EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
European Laboratory for Particle Physics

Large Hadron Collider Project

LHC Project Report 495

AN IMPROVED METHOD USING RBF NEURAL NETWORKS TO SPEED UP OPTIMIZATION ALGORITHMS

M. Bazan[1], M. Aleksa[2], S. Russenschuck[2]

Abstract

The paper presents a method using Radial Basis Function (RBF) neural networks to speed up deterministic search algorithms used for the optimization of superconducting magnets for the LHC accelerator project at CERN. The optimization of the iron yoke of the main LHC dipoles requires a number of numerical field computations per trial solution as the field quality depends on the excitation and local iron saturation in the yoke. This results in computation times of about 30 minutes for each objective function evaluation (on DEC-Alpha 600/333). In this paper we present a method for constructing an RBF neural network for a local approximation of the objective function. The computational time required for such a construction is negligible compared to the deterministic function evaluation, and thus yields a speed-up of the overall search process. The effectiveness of this method is demonstrated by means of two- and three-parametric optimization examples. The achieved speed-up of the search routine is up to 30 %.

1 Institute of Computer Science, University of Wroclaw, 51-151 Wroclaw, Poland
2 CERN, LHC Division, Geneva, Switzerland

Presented at the 13th Conference on the Computation of Electromagnetic Fields (COMPUMAG)
2-5 June 2001, Lyon-Evian, France

Administrative Secretariat
LHC Division
CERN
CH - 1211 Geneva 23
Switzerland

Geneva, 27 August 2001

An improved method using RBF neural networks to speed up optimization algorithms

M. Bazan*, M. Aleksa**, S. Russenschuck**
*Institute of Computer Science, University of Wrocław, 51-151 Wrocław, Poland,
**CERN, 1211 Geneva 23, Switzerland.

Abstract— **The paper presents a method using Radial Basis Function (RBF) neural networks to speed up deterministic search algorithms used for the optimization of superconducting magnets for the LHC accelerator project at CERN. The optimization of the iron yoke of the main LHC dipoles requires a number of numerical field computations per trial solution as the field quality depends on the excitation and local iron saturation in the yoke. This results in computation times of about 30 minutes for each objective function evaluation (on DEC-Alpha 600/333). In this paper we present a method for constructing an RBF neural network for a local approximation of the objective function. The computational time required for such a construction is negligible compared to the deterministic function evaluation, and thus yields a speed-up of the overall search process. The effectiveness of this method is demonstrated by means of two- and three-parametric optimization examples. The achieved speed-up of the search routine is up to 30 %.**

I. Introduction

The LHC accelerator project at CERN requires high field superconducting magnets with an extremely uniform field distribution in the aperture. For the optimization of these magnets deterministic search algorithms are used in combination with the BEM-FEM coupling method [1]. Since the gradient of the objective function cannot be calculated in the closed form, the algorithm EXTREM, a non-gradient algorithm proposed by Jacob [2], is used. Since the field in the magnet aperture depends non-linearly on the excitation (saturation effects in the iron yoke), the field quality has to be calculated over a large excitation range from injection field level of 0.53 T to the nominal field level of 8.33 T. This yields computation times of about 30 minutes per objective function evaluation on a DEC-Alpha 600/333.

In recent years RBF approximations were applied to the design optimization in computational electromagnetics. For global optimization, multiquadratic expansions were studied in the combination with simulated annealing [3] as well as with genetic algorithms [4]. Radial basis functions were applied to global optimization in connection with local deterministic procedures [5]. The application of RBF neural networks in optimization that we introduced in [6], concerns the speed-up of local optimization algorithms. The method is based on the observation that the search algorithm steps into regions of the search domain that have already been "visited" in previous iterations. For a search point which is generated in a region where a reliable RBF neural network can be constructed an expensive function evaluation can be replaced by an RBF approximation. In this paper, the method introduced in [6] is further developed and the number of user-defined parameters is reduced. A new strategy of choosing the ridge parameter λ, as well as a figure of merit measuring whether our evaluation point is well surrounded by data points, are proposed.

II. Radial Basis Function Neural Networks

Radial basis function neural networks [7] are two layer feed-forward networks with radial basis functions as activation functions in the hidden units, and linear activation functions in the output units. The network is fully connected and all connections from the hidden to the output units are weighted. Let us consider the RBF network with p inputs, one output and n Gaussian hidden units

$$\phi_i(\underline{\mathrm{x}}) = \exp\left(-||\underline{\mathrm{c}}_i - \underline{\mathrm{x}}||_2/(2r_i^2)\right) \qquad (i = 1, \ldots, n),$$

where $\underline{\mathrm{c}}_i \in \mathbb{R}^p$ are *centers* and r_i are the *widths* of the RBFs. For our design vector $\underline{\mathrm{x}} \in \mathbb{R}^p$, the output of the network is a linear combination of the basis functions

$$O(\underline{\mathrm{x}}) = \sum_{i=0}^{n} w_i \phi_i(\underline{\mathrm{x}}). \tag{1}$$

This model is analyzed in detail in [7] and it is motivated by the interpolation capabilities of RBF expansions [8]. Generally, as it was shown in [9], Gaussian RBF networks can approximate any continuous mapping $f : \mathbb{R}^p \to \mathbb{R}$, and hence can be used to model the continuous objective function in the optimization process

$$\min f(\vec{x}).$$

In practice, the approximation problem is given in a form of a set of n argument-value pairs $\mathbf{Z} = \{\underline{\mathrm{x}}_k, t_k\}_{k=1}^n$, i.e., by n conditions $f(\underline{\mathrm{x}}_k) = t_k$, $(k = 1, \ldots, n)$. If an RBF neural network is used to model the mapping f, conditions $f(\underline{\mathrm{x}}_k) = O(\underline{\mathrm{x}}_k)$ are requested. Under the assumption that the centers $\underline{\mathrm{c}}_i$ and widths r_i are known, the problem of establishing weights of the network reads in matrix form as

$$\Phi \mathbf{w} = \mathbf{t}, \tag{2}$$

where $\mathbf{t} = [t_1, \ldots, t_n]^T \in \mathbb{R}^n$, $\Phi = [\phi_i(\underline{\mathrm{x}}_k)]_{i,k=1,\ldots,n} \in \mathbb{R}^{n \times n}$ and $\mathbf{w} = [w_1, \ldots, w_n]^T \in \mathbb{R}^n$ is the vector of the final layer weights which are computed.

A learning algorithm of the RBF neural network in our application has two stages. In the first stage, the centers and widths of the RBFs are determined using only inputs $\underline{x}_k$. In the speed-up scheme presented below this stage is limited to assigning centers of RBFs to as many points from the training set as possible. The used procedure prevents setting two centers on points lying too close to each other. Hence the columns in matrix Φ are linearly independent [8] and the numerical rank (cf. [10]) of this matrix is increased. The widths r_i can be set arbitrarily to a fraction of the diameter of the training set, though rather broad widths are required to achieve a good generalization of the data. Here a width equal to the half of the diameter of the training set has been chosen.

The second stage is the calculation of the weights $\mathbf{w}$. Because all RBF centers are different, matrix Φ is non-singular [8]. Generally, however, Φ is very ill-conditioned, and the weights $\mathbf{w}$ obtained via matrix inversion from (2) yield a mapping $O(\underline{x})$ that exhibits oscillatory behaviour in between data points.

To resolve this problem we use the Tikhonov regularization technique [10]. The method replaces equation (2) by the *regularized least squares problem*. Its solution is a function of the regularization parameter λ :

$$\mathbf{w}_\lambda \equiv \min_{\mathbf{w}}\{||\Phi\mathbf{w}-\mathbf{t}||_2+\lambda^2||\mathbf{w}||_2\}. \tag{3}$$

In this formulation λ controls the norm of the solution. In statistics this method is called the *ridge regression*. For a prescribed value of the ridge parameter λ, the optimal weight vector is the solution of the *regularized normal equations* :

$$(\Phi^T\Phi+\lambda^2\mathbf{I})\mathbf{w}=\Phi^T.$$

In practice, however, this equation system – in case of Gaussian RBFs – is still very ill-conditioned and to calculate a regularized solution $\mathbf{w}_\lambda$, *regularized singular value decomposition (SVD)* of the matrix Φ has to be used [10]. The weight vector is expressed by the regularized pseudo-inverse obtained from SVD of Φ as

$$\mathbf{w}_\lambda=\Phi^\dagger\mathbf{t}=\mathbf{V}\Omega_\lambda^{-1}\mathbf{U}^T\mathbf{t},$$

where $\Omega_\lambda^{-1}=\mathbf{diag}(\sigma_1/(\sigma_1^2+\lambda^2),\ldots,\sigma_n/(\sigma_n^2+\lambda^2))$ is the regularized inverse of the singular value matrix $\mathbf{diag}(\sigma_1,\ldots,\sigma_n)$ where $\sigma_1\geq\cdots\geq\sigma_n$. Then the vector $\mathbf{w}_\lambda$ can be expressed as an expansion of the form

$$\mathbf{w}_\lambda=\sum_{i=1}^{n}\frac{\sigma_i}{\sigma_i^2+\lambda^2}(\mathbf{u}_i^T\mathbf{t})\mathbf{v}_i, \tag{4}$$

where the $\mathbf{u}_i$ and $\mathbf{v}_i$ are right and left singular vectors corresponding to the i-th singular value of matrix Φ. The aim of the regularization is to filter out big contributions of the vectors $\mathbf{v}_i$ corresponding to small singular values. The inversion of a small singular value, such as in the case of an ill-conditioned matrix, amplifies oscillations of the approximation (1). If $\lambda>\sigma_i$ the contribution of vectors $\mathbf{v}_i,\mathbf{v}_{i+1},\ldots,\mathbf{v}_n$ in expansion (4) is damped since

$$\frac{\sigma_l}{\sigma_l^2+\lambda^2}\ll\frac{1}{\sigma_l}\qquad(i<l\leq n).$$

The parameter λ can be chosen by means of the *Generalized Cross-Validation* method [11] which is a method, designed for the estimation of λ for mapping noisy data when the data set is big. For field calculation problems, however, statistical errors are negligible, and the data set is rather small. Hence a new method for choosing the ridge parameter λ is presented in the next section.

The advantage of this two-stage learning algorithm, in which parameters of RBFs are treated as hyperparameters, is that it is more efficient than the *back-propagation* method which iteratively optimizes parameters of RBFs and weights of the network (cf. [12]).

III. The Speed-up Scheme for Deterministic Search Algorithms

The RBF neural network approximation method has been implemented into the standard optimization routine of the field computation program [1] according to the following flowchart:

1. Perform some prescribed number of initial steps in the search algorithm.
2. Generate the j-th search point $\underline{x}^{(j)}$ by the search algorithm.
3. Generate a training set $\mathbf{Z}$ for point $\underline{x}^{(j)}$.
4. Judge if a reliable RBF approximator can be built:
   (a) If point $\underline{x}^{(j)}$ is in the reliable region of the search domain then train the network on the set $\mathbf{Z}$ for point $\underline{x}^{(j)}$. If the RBF network was successfully trained, then evaluate the approximation $\tilde{f}(\underline{x}^{(j)})$ and substitute $f(\underline{x}^{(j)})\leftarrow\tilde{f}(\underline{x}^{(j)})$.
   (b) If point $\underline{x}^{(j)}$ is not in the reliable region, or the network cannot be trained successfully, then evaluate $f(\underline{x}^{(j)})$ by means of a numerical field computation.
5. Increment j and go to 2.

The process is stopped when a local minimum of the objective function is found.

The speed-up is obtained by replacing the time consuming numerical calculations of the objective function value by neural network approximations that require only negligible time for construction and evaluation.

A. Initial steps of the search algorithm

The number of function evaluations using the deterministic search algorithm needed to obtain the first training set is determined by the prescribed number of neurons of the RBF approximator to be used. Experience shows that the number of neurons in this application is hardly dependent on the number of optimization parameters. Here $n\sim 30$ was found to be best suited. The number of initial steps was set to $2n$.

B. Construction of the training set

The number of points in the training set $\mathbf{Z}$ is determined by the number of RBF neurons of the network so that each RBF center is assigned to one of the points. In order to capture important information for the function approximation in point $\underline{\mathrm{x}}^{(j)}$, we first include the n nearest deterministically evaluated search points to the training set. Choosing the n nearest points as RBF centers may yield, however, an ill-conditioned matrix Φ since the matrix Φ, will contain two nearly equal columns, if some centers lie too close to each other. To avoid such a situation we seek the set of RBF centers for pairs of centers that lie too close to each other. From such pairs one center is excluded from the set of RBF centers and replaced by the next closest deterministically evaluated point that had not been included yet.

C. Reliability of the approximation

The crucial part of the above scheme is step 4. First it requires a method of detecting whether point $\underline{\mathrm{x}}^{(j)}$ lies in a region in which an accurate RBF approximation can be constructed. We define the following figure of merit :

$$S_{\mathrm{stat}}(\underline{\mathrm{x}}^{(j)}) = \sum_{k=1,i<k}^{n} a_{ki} W_{ki} \Big/ \sum_{k=1,i<k}^{n} W_{ki},$$

with $a_{ki} = \frac{d_{ki}}{(r_k+r_i)/2}$ and $W_{ki} = \frac{1}{r_k+r_i}$, where $d_{ki} = ||\underline{\mathrm{c}}_k - \underline{\mathrm{c}}_i||_2$ and $r_k = ||\underline{\mathrm{x}}^{(j)} - \underline{\mathrm{c}}_k||_2$ (the name S_{stat} stands for *surrounding statistics*). To compute $S_{\mathrm{stat}}(\underline{\mathrm{x}}^{(j)})$ the first stage of RBF construction has to be finished, i.e. all RBF centers have to be assigned. It can be seen that $S_{\mathrm{stat}}(\underline{\mathrm{x}}^{(j)})$ has the following properties: (a) its value is biggest if $\underline{\mathrm{x}}^{(j)}$ lies in a region of the search domain that is surrounded by RBF centers, (b) its value is affected most by centers that are close to point $\underline{\mathrm{x}}^{(j)}$, (c) it is dimensionless and scalable invariant.

From the above it follows that $S_{\mathrm{stat}}(\underline{\mathrm{x}}^{(j)})$ measures if point $\underline{\mathrm{x}}^{(j)}$ is surrounded well by RBF centers. Its value is within the range $(0, 2]$. The larger this number is the better the RBF centers are distributed around point $\underline{\mathrm{x}}^{(j)}$. In figure 1 the value of S_{stat} is plotted for a typical distribution of RBF centers.

D. Building an RBF approximator

For building an RBF approximator, the ridge parameter has to be chosen in a way that the mapping $O(\underline{\mathrm{x}})$ is as smooth as possible in the vicinity of point $\underline{\mathrm{x}}^{(j)}$. Due to the fact that the data in the set $\mathbf{Z}$ is non-noisy we can limit the choice of the parameter λ to the lower part of the singular value spectrum of the matrix Φ, e.g., $\lambda \in [\sigma_{n+1}, \sigma_{\lfloor n/2 \rfloor}]$. Thus this range is scanned and RBF approximators are generated for various values of λ. The SVD of the matrix Φ, however, is computed only once. Then the weight vector $\mathbf{w}_\lambda$ is calculated by means of *regularized singular vector expansion* (4) for each λ. To determine the quality of the constructed approximators we use two measures :

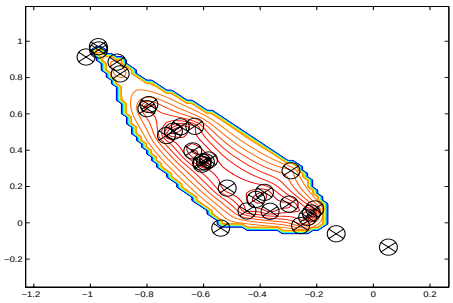

Fig. 1. Contour plot of the value of $S_{\mathrm{stat}}([x, y]) > 1.2$ for the first 50 neurons RBF approximator in a Rosenbrock function optimization (cf. [6]). The maximal value is attained for domain regions well surrounded by RBF centers.

1. the *Normalized Local Mean Square Error* to asses the fit quality:

$$NLMSE_\lambda(\underline{\mathrm{x}}^{(j)}) = \Big(\sum\nolimits_{k=1}^{n} \frac{[O_\lambda(\underline{\mathrm{x}}_k) - t_k]^2}{t_k^2 r_k^2} \Big/ \sum\nolimits_{k=1}^{n} \frac{1}{r_k^2}\Big)^{\frac{1}{2}},$$

2. the *Weighted Gradient Variance* to judge the generalization ability:

$$WGV_\lambda(\underline{\mathrm{x}}^{(j)}) = \sum\nolimits_{k=1}^{n} \frac{||\nabla O_\lambda(\underline{\mathrm{x}}_k) - \vec{M}_\lambda(\underline{\mathrm{x}}^{(j)}||_2^2}{r_k^2} \Big/ \sum\nolimits_{k=1}^{n} \frac{1}{r_k^2},$$

where $\vec{M}_\lambda(\underline{\mathrm{x}}^{(j)})$ is the "*mean gradient*" in point $\underline{\mathrm{x}}^{(j)}$, defined as

$$\vec{M}_\lambda(\underline{\mathrm{x}}^{(j)}) = \sum\nolimits_{k=1}^{n} \frac{\nabla O_\lambda(\underline{\mathrm{x}}_k)}{r_k^2} \Big/ \sum\nolimits_{k=1}^{n} \frac{1}{r_k^2}.$$

Note, that due to the weighting by $1/r_k^2$ all these quantities have a "local character"and depend on the position of $\underline{\mathrm{x}}^{(j)}$. The value of λ is chosen in a way that the corresponding approximator yields a fit quality less than the user-supplied $NLMSE$ threshold ($NLMSE_{\mathrm{thr}} \in [10^{-4}, 10^{-3}]$) and minimizes $WGV_\lambda(\underline{\mathrm{x}}^{(j)})$. If for $x^{(j)}$ the network with $NLMSE_\lambda(\underline{\mathrm{x}}^{(j)}) < NLMSE_{\mathrm{thr}}$ is not found we assume that the RBF network cannot be trained for this point (point 4b. of the scheme). This approach is a kind of discrepancy principle method (cf. [10]). Constraining $NLMSE$ guarantees a good local fit quality on the training set, whereas finding the minimum of WGV chooses the approximator which oscillates the least.

In such a way, objective function approximations with relative errors less than 10^{-3} were obtained in the examples presented below.

IV. Results

In this paper we present two examples of optimization problems showing the performance of the method. The

description of the optimization problem in the design of superconducting magnets for the LHC has been presented in [6]. The geometry consists of a superconducting dipole coil in a non-magnetic stainless steel collar within a ferromagnetic iron yoke. The design variables x_1, x_2, and x_3 denote the half axes of the ellipses defining the surface between collar and iron yoke. In the tables below, the performance of the proposed method is compared to the "purely" deterministic search algorithm. The first row corresponds to the purely deterministic optimization with the EXTREM routine, the second one corresponds to the combination of EXTREM with the RBF approximation scheme. The last column in all tables shows the number of deterministic steps plus the number of RBF approximations needed to converge.

The first problem was a 2-parametric optimization of the short sample field in the CERN coil test facility (CTF). The form of the objective function was given in [6]. The three user-supplied parameters were set to $n = 30$, $S_{\text{stat,thr}} = 1.25$, $NLMSE_{\text{thr}} = 5 \cdot 10^{-4}$. The achieved speed-up is of the order of 30 % (see fig. 2 and table I). As one can see in the table, exactly the same minima were found in both optimizations.

TABLE I
Comparison of the EXTREM algorithm and the speed-up scheme on a 2-parametric optimization (started from [130.,80.]) CTF optimization. Using the presented method (line 2), 31 neural-network approximations have been performed.

Algorithm	x_1	x_2	f_{obj}	No. f. eval.
EXTREM	68.53	75.87	-391.3003	167
RBF app.	68.57	75.86	-391.2974	111 (+31)

The 3-parametric example is an optimization of the half axes of the elliptical inner surface of the iron yoke of the LHC main dipole. For the LHC dipole magnets all field harmonics apart from the dipole component have to vanish. Hence the objective function has been defined as the sum of all "low-order" field harmonics using appropriate weighting factors. By minimizing this function an almost "pure" dipole field was obtained. The setup of parameters was $n = 27$, $S_{\text{stat,thr}} = 1.41$, $NLMSE_{\text{thr}} = 5 \cdot 10^{-4}$. The scheme converged to a point very close to the minimum found by the deterministic search yielding a slightly better objective function value. The achieved speed-up is of the order of 30 % (see table II). As one can see, the scheme allowed to save many deterministic function evaluations that were performed in the vicinity of the minimum by the deterministic search.

TABLE II
Comparison of the EXTREM algorithm and the speed-up scheme on a 3-parametric optimization (started from [88.0,105.0,79.0]) of the LHC main dipole. Using the presented method (line 2), 45 neural-network approximations have been performed.

Algorithm	x_1	x_2	x_3	f_{obj}	No. f. eval.
EXTREM	80.92	99.64	86.02	-23.941	301
RBF app.	80.70	99.40	85.74	-24.179	208 (+45)

V. Conclusions

An improved method using a Gaussian radial basis neural network to speed up search algorithms was presented.

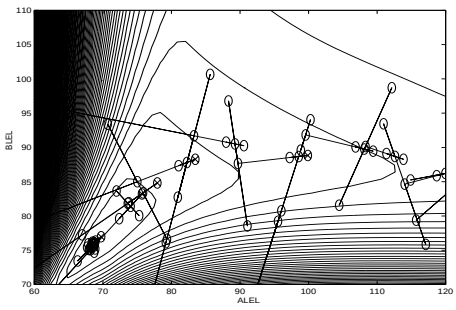

Fig. 2. The path of the 2-parametric doptimization. Started from [130., 80.], the algorithm converged to the minimum of the objective function [68.55, 75.87] after 111 deterministic function evaluations (empty circles) and 31 RBF approximations (crossed circles). The EXTREM algorithm without RBF approximation needed 167 function evaluations. The achieved speed-up is of the order of 30 %.

The method constructs an unbiased RBF approximator using the Tikhonov regularization technique for each point from the search path whose vicinity is sufficiently rich in data points generated in previous iterations. Since the time needed for the construction and evaluation of such an approximator is negligible compared to the deterministic objective function evaluation, a speed-up of the optimization process is achieved. In the paper, figures of merit to measure the local fit quality and the local oscillatory behaviour of the fit were introduced. A method was presented that allows to judge whether a reliable approximation can be obtained. We demonstrated the performance of the proposed scheme on two examples of magnet optimization. In both cases an absolute speed-up of about 30 % was achieved.

References

[1] S.Russenschuck "ROXIE: Routine for the Optimization of Magnet X-section, Inverse Field Calculation and Coil End Design", CERN Yellow Report 99-01, 1999.

[2] H.G.Jacob "Rechnergestützte Optimierung statischer und dynamischer Systeme", Springer-Verlag, Berlin 1982.

[3] A.Alotto *et al.* "A multiquadratics based algorithm for the acceleration of simulated annealing optimization procedures", IEEE Trans. Magn., Vol. 32, No. 3, September 1996, pp. 1198-1201.

[4] U.Pahner, K.Hameyer "Adaptive Coupling of Differential Evolution and Multiquadratics Approximation for the Tunning of the Optimization Process" IEEE Trans. on Magn., Vol. 36, No. 4, July; 2000; pp. 1047-1051.

[5] T.T.Ishikawa, M.Matsunami "A Combine Method for Global Optimization Using Radial Basis Function and Deterministic Approach", IEEE Trans. on Magn., Vol. 35, No. 3,1999.

[6] M.Bazan, S.Russenschuck "Using Neural Networks to Speed Up Optimization Algorithms", Euro. Phys. J., Vol. 12, No. 2, 2000.

[7] T.Poggio, F.Girosi "A Theory of Networks for Approximation and Learning", AI Memo 1140, AI Lab. MIT, 1989.

[8] Ch.A.Micchelli "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions", Constructive Approximation, Vol. 2, 1986, pp. 11-22.

[9] E.J.Hartman, *et al.* "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations", Neural Computation, Vol. 2, 1990, pp. 210-215.

[10] P.Ch.Hansen "Rank-deficient and Discrete Ill-posed Problems", SIAM, Philadelphia 1998.

[11] G.H.Golub, *et al.* "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter", Technometrics, Vol. 21, No. 2, 1979, pp. 215-223.

[12] J.Moody, Ch.Darken, "Fast learning in Networks of locally-Tuned Processing Units", Neural Computation, Vol. 1, 1989, pp. 281-294.