemp(j,i)=Xactual(j,i);

        end

    end

end
```

A.1.4 Function Selection

```
%%Function Selection
%%This function applies Selection operator to the
     choosen
%%solutions
%Function Name: Selection
%This function returns:
%   Xtemp: the solutions after the Selection
%

function [Xgenerada, NumEval, FunObjj]=Selection
          (Xgenerada,...Xanterior,strfitnessfct,
           NumEval, FunObjj, k)

 FunObjGenerada=feval(strfitnessfct, Xgenerada);

 NumEval=NumEval+1;

 if FunObjGenerada>= FunObjj;

    Xgenerada= Xanterior;

 end
```

A.1.5 Function AbsScatt

```
%%Function AbsScatt
%%This function applies Absorption-Scattering
     operator to
%%the choosen solutions
```

```
%Function Name: AbsScatt
%This function returns:
%   Xtemp: the solutions after the Absorption-
    Scattering
%

function [Xfinal, NumEval, FunObjj]=AbsScatt
          (Xgenerada,... Xanterior, Fbest,MaxIterAbs,
           a, b, strfitnessfct,... n,  k, FunObjj,
            NumEval)

 fa=feval(strfitnessfct,Xgenerada);

 NumEval=NumEval+1

 if fa<= FunObjj

     [Xfinal, FunObjj, NumEval]=Local
                       (Xgenerada,... MaxIterAbs,
                        a, b, n,... strfitnessfct,
                         NumEval);


 else

     if rand>1-(Fbest)/(fa)


         [Xfinal, FunObjj, NumEval] =Local
                           (Xgenerada,... MaxIterAbs,
                            a, b, n,...strfitnessfct,
                             NumEval);

     else

         Xfinal=Xanterior;

     end

 end
```

A.1.6 Function Update

```
%%Function Update
%%This function updates the best values for the new
%%population
%%
%Function Name: Update
%This function returns:
%   Xbest: best solution  from the population
%   Errorfinal: error for the  function value at
%               the best solution Xbest

function [Xbest, Errorfinal]=Update(Xactual,
           funcMin,... FunObj,z)


[Fbesttemp, I]=min(FunObj);

Xbest=Xactual(:,I);

Errorfinal=FunObj(I)-funcMin;
```

A.1.7 Function Local

```
%%Function Local
%%This function makes a search arround a solution
%%
%Function Name: Local
%This function returns:
%   Xbest: best solution  from the population
%   Fbest: function value at the best solution Xbest
%   Errorfinal: error for the  function value at the
%               best solution Xbest
function [Xfinal, temp2FO, Eval]=Local(Xgenerada,...
           MaxIterc,a, b, n, strfitnessfct, NumEval)

Xfinal=Xgenerada;

XfinalTemp=Xfinal;

temp2FO=feval(strfitnessfct, Xfinal);
```

```
for i=1:MaxIterc

    for j=1:n

        q2=0.8+ (1-0.8)*rand;

        q3=1.0+ (1.2-1.0)*rand;

        if q2*Xfinal(j)>=a

            Xlower=q2*Xfinal(j);

        else
            Xlower=a;

        end

         if q3*Xfinal(j)<=b

            Xupper=q3*Xfinal(j);

        else
            Xupper=b;

        end

        q1=rand;

        XfinalTemp(j)=Xfinal(j)+(Xupper-Xfinal(j))*
                      q1-...(Xlower-Xfinal(j))*(1-q1);

    end

    temp1FO=feval(strfitnessfct, XfinalTemp);

    NumEval=NumEval+1;

    if temp1FO <=  temp2FO

        Xfinal= XfinalTemp;

        temp2FO=temp1FO;

    end
end
```

Appendix B
Implementation in Matlab® of Particle Swarm Optimization with Memory (PSO-M)

B.1 Main Function: PSOM

```
function [ErrorFinal, Eval]=PSOM(n,z1,z2,C11,C12,...
          C21,C22,a,b,Cevap,Cinc,k,qo,itr1,itr2)
% Main function: Algorithm PSO-M
% Author: L\'{\i}dice Camps Echevarr\'{\i}a
% May-2017

% % Input / Entrada:
% n: dimension of the search space (number of
     variables)
% z1: number of particles in the first stage of PSO-M
% z2: number of particles in the second stage of PSO-M
% C11,C12: cognitive and social parameter values
%          in the first stage of PSO
% C21,C22: cognitive and social parameter values in
%          the second stage of PSO-M
% a,b: minimal and maximun allowed values for the
     variables
%       (It is considered the same cotes for each
     variable)
% Cevap: evaporation factor
% Cinc: incremental factor
% k: number of possible discrete values for each
     variable
% qo: control parameter in ACO (level of randomness
%      during the ant generation
% itr1: maximum number of iterations in the first
```


L. Camps Echevarría et al., *Fault Diagnosis Inverse Problems: Solution with Metaheuristics*, Studies in Computational Intelligence 763,
https://doi.org/10.1007/978-3-319-89978-7

```
%        stage of PSO-M
% itr2: maximum number of iterations in the second
%        stage of PSO-M

% % Output /Salida
% ErrorFinal: final error achieved by the algorithm
% Eval: number of final function evaluations performed
%        by the algorithm


% Choosing he function to minimize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    strfitnessfct ='f2';
    funcMin=1;

%%%%%%%%%%%%%%%%%
% Stop Criteria
%%%%%%%%%%%%%%%%%


   ErrorTarget=0.00000001;

 EvalMax=10000*n;

%%% ACO parameters and inicialization %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Discreet Values (DV) (1xk+1) (ACO)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      DV = a:((b-a)/k):b;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Pheromone matrix  (F) (ACO)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      F = 0.50*ones(n,k+1);

%%% PSO inicialization  for the first stage %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Initial position of particles in PSO  (X) (PSO)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        X = a + (b-a).*rand(n,z1);
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Initial  pi  of particles in PSO
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      Pi = X;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Initial velocity of particles in PSO (V)
        (nxz)(PSO)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      V = 1 + (b-1).*rand(n,z1); % Save velocity value

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %  Computing the remaining initial conditions
         for PSO
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      [Pg, ErrorPg, FunObjPi]=best(X, z1, funcMin,...
                              strfitnessfct);
      ErrorFinal=ErrorPg;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First Stage of PSO-M (z1,c1,c2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for itr=1:itr1

    for i=1:z1

        % Updating position and velocity
        [V,X] = UpdateVX(n,C11,C12,X,Pi,Pg,V,itr,i,
            itr1);

    end

    % Updating Pg and Pi

    [Pg,ErrorPg,Pi,FunObjPi]=UpdatePgPi(X,FunObjPi,
                             Pi,... strfitnessfct,z1,
                               funcMin);
    if ErrorPg < ErrorFinal
        ErrorFinal=ErrorPg;
    end
```

```
    if ErrorFinal < ErrorTarget
        break
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Updating pheromone matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    Ftemp=(1-Cevap)*F;
    Ftemp1=zeros(n,k+1);
    S=zeros(n,1);

    for t=1:n

        r = find( DV >= Pg(t,:));

        tam = size(r);

        if (tam(1,2) == 0)

            S(t,:) = DV(1,k+1);
            Ftemp1(t,k+1) = Cinc*F(t,k+1);

        else

            l = DV(1,r(1,1)) - Pg(t,:);

            m = ((b-a)/k)/2;

            if (l) > (m)

                if r(1,1) == 1

                    S(t,:) = DV(1,r(1,1));
                    Ftemp1(t,r(1,1)) = Cinc*
                       F(t,r(1,1));

                else

                   S(t,:) = DV(1,(r(1,1))-1);
                   Ftemp1(t,(r(1,1)-1)) = Cinc*F(t,
                        (r(1,1)-1));

                end
```

```
                else

                    S(t,:) = DV(1,r(1,1));
                    Ftemp1(t,r(1,1)) = Cinc*F(t,r(1,1));

                end

            end

        end

       F=Ftemp+Ftemp1;


  end




  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Initial parameters for the Second Stage of PSO
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    PC = UpdatePC(F,n,k);

    S = GenerateAnts(n,z2,qo,F,PC,DV);

    X = S;

    V = 1 + (b-1).*rand(n,z2);


    [PgNo, ErrorPgNo, FunObjPi]=best(X, z2, funcMin,...
                                    strfitnessfct);



  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Second Stage of  PSO-M  (z2,c1,c2)
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  for itr=1:itr2
```

```
    for i=1:z2

        [V,X] = UpdateVX(n,C21,C22,X,Pi,Pg,V,itr,
                i,itr2);

    end
    [Pg,ErrorPg,Pi,FunObjPi]=UpdatePgPi(X,FunObjPi,
                              Pi,... strfitnessfct,z2,
                                 funcMin);
    if ErrorPg < ErrorFinal
        ErrorFinal=ErrorPg;
    end

    Eval=itr1*z1+z2*itr;

    if Eval >= EvalMax
        break
    end

    if ErrorFinal < ErrorTarget

        break

    end

end
```

B.1.1 Function Best

```
%%Function best
%%This function updates the best values for a swarm
%%
%Function Name: best
%This function returns:
%   Pg: best particle  from the swarm
%   ErrorPg: error for the  function value at
%            the best particle Pg
%   FuncObjPi: updated vector with the values of the
%              function at the Pi of each particle
%
```

```
function [Pg, ErrorPg, FuncObjPi]=best(Xactual, z,...
           fmin, strfitnessfct)

FuncObjPi=zeros(z,1);

for i=1:z
    FuncObjPi(i)=feval(strfitnessfct, Xactual(:,i));
end

[Ftemp, I]=min(FuncObjPi);

Pg=Xactual(:,I);

FuncPg= FuncObjPi(I);

ErrorPg=norm(FuncPg-fmin);
```

B.1.2 Function UpdateVX

```
%%Function UpdateVX
%%This function updates the position and velocity
%%of a particle of the swarm
%%
%Function Name: UpdateVX
%This function returns:
%  V: the velocity of a particle
%  X: the position of a particle
%

function [V,X] = UpdateVX(n,C1,C2,X,Pi,Pg,V,itr,...
                          i,iteraciones)

       w=0.9-((0.9-0.4)/iteraciones)*itr;

       V(:,i) = w*V(:,i) + C1*(diag(rand(1,n)))*
                (Pi(:,i)... -X(:,i))+C2*(diag(rand
                (1,n)))*(Pg - X(:,i));

       X(:,i) = X(:,i) + V(:,i);

end
```

B.1.3 Function UpdatePgPi

```
%%
%Function Name: UpdatePgPi
%This function returns:
%   Pg: best particle  from the swarm
%   ErrorPg: error for the  function value at the
%            best particle Pg
%   Pi: position of the ith particle
%   FuncObjPi: updated vector with the values of the
%              function at the Pi of each particle
%

function[Pg,ErrorPg,Pi,FunObjPi]=UpdatePgPi(X,
                                  FunObjPi,... Pi,
                                  strfitnessfct,z,fmin)


for k=1:z

    temp=feval(strfitnessfct, X(:,k));

    if temp<FunObjPi(k)

        FunObjPi(k)=temp;
        Pi(:,k)=X(:,k);

    end


end

[Fbesttemp, I]=min(FunObjPi);

Pg=X(:,I);

FunObjPg=FunObjPi(I);


ErrorPg=norm(FunObjPg-fmin);
```

B.1.4 Function UpdatePC

```
%%Function UpdatePC
%%This function updates Accumulative Probability
     matrix
%%
%Function Name: UpdatePC
%This function receives three parameters:
%   F: Pheromone Matrix
%   n: Number of discreet variable
%   k: Number of interval
% and returns:
%   PC: Accumulative Probability matrix
%

function [PC] = UpdatePC(F,n,k)

   PC=zeros(n,k+1);

   for i=1:n  % Loop to move through row (Variable)

        for j=1:1:k+1   % Loop to move through col
                        %(Discreet Value)

            PC(i,j) = sum(F(i,1:j))/sum(F(i,:));

        end

    end


end
```

B.1.5 Function GenerateAnts

```
%%Function GenerateAnts
%%This function generates a  population of ants
%%
%Function Name: GenerateAnts
%This function receives seven parameters:
%   n: number of discreet variable
%   k: number of interval
```

```
%   qo: randomness parameter
%   z: number of ants
%   F:pPheromone Matrix
%   DV: discreet Values
%   PC: accumulative Probability matrix
% and returns:
%   S: matrix with a new population of z ants
%

function [S] = GenerateAnts(n,z,qo,F,PC,DV)

    S = ones(n,z);

    for i=1:z % Loop to generate z ants

        for j=1:n % Loop to generate n discreet values
                  % associate with n variables for
                    each ant

            qn = rand;

            if qn < qo

                [C,I] = max(F(j,:)); e
                S(j,i) = DV(1,I);

            else

                c = find(PC(j,:) >= qn);
                S(j,i) = DV(1,c(1,1));

            end
        end
    end

end
```

References

1. Acosta Diaz, C., Camps Echevarria, L., Prieto Moreno, A., Silva Neto, A.J., Llanes-Santiago, O.: A model-based fault diagnosis in a nonlinear bioreactor using an inverse problem approach. Chem. Eng. Res. Des. **114**, 18–29 (2016)
2. Angeli, C., Chatzinikolaou, A.: On-line fault detection techniques for technical systems: a survey. Int. J. Comput. Sci. Appl. **I**(I), 22–30 (2004)
3. Angeline, P.J.: Chapter Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. In: Evolutionary Programming VII: Proceeding of the Seventh Annual Conference on Evolutionary Programming (EP98). Lecture Notes in Computer Science, vol. 1447, pp. 601–611. Springer, New York (1998)
4. Baeck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford (1996)
5. Barbosa Muniz, W., Campos Velho, H.F., Manuel Ramos, F.: A comparison of some inverse methods for estimating the initial condition of the heat equation. J. Comput. Appl. Math. **103**, 145–163 (1999)
6. Bauer, F., Hohage, T., Munk, A.: Iteratively regularized Gauss-Newton method for nonlinear inverse problems with random noise. SIAM J. Numer. Anal. **47**, 1827–1846 (2009)
7. Becceneri, J.C., Zinober, A.: Extraction of energy in a nuclear reactor. In: XXXIII Simposio Brasileiro de Pesquisa Operacional. Campos do Jordão, SP, Brazil (2001)
8. Becceneri, J.C., Sandri, S., Luz, E.F.P.: Using ant colony systems with pheromone dispersion in the traveling salesman problem. In: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence. Sant Martí d'Empúries (2008)
9. Beck, J.V.: Combined parameter and function estimation in heat transfer with application to contact conductance. J. Heat Trans. **110**(4b), 1046–1058 (1998)
10. Beielstein, T., Parsopoulos, K.E., Vrahatis, M.N.: Tuning PSO parameters through sensitivity analysis. Tech. rep., Reihe Computational Intelligence CI 124/02, Collaborative Research Center (SFB 531), Department of Computer Science and University of Dortmund (2002)
11. Beni, G., Wang, J.: Swarm intelligence. In: Seventh Annual Meeting of the Robotics Society of Japan. RSJ Press, Tokyo (1989)
12. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont, MA (1999)
13. Blum, C.: Ant colony optimization: introduction and recent trends. Phys. Life Rev. **2**(4), 353–373 (2005)
14. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, Oxford (1999)

L. Camps Echevarría et al., *Fault Diagnosis Inverse Problems: Solution with Metaheuristics*, Studies in Computational Intelligence 763,
https://doi.org/10.1007/978-3-319-89978-7

15. Brest, J., Greiner, S., Boscovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(8), 646–657 (2006)
16. Camacho, O., Padilla, D., Gouveia, J.L.: Fault diagnosis based on multivariate statistical techniques. Rev. Téc. Ing. Univ. Zulia. **30**(3), 253–262 (2007)
17. Campos Velho, H.F.: Inverse Problems in Space Research. SBMAC, Sao Carlos, Brazil (2008)
18. Camps Echevarría, L., Llanes-Santiago, O., Silva Neto, A.J.: Chapter Fault diagnosis in industrial systems using bioinspired cooperative Strategies. In: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010. Studies in Computational Intelligence, vol. 284. Springer, New York (2010)
19. Camps Echevarría, L., Llanes-Santiago, O., Silva Neto, A.J.: A proposal to fault diagnosis in industrial systems using bio-inspired strategies. Ingeniare. Revista chilena de ingeniería **19**(2), 240–252 (2011)
20. Camps Echevarría, L., Silva Neto, A.J., Llanes-Santiago, O., Hernández Fajardo, J.A., Jiménez Sánchez, D.: A variant of the particle swarm optimization for the improvement of fault diagnosis in industrial systems via faults estimation. Eng. Appl. Artif. Intell. **28**, 36–51 (2014)
21. Camps Echevarria, L., Campos Velho, H.F., Becceneri, J.C., Silva Neto, A.J., Llanes-Santiago, O.: The fault diagnosis inverse problem with ant colony optimization and ant colony optimization with dispersion. Appl. Math. Comput. **227**(15), 687–700 (2014)
22. Cao, H., Haiwen, Y., Zhao M. Shi Jian, S., Limei, Z., Xin, G.: The fault diagnosis of aircraft power system based on inverse problem of fuzzy optimization. Part G: J. Aerosp. Eng. **230**(6), 1059–1074 (2016)
23. Carlisle, A., Dozier, G.: An off-the-self PSO. In: Proceedings of the Particle Swarm Optimization Workshop, Indiana, pp. 1–6 (2001)
24. Chen, J., Patton, R.J.: Robust Model-Based Fault Diagnosis for Dynamic Systems. Kluwer Academic Publishers, Dordrecht (1999)
25. Chow, E.Y., Willsky, A.: Analytical redundancy and the design of robust failure detection systems. IEEE Trans. Autom. Control **29**, 603–614 (1984)
26. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(2), 58–73 (2002)
27. Das, S., Abraham, A., Uday, K.C., Konar, A.: Differential evolution using a neighborhood-based mutation operator. IEEE Trans. Evol. Comput. **13**(3), 526–553 (2009)
28. Dawkins, R.: The Selfish Gene. Clarendon Press, Oxford (1976)
29. de Miguel, L.J., Blázquez, L.F.: Fuzzy logic-based decision-making for fault diagnosis in a DC motor. Eng. Appl. Artif. Intell. **18**(1), 423–450 (2005)
30. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical test as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
31. Ding, S.X.: Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. Springer, Berlin (2008)
32. Dorigo, M.: Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale. Ph.D. thesis, Politécnico di Milano (1992)
33. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**(2–3), 243–278 (2005)
34. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B **26**(1), 29–41 (1996)
35. Duarte, C., Quiroga, J.: PSO algorithm for parameter identification in a DC motor. Rev. Fac. Ing. Univ. Antioquia **55**, 116–124 (2010). (In Spanish)
36. Dulmage, A., Mendelsohn, N.: Coverings of bipartite graphs. Can. J. Math. **10**(5), 517–534 (1958)
37. Eberhart, R., Shi, Y.: Chapter Comparison between Genetic Algorithms and Particle Swarm Optimization. In: Evolutionary Programming VII: Proceeding of the Seventh Annual Conference on Evolutionary Programming (EP98). Lecture Notes in Computer Science, vol. 1447, pp. 611–619. Springer, Berlin (1998)

38. Eberhart, R.C., Shi, Y.H.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceeding of the IEEE Congress on Evolutionary Computation, pp. 84–88 (2001)
39. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer Academic Publishers, Dordrecht (1996)
40. Fliess, M., Join, C., Mounier, H.: An introduction to nonlinear fault diagnosis with an application to a congested internet router. In: Advances in Communication and Control Networks. Lecture Notes in Control and Information Sciences. Springer, Berlin (2004)
41. Florin Metenidin, M., Witczak, M., Korbicz, J.: A novel genetic programming approach to nonlinear system modelling: application to the DAMADICS benchmark problem. Eng. Appl. Artif. Intell. **17**(4), 363–370 (2004)
42. Frank, P.M.: Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. Automatica **26**(3), 459–474 (1990)
43. Frank, P.M.: Analytical and qualitative model-based fault diagnosis – a survey and some new results. Eur. J. Control **2**(1), 6–28 (1996)
44. Frank, P.M., Koeppen-Selinger, B.: New developments using AI in fault diagnosis. Eng. Appl. Artif. Intell. **10**(1), 3–14 (1997)
45. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the CEC 2005 Special Session on Real Parameter Optimization. J. Heuristics **15**(6), 617–644 (2009)
46. Glover, F.: Tabu search: a tutorial. Tech. rep., Center for Applied Artificial Intelligence, University of Colorado (1990)
47. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA (1989)
48. Gomez, Y.: Two step swarm intelligence to solve the feature selection problem. J. Univ. Comput. Sci. **14**(15), 2582–2596 (2008)
49. Gong, W., Cai, Z., Ling, C.X., Li, H.: A real-coded biogeography-based optimization with mutation. Appl. Math. Comput. **216**(9), 2749–2758 (2010)
50. Gong, W., Cai, Z., Ling, C.X.: DE/BBO a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput.: Fusion Found. Methodol. Appl. Arch. **15**(4), 645–665 (2010)
51. Hadamard, J.: Sur les problèmes aux dérivées partielles et leur signification physique, pp. 49–52. Princeton University Bulletin, Princeton, NJ (1902)
52. Hart, W.E., Krasnogor, N., Smith, J.E.: Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing. Springer (2005). http://books.google.com.br/books?id=LYf7YW4DmkUC
53. Henrique Terra, M., Tinós, T.R.: Fault detection and isolation in robotic manipulators via neural networks: a comparison among three architectures for residual analysis. J. Robot. Syst. **7**(18), 367–374 (2001)
54. Hoefling, T.: Detection of parameter variations by continuous-time parity equations. In: 12th IFACWorld-Congress, pp. 511–516 (1993)
55. Hoefling, T., Isermann, R.: Fault detection based on adaptive parity equations and single-parameter tracking. Control Eng. Pract. **4**(10), 1361–1369 (1996)
56. Hohmann, H.: Automatische ueberwachung und fehlerdiagnose am werkzeugmaschinen. Ph.D. thesis, Technische Hochschule Darmstadt (1987)
57. Isermann, R.: Process fault detection based on modelling and estimation methods – a survey. Automatica **20**(4), 387–404 (1984)
58. Isermann, R.: Fault diagnosis of machines via parameter estimation and knowledge processing. Automatica **29**(4), 815–835 (1993)
59. Isermann, R.: Model based fault detection and diagnosis methods. In: American Control Conference, pp. 1605–1609 (1995)
60. lsermann, R.: Supervision, fault-detection and fault-diagnosis methods- an introduction. Control Eng. Pract. **5**(5), 639–652 (1997)

61. Isermann, R.: Model based fault detection and diagnosis. Status and applications. Annu. Rev. Control **29**(1), 71–85 (2005)
62. Isermann, R.: Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Springer, Berlin (2006)
63. Isermann, R.: Fault-Diagnosis Applications: Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-tolerant Systems. Springer, Berlin (2011)
64. Isermann, R., Ballé, P.: Trends in the application of model-based fault detection and diagnosis of technical processes. Control Eng. Pract. **5**(5), 709–719 (1997)
65. Kameyama, K.: Particle swarm optimization – a survey. IEICE Trans. Inf. Syst. **E92-D**(7), 1354–1361 (2009)
66. Kanović, Z., Rapaić, M.R., Jeliićc, Z.D.: Generalized particle swarm optimization algorithm – theoretical and empirical analysis with application in fault detection. Appl. Math. Comput. **217**(24), 10175–10186 (2011)
67. Karaboga, D., Akay, B.: A survey: algorithms simulating bee swarm intelligence. Artif. Intell. Rev. **31**(1), 61–85 (2009)
68. Keller, J.B.: Inverse problems. Am. Math. Mon. **83**, 107–118 (1976)
69. Kennedy, J.: The particle swarm: social adaptation of knowledge. In: IEEE International Conference on Evolutionary Computation, pp. 303–308. IEEE, Piscataway, NJ (1997)
70. Kennedy, J.: Chapter The behavior of particles. In: Evolutionary Programming VII: Proceeding of the Seventh Annual Conference on Evolutionary Programming (EP98). Lecture Notes in Computer Science, vol. 1447, pp. 581–590. Springer, New York (1998)
71. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE, Perth (1995)
72. Kiran, M., Ozceylan, E., Gunduz, M., Paksoy, T.: A novel hybrid approach based on particle swarm optimization and ant colony algorithm to forecast energy demand of Turkey. Energy Convers. Manag. **53**(2), 75–83 (2012)
73. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
74. Knupp, D.C., Silva Neto, A.J., Sacco, W.F.: Estimation of radiative properties with the particle collision algorithm. In: Inverse Problems, Design and Optimization Symposium, Miami, Florida (2007)
75. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans. Evol. Comput. **9**(5), 474–488 (2005)
76. Krishnaswami, V., Luh, G.C., Rizzoni, G.: Nonlinear parity equation based residual generation for diagnosis of automotive engine faults. Control Eng. Pract. **3**(10), 1385–1392 (1995)
77. Krysander, M., Aslund, J., Frisk, E.: An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**(1), 197–206 (2008)
78. Krysander, M., Frisk, E.: Sensor placement for fault diagnosis. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**(6), 1398–1410 (2008)
79. Larrañaga, P., Lozano, J.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Genetic Algorithms and Evolutionary Computation, vol. 2. Kluwer Academic Publishers, Boston (2002)
80. Li, Z., Dahhou, B.: A new fault isolation and identification method for nonlinear dynamic systems: application to a fermentation process. Appl. Math. Model. **32**, 2806–2830 (2008)
81. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
82. Liu, Q., Wenyuan, L.: The study of fault diagnosis based on particle swarm optimization algorithm. Comput. Informa. Sci. **2**(2), 87–91 (2009)
83. Liu, L., Yang, S., Wang, D.: Particle Swarm Optimization with Composite Particle in Dynamic Environments. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **40**(6), 1634–1638 (2010)

84. Lobato, F.S., Steffen, V., Silva Neto, A.J.: Solution of inverse radiative transfer problems in two-layer participating media with differential evolution. Inverse Prob. Sci. Eng. **18**(2), 183–195 (2009)
85. Lobato, F.S., Steffen, V., Silva Neto, A.J.: Solution of the coupled inverse conduction-radiation problem using multi-objective optimization differential evolution. In: 8th World Congress on Structural and Multidisciplinary Optimization. Lisboa, Portugal (2009)
86. Lunze, J.: Laboratory three tanks system -benchmark for the reconfiguration problem. Tech. rep., Technical University of Hamburg-Harburg, Institute of Control Engineering, Germany (1998)
87. Luz, E.F.P., Becceneri, J.C., De Campos Velho, H.F.F.: A new multiparticle collision algorithm for optimization in a high-performance environment. J. Comput. Interdiscip. Sci. **1**(1), 3–10 (2008)
88. Mauryaa, M.R., Rengaswamy, R., Venkatasubramaniana, V.: Fault diagnosis using dynamic trend analysis: a review and recent developments. Eng. Appl. Artif. Intell. **20**(2), 133–146 (2007)
89. Metenidin, M.F., Witczak, M., Korbicz, J.: A novel genetic programming approach to nonlinear system modelling: application to the DAMADICS benchmark problem. Eng. Appl. Artif. Intell. **24**, 958–967 (2011)
90. Metropolis, N., Rosenbluth, A.W., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
91. Mezura-Montes, E., Velázquez-Reyes, J., Coello-Coello, C.: A comparative study of differential evolution variants for global optimization. In: GECCO 06, Seattle, Washington (2006)
92. Morozov, V.A.: Regularization Methods for Ill- Posed Problems. CRC Press, Boca Raton, FL (1993)
93. Mosegaard, K., Tarantola, A.: Chapter Probabilistic approach to inverse problems. In: International Handbook of Earthquake and Engineering Seismology, vol. A, pp. 237–265. Academic, London (2001)
94. Moura Neto, F.D., Silva Neto, A.J.: An Introduction to Inverse Problems with Applications. Springer, New York (2012)
95. Narasimhana, S., Vachhani, P., Rengaswamya, R.: New nonlinear residual feedback observer for fault diagnosis in nonlinear systems. Automatica **44**, 2222–2229 (2008)
96. Odgaard, P.F., Matajib, B.: Observer-based fault detection and moisture estimating in coal mills. Control Eng. Pract. **16**, 909–921 (2008)
97. Ogata, K.: Modern Control Engineering. Prentice-Hall, Englewood Cliffs, NJ (1998)
98. Parker, R.L.: Geophysical Inverse Theory. Princeton University Press, Princeton, NJ (1994)
99. Patton, R.J., Chen, J.: A review of parity space approaches to fault diagnosis. In: IFAC SAFEPROCESS Symposium (1991)
100. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)
101. Puris, A., Bello, R., Herrera, F.: Analysis of the efficacy of a two-step methodology for ant colony optimization: case of study with TSP and QAP. Expert Syst. Appl. **37**(7), 5443–5453 (2010)
102. Qin, A., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
103. Ramos, F.M., Velho, H.F.C.: Reconstruction of geoelectric conductivity distributions using a first-order minimum entropy technique. In: Proceeding of the 2nd International Conference on Inverse Problems in Engineering: Theory and Practice, vol. 2, pp. 195–206. Le Croisic, France (1996)
104. Sacco, W.F., Oliveira, C.R.E.: A new stochastic optimization algorithm based on particle collisions. In: 2005 ANS Annual Meeting, Transactions of the American Nuclear Society (2005)
105. Sacco. W.F. Oliveira, C.R.E., Pereira, C.M.N.A.: Two stochastic optimization algorithms applied to nuclear reactor core design. Prog. Nucl. Energy **48**(6), 525–539 (2006)

106. Samanta, B., Nataraj, C.: Use of particle swarm optimization for machinery fault detection. Eng. Appl. Artif. Intell. **22**(2), 308–316 (2009)
107. Seckiner, S., Eroglu, Y., Emrullah, M., Dereli, T.: Ant colony optimization for continuous functions by using novel pheromone updating. Appl. Math. Comput. **219**, 4163 (2013)
108. Shah-Hosseini, H.: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput. **1**(1/2), 71–79 (2009)
109. Sharkey, A.J.C., Sharkey, N.E., Gopinath, O.C.: Diverse neural net solutions to a fault diagnosis problem. Tech. rep., Department of Computer Science, University of Sheffield (1999)
110. Shelokar, P.S., Siarry, P., Jayaraman, V.K., Kulkarni, B.D.: Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Appl. Math. Comput. **188**(1), 129–142 (2007)
111. Silva Neto, A.J., Becceneri, J.C., Campos Velho, H.F. (eds.): Computational Intelligence Applied to Inverse Problems in Radiative Transfer. EdUERJ, Rio de Janeiro, Brazil (2016)
112. Silva Neto, A.J., Lugon Jr., J., Soliro, F.J.C.P., Biondi Neto, L., Santana, C.C., Lobato, F.S., Steffen Jr., V., Campos Velho, H.F., Souza, A.F., Camara L, D.T., Assis, E.G., Silva, F.M., Oliveira, G.P., Camps Echevarría, L., Llanes-Santiago, O.: Direct and Inverse Problems with Applications in Engineering – Research Collection. InTech, Rijeka, Croatia (2016)
113. Silva Neto, A.J., Llanes-Santiago, O., Silva, G.N. (eds.): Mathematical Modelling and Computational Intelligence in Engineering Applications. Springer, Cham (2016)
114. Simani, S., Patton, R.J.: Fault diagnosis of an industrial gas turbine prototype using a system identification approach. Control Eng. Pract. **16**(7), 769–786 (2008)
115. Simani, S., Fantuzzi, C., Patton, R.J.: Model-Based Fault Diagnosis in Dynamic Systems Using Identification Techniques. Springer, New York (2002)
116. Smith, C.R., Grandy, W.T.: Maximum-Entropy and Bayesian Methods in Inverse Problems, Fundamental Theories of Physics. Springer, Dordrecht (1985)
117. Snieder, R., Trampert, J.: Inverse Problems in Geophysics. Samizdat Press (1999)
118. Socha, K.: Ant colony optimization for continuous and mixed-variable domains. Ph.D. thesis, Université Libre de Bruxelles (2008)
119. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**(3), 1155–1173 (2008)
120. Sorsa, T., Koivo, H., Koivisto, H.: Neural networks in process fault diagnosis. IEEE Trans. Syst. Man Cybern. **21**(4), 815–825 (1991)
121. Sousa, F.L.: Generalized extremal optimization: a new stochastic algorithm for optimal design. Ph.D. thesis, Graduate Program in Applied Computation, Instituto Nacional de Pesquisas Espaciais, INPE-9564-TDI/836 (2003). (In Portuguese)
122. Souto, R.P., Stephany, S., Becceneri, J.C., Campos Velho, H.F., Silva Neto, A.J.: On the use of the ant colony system for radiative properties estimation. In: 5th International Conference on Inverse Problems in Engineering – Theory and Practice (V ICIPE), vol. 3, pp. 1–10. Leeds University Press, Leeds, Inglaterra (2005)
123. Storn, R., Price, K.: Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. 12, International Computer Science Institute (1995)
124. Storn, R., Price, K.: Differential evolution: a simple and efficient adaptive heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
125. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Tech. rep., Nanyang Technological University (2005)
126. Tarantola, A.: Inverse Problem Theory. Elsevier, New York (1987)
127. Tarantola, A.: Inverse Problem Theory and Model Parameter Estimation. SIAM, Philadelphia, PA (2004)
128. Tarantola, A., Valette, B.: Inverse problems=quest for information. J. Geophys. **50**(3), 159–170 (1982)

129. Tvrdik, J.: Adaptation in differential evolution: a numerical comparison. Appl. Soft Comput. **9**(3), 1149–1155 (2009)
130. Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N.: A review of process fault detection and diagnosis-part I: quantitative model-based methods. Comput. Chem. Eng. **27**(3), 293–311 (2003)
131. Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N.: A review of process fault detection and diagnosis-part II: qualitative model-based methods and search strategies. Comput. Chem. Eng. **27**(3), 313–326 (2003)
132. Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N.: A review of process fault detection and diagnosis-part III: process history based methods. Comput. Chem. Eng. **27**(3), 327–346 (2003)
133. Wang, L., Niu, Q., Fei, M.: A novel quantum ant colony optimization algorithm and its application to fault diagnosis. Trans. Inst. Meas. Control **30**(3/4), 313–329 (2008)
134. Willsky, A.S.: A survey of design methods for failure detection systems. Automatica **12**(6), 601–611 (1976)
135. Witczak, M.: Advances in model based fault diagnosis with evolutionary algorithms and neural networks. Int. J. Appl. Math. Comput. Sci. **16**(1), 85–99 (2006)
136. Witczak, M.: Modelling and Estimation Strategies for Fault Diagnosis of Non-Linear Systems From Analytical to Soft Computing Approaches. Springer, New York (2007)
137. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
138. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
139. Yang, X.: Nature – Inspired Optimization Algorithms. Elsevier, Amsterdam (2014)
140. Yang, X.S., Deb, S.: Cuckoo search via Levy flights. In: World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), pp. 210–214. IEEE Publications, Piscataway, NJ (2009)
141. Yang, X.S., Deb, S.: Engineering Optimisation by Cuckoo Search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
142. Yang, S.H., Chen, B.H., Wang, X.Z.: Neural network based fault diagnosis using unmeasurable inputs. Eng. Appl. Artif. Intell. 1 **13**(31), 345–356 (2000)
143. Yang, E., Xiang, H., Guand, D., Zhang, Z.: A comparative study of genetic algorithm parameters for the inverse problem-based fault diagnosis of liquid rocket propulsion systems. Int. J. Autom. Comput. **4**(3), 255–261 (2007)
144. Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: IEEE Congress on Evolutionary Computation (CEC2008), Hong Kong, pp. 1110–1116 (2008)
145. Yew-Soon, O., Meng-Hiot, L., Ning, Z., Kok-Wai, W.: Classification of adaptive memetic algorithms: a comparative study. Syst. Man Cybern. Part B: Cybern. **36**(1), 141–152 (2006)
146. Zaharie, D.: Influence of crossover on the behavior of differential evolution algorithms. Appl. Soft Comput. **9**(3), 1126–1138 (2009)
147. Zhan, Z.H., Zhang, J., Li, Y., Shi, Y.H.: Orthogonal learning particle swarm optimization. IEEE Trans. Evol. Comput. **15**(6), 832–847 (2011)
148. Zhang, J.: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)