Multiple Roots of Systems of Equations by Repulsion Merit Functions

Gisela C.V. Ramadas[1], Edite M.G.P. Fernandes[2], and Ana Maria A.C. Rocha[2]

[1] Department of Mathematics, School of Engineering, Polytechnic of Porto, 4200-072 Porto, Portugal
gcv@isep.ipp.pt

[2] Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal
emgpf@dps.uminho.pt
arocha@dps.uminho.pt

Abstract. In this paper we address the problem of computing multiple roots of a system of nonlinear equations through the global optimization of an appropriate merit function. The search procedure for a global minimizer of the merit function is carried out by a metaheuristic, known as harmony search, which does not require any derivative information. The multiple roots of the system are sequentially determined along several iterations of a single run, where the merit function is accordingly modified by penalty terms that aim to create repulsion areas around previously computed minimizers. A repulsion algorithm based on a multiplicative kind penalty function is proposed. Preliminary numerical experiments with a benchmark set of problems show the effectiveness of the proposed method.

Keywords: System of equations; multiple roots; penalty function; repulsion; harmony search

1 Introduction

In this paper, we aim to investigate the performance of a repulsion algorithm that is based on a multiplicative kind penalty merit function, combined with a metaheuristic optimization algorithm, the harmony search (HS), to compute multiple roots of a system of nonlinear equations of the form

$$f(x) = 0, \tag{1}$$

where $f(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, each $f_i : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n$ is a continuous possibly nonlinear function in the search space and Ω is a closed convex set, herein defined as $[l, u] = \{x : -\infty < l_i \leq x_i \leq u_i < \infty, i = 1, \ldots, n\}$. The functions $f_i(x)$, $i = 1, \ldots, n$ are not necessarily differentiable implying that analytical and numerical derivatives may not be used. The work herein presented comes in the sequence of the study published in [1,2]. To compute a solution of

a nonlinear system of equations is equivalent to compute a global minimizer of the optimization problem

$$\min_{x \in \Omega \subset \mathbb{R}^n} \mathcal{M}(x) \equiv \sum_{i=1}^{n} f_i(x)^2, \tag{2}$$

in the sense that they have the same solutions. Thus, a global minimizer and not just a local one, of the function $\mathcal{M}(x)$, known as merit function, in the set Ω, is required. Problem (2) is similar to the usual least squares problem for which many iterative methods have been proposed. They basically assume that the objective function is twice continuously differentiable. However, the objective $\mathcal{M}$ in (2) is only once differentiable if some, or just one, of the f_i, $(i = 1, \ldots, n)$ are not differentiable. Thus, the most popular Newton-type and Quasi-Newton methods should be avoided [3,4,5,6]. Furthermore, their convergence and practical performance are highly sensitive to the user provided initial approximation. Additionally, they are only capable of finding one root at each run of the algorithm. Since a global minimizer of problem (2) is required, classical optimization techniques with guaranteed convergence to local minimizers cannot be applied. When the optimization problem is nonlinear and non-convex, metaheuristics are able to avoid convergence to local minimizers and to generate good quality solutions in less time than most classical techniques. Metaheuristics are general heuristic methods which can be applied to a wide variety of optimization problems. In 2001 emerged the HS algorithm that relies on a set of points and is inspired by natural phenomena [7]. It draws its inspiration not from a biological or physical process like most metaheuristic optimization techniques, but from an artistic one – the improvisation process of musicians seeking a wonderful harmony. HS has efficient strategies for exploring the entire search space, as well as techniques to exploit locally a promising region to yield a high quality solution in a reasonable time. The dynamic updating of two important parameters in the HS algorithm has improved the efficiency and robustness of the metaheuristic [8]. Until today, the HS paradigm has been implemented in many areas, such as in engineering, robotics, telecommunications, health and energy [9,10], in scheduling problems [11], in transportation problems [12], and in seismic isolation systems [13].

Although finding a single root of a system of nonlinear equations is a trivial task, finding all roots is one of the most demanding problems. Multistart methods are stochastic techniques that have been used to compute multiple solutions to problems [14,15,16]. In a multistart strategy, a search procedure is applied to a set of randomly generated points of the search space to converge sequentially along the iterations to the multiple solutions of the problem, in a single run. However, the same solutions may be located over and over again along the iterations and the computational effort turns out to be quite heavy. Other approaches that combine metaheuristics with techniques that modify the objective function in problem (2) have been reported in the literature [17,18,19,20]. The technique in [20] relies on the assignment of a penalty term to each previously computed root so that a repulsion area around the root is created. In [19], an evolutionary optimization algorithm is used together with a type of polarization technique

to create a repulsion area around each previously computed root. The repulsion areas force the algorithm to move to other areas of the search space and look for other roots thus avoiding repeated convergence to already located solutions.

In this study, we further explore this penalty-type approach to create repulsion areas around previously detected roots and propose a repulsion algorithm that is capable of computing multiple roots of a system of nonlinear equations through the invoking of the HS algorithm with modified merit functions. We propose a multiplicative kind penalty function based on the inverse of the 'erf' function, known as error function.

The proposed algorithm is tested on 13 benchmark systems of nonlinear equations and the obtained results are compared to the results produced by other penalty type functions that have been recently proposed in the literature. It is shown that the proposed 'erf' penalty function is competitive with other penalties in comparison.

The paper is organized as follows. Section 2 reports on penalty type functions and describes the proposed repulsion algorithm and Section 3 addresses the HS metaheuristic to compute global minimizers of merit functions with accuracy and efficiency. Then, some numerical experiments are shown in Section 4 and we conclude the paper in Section 5.

2 Repulsion Merit Functions

This section aims to discuss the implementation of penalty type functions to create repulsion areas around previously computed solutions of a system of nonlinear equations, thus avoiding the convergence to already located solutions. The proposed repulsion algorithm solves a sequence of global optimization problems by invoking a solver to locate a global minimizer of a sequentially modified merit function. The first call to the global solver considers the original merit function (2). Thereafter the merit function needs to be modified to avoid locating previously computed minimizers. Let the first located minimizer be ξ_1. The idea is to define a repulsion area around ξ_1 so that it will be no more a global minimizer of the modified merit function. The minimization problem is then based on the modified merit function $\bar{\mathcal{M}}$ with a repulsion area created around ξ_1 so that the solver will not find it again. We now show two modified objective functions available in the literature. The first one, presented in [20], is:

$$\min_{x \in \Omega \subset \mathbb{R}^n} \bar{\mathcal{M}}(x) \equiv \mathcal{M}(x) + \beta e^{-\|x-\xi_1\|} \tag{3}$$

where β is a large positive parameter and aims to scale the penalty for approaching the root ξ_1. Thus, after k roots of the problem (1) having been identified, herein denoted by $\xi_1, \xi_2, \ldots, \xi_k$, the repulsion modified merit function is

$$\bar{\mathcal{M}}(x) = \mathcal{M}(x) + \sum_{i=1}^{k} P^A(x; \xi_i, \beta, \rho), \tag{4}$$

where the superscript A stands for 'additive-type penalty' and each penalty term is given by

$$P^A(x;\xi_i,\beta,\rho) = \begin{cases} \beta e^{-\|x-\xi_i\|}, & \text{if } \|x-\xi_i\| \leq \rho \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $\rho = \min\{0.1,\, \|\xi_j - \xi_l\|/2 : j \neq l \text{ and } j,l = 1,\ldots,k\}$ is a small problem dependent parameter and defines the radius of the repulsion area, so that other solutions not yet identified, and outside the repulsion area, are not penalized in $\bar{\mathcal{M}}$ [20]. We note that an additive penalty term like (5) satisfies the following properties:

- $P^A(x;\xi_i,\beta,\rho) \to \beta$ when $x \to \xi_i$, and increases with parameter β, in a way that ξ_i is no longer a minimizer of $\bar{\mathcal{M}}(x)$;
- $P^A(x;\xi_i,\beta,\rho) \to 0$ when x moves away from ξ_i, such that the merit function is not affected outside the repulsion area.

Other type of penalty term aiming to create a repulsion area around a previously computed minimizer, say ξ_i, but with a distinctive behavior, is presented in [19]:

$$P^M(x;\xi_i,\alpha) = \left|\coth(\alpha\|x-\xi_i\|)\right|, \tag{6}$$

where coth is the hyperbolic cotangent function and α is a positive parameter greater or equal to one, called density factor and used to adjust the radius of the repulsion area. Here, the superscript M means that the penalty is of a 'multiplicative-type'. The main properties in this repulsion context are the following:

- $P^M(x;\xi_i,\alpha) \to \infty$ when $x \to \xi_i$, so that ξ_i is no longer a minimizer of $\bar{\mathcal{M}}(x)$;
- $P^M(x;\xi_i,\alpha) \to 1$ when x moves away from ξ_i, in a way that the merit function is not affected outside the repulsion area.

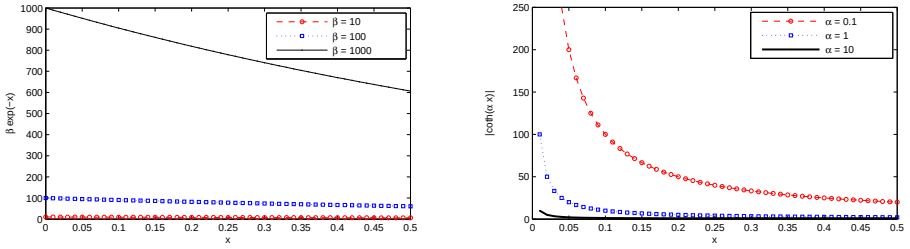
Similar arguments may be used to create a sequence of global minimization problems based on the modified merit function $\bar{\mathcal{M}}$ [19] which creates repulsion areas around the located global minimizers ξ_i, $i = 1,\ldots,k$ so that the solver will not converge again to the same solutions:

$$\bar{\mathcal{M}}(x) = \mathcal{M}(x)\prod_{i=1}^{k} P^M(x;\xi_i,\alpha). \tag{7}$$

We now illustrate the behavior of the above referred penalty terms (5) and (6), as the corresponding parameters β and α increase. See Figure 1. While the parameter α, in the penalty $|\coth(\alpha\|x-\xi_i\|)|$, aims to define the radius of the repulsion area (figure on the right), the parameter β in penalty $\beta e^{-\|x-\xi_i\|}$ aims to scale the penalty created by moving close to a previously located solution. The radius of the repulsion area is defined by the parameter ρ.

We now present the main ideas behind the new repulsion merit function. It uses the error function, denoted by 'erf', which is a mathematical function defined by the integral

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt,$$



(a) βe^{-x} penalty (b) $|\coth(\alpha x)|$ penalty

Fig. 1. Penalty terms in $(0, 0.5]$, for different values of the parameters

satisfies the following properties

$$\operatorname{erf}(0) = 0,\ \ \operatorname{erf}(-\infty) = -1,\ \ \operatorname{erf}(+\infty) = 1,\ \ \operatorname{erf}(-x) = -\operatorname{erf}(x), \tag{8}$$

and has a close relation with the normal distribution probabilities. When a series of measurements are described by a normal distribution with mean 0 and standard deviation σ, the erf function evaluated at $\frac{x}{\sigma\sqrt{2}}$, for a positive x, gives the probability that the error of a single measurement lies in the interval $[-x, x]$. In a penalty function context aiming to prevent convergence to a located root ξ_i, thus defining a repulsion area around it, we propose the multiplicative inverse 'erf' penalty function:

$$P_e^M(x; \xi_i, \delta, \bar{\rho}) = \begin{cases} |\operatorname{erf}(\delta \|x - \xi_i\|)|^{-1}, & \text{if } \|x - \xi_i\| \leq \bar{\rho} \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

which depends on the parameter $\delta > 0$ to scale the penalty for approaching the already computed solution ξ_i, and on the parameter $\bar{\rho}$ to adjust the radius of the repulsion area, where $\bar{\rho} = 0.1 \min_{i=1,\ldots,n}(u_i - l_i)$. We note that the penalty term tends to $+\infty$ when x approaches the root ξ_i, meaning that ξ_i is no longer a minimizer of the modified penalty merit function. According to the properties in (8), as $x \to \infty$, the penalty $P_e^M(x; \xi_i, \delta, \bar{\rho}) \to 1$ meaning that the modified merit function is of 'multiplicative-type' and thus it is not affected when far from previous located roots:

$$\bar{\mathcal{M}}(x) = \mathcal{M}(x) P_e^M(x; \xi_i, \delta, \bar{\rho}). \tag{10}$$

We include Figure 2 to show how the penalty behaves with the parameter δ. Our proposal for the implementation of the penalty (9) is the following:

- the parameter δ is used to scale the penalty in the neighborhood of ξ_i, i.e., when $\|x - \xi_i\| \approx 0$, noting that the penalty increases as δ decreases;
- the parameter $\bar{\rho}$ is used to adjust the radius of the repulsion area, and this may depend closely on the problem at hand.

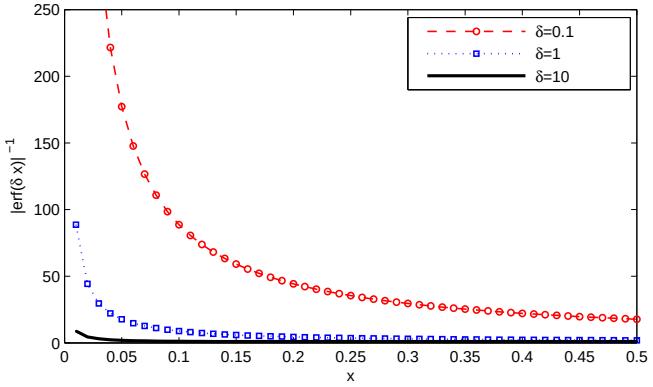


Fig. 2. $|\operatorname{erf}(\delta x)|^{-1}$ penalty in $(0, 0.5]$, for $\delta = 0.1$, $\delta = 1$ and $\delta = 10$

Algorithm 1 contains the main steps of the repulsion algorithm. The set Ξ, empty at the beginning of the iterative process, contains the roots that are computed and are different from the previous ones. To check if a computed root ξ has been previously located the following conditions

$$|\mathcal{M}(\xi) - \mathcal{M}(\xi_i)| \leq \epsilon \text{ and } \|\xi - \xi_i\| \leq \epsilon \tag{11}$$

must hold, for all $\xi_i \in \Xi$ and a small positive ϵ. Although the repulsion strategy is rather successful in locating multiple roots and avoiding repeated convergence to previously located solutions, these may be occasionally recovered suggesting that the penalty could be increased. The algorithm stops when five unsuccessful iterations (counter It_{uns} in the algorithm) are encountered. An iteration is considered unsuccessful when the solution produced by Algorithm 2 is not a global minimizer of the merit function. In practice, and after an empirical study, we consider that a solution is not a global one if $\mathcal{M} > 10^{-10}$. Anyway, Algorithm 1 is allowed to run for $It_{\max}$ iterations, the user provided threshold.

We now consider an example to illustrate the behavior of the three above described penalty functions. This is a system with many roots in the considered search space Ω.

Example 1. Let the function $\mathcal{M}$ illustrated in Figure 3 be the merit function of the system

$$f(x) = \begin{cases} \cos(x_1) = 0 \\ \sin(x_2) = 0 \end{cases}.$$

In $[-5, 5]^2$, the merit function has 12 global minimizers [18]. Table 1 shows the average number of roots, $N.\text{roots}_{\text{avg}}$, the average number of function evaluations, NFE_{avg}, and time (in seconds), T_{avg}, found in five experimental runs produced by our algorithm, where we implemented:

– the 'exp' penalty term as described in (5) with $\beta = 1000$;

Algorithm 1 Repulsion algorithm

Require: $It_{\max} > 0$, $\epsilon > 0$;
Set $\Xi = \emptyset$, $It = 0$, $It_{\text{uns}} = 0$, $k = 0$;
Compute $\xi_1 = \arg\min_{x \in \Omega} \bar{\mathcal{M}}(x)$ using Algorithm 2;
if $\mathcal{M}(\xi_1) \leq 10^{-10}$ **then**
  Set $k = 1$, $r_1 = 1$, $\Xi = \Xi \cup \xi_1$;
else
  Set $It_{\text{uns}} = It_{\text{uns}} + 1$;
end if
while $It_{\text{uns}} \leq 5$ and $It \leq It_{\max}$ **do**
  Compute $\xi = \arg\min_{x \in \Omega} \bar{\mathcal{M}}(x)$ using Algorithm 2;
  if $\mathcal{M}(\xi) \leq 10^{-10}$ **then**
    if $|\mathcal{M}(\xi) - \mathcal{M}(\xi_i)| > \epsilon$ or $\|\xi - \xi_i\| > \epsilon$, for any $i = 1, \ldots, k$ **then**
      Set $k = k + 1$, $\xi_k = \xi$, $r_k = 1$, $\Xi = \Xi \cup \xi_k$;
    else
      Set $r_l = r_l + 1$ ($\xi_l \in \Xi$);
    end if
  else
    Set $It_{\text{uns}} = It_{\text{uns}} + 1$;
  end if
  Set $It = It + 1$;
end while
return k (number of located roots), Ξ (located roots), r_i, $i = 1, \ldots, k$ (number of times each root was recovered)

– the 'coth' penalty as shown in (6) with $\alpha = 10$;
– the 'erf' penalty with $\delta = 10$ but without the condition with the parameter $\bar{\rho}$ (aiming to work like the coth function) (erf_1);
– the 'erf' penalty using $\delta = 0.1$ and $\bar{\rho} = \rho$ as defined to be used in (5) (erf_2);
– and finally, the 'erf' penalty using $\delta = 0.1$ and $\bar{\rho}$ as proposed to be used in (9) (erf_3).

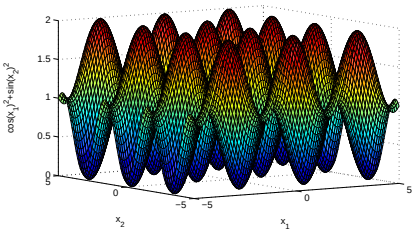


Fig. 3. $\mathcal{M}$ function of Example 1

Table 1. Comparison of penalty repulsion functions

penalty	$N.\text{roots}_{\text{avg}}$	NFE_{avg}	T_{avg}
exp	10.6	14204	1.562
coth	11.2	20695	1.860
erf_1	9.4	15426	2.344
erf_2	11.4	15334	1.894
erf_3	12.0	16728	1.313

Based on this example we conclude that the proposed methodology, summarized by variant erf_3, performed better for this experiment. For completeness, we report now the roots produced by one of the five experimental runs of erf_3:

(1.57080295e+00, -3.14159058e+00), (-4.71238170e+00, -3.14159195e+00), (-4.71239481e+00, 1.99367704e-06), (4.71238471e+00, 3.14158920e+00), (-1.57080249e+00, -3.14159349e+00), (1.57080522e+00, -3.68619544e-06), (-1.57079403e+00, 7.33762541e-06), (-1.57079621e+00, 3.14160097e+00), (-4.71239131e+00, 3.14159943e+00), (4.71239663e+00, -3.14159506e+00), (4.71237915e+00, 1.11265112e-06), (1.57079907e+00, 3.14158507e+00).

3 Improved Harmony Search

The HS algorithm was developed to solve global optimization problems in an analogy with the music improvisation process where music players improvise the pitches of their instruments to obtain better harmony [7,9]. An overview of the existing variants of the HS is presented by Alia and Mandava in [21]. Here, the improved harmony search (I-HS) variant [8] is used to compute a global solution of the problem (2).

At each iteration, the I-HS algorithm provides a set of solution vectors from which the best and the worst solutions, in terms of their fitness - objective function values - are selected. The candidate solutions are saved in the harmony memory (HM). Throughout the iterative process there are HMS (the size of the HM) solutions. After generating the HM randomly in the search space Ω, x^j, $j = 1, \ldots,$HMS, the vectors are evaluated and the best harmony, x^{best}, and the worst, x^{worst}, in terms of objective/merit function value are selected. Thereafter, a new harmony is improvised meaning that a new vector y is generated using three improvisation operators:

- O_1: HM operator
- O_2: random selection operator
- O_3: pitch adjustment operator.

A harmony memory considering rate (HMCR) represents the probability of choosing the component of the new harmony/vector from the HM (operator O_1). Otherwise, the component is randomly generated in Ω (operator O_2):

$$y_i = \begin{cases} x_i^j, j \text{ random } \in \{1, \ldots, \text{HMS}\}, & \text{if } \tau_1 < \text{HMCR} \\ l_i + \tau_2(u_i - l_i), & \text{otherwise} \end{cases} \tag{12}$$

for $i = 1, \ldots, n$, where τ_1, τ_2 are uniformly distributed random variables in $[0, 1]$. Based on a pitch adjusting rate (PAR), the operator O_3 is subsequently applied to refine only the components i produced by O_1, as follows:

$$y_i = \begin{cases} y_i \pm \tau \text{BW}, & \text{if } \tau < \text{PAR} \\ y_i, & \text{otherwise} \end{cases} \tag{13}$$

where BW is an arbitrary distance bandwidth and τ is a random number in the range $[0, 1]$. Finally, the HM is updated. The new harmony is compared with the

worst harmony in the HM, in terms of $\mathcal{M}$ values. The new harmony is included in the HM, replacing the worst one if it is better than the worst harmony.

As shown in (13), the classical HS algorithm uses fixed value for both PAR and BW. However, small values of PAR with large values of BW can considerably increase the number of iterations required to converge to an optimal solution of (2). Experience has shown that BW must take large values at the beginning of the iterative process to enforce the algorithm to increase the diversity of solution vectors. However, small BW values in the final iterations increase the fine-tuning of solution vectors. Furthermore, large values of PAR combined with small values of BW usually cause the improvement of best solutions in the final stage of the process. To eliminate some of the drawbacks due to fixed values of PAR and BW, the I-HS variant [8] dynamically defines parameter values that depend on the iteration counter It_{HS} of the algorithm:

$$\mathrm{PAR}(It_{\mathrm{HS}}) = \mathrm{PAR}_{\min} + It_{\mathrm{HS}} \frac{(\mathrm{PAR}_{\max} - \mathrm{PAR}_{\min})}{It_{\mathrm{HS}_{\max}}} \tag{14}$$

where $It_{\mathrm{HS}_{\max}}$ represents the allowed maximum number of iterations, $\mathrm{PAR}_{\min}$ and $\mathrm{PAR}_{\max}$ are the minimum and maximum pitch adjusting rate respectively, and

$$\mathrm{BW}(It_{\mathrm{HS}}) = \mathrm{BW}_{\max} e^{c\, It_{\mathrm{HS}}}, \quad \text{for } c = \frac{\ln(\frac{\mathrm{BW}_{\min}}{\mathrm{BW}_{\max}})}{It_{\mathrm{HS}_{\max}}} \tag{15}$$

where $\mathrm{BW}_{\min}$ and $\mathrm{BW}_{\max}$ are the minimum and maximum bandwidth respectively. The main steps of the I-HS algorithm are as represented in Algorithm 2 below:

Algorithm 2 I-HS algorithm

Require: k and ξ_i, $i = 1, \ldots, k$ (from Algorithm 1), HMS, Ω, $It_{\mathrm{HS}_{\max}} > 0$;
1: Set $It_{\mathrm{HS}} = 1$;
2: **for** $j = 1, \ldots,$HMS **do**
3: Randomly generate $x^j \in \Omega$;
4: Compute $\bar{\mathcal{M}}(x^j)$;
5: **end for**
6: Based on $\bar{\mathcal{M}}$, select x^{best} and x^{worst};
7: **while** $It_{\mathrm{HS}} \leq It_{\mathrm{HS}_{\max}}$ and $\bar{\mathcal{M}}(x^{\mathrm{best}}) > 10^{-10}$ **do**
8: Improvise a new harmony $y \in \Omega$;
9: Compute $\bar{\mathcal{M}}(y)$;
10: Based on $\bar{\mathcal{M}}$, update HM and select x^{best} and x^{worst};
11: Set $It_{\mathrm{HS}} = It_{\mathrm{HS}} + 1$;
12: **end while**
13: **return** $\xi \leftarrow x^{\mathrm{best}}$ (for Algorithm 1)

Table 2. Problems set

NonD2	$f_1 = x_1^2 - x_2^2$ $f_2 = 1 - \lvert x_1 - x_2 \rvert$ 2 roots in $[-3, 3]^2$
Trans	$f_1 = x_1^2 - x_2 - 2$ $f_2 = x_1 + \sin(\pi x_2/2)$ 3 roots in $[-3, 3]^2$
Himmelblau	$f_1 = 4x_1^3 + 4x_1x_2 + 2x_2^2 - 42x_1 - 14$ $f_2 = 4x_2^3 + 4x_1x_2 + 2x_1^2 - 26x_2 - 22$ 9 roots in $[-5, 5]^2$
Geometry	$f_1 = x_1x_2 + (x_1 - 2x_3)(x_2 - 2x_3) - 165$ $f_2 = (x_1x_2^3)/12 - (x_1 - 2x_3)(x_2 - 2x_3)^3/12 - 9369$ $f_3 = \left(2(x_2 - x_3)^2(x_1 - x_3)^2x_3\right) / (x_1 + x_2 - 2x_3) - 6835$ 2 roots in $[0, 50]^3$
Floudas	$f_1 = (0.25/\pi)x_2 + 0.5x_1 - 0.5\sin(x_1x_2)$ $f_2 = (e/\pi)x_2 - 2ex_1 + (1 - 0.25/\pi)(e^{2x_1} - e)$ 2 roots in $[0.25, 1] \times [1.5, 2\pi]$
Merlet	$f_1 = -\sin(x_1)\cos(x_2) - 2\cos(x_1)\sin(x_2)$ $f_2 = -\cos(x_1)\sin(x_2) - 2\sin(x_1)\cos(x_2)$ 13 roots in $[0, 2\pi]^2$
Reactor	$f_1 = (1 - R)\,(D/(10(1 + B_1)) - x_1)\,e^{10x_1/(1+10x_1/\gamma)} - x_1$ $f_2 = (1 - R)\,(D/10 - B_1x_1 - (1 + B_2)x_2))e^{10x_2/(1+10x_2/\gamma)} + x_1$ $-(1 + B_2)x_2$ with $D = 22, B_1 = B_2 = 2, R = 0.960$ and $\gamma = 1000$ 7 roots in $[0, 1]^2$
P1syst	$f_1 = x_1 + x_2 - 3$ $f_2 = x_1^2 + x_2^2 - 9$ 2 roots in $[-3, 3]^2$
Papersys	$f_1 = x_1 - \sin(2x_1 + 3x_2) - \cos(3x_1 - 5x_2)$ $f_2 = x_2 - \sin(x_1 - 2x_2) + \cos(x_1 + 3x_2)$ 3 roots in $[-3, 3]^2$
Casestudy5	$f_1 = e^{x_1^2} - 8x_1\sin(x_2)$ $f_2 = x_1 + x_2 - 1$ $f_3 = (x_3 - 1)^3$ 2 roots in $[0, 1]^3$
Casestudy7	$f_1 = x_1^3 - 3x_1x_2^2 - 1$ $f_2 = 3x_1^2x_2 - x_2^3 + 1$ 3 roots in $[-1, 2]^2$
Manipulator	$f = 3.9852 - 10.039x^2 + 7.2338x^4 - 1.17775x^6$ $+(-8.8575x + 20.091x^3 - 11.177x^5)\sqrt{1 - x^2}$ 6 roots in $[-1, 1]$
Trigonometric	$f = \sin(0.2x)\cos(0.5x)$ 19 different roots in $[-50, 50]$

4 Computational Experiments

The experiments were carried out on a PC Intel Core 2 Duo Processor E7500 with 2.9GHz and 4Gb of memory. The algorithms were coded in Matlab Version 8.0.0.783 (R2012b). In this study, the thirteen problems used for benchmark [16,17,19,20,22,23] are listed in Table 2 that also contains the number of roots in the search space Ω. These are the values set to the parameters: $\epsilon = 0.005$, $It_{\max} = 30$, HMS=10 when $n = 1, 2$ and 12 for $n = 3, 4$, HMCR=0.95, $\text{PAR}_{\min}$=0.35, $\text{PAR}_{\max} = 0.99$, $\text{BW}_{\min} = 10^{-6}$ and $\text{BW}_{\max} = 5$ [8]. We remark that the maximum number of iterations allowed in Algorithm 2, $It_{\text{HS}_{\max}}$, varies with the problem: 1000 in NonD2, 2000 in Merlet and P1syst, 5000 in Trans,

Floudas, Casestudy5, Casestudy7, Manipulator and Trigonometric, and 10000 in Himmelblau, Geometry, Reactor and Papersys.

Tables 3 – 5 report the average results produced by the proposed Algorithm 1 using:

- the 'exp' penalty with $\beta = 1000$ and ρ, as defined in (5),
- the 'coth' penalty with $\alpha = 10$, as described in (6), and
- the 'erf' penalty with $\delta = 0.1$ and $\bar{\rho}$, as defined in (9),

respectively, where the columns show:

- the name of the problem, Prob.;
- percentage of runs (out of 30) where all the roots were located, SR (%);
- the average number of located roots per run, $N.\text{roots}_{\text{avg}}$;
- the average number of merit function evaluations per run, NFE_{avg};
- the average time in seconds per run, T_{avg};
- the average number of function evaluations required to locate a root, NFE_{root};
- the average time required to locate a root, T_{root}.

These preliminary results are very encouraging. Numerical results suggest that the new 'erf' penalty function clearly has some advantages over 'coth' penalty function and the 'exp' penalty function. Overall, 'coth' and 'exp' penalties produce fairly similar results for most problems. We observed that the 'erf' penalty function demonstrated to be a promising and viable tool for computing multiple roots of systems of nonlinear equations.

Table 3. Numerical results from 'exp' penalty with $\beta = 1000$, considering (4) and (5)

Prob.	SR	$N.\text{roots}_{\text{avg}}$	NFE_{avg}	T_{avg}	NFE_{root}	T_{root}
NonD2	100	2.0	1507	0.109	753	0.055
Trans	97	3.0	9688	0.684	3266	0.231
Himmelblau	0	5.9	37664	3.146	6420	0.536
Geometry	43	1.4	8865	0.633	6332	0.452
Floudas	100	2.0	6290	0.426	3145	0.213
Merlet	20	11.4	10311	1.570	902	0.137
Reactor	7	5.7	157388	12.857	27451	2.243
P1syst	97	2.0	2240	0.157	1139	0.080
Papersys	60	2.4	16479	1.135	6866	0.473
Casestudy5	100	2.0	6941	0.505	3471	0.252
Casestudy7	90	2.9	10207	0.713	3520	0.246
Manipulator	0	5.0	12505	0.889	2501	0.178
Trigonometric	0	8.7	12760	1.383	1467	0.159

Table 4. Numerical results from 'coth' penalty with $\alpha = 10$, considering (6) and (7)

Prob.	SR	$N.\text{roots}_{\text{avg}}$	NFE_{avg}	T_{avg}	NFE_{root}	T_{root}
NonD2	100	2.0	1734	0.141	867	0.070
Trans†	90	2.9	30817	2.416	10627	0.833
Himmelblau†	0	6.2	58940	4.909	9456	0.788
Geometry	63	1.6	16779	1.324	10273	0.811
Floudas†	100	2.0	16385	1.262	8193	0.631
Merlet†	10	10.1	11577	1.119	1146	0.111
Reactor†	3	6.0	467962	39.799	77563	6.597
P1syst†	100	2.0	12303	0.951	6151	0.476
Papersys	60	2.4	17523	1.309	7301	0.545
Casestudy5	23	3.1	13617	1.124	4393	0.363
Casestudy7	90	2.9	10509	0.816	3624	0.282
Manipulator†	100	6.0	105303	7.353	17551	1.225
Trigonometric†	0	13.2	33123	2.585	2509	0.196

† - problems where some or all roots were recovered more than once (not necessarily in all runs).

Table 5. Numerical results from 'erf' function with $\delta = 0.1$, considering $\bar{\rho} = 0.1 \min_i(u_i - l_i)$ in (9)

Prob.	SR	$N.\text{roots}_{\text{avg}}$	NFE_{avg}	T_{avg}	NFE_{root}	T_{root}
NonD2	100	2.0	1495	0.114	748	0.057
Trans	90	2.9	10074	0.781	3474	0.269
Himmelblau	60	8.6	56018	4.319	6539	0.504
Geometry	53	1.4	8598	0.659	6291	0.482
Floudas	97	2.0	6306	0.457	3206	0.232
Merlet	100	13.0	12574	1.048	967	0.081
Reactor	10	5.9	165713	12.577	28087	2.132
P1syst†	100	2.0	4361	0.366	2181	0.183
Papersys	7	1.7	11955	0.863	6897	0.498
Casestudy5	83	2.1	7938	0.620	3780	0.295
Casestudy7	100	3.0	10523	0.768	3508	0.256
Manipulator	100	6.0	21563	1.689	3594	0.281
Trigonometric†	53	18.6	45499	4.781	2446	0.257

† - problems where some or all roots were recovered more than once

5 Conclusions

A repulsion algorithm is presented for locating multiple roots of a system of nonlinear equations. The proposed algorithm relies on a multiplicative kind penalty merit function that depends on two parameters. One aims to scale the penalty and the other adjusts the radius of the repulsion area, so that convergence to previously located solutions is avoided. The algorithm has been successfully applied and tested with a benchmark set of problems. The numerical experiments

also lead us to allege that the 'erf' penalty function is indeed more accurate, reliable, and efficient at locating multiple roots than the other alternatives in comparison.

Acknowledgments. The authors wish to thank two anonymous referees for their valuable comments and suggestions to improve the paper. This work has been supported by CIDEM (Centre for Research & Development in Mechanical Engineering, Portugal) and FCT - Fundação para a Ciência e Tecnologia within the Projects Scope: PEst-OE/EME/UI0615/2014 and PEst-OE/EEI/UI0319/2014.

References

1. Ramadas, G.C.V., Fernandes, E.M.G.P.: Solving systems of nonlinear equations by harmony search. 13th International Conference Computational and Mathematical Methods in Science and Engineering 2013; Aguiar JV et al. (eds), ISBN: 978-84-616-2723-3 Vol. IV, 1176–1186 (2013)
2. Ramadas, G.C.V., Fernandes, E.M.G.P.: Combined mutation differential evolution to solve systems of nonlinear equations, 11th International Conference of Numerical Analysis and Applied Mathematics 2013 AIP Conf. Proc. Vol. 1558, 582–585 (2013)
3. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall Inc, 1983
4. González-Lima, M.D., Oca, F.M.: A Newton-like method for nonlinear system of equations. Numer. Algorithms 52, 479–506 (2009)
5. Martínez, J.M.: Practical Quasi-Newton methods for solving nonlinear systems. J. Comput. Appl. Math. 124, 97–122 (2000)
6. Nowak, U., Weimann, L.: A family of Newton codes for systems of highly nonlinear equations. Technical Report Tr-91-10, K.-Z.-Z. Inf. Berlin, 1991
7. Geem, Z.W., Kim, J.H., Loganathan, G.: A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68 (2001)
8. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. 188, 1567–1579 (2007)
9. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Method Appl. M. 194, 3902–3933 (2004)
10. Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M.N., Salcedo-Sanz, S., Geem, Z.W.: A survey on applications of the harmony search algorithm, Eng. Appl. Artif. Intel. 26(8), 1818–1831 (2013)
11. Estahbanati, M.J.: Hybrid probabilistic-harmony search algorithm methodology in generation scheduling problem, J. Exp. Theor. Artif. Intell. DOI:10.1080/0952813X.2013.861876 (2014).
12. Hosseini, S.D., Shirazi, M.A., Ghomi, S.M.T.F.: Harmony search optimization algorithm for a novel transportation problem in a consolidation network, Eng. Optimiz. DOI:10.1080/0305215X.2013.854350 (2013)
13. Nigdeli, S.M., Bekdas, G., Alhan, C.: Optimization of seismic isolation systems via harmony search, Eng. Optimiz. DOI:10.1080/0305215X.2013.854352 (2013)
14. Tsoulos, I.G., Lagaris, I.E.: MinFinder: Locating all the local minima of a function. Comput. Phys. Commun. 174, 166–179 (2006)

15. Voglis, C., Lagaris, I.E.: Towards "Ideal Multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain. Appl. Math. Comput. 213, 1404–1415 (2009)
16. Tsoulos, I.G., Stavrakoudis, A.: On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods. Nonlinear Anal.-Real 11, 2465–2471 (2010)
17. Hirsch, M.L., Pardalos, P.M., Resende, M.: Solving systems of nonlinear equations with continuous GRASP. Nonlinear Anal.-Real 10, 2000–2006 (2009)
18. Parsopoulos, K.E., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization, IEEE Trans. Evol. Comput. 8(3), 211–224 (2004)
19. Pourjafari, E., Mojallali, H.: Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. Swarm and Evolutionary Computation 4, 33–43 (2012)
20. Silva, R.M.A., Resende, M.G.C., Pardalos, P.M.: Finding multiple roots of a box-constrained system of nonlinear equations with a biased random-key genetic algorithm. J. Global Optim. doi:10.1007/s10898-013-0105-7, online September 2013.
21. Alia, O.M., Mandava, R.: The variants of the harmony search algorithm: an overview. Artif. Intell. Rev. 36, 49–68 (2011)
22. Grosan, C., Abraham, A.: A new approach for solving nonlinear equations systems. IEEE T. Syst. Man. Cy. A 38, 698–714 (2008)
23. Jaberipour, M., Khorram, E., Karimi, B.: Particle swarm algorithm for solving systems of nonlinear equations. Comput. Math. Appl. 62, 566–576 (2011)