Newton Iteration with Multiquadrics for the Solution of Nonlinear PDEs

Gregory E. Fasshauer

Abstract. Newton iteration is a standard tool for the numerical solution of nonlinear partial differential equations. We show how globally supported multiquadric radial basis functions can be used for this task. One of the insights gained is that the use of coarse meshes during the initial iterations along with a multiquadric parameter which is adjusted with the meshsize increases the efficiency and stability of the resulting algorithm. Some experiments with Nash iteration are also included.

1. Introduction

Since the radial basis function (RBF) method is a meshless method which is comparatively easy to implement it is slowly making its way into the toolbox of researchers and scientists dealing with the numerical solution of partial differential equations. This special issue is a much-needed effort to collect the state-of-the-art in this field.

RBFs are usually grouped into two categories: globally supported and locally supported functions. The advantages of one category over the other have been discussed in many papers (see, e.g., [4, 17] and references therein). In this paper we will concentrate on the use of globally supported functions, and multiquadrics in particular. Table 1 lists some of the "classical" globally supported RBFs. As usual we use $r = \|\cdot\|$ (the Euclidean norm), and c is a parameter to be set by the user. It is well-known that those functions which include the parameter c suffer from a *trade-off principle* (see [18]), i.e., one can adjust the parameter c to increase the approximation power of the basis, but one pays for this by making the problem more and more ill-conditioned. These effects were initially observed for scattered data interpolation, but we will see below that they also occur in the numerical solution of PDEs.

$\phi(r) = \sqrt{r^2 + c^2}$	multiquadrics
$\phi(r) = (r^2 + c^2)^{-1/2}$	inverse multiquadrics
$\phi(r) = r^2 \ln r$	thin plate splines
$\phi(r) = e^{-c^2 r^2}$	Gaussians

Tab. 1. "Classical" RBFs.

We are interested in the numerical solution of a generic nonlinear (elliptic) PDE of the form $\mathcal{L}u = f$ on some domain $\Omega \subset \mathbb{R}^d$. More precisely, we will use nonsymmetric collocation with multiquadrics within an operator Newton iteration to accomplish this. The nonsymmetric collocation method for solving (elliptic) PDEs is due to Kansa [15] and we now briefly review this method.

Let us assume we are given a linear(ized) boundary value problem of the form

$$\begin{aligned} Lu &= f \qquad \text{in } \Omega \subset \mathbb{R}^d, \\ Bu &= g \qquad \text{on } \partial\Omega. \end{aligned} \tag{1}$$

Other boundary conditions can also be added, e.g., it is possible to use Neumann conditions on one part of the boundary and Dirichlet conditions on the remaining part. Kansa (see [15]) proposed to find an approximate solution u^* from the function space $\mathcal{S} = \text{span}\{\phi(\|\cdot - x_j\|) : x_j \in \mathcal{X}\}$, where ϕ is a globally supported RBF and $\mathcal{X}$ is some set of n (scattered) nodes or *collocation points* in Ω, i.e., u^* is given by the expansion

$$u^*(x) = \sum_{j=1}^{n} \alpha_j \phi(\|x - x_j\|). \tag{2}$$

The coefficients α_j are found by solving a system of linear equations of the form $A\boldsymbol{\alpha} = \boldsymbol{b}$, where the matrix A, the coefficient vector $\boldsymbol{\alpha}$ and the right-hand side vector $\boldsymbol{b}$ have the block forms

$$A = \begin{bmatrix} L\Phi \\ B\Phi \end{bmatrix}, \qquad \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_I \\ \boldsymbol{\alpha}_B \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}. \tag{3}$$

If n_I denotes the number of collocation points in the interior of Ω and n_B the number of those on the boundary then $L\Phi$ is an $n_I \times n$ matrix with entries $L\Phi_{ij} = L\phi(\|x - x_j\|)|_{x=x_i}$, $B\Phi$ is an $n_B \times n$ matrix with entries $B\Phi_{ij} = B\phi(\|x - x_j\|)|_{x=x_i}$, and $\boldsymbol{\alpha}_I$ and $\boldsymbol{\alpha}_B$ are those coefficients in (2) corresponding to interior and boundary collocation points, respectively. The pieces $\boldsymbol{f}$ and $\boldsymbol{g}$ of $\boldsymbol{b}$ are obtained analogously by evaluating f and g at the collocation points. In the literature this method is called a *collocation method* because the conditions (1) are enforced pointwise at the points of $\mathcal{X}$. We will therefore refer to A as the *collocation matrix*.

In [2] we compared this nonsymmetric collocation approach with a symmetric one for which it is known that the matrix A is invertible for (distinct) collocation points in arbitrary positions provided ϕ is taken from an appropriate class of functions (e.g., strictly positive definite functions, but also multiquadrics or thin plate splines if additional care is taken). For the nonsymmetric method it was recently shown that there exist configurations of collocation points for which A is singular [13]. Since the nonsymmetric approach is easier to implement (especially for nonlinear problems), and since singular configurations seem to be quite rare (cf. [13]), we have decided to use this method in the present paper. The characterization of certain configurations of collocation points which guarantee nonsingularity of the collocation matrix for the nonsymmetric approach is an interesting open problem.

In Sect. 2 we outline the operator Newton iteration and also explain the concept of Nash iteration. We specialize this discussion to linear problems in Sect. 3 and show there that a multilevel implementation is the wrong tool if one intends to use globally supported RBFs with fixed parameter c. For linear problems the multilevel method needs to be combined with either locally supported functions (varying the support size from level to level, see, e.g., [4, 6]), or with globally supported functions which contain a parameter that can be varied with the levels (cf. Tables 3 and 4 below). The situation is different, however, for nonlinear problems and we study these in the remainder of the paper. In Sect. 4 we formulate our algorithm for the operator Newton method and introduce the nonlinear PDE we use as a test problem. Numerical tests with single-level and multilevel Newton iteration are reported in Sect.5. We show there that multilevel Newton iteration is superior to iteration on a single mesh. In particular, the use of coarse meshes during initial iterates results in a faster and more accurate method. Also, scaling the multiquadric parameter c with the meshsize has a positive effect on the stability of the iteration. In Sect. 6 we add a few experiments with Nash iteration in which a smoothing operation is added to the Newton algorithm. This is shown to be beneficial, enabling us to carry the iteration to finer meshes than without the smoothing in which case the algorithm brakes down due to ill-conditioning (cf. Tables 9 and 13). The paper concludes with some remarks in Sect. 7.

2. Operator Newton Iteration

In [9] we presented a detailed discussion of a general framework for operator Newton iteration. That work was based on earlier results by Moser [16], Hörmander [12] and Jerome [14].

2.1. Newton Iteration

Given the nonlinear elliptic PDE $\mathcal{L}u = f$ one starts with an initial guess u_0 for the solution and then obtains a sequence $\{u_k\}$ of approximate solutions via the following Newton iteration formula

$$u_k = u_{k-1} - T_{h_k}(u_{k-1})F(u_{k-1}), \qquad k \geq 1. \tag{4}$$

Here the operator F defined by $F(u) = \mathcal{L}u - f$ determines the *residual*, and T_{h_k} is a *numerical inversion operator* (i.e., T_{h_k} approximates $(F')^{-1}$ on a mesh with meshsize h_k and may depend – especially for nonlinear problems – on the previous iterate u_{k-1}). Thus the Newton method is used to find the zero of the operator

F. The discussion in [9, 14] is quite general and allows for many different kinds of numerical inversion techniques. As mentioned in the introduction we will realize T_{h_k} via nonsymmetric multiquadric RBF collocation in this paper.

2.2. Nash Iteration

The main result of [9, 14] is that for many numerical inversion methods the Newton method (4) will suffer from a *loss of derivatives*, i.e., not achieve full (quadratic) convergence, and that if an appropriate smoothing is incorporated at each iteration, then almost full (superlinear) convergence can be achieved. Thus, Newton iteration with smoothing is described by

$$u_k = u_{k-1} - S_{t_k} T_{h_k}(u_{k-1}) F(u_{k-1}), \qquad k \geq 1, \tag{5}$$

where S_{t_k} is an appropriate *smoothing operator* with smoothing parameter t_k.

The smoothed version (5) was referred to in our earlier papers [4, 7, 9] as *Nash iteration* and in [7] a number of different interpretations and implementations of this general concept were studied. The algorithms used in the present paper correspond to the so-called “simple algorithm” of [7, Alg. 2.1] and a slight modification thereof in which the computational meshes $\mathcal{X}_k$ are all identical.

3. Globally Supported RBFs and Linear Problems

In this section we briefly consider linear problems of the form $Lu = f$. In particular, we will give some numerical examples below with $L = I$ the identity operator, i.e., for scattered data interpolation.

3.1. Multilevel Newton Iteration and Globally Supported RBFs

We showed in [9] that the operator Newton method (4) for linear problems can be viewed (among other possibilities) as a multilevel interpolation algorithm as first proposed for locally supported RBFs by Floater and Iske [10]. Here is a description of this algorithm.

Algorithm 1.

1) *Create nested point sets* $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_K \subset \Omega$, *and let* $u_0 = 0$.
2) *For* $k = 1, 2, \ldots, K$
   a) *Solve the linear problem*

   $$Lv = f - Lu_{k-1} \quad \text{on } \mathcal{X}_k.$$

   b) *Perform the Newton update of the* k*-th iterate*

   $$u_k = u_{k-1} + v.$$

By linearity it is clear that $Lu_K = f$ on $\mathcal{X}_K$. In our earlier work on multilevel RBF approximation [3, 4, 5, 6, 7, 9] we concentrated on the use of locally supported functions within Algorithm 1. One of the reasons for this is that, for linear problems, one does not gain anything by applying globally supported functions in a multilevel framework. More precisely, if the meshes $\mathcal{X}_k$ at the different levels are nested and the parameter c in the globally supported functions (see Table 1) is kept fixed through all levels k, then the function spaces $\mathcal{S}_k = \text{span}\{\phi(\|\cdot - x_j^{(k)}\|) : x_j^{(k)} \in \mathcal{X}_k\}$ are also nested. (For this discussion thin plate splines can be treated the same as the other globally supported RBFs with fixed c.) Consequently, the richest space $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$, which is used when all updates have been performed at the finest level, is equal to the space $\mathcal{S}_K$ on the finest mesh $\mathcal{X}_K$. Thus, a direct fit at the finest level K uses the same approximation space, and will therefore yield the same quality of fit, as the multilevel algorithm using all of the meshes $\mathcal{X}_k$, $k = 1, \ldots, K$. However, the multilevel algorithm requires all the additional (unnecessary) work on the coarser meshes.

It therefore follows

Theorem (Rule 1). *Consider a linear problem of the form $Lu = f$ on Ω, let $\mathcal{X}_1, \ldots, \mathcal{X}_K$ be a nested sequence of sets of collocation points in Ω, and let ϕ be one of the globally supported RBFs of Table 1 with fixed parameter c for all $k = 1, \ldots, K$ (or a thin plate spline). Then the approximate solution u_K obtained by Algorithm 1 is the same as the solution of the problem $Lu = f$ on the single mesh $\mathcal{X}_K$ using the space $\mathcal{S}_K$, i.e., the use of globally supported RBFs with fixed value of c within a multilevel Newton algorithm for linear problems is pointless.*

If one varies the parameter c with the levels k then the function space $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$ used for the final fit with a multilevel algorithm will be richer than the space $\mathcal{S}_K$ used directly for the finest level $\mathcal{X}_K$ alone. This is clear since the spaces $\mathcal{S}_k$, $k = 1, \ldots, K$, are no longer nested. This implies that, for a "good" sequence of c-values, one can expect to obtain more accurate fits using the multilevel framework.

This is summarized in

Corollary (Rule 2). *The multilevel Newton algorithm for linear problems has the potential of being more accurate than a direct fit if the parameter c is varied with the levels.*

There has been much work on finding an "optimal" value for the parameter c in the literature, but so far this is still an open question. For the numerical examples later on we have chosen one value of c that works reasonably well compared to a limited number of other choices. Kansa (see e.g., [15]) is quite successful using values of c which vary from node to node. This, however, leads to an approximation space which is no longer generated by a single basis function, and thus much more difficult to analyze theoretically.

3.2. Numerical Illustrations

We now illustrate Rules 1 and 2 with a scattered data fitting problem in $\mathbb{R}^2$. We use the well-known Franke function

$$
\begin{aligned}
f(\xi,\eta) = {} & \frac{3}{4}\exp\left(-\frac{(9\xi-2)^2+(9\eta-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9\xi+1)^2}{49} - \frac{(9\eta+1)^2}{10}\right) \\
& + \frac{1}{2}\exp\left(-\frac{(9\xi-7)^2+(9\eta-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9\xi-4)^2-(9\eta-7)^2\right)
\end{aligned}
$$

on the unit square as our test function. Here we have used (ξ,η) to denote the two Cartesian coordinates of the point $x \in \mathbb{R}^2$. The approximate fit to the test function is obtained by letting the Newton update v in iteration k of Algorithm 1 (where $L = I$ for this example) be of the form

$$
v(x) = \sum_{j=1}^{n^{(k)}} \alpha_j^{(k)} \phi(\|x - x_j^{(k)}\|), \tag{6}
$$

with ϕ a multiquadric RBF. The point sets $\mathcal{X}_k$ are given by $(2^k+1)^2$ equally spaced points in the unit square and are therefore nested with "meshsize" $h_k = 1/2^k$.

In all of our numerical examples we will list ℓ_2-errors calculated on a fine evaluation mesh via

$$
e_k = \left[\frac{1}{m}\sum_{j=1}^{m} |u(x_j) - u_k(x_j)|^2\right]^{1/2}, \qquad k = 1,\ldots,K,
$$

where m is the number of points in the evaluation mesh (usually one more refinement of the mesh $\mathcal{X}_K$). There are two different "rates" stated in the experiments reported in this paper. The h-rates are computed via

$$
h\text{-rate}_k = \ln\left(\frac{e_{k-1}}{e_k}\right)/\ln 2, \qquad k = 2,\ldots,K,
$$

and they therefore indicate the rate of convergence with respect to the meshsize, i.e., the algorithm can be observed to be $\mathcal{O}(h^{\text{rate}})$ convergent (since $h_{k-1}/h_k = 2$). The k-rates are computed via

$$
k\text{-rate}_k = \frac{\ln e_k}{\ln e_{k-1}}, \qquad k = 2,\ldots,K,
$$

and they therefore indicate the rate of convergence with respect to the iteration, i.e., errors of successive approximations are related via $e_k = e_{k-1}^{\text{rate}}$, and so k-rate > 1 indicates a superlinearly convergent algorithm.

For the first example (see Table 2 and Figures 1 and 2) we let the multiquadric parameter $c = 0.3$ throughout. The errors and rates in Table 2 indicate the well-known convergence behavior of multiquadrics. The last row in the table also shows that the parameter c is too large for this point set and the matrix is so ill-conditioned that the approximation is starting to be contaminated by roundoff errors. According to Rule 1 there is no difference between using the multilevel Algorithm 1 and a direct fit on n points (this can also be observed numerically).

n	ℓ_2-error	h-rate
9	$1.556543\,10^{-1}$	
25	$2.524195\,10^{-2}$	2.62
81	$3.860265\,10^{-3}$	2.71
289	$3.765404\,10^{-5}$	6.68
1089	$6.791556\,10^{-8}$	9.11
4225	$7.036481\,10^{-7}$	-3.37

Tab. 2. MQ fit to Franke's function with fixed value of $c = 0.3$.

The surfaces in Figures 1 and 2 are rendered on the next finer grid, respectively, i.e., the 9 point fit is rendered on 25 points, the 25 point fit on 81, etc.. We have only shown the first four fits since once we reach 289 points there is no visual difference between the approximate fits and the exact function.

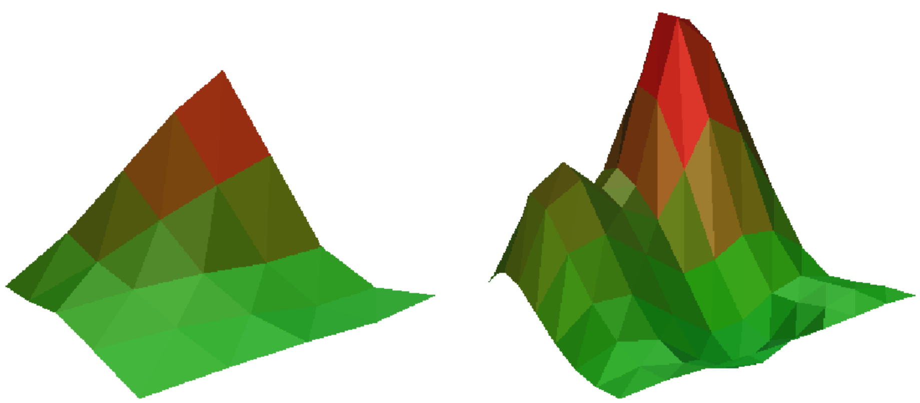

Fig. 1. MQ fit with $c = 0.3$ to Franke's function with 9 and 25 points.

In order to illustrate Rule 2 we repeat the above example, but now take $c = 2/\sqrt{n}$. This time we list what happens with the multilevel algorithm (Table 3) and compare this to the results obtained by computing the approximation directly in one step on the set $\mathcal{X}_k$ (listed in Table 4). Since the results do not differ much from the previous example we did not include any figures.

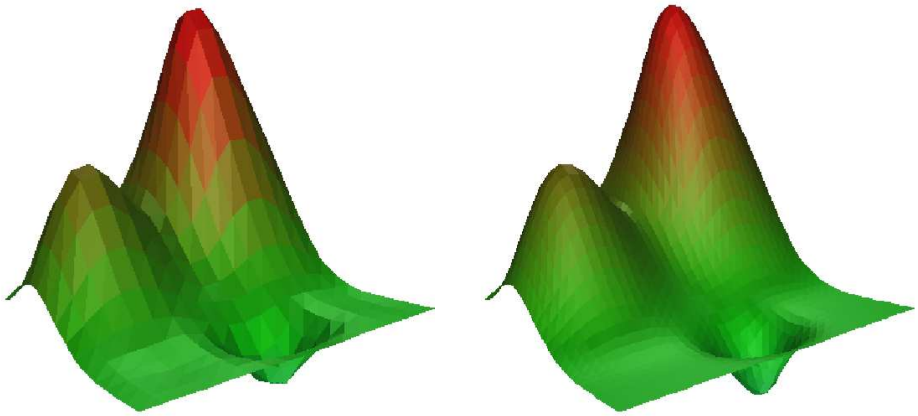

Fig. 2. MQ fit with $c = 0.3$ to Franke's function with 81 and 289 points.

n	c	ℓ_2-error	h-rate
9	.667	$1.589896\,10^{-1}$	
25	.400	$2.800912\,10^{-2}$	2.50
81	.222	$4.120900\,10^{-3}$	2.76
289	.118	$1.541303\,10^{-4}$	4.74
1089	.061	$1.946314\,10^{-5}$	2.99
4225	.031	$2.595578\,10^{-6}$	2.91

Tab. 3. Multilevel MQ fit to Franke's function with variable value of $c = 2/\sqrt{n}$.

n	c	ℓ_2-error	h-rate
9	.667	$1.589896\,10^{-1}$	
25	.400	$2.778591\,10^{-2}$	2.52
81	.222	$3.893510\,10^{-3}$	2.84
289	.118	$9.704496\,10^{-5}$	5.33
1089	.061	$3.015079\,10^{-5}$	1.69
4225	.031	$8.477579\,10^{-6}$	1.83

Tab. 4. Direct MQ fit to Franke's function with variable value of $c = 2/\sqrt{n}$.

Note that, with the varying parameter c, up to 289 points the direct approach is more accurate, but then the multilevel approach does a better job. This is due to the fact that the matrices for the denser point sets become increasingly ill-conditioned (even with the adjusted c-value) and therefore the direct fits with 1089 or 4225 points are likely to be inaccurate. With the multilevel algorithm the fits on the finer grids act only as "corrections" to the coarse grid fits computed earlier. This agrees with the philosophy of the multilevel algorithm, and here is where the richer function space pays off.

In a sense our earlier work with locally supported functions falls into the situation described by Rule 2, also. Only there the role of the parameter c is played by the support radius.

4. Newton Iteration for Nonlinear PDEs and the Test Problem

4.1. The Algorithm

We now state an algorithmic version of the smoothed Newton iteration (5) for a nonlinear PDE $\mathcal{L}u = f$.

Algorithm 2.

1) *Create nested point sets $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_K \subset \Omega$, and pick an initial guess u_0.*
2) *For $k = 1, 2, \ldots, K$*
   a) *Solve the linearized problem*

   $$L_{u_{k-1}} v = f - \mathcal{L}u_{k-1} \quad \text{on } \mathcal{X}_k.$$

   b) *Perform the smoothing of the Newton update*

   $$\widetilde{v} = S_{t_k} v.$$

   c) *Update the previous approximation*

   $$u_k = u_{k-1} + \widetilde{v}.$$

Here $L_{u_{k-1}}$ is the linearization of the nonlinear differential operator $\mathcal{L}$ at u_{k-1} and S_{t_k} is the smoothing operator mentioned in Sect. 2. In this section we will not employ any smoothing (i.e., $S_{t_k} = I$) and in that case refer to the algorithm as Newton iteration. In the next section, however, we will add the smoothing step and refer to the procedure as Nash iteration. More details of the smoothing operator will be presented in Sect. 5.

4.2. The Test Problem

For the numerical experiments presented in the remainder of this paper we use the nonlinear PDE

$$\begin{aligned} -\varepsilon^2 \nabla^2 u - u + u^3 &= f \qquad \text{in } \Omega = (0,1) \times (0,1), \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{7}$$

This equation was also used in [6, 8]. The right-hand side f is chosen so that (7) has an analytic solution of the form

$$u(\xi, \eta) = \psi(\xi)\psi(\eta) \tag{8}$$

with

$$\psi(t) = 1 + e^{-1/\varepsilon} - e^{-t/\varepsilon} - e^{(t-1)/\varepsilon}. \tag{9}$$

Again (ξ, η) denote the Cartesian coordinates of $x \in \mathbb{R}^2$, and the parameter ε determines the size of the boundary layers near the edges of the domain Ω. We use a value of $\varepsilon = 0.1$ in our experiments below. A more detailed discussion of this problem is given in [8].

For this model problem the linearization $L_{u_{k-1}}$ of $\mathcal{L}$ is given by

$$L_{u_{k-1}} v = -\varepsilon^2 \nabla^2 v + (3u_{k-1}^2 - 1)v \tag{10}$$

and therefore the equation to be solved in step 2a of Algorithm 2 is of the form

$$-\varepsilon^2 \nabla^2 v + (3u_{k-1}^2 - 1)v = f + \varepsilon^2 \nabla^2 u_{k-1} + u_{k-1} - u_{k-1}^3.$$

Each time we perform step 2a we have to solve the following linear system arising from the nonsymmetric collocation procedure (1)–(3):

$$\begin{aligned} \sum_{j=1}^{n^{(k)}} \alpha_j^{(k)} L_{u_{k-1}} \left[\phi(\|x - x_j^{(k)}\|)\right]_{x = x_i^{(k)}} &= f + \mathcal{L} u_{k-1}(x_i^{(k)}), \quad i = 1, \ldots, n_I^{(k)}, \\ \sum_{j=1}^{n^{(k)}} \alpha_j^{(k)} \phi(\|x_i^{(k)} - x_j^{(k)}\|) &= 0, \quad i = n_I^{(k)} + 1, \ldots, n^{(k)}. \end{aligned} \tag{11}$$

In order to keep the following manageable we introduce the notation

$$\begin{aligned} \phi_{ij} &= \phi(\|x_i^{(k)} - x_j^{(k)}\|), \\ [\nabla^2 \phi]_{ij} &= \left(\frac{\partial^2}{\partial \xi^2} + \frac{\partial^2}{\partial \eta^2}\right) \phi(\|x - x_j^{(k)}\|)\big|_{x = x_i^{(k)}}, \\ u_i &= u_{k-1}(x_i^{(k)}), \\ [\widetilde{\nabla}^2 u]_i &= \frac{u_i^{(-x)} + u_i^{(-y)} - 4u_i + u_i^{(+x)} + u_i^{(+y)}}{h^2}. \end{aligned} \tag{12}$$

In the last formula we have used additional abbreviations of the form $u_i^{(+x)}$. This denotes u_{k-1} evaluated at the neighbor (in the positive x-direction on the evaluation mesh) of $x_i^{(k)}$. Therefore, the last equation in (12) simply denotes the usual discretization of the Laplacian of u_{k-1} computed on the (very fine) evaluation mesh assumed to have meshsize h here.

Now, using (10) and (12), (11) becomes

$$\begin{aligned} \sum_{j=1}^{n} \alpha_j \left[-\varepsilon^2 [\nabla^2 \phi]_{ij} + (3u_i^2 - 1)\phi_{ij}\right] &= f + \varepsilon^2 [\widetilde{\nabla}^2 u]_i + u_i - u_i^3, \quad i = 1, \ldots, n_I, \\ \sum_{j=1}^{n} \alpha_j \phi_{ij} &= 0, \quad i = n_I + 1, \ldots, n_I + n_B = n, \end{aligned} \tag{13}$$

where, for transparency, we have omitted the index k on the quantities n, n_I, n_B, and α_j. The Newton update $\widetilde{v} = v$ used in step 2c of Algorithm 2 is then given by

$$v(x) = \sum_{j=1}^{n^{(k)}} \alpha_j^{(k)} \phi(\|x - x_j^{(k)}\|).$$

In our experiments we use multiquadrics (see Table 1), and take the function

$$u_0(\xi, \eta) = 16\xi(1-\xi)\eta(1-\eta) \tag{14}$$

as our initial guess. The sets of collocation points $\mathcal{X}_k$ are as in Sect. 3.

We would also like to point out that the discretization of the Laplacian introduced in (12) is designed for a grid. Note that this, however, does not mean that the computational meshes can not consist of scattered points. The discretized Laplacian is only required on a fine evaluation mesh which is independent of the computational meshes. If for some reason one desires to have a completely "mesh-free" implementation, then it is also possible to implement this part exactly by differentiating the basis functions in the expansion of u_{k-1} and then evaluating the resulting expansion at x_i.

5. Numerical Experiments for Nonlinear PDEs using Newton Iteration

It is clear that Rules 1 and 2 do not apply to nonlinear problems. Now one needs to use a Newton method to overcome the nonlinearity of the problem. There are, however, many possible ways of implementing the iterations (cf. [7]). We study two of these possibilities:

- iteration on a single fixed mesh,
- multilevel iteration for which the mesh is refined from one iteration to the next.

5.1. Newton Iteration on a Single Mesh

The standard approach to doing Newton iteration for a nonlinear PDE is to fix a computational mesh and then iteratively improve an initial guess via Newton's method (4) on this mesh. We present the results of two sets of five experiments each following this philosophy. In the first set (see Tables 5 and 6) the multiquadric parameter is kept fixed at $c = 0.3$. For the second set (see Tables 7 and 8 as well as Figures 3 and 4) the multiquadric parameter is adjusted according to the number of points used, i.e., $c = 2/\sqrt{n}$.

Tables 5 and 7 each contain the information for five separate experiments. In each experiment we fixed the computational mesh $\mathcal{X}$ to have n points. Then we run Algorithm 2 (using only one point set) until convergence. The entries in the K-column indicate at which iteration this occurs. The ℓ_2-error listed is the resulting error in that iteration. The column labeled "smallest ℓ_2-error" along with the k column shows that a slightly smaller error is usually reached at an earlier stage of the iteration. This is a clear sign that the Newton iteration is not converging optimally. The "rate" column compares the different experiments in the $\mathcal{O}(h^{\text{rate}})$ sense.

n	ℓ_2-error	K	h-rate	smallest ℓ_2-error	k	L_2-cond(A)
9	$2.121533\,10^{-1}$	5		$2.092175\,10^{-1}$	1	$\mathcal{O}(10^2)$
25	$1.283499\,10^{-1}$	6	0.73	$1.049715\,10^{-1}$	1	$\mathcal{O}(10^3)$
81	$4.896277\,10^{-2}$	7	1.39	$3.189209\,10^{-2}$	2	$\mathcal{O}(10^5)$
289	$6.878832\,10^{-3}$	7	2.83	$6.684559\,10^{-3}$	4	$\mathcal{O}(10^{10})$
1089	NA	NA	NA	$8.123789\,10^{-2}$	2	$\mathcal{O}(10^{19})$

Tab. 5. Newton iteration on one grid with fixed value of $c = 0.3$.

We can observe that the single grid iteration idea basically works. However, the choice of c causes the collocation matrix for 1089 points to have an L_2-condition number on the order of 10^{19} (see the rightmost column of Table 5). Therefore the breakdown of the algorithm (indicated by the NA entries) is not surprising. The individual iterations on 1089 points are listed separately in Table 6. We can see a nice improvement from step one to step two, but then the algorithm blows up.

k	ℓ_2-error	k-rate
1	$2.569520\,10^{-1}$	
2	$8.123789\,10^{-2}$	1.84
3	$1.605847\,10^{1}$	-1.11

Tab. 6. Newton iteration on 1089 points with $c = 0.3$.

In the next set of experiments we vary the choice of $c = 2/\sqrt{n}$ with the mesh (see Tables 7 and 8).

n	c	ℓ_2-error	K	h-rate	smallest ℓ_2-error	k	L_2-cond(A)
9	.667	$2.153155\,10^{-1}$	5		$2.119097\,10^{-1}$	1	$\mathcal{O}(10^2)$
25	.400	$1.227111\,10^{-1}$	6	0.81	$9.526223\,10^{-2}$	1	$\mathcal{O}(10^4)$
81	.222	$5.444689\,10^{-2}$	7	1.17	$3.475364\,10^{-2}$	2	$\mathcal{O}(10^5)$
289	.118	$1.631439\,10^{-2}$	7	1.74	$1.173533\,10^{-2}$	3	$\mathcal{O}(10^5)$
1089	.061	$4.431649\,10^{-3}$	7	1.88	$4.249423\,10^{-3}$	4	$\mathcal{O}(10^6)$

Tab. 7. Newton iteration on one grid with variable value of $c = 2/\sqrt{n}$.

In most cases the entries in Table 7 are similar to those in Table 5. The errors are slightly higher. However, the problems are better conditioned, and thus we also obtain a feasible solution on 1089 points. This case is again separately illustrated in Table 8. Superlinear convergence behavior is evident in the first few steps.

k	ℓ_2-error	k-rate
1	$2.444807\,10^{-1}$	
2	$7.297524\,10^{-2}$	1.86
3	$8.079207\,10^{-3}$	1.84
4	$4.249423\,10^{-3}$	1.13
7	$4.431649\,10^{-3}$	0.99

Tab. 8. Newton iteration on 1089 points with $c = 0.061$.

The approximate solutions on 1089 points using $c = 0.061$ are illustrated by Figures 3 and 4 which also contain a plot of the initial guess (14) and the solution (8) and (9).

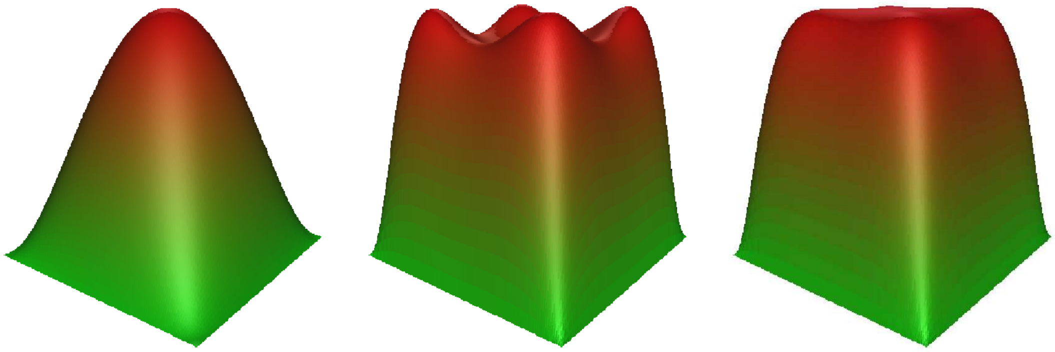

Fig. 3. Newton iteration on 1089 points with $c = 0.061$: initial guess and iterates 1 and 2.

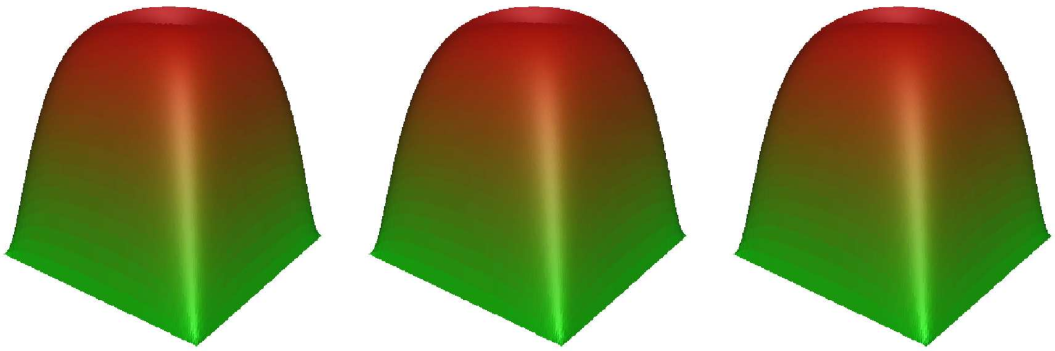

Fig. 4. Newton iteration on 1089 points with $c = 0.061$: iterates 3, 4 and exact solution.

5.2. Multilevel Newton Iteration

A less traditional interpretation of Newton iteration is one for which the computational grids change from one iteration to the next as stated in Algorithm 2. This is what the theory for Nash iteration in [9, 14] requires. We will see that this approach is superior to the standard approach presented in Sect. 5.1 in that it is more efficient and produces more accurate results.

Again we consider the case of fixed $c = 0.3$ and variable $c = 2/\sqrt{n}$. In Table 9 and Figures 5 and 6 we present the results for the fixed c case. Now the entries in the table represent one single experiment. Each row can be interpreted as the result of one iteration of the Newton method. As before we have listed errors and condition numbers. However, now we include both h- and k-rates. As earlier, with $c = 0.3$, the matrix for 4225 points is so ill-conditioned that the process breaks down. Also, the time to solve the last iteration on a desktop PC approaches 2 hours. We note that the times listed here are representative for computations with the same matrix size in other sections of this paper.

k	n	ℓ_2-error	h-rate	time	L_2 cond(A)	k-rate
1	9	$2.395110\,10^{-1}$		0	$1.103085\,10^{2}$	
2	25	$1.010451\,10^{-1}$	1.25	0	$2.622604\,10^{3}$	1.60
3	81	$4.551377\,10^{-2}$	1.15	0	$7.922859\,10^{5}$	1.35
4	289	$5.049607\,10^{-3}$	3.17	1	$2.408632\,10^{10}$	1.71
5	1089	$1.154054\,10^{-4}$	5.45	63	$2.833023\,10^{19}$	1.71
6	4225	$4.851815\,10^{-2}$	-8.72	6271	NA	0.33

Tab. 9. Multilevel Newton iteration with fixed value of $c = 0.3$.

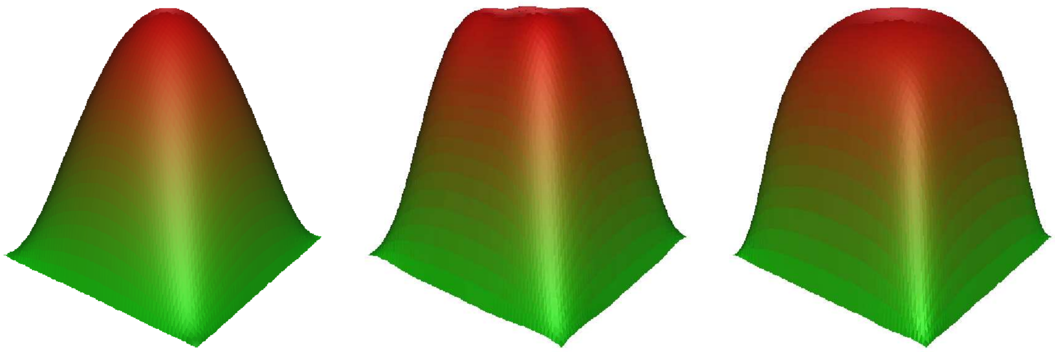

Fig. 5. Multilevel Newton iteration with $c = 0.3$: iterates 1, 2 and 3.

We can see that the more stable algorithm is obtained by adapting the multiquadric parameter c to the mesh size. Moreover, the algorithm has superlinear convergence as the k-rate columns indicate (see Table 10).

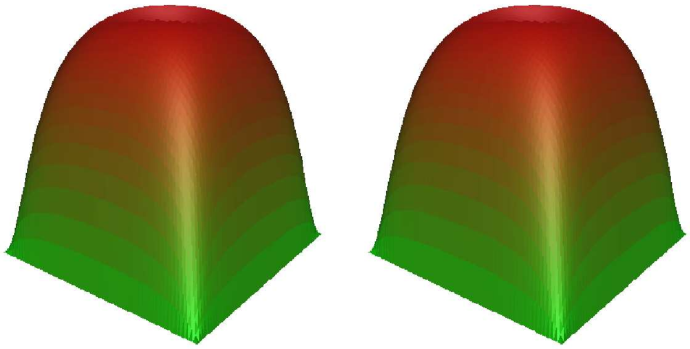

Fig. 6. Multilevel Newton iteration with $c = 0.3$: iterates 4 and 5.

k	n	c	ℓ_2-error	h-rate	time	L_2 cond(A)	k-rate
1	9	.667	$2.423852\,10^{-1}$		0	$9.105692\,10^{2}$	
2	25	.400	$9.260243\,10^{-2}$	1.39	0	$1.015451\,10^{4}$	1.68
3	81	.222	$5.068026\,10^{-2}$	0.87	0	$1.062094\,10^{5}$	1.25
4	289	.117	$1.343129\,10^{-2}$	1.92	1	$9.078648\,10^{5}$	1.45
5	1089	.061	$3.196261\,10^{-3}$	2.07	69	$8.745925\,10^{6}$	1.33
6	4225	.031	$6.223060\,10^{-4}$	2.36	6359	NA	1.28

Tab. 10. Multilevel Newton iteration with variable value of $c = 2/\sqrt{n}$.

The most interesting insight of this section comes when we compare the entries in Tables 9 and 10 with those of Tables 5 and 7. We see that the multilevel algorithm performs better than the one-grid Newton iteration. Not only are the resulting errors smaller, but the amount of work needed to obtain these smaller errors is less. For example with $c = 0.061$ on 1089 points it takes the multilevel Newton iteration 69 seconds to find an approximate solution whose ℓ_2-error is $3.196261\,10^{-3}$ (see Table 10). With the single-grid version the error on 1089 points is $4.431649\,10^{-3}$ and it requires 7 iterations, each taking about 65 seconds (see Table 7). For $c = 0.3$ the situation is even more extreme since the multilevel algorithm with $n = 1089$ produces a very accurate solution with error $1.154054\,10^{-4}$ in only 63 seconds (see Table 9), whereas the single-grid version blows up after 3 iterations and about 200 seconds (see Table 5).

This offers the following insight:

Observation. *For nonlinear problems it is possible to use coarse grids during the initial iterates without sacrificing accuracy. This leads to a more efficient implementation of the operator Newton algorithm.*

6. Numerical Experiments with Nash Iteration

In our earlier paper [9] we showed that the convergence of the Newton algorithm can be improved by the use of an additional smoothing of the Newton update in each iteration (see Algorithm 2). That method is then called Nash iteration. In this section we present a few experiments using a smoothing based on explicit time stepping.

6.1. Harmonic Smoothing

To obtain the harmonic smoothing operator used below we start with the classical heat equation in $\mathbb{R}^2$

$$s_t(x,t) = \nabla_x^2 s(x,t), \qquad x \in \Omega = (0,1)^2, \quad t > 0, \tag{15}$$

to which we add the Dirichlet boundary condition $s(x,t) = 0$ for x on the boundary of Ω. Since we intend to do a smoothing of the Newton update v (see step 2b of Algorithm 2 or (5)) we choose the initial condition

$$s(x,0) = v(x), \qquad x \in [0,1]^2.$$

In our implementation we perform the smoothing operation on the evaluation grid and therefore can discretize the Laplacian in (15) with the classical stencil as in (12). If we use a forward Euler method for the time discretization we obtain the iteration formula

$$\frac{s(x,t_{i+1}) - s(x,t_i)}{\Delta t} = [\widetilde{\nabla}^2 s](x,t_i), \qquad i = 0,1,\ldots, \tag{16}$$

where $\Delta t = t_{i+1} - t_i$ and $t_0 = 0$.

Now the smoothing of v is accomplished by performing a certain number of Euler steps, and the resulting quantity is assigned to $\widetilde{v}(x)$. If N smoothing steps (=Euler steps) have been performed then from (16)

$$\widetilde{v}(x) = s(x, N\Delta t) = \left(I + \Delta t \widetilde{\nabla}^2\right)^N s(x,t_0) = \left(I + \Delta t \widetilde{\nabla}^2\right)^N v(x).$$

Therefore, in the sense of (5) in Sect. 2, we have

$$S_{t_k} = \left(I + \Delta t \widetilde{\nabla}^2\right)^N, \tag{17}$$

and the smoothing parameter t_k corresponds to the step size Δt chosen for the Euler method. The characteristics of the smoothing are determined by the ratio of $\Delta t/h^2$. An analysis similar to the one given in [7] shows that $\Delta t/h^2 = 1/6$ leads to a maximum smoothing of the high-frequency modes. This corresponds to the "damped Jacobi" smoother used in some multigrid algorithms [11]. More details of this (as well as a related biharmonic) smoother are given in [7].

6.2. Nash Iteration on a Single Grid

As explained earlier, the difference between Newton iteration and Nash iteration is the addition of an appropriate smoothing step as postconditioner at each iteration of the Newton method (cf. (4) and (5)). We now take a look at how the harmonic smoothing (17) affects the single-grid Newton iteration (cf. Tables 7 and 8 in Sect. 5.1). The basic setup is the same as in Sect. 5.1. We use nonsymmetric multiquadric collocation with $c = 0.061$ and pick the computational grid $\mathcal{X}$ to consist of $n = 1089$ equally spaced points in the unit square. Then we perform 10 steps of the single-grid version of Algorithm 2. The resulting ℓ_2-errors and k-rates are displayed in Tables 11 and 12.

In the first version of the experiment we pick a constant number of $N = 100$ smoothing steps in each iteration of the Newton/Nash method. The effects of this smoothing are minor. Without smoothing the algorithm reaches its smallest ℓ_2-error in iteration $k = 4$ and then the approximation deteriorates slightly until it reaches its limit in iteration 7 (see Table 8). With Nash iteration (see Table 11) the deterioration is prevented (except for one small increase of the error in iteration 5), and the resulting approximation is slightly better than that obtained in Sect. 5.1.

k	N	ℓ_2-error	k-rate
1	100	$2.232966\,10^{-1}$	
2	100	$7.142212\,10^{-2}$	1.76
3	100	$1.222270\,10^{-2}$	1.67
4	100	$4.547897\,10^{-3}$	1.22
5	100	$4.661234\,10^{-3}$	1.00
6	100	$4.508951\,10^{-3}$	1.01
7	100	$4.381574\,10^{-3}$	1.01
8	100	$4.292863\,10^{-3}$	1.00
9	100	$4.230374\,10^{-3}$	1.00
10	100	$4.184892\,10^{-3}$	1.00

Tab. 11. Nash iteration on 1089 points with $c = 0.061$ and harmonic smoothing.

In [9] a formula for the smoothing parameters t_k was given which involved a number of additional parameters:

$$t_k = \rho^{\theta\beta^k}, \qquad k = 1, 2, \ldots. \tag{18}$$

Here ρ is some positive real number, β is a *superlinear convergence parameter* in $(1, 2)$, and $\theta > 1$ is referred to as *acceleration parameter*. If we let $\rho = 2.0$, $\beta = 1.3$

and $\theta = 1.2$ then we can compute smoothing parameters t_k via (18). By scaling the resulting smoothing parameters t_k (with a factor of $\kappa = 95000$) we obtain the values for N (the number of smoothing steps) in our following experiment (see Table 12).

k	N	ℓ_2-error	k-rate
1	242	$1.988494\,10^{-1}$	
2	186	$6.454938\,10^{-2}$	1.70
3	135	$1.333571\,10^{-2}$	1.58
4	93	$6.140058\,10^{-3}$	1.18
5	59	$4.922886\,10^{-3}$	1.04
6	34	$4.190522\,10^{-3}$	1.03
7	18	$3.949033\,10^{-3}$	1.01
8	8	$3.947686\,10^{-3}$	1.00
9	3	$3.972074\,10^{-3}$	1.00
10	1	$3.975491\,10^{-3}$	1.00

Tab. 12. Nash iteration on 1089 points with $c = 0.061$ and harmonic smoothing.

This version of Nash iteration performs better than the previous one. The final error represents a slight improvement, and the error decreases monotonically throughout.

6.3. Multilevel Nash Iteration

In our final set of experiments we consider what happens to the multilevel Newton iteration of Sect. 5.2 when harmonic smoothing is added at each step. We compare to both the multilevel multiquadric collocation algorithm with $c = 0.3$ and with $c = 2/\sqrt{n}$ (cf. Tables 9 and 10).

For both experiments we take $\rho = 2.5$, $\beta = 1.3$ and $\theta = 1.2$ and $\kappa = 2000$ to obtain the values of N as explained above.

k	n	N	ℓ_2-error	h-rate	k-rate
1	9	480	$2.382085\,10^{-1}$		
2	25	310	$1.118652\,10^{-1}$	1.09	1.53
3	81	180	$5.430229\,10^{-2}$	1.04	1.33
4	289	90	$1.428969\,10^{-2}$	1.93	1.46
5	1089	30	$2.748293\,10^{-3}$	2.38	1.39
6	4225	10	$2.549148\,10^{-3}$	0.11	1.01

Tab. 13. Multilevel Nash iteration with fixed value of $c = 0.3$ and harmonic smoothing.

For the fixed-c version the effects of the smoothing are remarkable. Even though the errors using Nash iteration (Table 13) vs. Newton iteration (Table 9) are initially larger, the main benefit of the smoothing is evident in the later stages of the iteration. Whereas the algorithm broke down with the rough residuals encountered without smoothing (see Table 9 and Figure 7), the added smoothing allows us to maintain convergence throughout. The effects of the smoothing on the residuals are illustrated graphically in Figure 7 where we have displayed the residuals in the 5th iteration ($n = 1089$ points) without and with smoothing for the case $c = 0.3$. The view is different from that in the earlier figures. Due to the large "ridge" of the residual along the boundary of the domain it is better to view these graphs "from below". The "ridge" near the boundary is characteristic for all our experiments. It also occurs when using locally supported RBFs instead of multiquadrics (see [6]). One possibility to overcome this problem and improve the solutions would be to refine the meshes nonuniformly and add some more collocation points near the boundary.

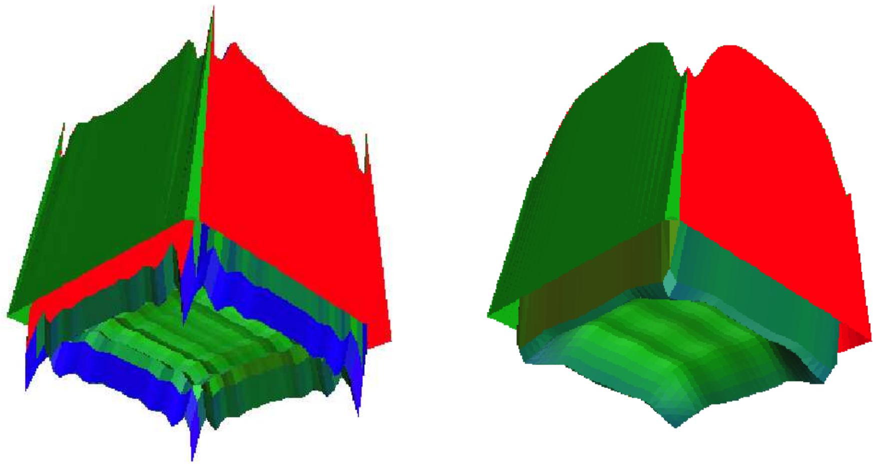

Fig. 7. Residuals for multilevel Newton iteration (left) and multilevel Nash iteration (right) with $c = 0.3$ and $n = 1089$.

n	c	N	ℓ_2-error	h-rate	k-rate
9	.667	480	$2.408171\, 10^{-1}$		
25	.400	310	$1.028604\, 10^{-1}$	1.23	1.60
81	.222	180	$5.918389\, 10^{-2}$	0.80	1.24
289	.118	90	$2.036975\, 10^{-2}$	1.54	1.38
1089	.061	30	$5.513894\, 10^{-3}$	1.89	1.34
4225	.031	10	$9.293965\, 10^{-4}$	2.57	1.34

Tab. 14. Multilevel Nash iteration with variable value of $c = 2/\sqrt{n}$ and harmonic smoothing.

With the variable-c algorithm (compare Tables 14 and 10) the situation is similar. If one compares only errors then the smoothing does not provide an improvement. However, looking at the rate columns one can observe that the smoothing does seem to “kick in” towards the end of the experiment, and it is conceivable that the benefits would become more obvious in the next few iterations. Due to the problem size (16641×16641 dense matrix for the next iteration) we are not able to take this experiment any further on a desktop PC.

7. Concluding Remarks

We have shown that the use of globally supported RBFs within a multilevel Newton framework is a viable approach to the numerical solution of nonlinear PDEs. RBFs are one of several emerging meshless methods being investigated for this purpose and the nonsymmetric collocation method used above has the advantage of being one of the easiest to implement.

Moreover, the use of globally supported RBFs leads to a highly accurate (and comparably efficient) method. In fact, the errors obtained with only 1089 collocation points and multiquadrics (see Table 9) were smaller than those for 16641 points and locally supported RBFs (see [6, Table 4]) (and computed at a fraction of the cost). Using piecewise linear finite elements a comparable accuracy was also reached with 16384 points (see [8, Table 2]).

Another insight presented above is the fact that – for nonlinear problems – multilevel Newton iteration is superior to iteration on a fixed level. The multilevel method is computationally more efficient as well as more accurate.

The experiments presented here are only a first attempt at using globally supported RBFs for the solution of nonlinear PDEs. So far, we have not overcome the restrictions imposed by the global support (i.e., limited problem size due to dense matrices). However, investigations using the ideas of [1] are under way.

The addition of a smoothing step to the Newton iteration, i.e., Nash iteration, has also produced some promising results. It enabled us to overcome the breakdown of the algorithm in the case involving a highly ill-conditioned 4225×4225 matrix. Much more work, however, is needed in this direction.

Finally, as suggested in [7], one can also try other implementations of the Newton method, such as nested iterations (which may themselves act as a smoother). For nonlinear problems these possibilities seem even greater than for linear problems.

Acknowledgments. This research was partially supported by the National Science Foundation under grant DMS-0073636.

References

1. Beatson, R. K., J. B. Cherrie, and C. T. Mouat, Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration, Adv. in Comput. Math. **11** (1999), 253–270.
2. Fasshauer, G. E., Solving partial differential equations by collocation with radial basis functions, in *Surface Fitting and Multiresolution Methods*, A. LeMéhauté, C. Rabut, and L. L. Schumaker (eds), Vanderbilt University Press, Nashville TN, 1997, 131–138.
3. Fasshauer, G. E., On smoothing for multilevel approximation with radial basis functions, in *Approximation Theory IX, Vol. 2: Computational Aspects*, Charles K. Chui and Larry L. Schumaker (eds), Vanderbilt University Press, Nashville TN, 1998, 55–62.
4. Fasshauer, G. E., Solving differential equations with radial basis functions: Multilevel methods and smoothing, Adv. in Comput. Math. **11** (1999), 139–159.
5. Fasshauer, G. E., On the numerical solution of differential equations with radial basis functions, in *Boundary Element Technology XIII*, C. S. Chen, C. A. Brebbia, and D. W. Pepper (eds), WIT Press, 1999, 291–300.
6. Fasshauer, G. E., Nonsymmetric multilevel RBF collocation within an operator Newton framework for nonlinear PDEs, in *Trends in Approximation Theory*, K. Kopotun, T. Lyche and M. Neamtu (eds), Vanderbilt University Press, Nashville, 2001, submitted.
7. Fasshauer, G. E., E. C. Gartland, and J. W. Jerome, Algorithms Defined by Nash iteration: Some implementations via multilevel collocation and smoothing, J. Comput. Applied Math. **119** (2000), 161–183.
8. Fasshauer, G. E., E. C. Gartland, and J. W. Jerome, Newton iteration for partial differential equations and the approximation of the identity, Numerical Algorithms, to appear.
9. Fasshauer, G. E. and J. W. Jerome, Multistep approximation algorithms: Improved convergence rates through postconditioning with smoothing kernels, Adv. in Comput. Math. **10** (1999), 1–27.
10. Floater, M. S. and A. Iske, Multistep scattered data interpolation using compactly supported radial basis functions, J. Comput. Applied Math. **73** (1996), 65–78.
11. Hackbusch, W., *Multi-Grid Methods and Applications,* Springer, Berlin, 1985.
12. Hörmander, L., The boundary problems of physical geodesy, Arch. Ration. Mech. Anal. **62** (1976), 1–52.

13. Hon, Y. C. and R. Schaback, On unsymmetric collocation by radial basis functions, Appl. Math. Comput., to appear.

14. Jerome, J. W., An adaptive Newton algorithm based on numerical inversion: regularization as postconditioner, Numer. Math. **47** (1985), 123–138.

15. Kansa, E. J., Multiquadrics – A scattered data approximation scheme with applications to computational fluid-dynamics – II: Solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl. **19** (1990), 147–161.

16. Moser, J., A rapidly convergent iteration method and nonlinear partial differential equations I., Ann. Scuola Norm. Pisa **XX** (1966), 265–315.

17. Schaback, R., Creating surfaces from scattered data using radial basis functions, in *Mathematical Methods for Curves and Surfaces,* Morten Dæhlen, Tom Lyche, Larry L. Schumaker (eds), Vanderbilt University Press, Nashville & London, 1995, 477–496;.

18. Schaback, R., On the efficiency of interpolation by radial basis functions, in *Surface Fitting and Multiresolution Methods,* A. LeMéhauté, C. Rabut, and L. L. Schumaker (eds), Vanderbilt University Press, Nashville TN, 1997, 309–318.

Gregory E. Fasshauer
Department of Applied Mathematics
Illinois Institute of Technology
Chicago, IL 60616
`fass@amadeus.math.iit.edu`
`http://amadeus.math.iit.edu/~fass`