MESHFREE EXPONENTIAL INTEGRATORS

MARCO CALIARI[†], ALEXANDER OSTERMANN[‡], AND STEFAN RAINER[‡]

Abstract. For the numerical solution of time-dependent partial differential equations, a class of meshfree exponential integrators is proposed. These methods are of particular interest in situations where the solution of the differential equation concentrates on a small part of the computational domain which may vary in time. For the space discretization, radial basis functions with compact support are suggested. The reasons for this choice are the stability and robustness of the resulting interpolation procedure. The time integration is performed with an exponential Rosenbrock method. The required matrix functions are computed by Newton interpolation based on Leja points. The proposed integrators are fully adaptive in space and time. Numerical examples that illustrate the robustness and the good stability properties of the method are included.

Key words. meshfree integrators, radial basis functions, exponential integrators, Leja point interpolation, evolution equations

AMS subject classifications. 65M70, 65L05, 65D05

DOI. 10.1137/100818236

1. Introduction.

In this paper we are concerned with the numerical solution of (nonlinear) time-dependent partial differential equations (PDEs). We are particularly interested in situations where the *essential support* of the solution is small and varying in time. By the essential support of a function we mean the closure of the set of points for which the magnitude of the function is greater than some prescribed threshold value. The essential support is called small if its size is small in comparison with the spatial domain of interest. We always assume that the solution is sufficiently smooth in time and space, and that its essential support is bounded away from the borders of the domain of interest. A typical example of such a function is a Gaussian pulse in the plane. We emphasize that the domain of interest does not need to be bounded, in principle.

In the above situation, classical space discretizations based on a fixed grid or mesh are computationally inefficient, in general. In order to keep the number of degrees of freedom small, one should rather consider an adaptive mesh where the majority of discretization points is contained in the essential support of the solution. In the last few decades, new approximation methods were developed that do not even require a mesh. Among these so-called *meshfree* (or *meshless*) methods, we mention the approaches based on moving least squares or on radial basis functions (RBFs). The latter are particularly suited for computations with high-dimensional data, for problems involving moving fronts such as traveling waves, and for multiscale resolution. A great advantage of meshfree methods lies in their flexibility to add or delete discretization points. Compactly supported radial basis functions (CSRBFs) were developed around 1995 in a series of papers by Schaback, Wendland, and Wu;

[†]Dipartimento di Informatica, Università di Verona, I-37134 Verona, Italy (marco.caliari@univr.it).

[‡]Institut für Mathematik, Universität Innsbruck, A-6020 Innsbruck, Austria (alexander.ostermann@uibk.ac.at, stefan.rainer@uibk.ac.at).

see [27, 33, 37]. They are of particular interest for our purpose, because of their good stability properties and their computational efficiency.

For the time integration of differential equations, there exists a large number of efficient integrators. For stiff problems resulting from space discretizations of PDEs and, in particular, for problems with dominating advection, exponential integrators have turned out to be reliable and efficient; see [8, 9, 16, 31]. Therefore, we will focus on these methods here. We emphasize, however, that the ideas developed in this paper are likewise applicable to other time integration schemes, as well as to other spatial discretizations.

Both employed discretization techniques, space discretizations based on (compactly supported) RBFs and exponential integrators for time integration, are well understood on their own. Their combination, however, has not been investigated in detail yet. In this paper we construct a fully adaptive meshfree exponential integrator based on an efficient error control in space and time.

The outline of this paper is as follows. We start in section 2 with the general description of a meshfree integrator. In section 3 we give a brief introduction to RBF interpolation and we introduce compactly supported RBFs in more detail. In the last part of this section we answer the question how the RBF approach can be applied to the numerical solution of PDEs.

Section 4 is devoted to the time integration of PDEs using exponential integrators. The main focus will be on exponential Rosenbrock integrators, since they are well suited for the solution of nonlinear time-dependent PDEs. For the numerical computation of the arising operator functions, the Leja point method will be employed. A special adaptation for problems with dominating advection part will be discussed there as well. This approach combines very well with the RBF discretization.

In section 5, we give further details on the particular meshfree integrator that we propose. The employed strategy for finding appropriate interpolation and check points is discussed, and the combination of exponential integrators with the RBF approach is elaborated. Section 6 is devoted to numerical experiments that illustrate the stability and robustness of the introduced method. We test our method on a two-dimensional pure advection equation, the Molenkamp–Crowley test, and on a three-dimensional extension of it. This example models the flow around a center. Further, we apply the integrator to an advection-reaction-diffusion problem in two dimensions. For these illustrations, we have implemented an experimental code in MATLAB which can be downloaded from MathWorks.[1]

2. The concept of a meshfree integrator.

Throughout this paper, we consider a time-dependent PDE of the form

$$\frac{\partial}{\partial t}u(t,\xi) = F\left(t,\xi,u(t,\xi),\frac{\partial}{\partial \xi}u(t,\xi),\dots\right), \qquad t\in[t_0,T], \quad \xi\in\Omega\subset\mathbb{R}^d, \tag{2.1}$$

subject to appropriate initial and boundary conditions. For its numerical solution, we propose a meshfree integrator that provides a numerical solution at discrete times $t_0 < t_1 < t_2 < \cdots < t_N = T$. This numerical approximation at time t_n will be denoted henceforth by $u_n(\xi)$.

A time step of a meshfree integrator consists of three parts: a procedure that discretizes a given function (the initial value), a method that integrates this discretization in time over a given time interval, and, finally, a procedure that reconstructs the solution from the fully discrete result. Moreover, it is vital to provide the possibility to

[1] http://www.mathworks.com/matlabcentral

Table 1
Principle of a meshfree integrator.

```
REPEAT   % time stepping loop
         * Choose a set of candidate interpolation points.
         REPEAT   % residual subsampling
             * Interpolate the current solution with respect to the actual set of candidate
               interpolation points.
             * Check error and add/remove points using the thresholds $\theta_r$, $\theta_c$.
             * Update the set of candidate interpolation points.
         UNTIL the set of candidate interpolation points remains fixed.
         * Take the set of candidate interpolation points as the set of inter-
           polation points for the time step.
         REPEAT   % carry out a single time step
             * Compute the check points.
             * Evaluate the current solution at the set of integration points.
             * Perform the time step and control the error.
             * If necessary, add new interpolation points using the monitor function and $\theta_r$.
         UNTIL the set of interpolation points remains fixed.
         * Accept time step and new solution.
UNTIL the final time $T$ is reached.
```

control the error both in time and in space. Before describing the full time step in detail, we discuss its single building blocks. The algorithm stated in Table 1 summarizes the whole concept.

2.1. Space discretization. As we are interested in problems whose solutions have a strongly localized essential support varying in time, we need a highly adaptive space discretization. For this reason, we propose a discretization based on interpolation. Let $w : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ be a given function. For an appropriate set of *interpolation points* $X = \{x_i\} \subset \Omega$ and suitable *basis functions* $\{\Phi_i\}$, we replace $w(\xi)$ by its interpolant

$$p(\xi) = \sum_{j=1}^{m} \lambda_j \Phi_j(\xi), \qquad p(x_i) = w(x_i), \quad i = 1, \ldots, m. \tag{2.2}$$

Here λ_i denote the corresponding coefficients which are determined from the interpolation conditions $p(x_i) = w(x_i)$, $i = 1, \ldots, m$. The interpolation error

$$\max_{\xi \in \Omega} |w(\xi) - p(\xi)|$$

is the *spatial discretization* error with respect to the given set of interpolation points and basis functions. Note that the interpolant $p(\xi)$ can easily be reconstructed from its discrete values $p(x_i)$ at the interpolation points. The spatial discretization error is controlled by the choice of the interpolation points.

2.2. Residual subsampling method. Efficient meshfree methods need a smart procedure for adding and deleting interpolation points. For this purpose, we use the standard *residual subsampling* method, based on the ideas presented in [11]. This method is used to determine a suitable set of interpolation points to efficiently represent a given function $w(\xi)$. This is achieved as follows.

Together with a set of *candidate interpolation points* we consider a set of *check points*. Every candidate interpolation point requires at least one check point. In addition we make use of a *monitor function* θ that gives us information about the error at each check point. Starting from the set of candidate interpolation points, we compute an interpolant $p(\xi)$ of $w(\xi)$ and evaluate it at the check points. If the value of the monitor function at a check point is greater than a *refining threshold* θ_r, that point is added to the candidate interpolation point set *(refinement procedure)*. On the other hand, if $\theta(y)$ is smaller than a *coarsening threshold* θ_c for all check points y associated to a candidate interpolation point, this point is removed from the candidate interpolation point set *(coarsening procedure)*. Note that $\theta_\mathrm{c} < \theta_\mathrm{r}$. If the actual set of candidate interpolation points has changed during this procedure, we update the check point set, interpolate again, and iterate this procedure until no more points are added or removed. The final set of candidate interpolation points forms the set of interpolation points.

The initial set of candidate interpolation points is derived from the predicted essential support. A simple choice can be a coarse grid containing the essential support of w. In the present situation of a differential equation, the candidate interpolation points can be predicted from the interpolation points used at previous time steps. The subsampling method should eventually provide a reasonable set of interpolation points which appropriately reflects the behavior of the solution of the differential equation from step to step.

2.3. Time integration. We are now in position to describe the integration step which computes the numerical solution $u_{n+1}(\xi)$ at time t_{n+1} starting from $u_n(\xi)$ at time t_n. Given $u_n(\xi)$, we determine an appropriate set of interpolation points X_n by the residual subsampling method. Let Y_n denote the corresponding set of check points, and let $\tilde{u}_n(\xi)$ be the interpolant of $u_n(\xi)$ with respect to the points X_n.

In order to perform the time step, we consider a set of *integration points* $\bar{X}_n$. For the sake of simplicity we take $\bar{X}_n = X_n \cup Y_n$, which is the union of the current interpolation and check points. Note, however, that this choice introduces a natural restriction on the length of the time step $\Delta t_n = t_{n+1} - t_n$, since the support of the solution from time t_n to time t_{n+1} has to be covered by the set $\bar{X}_n$ itself.

Discretizing the differential equation at these points results in a stiff system of ordinary differential equations

$$v'(t) = F_{\bar{X}_n}\bigl(t, v(t)\bigr), \qquad v(t_n) = \bigl[u_{n,X_n}, u_{n,Y_n}\bigr]^\mathsf{T}. \tag{2.3}$$

Here, the vectors u_{n,X_n} and u_{n,Y_n} contain the function values $\tilde{u}_n(x)$ for $x \in X_n$ and $x \in Y_n$, respectively (for a concrete discretization based on RBFs, we refer the reader to section 3.2). Integrating this system with step size Δt_n results in a numerical approximation at time $t_{n+1} = t_n + \Delta t_n$, denoted by

$$v_{n+1} = \bigl[u_{n+1,X_n}, u_{n+1,Y_n}\bigr]^\mathsf{T} \approx v(t_{n+1}).$$

After performing the time step, the local time error is estimated in a standard way as described, e.g., in [14, sect. II.4]. If this error is less than a given tolerance, the time step size is accepted. Otherwise, the step is repeated with a smaller step size.

In order to estimate the spatial error, the discrete solution u_{n+1,X_n} corresponding to the point set X_n is interpolated. The resulting function $u_{n+1}(\xi)$ is then evaluated at the check points and compared to the results that are provided directly by the integrator, namely the values u_{n+1,Y_n}. The monitor function θ is thus given by

$$\theta(y) = |u_{n+1}(y) - [u_{n+1,Y_n}]_y|, \qquad y \in Y_n.$$

Each check point $y \in Y_n$, which fails the criterion $\theta(y) \le \theta_{\mathrm{r}}$, is added to the set of interpolation points. If this set has changed, then the corresponding set of check points is updated and the time step is repeated with these new sets. The whole procedure is iterated, until no more interpolation points are added. Finally, the corresponding numerical solution $u_{n+1}(\xi)$ is accepted as numerical approximation to the solution $u(t_{n+1}, \xi)$. This iterative procedure is proposed to ensure that enough interpolation points are used during the time step where the essential support of the solution might change.

3. RBF interpolation.

Interpolation in arbitrary dimension on an *arbitrary* set of collocation points is a challenging task. In this section, we briefly describe the idea of using radial basis functions (RBFs) to face this problem. For a more detailed discussion on RBFs and their properties, we refer the reader to [5, 6, 12, 36].

Let $X = \{x_1, \dots, x_m\} \subset \Omega \subset \mathbb{R}^d$ be a given set of pairwise distinct interpolation points, let $v = [v_1, \dots, v_m]^\mathsf{T}$ be a corresponding vector of data, and let $\Phi : \mathbb{R}^d \to \mathbb{R}$ be a radial function, i.e., $\Phi(\xi) = \phi(\|\xi\|)$ for some $\phi : \mathbb{R}_+ \to \mathbb{R}$. We set $\Phi_j(\xi) = \Phi(\xi - x_j) = \phi(\|\xi - x_j\|)$ and look for an interpolant p of the form

$$p(\xi) = \sum_{j=1}^{m} \lambda_j \Phi_j(\xi), \qquad \xi \in \mathbb{R}^d.$$

The coefficients $\lambda = [\lambda_1, \dots, \lambda_m]^\mathsf{T}$ are chosen such that p interpolates the given data

$$p(x_i) = v_i, \quad i = 1, \dots, m.$$

Finding the coefficients is equivalent to solving the linear system $A\lambda = v$ with the symmetric matrix A with entries $A_{ij} = \phi(\|x_i - x_j\|)$, $i, j = 1, \dots, m$. Depending on the particular choice of the radial basis function, this matrix will be positive definite and sparse.

Since the point set X will vary in our applications from step to step, we will use a slightly different notation henceforth. The vector of the coefficients will be denoted by $[\lambda_x]_{x \in X}$ and the summation over the set X will be rewritten as $\sum_{x \in X}$. In this notation the interpolant has the form

$$p(\xi) = \sum_{x \in X} \lambda_x \Phi_x(\xi), \qquad \Phi_x(\xi) = \Phi(\xi - x). \tag{3.1}$$

In the past 30 years several classes of basis function were proposed. One class is of particular interest for our purpose, namely the class of compactly supported RBFs, because it leads to a sparse and symmetric positive definite interpolation matrix A. Although this class has inferior convergence properties, it is numerically more stable (see, for instance, [36]), the solution of the linear system is not as time consuming, and it is more flexible for approximating solutions changing shape during time evolution. Therefore our main attention will be on compactly supported RBFs. Note, however, that the ideas presented here are valid for other types of basis functions as well.

3.1. Compactly supported RBFs. Within the class of compactly supported RBFs, we concentrate on Wendland's functions $\Phi_{d,k}$; see [33]. In space dimension d they are defined by

$$\Phi_{d,k}(\xi) = \phi_{d,k}(\|\xi\|), \qquad 0 \le k \le d,$$

where

$$\phi_{d,k} = \beta I^k \phi_{\lfloor d/2 \rfloor + k + 1}, \qquad \phi_\ell(r) = (1-r)_+^\ell, \qquad (I\phi)(r) = \int_r^1 t\phi(t)\mathrm{d}t.$$

Here $\lfloor \mu \rfloor$ denotes the largest integer less than or equal to $\mu \in \mathbb{R}$. The scaling factor β is usually chosen such that $\phi_{d,k}(0) = 1$. Some examples are listed in Table 2.

We summarize some important properties of CSRBFs which will be useful for the next sections. For more details, we refer the reader to [35, 38]. First note that the function $\Phi_{d,k}$ is $2k$ times continuously differentiable and strictly positive definite on $\mathbb{R}^s$, $s \le d$. This means that the corresponding interpolation matrix is positive definite, independent of the choice of the interpolation points. We are interested in local error estimates for the interpolant. For this purpose we define the local fill distance of the interpolation points $x \in X$ near a given point $y \in \Omega$ by

$$h_\rho(y) = \max_{\xi \in B_\rho(y)} \min_{x \in X} \|\xi - x\|_2, \qquad \rho > 0. \tag{3.2}$$

Let f be a given function and p its interpolant, computed by CSRBFs. Then the local interpolation error at the point y is bounded by

$$|f(y) - p(y)| \le C h_\rho^{k+1/2}(y) \tag{3.3}$$

for $h_\rho(y)$ sufficiently small. The constant C depends, in particular, on an appropriate Sobolev norm of f. This estimate tells us that the interpolation error will become locally smaller if we decrease the fill distance around the point y. Still we have to keep stability in mind. The stability of the interpolation process is determined by the condition number of the corresponding interpolation matrix A. This number can be bounded in terms of the separation distance

$$q_X = \min_{x_i, x_j \in X} \left\{ \frac{1}{2}\|x_i - x_j\|_2 \colon x_i \ne x_j \right\}, \tag{3.4}$$

namely by

$$\mathrm{cond}_2 A \le C q_X^{-d-2k-1}. \tag{3.5}$$

The bounds (3.3) and (3.5) are crucial in order to select *adequate* interpolation points. In section 5.1 we will treat in more detail the question of selecting the points.

TABLE 2
Wendlands's CSRBF $\Phi_{d,k}$ for various choices of d and k.

d	$\phi_{d,k}(r)$	Smoothness of $\Phi_{d,k}$
1	$\phi_{1,0}(r) = (1-r)_+$	C^0
	$\phi_{1,1}(r) = (1-r)_+^3(3r+1)$	C^2
3	$\phi_{3,0}(r) = (1-r)_+^2$	C^0
	$\phi_{3,1}(r) = (1-r)_+^4(4r+1)$	C^2
	$\phi_{3,2}(r) = (1-r)_+^6(35r^2+18r+3)$	C^4
	$\phi_{3,3}(r) = (1-r)_+^8(32r^3+25r^2+8r+1)$	C^6

3.2. RBF discretizations of differential equations. The idea of using RBFs for discretizing time-invariant PDEs dates back to [20, 21]. Later the idea was extended to time-dependent PDEs, e.g., in [1, 2, 26]. The concept is similar to the pseudospectral approach. We explain the basic ideas first with the help of a simple example. Consider the semilinear differential equation

$$\frac{\partial}{\partial t}u(t,\xi) = Lu(t,\xi) + f\bigl(\xi, u(t,\xi)\bigr), \qquad u(t_n,\xi) = \tilde{u}_n(\xi), \tag{3.6}$$

which has to be integrated from t_n to $t_{n+1} = t_n + \Delta t_n$ for a given initial value $\tilde{u}_n(\xi)$. Here L denotes a spatial differential operator whose coefficients do not depend on t. For example, L could be a second-order differential operator with variable coefficients. We show how to discretize this problem on a set $\bar{X}$ of integration points.

To this aim, we replace $u(t,\xi)$, $\xi \in \mathbb{R}^d$ by its interpolant $p(t,\xi) = \sum_{x\in\bar{X}} \lambda_x \Phi_x(\xi)$ and approximate $Lu(t,\xi)$ by

$$Lp(t,\xi) = L\sum_{x\in\bar{X}} \lambda_x \Phi_x(\xi) = \sum_{x\in\bar{X}} \lambda_x \bigl(L\Phi_x\bigr)(\xi).$$

Here, the coefficients λ are the solution of the linear system $\bar{A}\lambda = u(t,\cdot)|_{\bar{X}}$. Note that the function ϕ has to be chosen in such a way that $L\Phi_x$ is well defined. In particular, Φ has to possess sufficiently many derivatives.

From the above considerations, we get the representation

$$Lp(t,\xi) = \sum_{x\in\bar{X}} \left[\bar{A}^{-1}u(t,\cdot)|_{\bar{X}}\right]_x \bigl(L\Phi_x\bigr)(\xi).$$

Using the fact that $u(t,\xi) = p(t,\xi)$ for all $\xi \in \bar{X}$, we can rewrite this as

$$Lp(t,\cdot)|_{\bar{X}} = \bar{A}_L\bar{A}^{-1}p(t,\cdot)|_{\bar{X}} \tag{3.7}$$

with $\bar{A}_L = \{L\Phi_x(\xi)\}_{\xi,x\in\bar{X}}$. Note that the arising matrices can be computed once and for all if the set of integration points $\bar{X}$ remains fixed during the time integration. The nonlinearity f is discretized as

$$f_{\bar{X}}\bigl(p(t,\cdot)|_{\bar{X}}\bigr) = \bigl[f\bigl(x, p(t,x)\bigr)\bigr]_{x\in\bar{X}}.$$

By the proposed RBF discretization, the PDE (3.6) thus becomes a stiff system of ordinary differential equations

$$v'(t) = \bar{A}_L\bar{A}^{-1}v(t) + f_{\bar{X}}\bigl(v(t)\bigr), \qquad v(t_n) = \tilde{u}_n(\cdot)|_{\bar{X}}.$$

Note that the above procedure corresponds to discretizing the advection term

$$\frac{\partial}{\partial\xi}\bigl(a(\xi)u(t,\xi)\bigr) = \frac{\partial a}{\partial\xi}(\xi)u(t,\xi) + a(\xi)\frac{\partial u}{\partial\xi}(t,\xi)$$

by restricting the function

$$\frac{\partial a}{\partial\xi}\sum_{x\in\bar{X}} \lambda_x\Phi_x + a\sum_{x\in\bar{X}} \lambda_x\frac{\partial\Phi_x}{\partial\xi}$$

to the points in $\bar{X}$. The advection term can also be discretized in *conservative* form as

$$\frac{\partial}{\partial x}\big(a(x)u(t,x)\big) \approx \sum_{x\in\bar{X}} \widetilde{\lambda}_x \frac{\partial}{\partial x}\Phi_x(\cdot)\Big|_{\bar{X}},$$

where the coefficients $\widetilde{\lambda}$ are now defined by $\bar{A}\widetilde{\lambda} = a(\cdot)u(t,\cdot)|_{\bar{X}}$. This discretization is particularly useful for mass conservation schemes.

In general, our space discretization at time t_n gives the nonlinear system

$$v'(t) = F_{\bar{X}}(t, v(t)), \qquad v(t_n) = \tilde{u}_n(\cdot)|_{\bar{X}}. \tag{3.8}$$

In the next section, we discuss an appropriate time discretization of this system of ordinary differential equations.

4. Exponential integrators for time-dependent evolution equations.

For the time discretization of the nonlinear initial value problem

$$v'(t) = G\big(t, v(t)\big), \qquad v(t_0) = v_0, \tag{4.1}$$

we consider a particular class of exponential integrators, namely exponential Rosenbrock methods; see [8, 17] and the review article [16].

Rosenbrock methods are based on a continuous linearization of the vector field along the numerical solution. For a given approximation $v_n \approx v(t_n)$, we linearize (4.1) in the following way:

$$v'(t) = J_n v(t) + k_n t + g_n\big(t, v(t)\big), \tag{4.2a}$$

$$J_n = \frac{\partial G}{\partial u}(t_n, v_n), \quad k_n = \frac{\partial G}{\partial t}(t_n, v_n), \quad g_n(t,v) = G(t,v) - J_n v - k_n t \tag{4.2b}$$

with g_n denoting the nonlinear remainder.

Let $v(t_{n+1})$ denote the exact solution of (4.2) at $t_{n+1} = t_n + \Delta t_n$ with initial value $v(t_n) = v_n$. Using the variation-of-constants formula, it can be represented as

$$\begin{aligned} v(t_{n+1}) = \mathrm{e}^{\Delta t_n J_n} v_n + \Delta t_n \varphi_1(\Delta t_n J_n) t_n k_n + \Delta t_n^2 \varphi_2(\Delta t_n J_n) k_n \\ + \int_0^{\Delta t_n} \mathrm{e}^{(\Delta t_n - \tau)J_n} g\big(t_n + \tau, v(t_n+\tau)\big)\,\mathrm{d}\tau \end{aligned} \tag{4.3}$$

with the entire functions

$$\varphi_k(z) = \int_0^1 \mathrm{e}^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!}\,\mathrm{d}\theta, \quad k \geq 1. \tag{4.4}$$

These functions satisfy $\varphi_k(0) = 1/k!$ and the recurrence relation

$$\varphi_0(z) = \mathrm{e}^z, \qquad \varphi_{k+1}(z) = \frac{\varphi_k(z) - \varphi_k(0)}{z}, \quad k \geq 0. \tag{4.5}$$

Replacing $g\big(t_n + \tau, v(t_n + \tau)\big)$ in (4.3) by $g(t_n, v_n)$ defines a second-order numerical scheme, the well-known *exponential Rosenbrock–Euler* method

$$v_{n+1} = v_n + \Delta t_n \varphi_1(\Delta t_n J_n) G(t_n, v_n) + \Delta t_n^2 \varphi_2(\Delta t_n J_n) k_n. \tag{4.6}$$

This scheme is explicit and thus computationally attractive. For autonomous problems where $k_n = 0$, it requires only one evaluation of a matrix function per step.

Higher-order exponential Rosenbrock methods for the numerical solution of (4.1) have the general format

$$\begin{aligned} V_{ni} &= v_n + \Delta t_n c_i \varphi_1(c_i \Delta t_n J_n) G(t_n, v_n) \\ &\quad + \Delta t_n^2 c_i^2 \varphi_2(c_i \Delta t_n J_n) k_n + \Delta t_n \sum_{j=2}^{i-1} a_{ij}(\Delta t_n J_n) D_{nj}, \end{aligned} \tag{4.7a}$$

$$\begin{aligned} v_{n+1} &= v_n + \Delta t_n \varphi_1(\Delta t_n J_n) G(t_n, v_n) \\ &\quad + \Delta t_n^2 \varphi_2(\Delta t_n J_n) k_n + \Delta t_n \sum_{i=2}^{s} b_i(\Delta t_n J_n) D_{ni}, \end{aligned} \tag{4.7b}$$

where $D_{ni} = g_n(t_n + c_i \Delta t_n, V_{ni}) - g_n(t_n, v_n)$. Without further mentioning, we will assume that the methods fulfill $c_1 = 0$ and consequently $V_{n1} = u_n$. Note that the vectors D_{ni} are expected to be small in norm, since

$$\frac{\partial g_n}{\partial u}(t_n, v_n) = 0, \qquad \frac{\partial g_n}{\partial t}(t_n, v_n) = 0.$$

Each stage of (4.7) consists of a perturbed exponential Rosenbrock–Euler step (4.6) and these additional corrections can be cheaply computed.

For our numerical experiments below, we consider the third-order exponential Rosenbrock method with $s = 2$ stages and coefficients $c_2 = 1$ and $b_2(z) = \varphi_3(z)$. The method thus takes the form

$$\begin{aligned} V_{n2} &= v_n + \Delta t_n \varphi_1(\Delta t_n J_n) G(t_n, v_n) + \Delta t_n^2 \varphi_2(\Delta t_n J_n) k_n, \\ v_{n+1} &= V_{n2} + \Delta t_n \varphi_3(\Delta t_n J_n) \big(g_n(t_n + \Delta t_n, V_{n2}) - g_n(t_n, v_n) \big). \end{aligned} \tag{4.8}$$

Note that the stage V_{n2} is just an exponential Rosenbrock–Euler step and thus a second-order approximation. Consequently, the difference

$$v_{n+1} - V_{n2} = \Delta t_n \varphi_3(\Delta t_n J_n) \big(g_n(t_n + \Delta t_n, V_{n2}) - g_n(t_n, v_n) \big) \tag{4.9}$$

can be used as an error estimate; see [8]. The resulting embedded method will be called `exprb32`. The convergence properties of this method were analyzed in [16].

4.1. Newton interpolation at Leja points. In order to compute $\varphi_k(\Delta t J) w$, we use the *real Leja points method* which is based on Newton's interpolation formula for the scalar function $\varphi_k(\Delta t z)$ at a sequence of Leja points $\{z_i\}$. This method was first proposed in [9] for evaluating φ_1. It possesses very attractive computational features. Note that the analysis given in [9] extends to other φ-functions in a straightforward way (see [7, 8]). So far the method has been used mainly for matrices J discretizing operators with a moderate advection part: here we extend it to operators with a strong advection part or even to pure advection operators.

The starting point is a cheap estimate of a rectangle containing the spectrum of the operator J. Following [28], it can be obtained by computing the largest eigenvalues in magnitude of the symmetric part and the skew-symmetric part of the operator J separately. It is worth noting that only a rough estimate of the spectrum is necessary (see [9]), which can be obtained at a low computational cost. For the problems we have in mind, J is a real operator and its smallest eigenvalue in magnitude is simply estimated by 0. Therefore, the estimate of the spectrum results in a rectangle with vertices $(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (-a, \mathrm{i}b)$, $a, b \geq 0$.

We distinguish the cases $a \geq b$ and $a < b$. The first case, corresponding to moderate advection, is treated with a sequence of standard (real) Leja points $\{z_i\}$ on the interval $D = [-a, 0]$ and is well described in [7, 8, 9].

In the second case, $a < b$, corresponding to strong advection, standard Leja points on the domain $D = \{z \in \mathbb{C} \colon \operatorname{Re} z = -a/2, \operatorname{Im} z \in [-b, b]\}$ would require complex arithmetic. We prefer considering *conjugate pairs* of Leja points instead. They are defined by $z_0 = -a/2$ and

$$z_m \in \arg\max_{z \in D} \prod_{i=0}^{m-1} |z - z_i|, \qquad z_{m+1} = \overline{z}_m \qquad \text{for } m \text{ odd.}$$

Let $c = -a/2$ and $\gamma = b/2$. The polynomial $p_m(\Delta t J)w$ of degree m in the Newton form which approximates $\varphi_k(\Delta t J)w$ can be written in *real* arithmetic (see [30]) as

$$\begin{aligned} p_m(\Delta t J)w &= p_{m-2}(\Delta t J)w + (\operatorname{Re} d_{m-1}) r_{m-2} + d_m q_m, \quad m > 0 \text{ even}, \\ r_m &= \big((J - cI)/\gamma\big) q_m + \big(\operatorname{Im} \xi_{m-1}\big)^2 r_{m-2}, \\ q_m &= \big((J - cI)/\gamma\big) r_{m-2}, \end{aligned} \tag{4.10a}$$

where

$$p_0(\Delta t J)w = d_0 w, \quad r_0 = \big((J - cI)/\gamma\big) w, \tag{4.10b}$$

and $\{d_i\}_{i=0}^m$ are the divided differences (real if i even) of the function $\varphi_k\big(\Delta t(c + \gamma\xi)\big)$ at the conjugate pairs of Leja points $\{\xi_i\}$, $\xi_0 = 0$, of the reference domain $\mathrm{i}[-2, 2]$. For the implementation of (4.10) it is sufficient to use three vectors $p = p_m$, $r = r_m$, and $q = q_m$ and to update them in each iteration. Moreover, a convenient estimate of the interpolation error is given by

$$\begin{aligned} \|e_m\| = \|p_{m+2}(\Delta t J)w - p_m(\Delta t J)w\| &= \|(\operatorname{Re} d_{m+1}) r_m + d_{m+2} q_{m+2}\| \\ &\approx \|p_m(\Delta t J)w - \varphi_k(\Delta t J)w\|. \end{aligned} \tag{4.11}$$

The attractive computational features of the method are clear: there is no Krylov subspace to store and the complexity of the recurrences (4.10a) is linear in m, whereas the long-term recurrence in the standard Krylov method is quadratic in m for non-symmetric operators $\Delta t J$; see, e.g., [4, 13, 15]. Moreover, the computations can be made in real arithmetic, and it is not necessary to solve real or complex linear systems with the operator J, as in rational Krylov approximations [25, 32] or Carathéodory–Fejér and contour integrals approximations [29]. Finally, the real Leja points method is very well structured for a parallel implementation, as shown in [3, 23] for the φ_1-function. When the expected degree m for convergence is too large, the original time step Δt has to be split into a certain number of substeps, say ℓ, and the approximation of $\varphi_k(\Delta t J)w$ is recovered from $\varphi_k(\tau \Delta t J)w$ with $\tau = 1/\ell$.

5. Meshfree exponential integrators.

We are now in position to combine the meshfree approximation based on compactly supported RBFs with time integration based on exponential integrators. The resulting method will be an efficient and reliable integrator for certain classes of PDEs that possess solutions with a well-localized essential support. For the space discretization, we have chosen Wendland's function

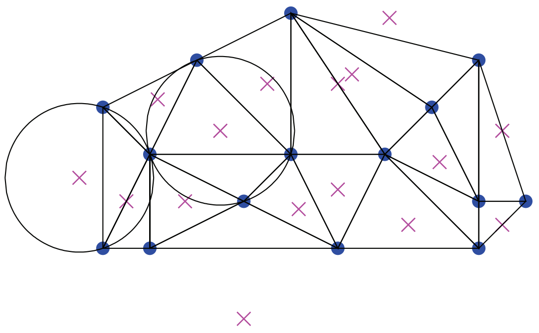

Fig. 1. *Example of interpolation points (dots) and check points (crosses). The check points are the centers of the circumcircles of the Delaunay triangulation based on the interpolation points.*

$\phi_{3,2}$ of Table 2. This function is four times continuously differentiable and can be used for problems in space dimensions 1, 2, and 3.

In order to obtain a robust method, we need an elaborated strategy to select the corresponding sets of interpolation and check points introduced in section 2, as well as a strategy to accept or reject time steps. In fact, standard techniques for time step rejection do not take into account the spatial error which, in the standard method of lines with a fixed grid or mesh discretization, is introduced at the beginning of the integration and later often ignored. Henceforth, we denote by Tol the user-supplied tolerance that will be used in the error control.

5.1. Selection of interpolation and check points. The flexibility of RBFs allows the use of an arbitrary set of distinct interpolation points in principle. However, in order to obtain an accurate and robust method, the sets have to be chosen with care. In particular, the bounds for the interpolation error (3.3) and the condition number (3.5) should be kept as small as possible.

In order to interpolate a given smooth function $w : \mathbb{R}^d \to \mathbb{R}$, we start from a set of candidate interpolation points. There are various possibilities for defining such a set. In our application, w is the numerical solution of problem (2.1) at time t_n. For the initial value ($n = 0$), the set of candidate interpolation points will just consist of a coarse grid containing the essential support of w. In the general case, when stepping from t_n to t_{n+1} we already know an appropriate set of interpolation points from the previous step. This set, transported by the advection term of (2.1), defines a set of candidate interpolation points.

The candidate set is then adapted to the given interpolant w using the residual subsampling method as described in section 2. The threshold values are chosen as

$$\theta_{\mathrm{r}} = \text{Tol}, \qquad \theta_{\mathrm{c}} = 10^{-3}\text{Tol}.$$

While the choice of the refining threshold θ_{r} is just the interpretation of the user-prescribed tolerance Tol, the coarsening threshold θ_{c} must be chosen considerably smaller in order to avoid unwanted iterations in the subsampling method. In order to generate the set of check points, we compute a Delaunay triangulation of the interpolation points and choose the centers of the circumspheres of the resulting simplices, the so-called Voronoi points, as check points; see Figure 1. The reason for this choice is twofold. The Delaunay triangulation has the property that no further interpolation points are contained in the interior of the circumspheres. Therefore, their centers are points of local maxima for the distance

$$\mathrm{d}(\xi, X) = \min_{x\in X} \|\xi - x\|_2 \tag{5.1}$$

to the interpolation point set X. This means that they maximize the error bound (3.3). Second, if a refinement is needed, adding a point of such a check point set to the interpolation point set minimizes the growth of the condition number bound (3.5) for the interpolation matrix.

Note that adding or removing an interpolation point affects a Delaunay triangulation only locally. Hence, after updating the interpolation point set, the computation of the check points has constant complexity.

5.2. Performing the time step. We next describe how a step with the exponential integrator is carried out. Let $\bar{X}_n$ denote the integration points at time t_n. For these points, we consider an RBF discretization of (2.1), as detailed in subsection 3.2. This results in a stiff system of ordinary differential equations

$$v'(t) = F_{\bar{X}_n}(t, v(t)), \qquad v(t_n) = [u_{n,X_n}, u_{n,Y_n}]^{\mathsf{T}}. \tag{5.2}$$

In order to apply an exponential integrator of Rosenbrock type, we have to compute the Fréchet derivative J_n of $F_{\bar{X}_n}\big(t, v(t)\big)$ at $(t_n, u_{n,X_n}, u_{n,Y_n})$.

With this derivative at hand we can approximate the expressions $\varphi_k(\Delta t_n J_n)v$ by Newton interpolation, as described in section 4.1. Since the (symmetric) interpolation matrix $A_{\bar{X}_n}$ does not change during the step from time t_n to time t_{n+1}, it has to be factorized only once per time step. When carrying out the Newton interpolation at Leja points, we have to compute many times the application of the operator J_n to a function $w(\xi)$, which is obtained from a vector of discrete values by RBF interpolation. For this aim, it is vital that the RBF interpolation has good stability properties. This is the main reason why we have chosen compactly supported RBFs. Moreover, as the corresponding interpolation matrix $A_{\bar{X}_n}$ is positive definite and sparse, it can be factorized in a stable and efficient way.

The time step Δt_n is chosen in a standard way, based on the local error estimate (4.9); see [14, sect. II.4]. If the estimated error is less than a prescribed temporal tolerance θ_{t}, the time step size is accepted, otherwise it is rejected. Numerical experiments in [8] indicate that θ_{t} should be smaller than $\textsc{Tol}$. In our experiments, we have chosen

$$\theta_{\mathrm{t}} = 10^{-1}\textsc{Tol}.$$

All the procedures and considerations mentioned above yield the *meshfree exponential integrator* scheme summarized in Table 3.

5.3. Computational costs. In order to analyze the computational costs of our method, we consider separately the two main ingredients, space and time discretization on the one hand and the coupling of them on the other.

For the RBF approximation, three basic operations are of interest: the computation of the interpolant of a function, the evaluation of the interpolant at a given set of points, and the approximation of a linear operator applied to a certain function. The computation of the interpolation coefficients λ_x in (3.1) requires the solution of a sparse linear system with a symmetric positive definite matrix with entries $A_{ij} = \phi(\|x_i - x_j\|)$, $i, j = 1, \ldots, m$. To this purpose, we use the direct sparse Cholesky

TABLE 3
Meshfree exponential integrator.

```
Let $n = 0$.
REPEAT
    * Given $u_n(\xi)$, the current approximation of the exact solution $u(t_n, \xi)$, and the
      set of candidate interpolation points, apply the residual subsampling to find an
      appropriate set of interpolation points $X_n$.
    * Define a new approximation $\tilde{u}_n(\xi)$ by
          $$\tilde{u}_n(\xi) = \sum_{x \in X_n} \lambda_x \Phi_x(\xi),$$
      where $A_{X_n} \lambda = u_n(\cdot)|_{X_n}$.
    REPEAT
        * Compute the set of check points $Y_n$ and define the set of integration points
          $\bar{X}_n = X_n \cup Y_n$.
        * Evaluate $\tilde{u}_n(\cdot)|_{X_n} = u_{n,X_n}$ and $\tilde{u}_n(\cdot)|_{Y_n} = u_{n,Y_n}$.
        * Discretize the right-hand side of the differential equation with respect to $\bar{X}_n$
          to obtain
              $$v'(t) = F_{\bar{X}_n}\big(t, v(t)\big), \qquad v(t_n) = [u_{n,X_n}, u_{n,Y_n}]^{\mathsf{T}}.$$
        * Integrate this initial value problem to $t_{n+1} = t_n + \Delta t_n$ to obtain a numerical
          approximation $[u_{n+1,X_n}, u_{n+1,Y_n}] \approx v(t_{n+1})$.
        * Check the local integration error and repeat the step if the error is larger
          than $\theta_{\mathrm{t}}$.
        * Interpolate the values $u_{n+1,X_n}$ with respect to the points $X_n$ to get
              $$u_{n+1}(\xi) = \sum_{x \in X_n} \lambda_x(u_{n+1,X_n}) \Phi_x(\xi).$$
        * Estimate the interpolation error at the check points, i.e., set
              $$\theta(y) = \Big\| u_{n+1}(y) - \big[u_{n+1,Y_n}\big]_y \Big\|, \qquad y \in Y_n.$$
        * Add those check points where the error is too large to the set of interpolation
          points.
    UNTIL the set of interpolation points remains fixed.
    * Set $n = n + 1$ and accept $u_{n+1}(\xi)$ as numerical solution at time $t_{n+1}$.
UNTIL $t_n = T$.
```

method. In our experiments the fraction of nonzero elements turned out to be about 20%. The evaluation at a certain point ξ requires a linear combination of m shifted basis functions $\Phi_x(\xi)$; see (3.1). The approximation of a linear operator applied to a function at the set of integration points $\bar{X}$ requires the computation of the right-hand side of (3.7). In order to make it possible to apply the operator to different functions defined on the same set $\bar{X}$, we compute the sparse Cholesky factors of $\bar{A}$ and evaluate (3.7) using sparse backward substitutions.

The main cost for performing a time step from t_n to t_{n+1} by an exponential integrator is the computation of the action of the φ_k functions. Any iterative polynomial method, such as the real Leja points method, requires successive applications of the linear operator J_n (see (4.2)). In practice, as seen above, each application is a sparse backward substitution followed by a sparse matrix-vector product, since the set $\bar{X}_n$ remains unchanged during the calculation of φ_k. The number of iterations for the approximation of the action of $\varphi_k(\Delta t_n J_n)$ depends roughly linearly on the stiffness of $\Delta t_n J_n$.

In addition, our integrator needs to compute a set of check points and update interpolation points at each time step. The calculation of the check points requires a Delaunay triangulation which requires $\mathcal{O}\big(m^{\lceil\frac{d}{2}\rceil}\big)$ operations (see, e.g., [18]), where m is the number of interpolation points. Here $\lceil\mu\rceil$ denotes the smallest integer greater than or equal to $\mu \in \mathbb{R}$. Therefore, the number of check points is asymptotically $\mathcal{O}\big(m^{\lceil\frac{d}{2}\rceil}\big)$. Since our set of integration points consists of the union of interpolation and check points, we have $\mathcal{O}(m + m^{\lceil\frac{d}{2}\rceil})$ integration points. This means that the cost of a Delaunay triangulation is asymptotically the same as a single sparse matrix-vector multiplication with the matrix $\bar{A}_L$. Although, as already mentioned, this choice for the set of check points seems to be optimal with respect to the minimization of the local error bound and the control of the growth of the condition number of the interpolation matrix, the memory requirements can be quite large, especially in higher dimensions. A possibility of reducing the storage requirements is to use different strategies for computing check points. For example, nonoverlapping boxes (see [11]) introduce only $N \cdot 2^d$ check points at most, where N denotes the number of interpolation points. However, we find the Delaunay triangulation to be more flexible; moreover, it is already efficiently implemented for arbitrary dimensions.

We note that the costs involved in the RBF approximation are the same for any method based on this type of spatial discretization. When combined with an exponential integrator, the most expensive part is the application of the linear operator, which has to be done several times in order to approximate the φ_k functions. However, even implicit methods require several applications of the linear operator when an iterative method is chosen to solve the arising linear systems. Moreover, implicit methods usually require good preconditioners, which have to be recomputed after each time step, since the interpolation points change. On the other hand, standard explicit methods may have severe restrictions on the time step length when applied to stiff problems, and thus the number of applications of the linear operator can become very high as well.

6. Numerical experiments in MATLAB.

In this section we perform some numerical tests that demonstrate the strength of meshfree exponential integrators. For this purpose, we have implemented a meshfree integrator in MATLAB as detailed in Table 3. This code is available at MathWorks. Interpolation in space is carried out with the Wendland function $\phi_{3,2}$, scaled in such a way that its support is $[-1/2, 1/2]$. For the evolution in time we use the exponential integrator `exprb32` with the standard step size selection as described in [14, sect. II.4]. Newton interpolation at Leja points is used for the approximation of the operator functions. The estimate of the spectrum is computed with the command `eigs` in MATLAB (which is based on ARPACK [22]), applied to the symmetric part and to the square of the skew-symmetric part of the operator with the argument `SIGMA` set to `LM`. In order to keep the computational costs low, a small maximum number of iterations and a quite large tolerance are used.

We present two examples, a pure advection problem and a reaction-diffusion-advection problem. In both examples, the solution has a small essential support. The errors are measured by comparing the numerical approximation with the exact solution in a discrete L^∞ norm on a fixed grid of size 50×50.

6.1. The Molenkamp–Crowley test.

As a first example we consider a pure advection problem, the Molenkamp–Crowley test [10, 24]. This is a standard two-dimensional (2D) test problem in meteorology that models the flow around a center.

The describing differential equation is

$$(6.1)\quad \frac{\partial}{\partial t}u(t,x) + \frac{\partial}{\partial x_1}\big(a_1(x)u(t,x)\big) + \frac{\partial}{\partial x_2}\big(a_2(x)u(t,x)\big) = 0, \qquad -2 \le x_1, x_2 \le 2,$$

with velocity field

$$a_1(x_1,x_2) = 2\pi x_2, \qquad a_2(x_1,x_2) = -2\pi x_1$$

as in [19, Chap. IV]. The velocity field defines a rotation of period one around the origin. As an initial value we have chosen a Gaussian pulse

$$(6.2)\qquad u(0,x_1,x_2) = \exp\big(-10(x_1-0.2)^2 - 10(x_2-0.2)^2\big)$$

with center $(0.2, 0.2)$. The shape of the resulting solution does not change for $0 \le t \le T$, just the position of the pulse varies during the time evolution.

Further, we consider a three-dimensional (3D) version of (6.1) on the domain $-2 \le x_1, x_2, x_3 \le 2$ with coefficients

$$a_1(x_1,x_2,x_3) = 2\pi x_2, \qquad a_2(x_1,x_2,x_3) = -2\pi x_1 \qquad a_3(x_1,x_2,x_3) = 0$$

and initial value

$$(6.3)\qquad u(0,x_1,x_2,x_3) = \exp\big(-10(x_1-0.2)^2 - 10(x_2-0.2)^2 - 10(x_3-0.2)^2\big).$$

As (6.1) is a linear problem, it is integrated in time exactly by an exponential integrator. Thus any error occurring during computation comes from the discretization in space. The absence of diffusion might cause numerical instabilities. Usually flux-corrected schemes and special treatment of outflow boundary conditions are used in order to avoid severe oscillations. In any case, since the essential support of the solution is quite small with respect to the computational domain, local spatial refinements are suggested (see [19]). The Molenkamp–Crowley test, in particular its long-time integration, is considered as a difficult test problem.

In a first experiment we show that the spatial error estimate works as desired. For a given local tolerance in space, we integrate both problems up to $T = 1$ and compare the difference to the exact solution, i.e., the initial value. As already mentioned, exponential integrators can integrate linear problems with arbitrarily large time steps. However, since the essential support of the solution for the time step $\Delta t_n = t_{n+1} - t_n$ has to be covered by the integration point set $\bar{X}_n$, a meshfree integrator should not take too large time steps. For this reason we restrict the step size $\Delta t_n = \Delta t$ to 10^{-2} for this experiment. The achieved results are displayed in Figure 2. The left figure clearly shows that the error estimate in space performs as desired. The right figure displays the number of required interpolation points as a function of the given tolerance TOL. The figure shows that the error is proportional to N^{-3} in the 2D case and to $N^{-6.5/3}$ in the 3D example, where N is the average number of interpolation points.

To show that the computation remains stable during time evolution, we perform a long-time integration. We integrate (6.1) up to $T = 100$, which means that the pulse performs 100 turns around the origin, with a spatial tolerance of $\mathrm{TOL} = 10^{-5}$. Again we restrict the step size to $\Delta t = 10^{-2}$. As we allow in each time step an interpolation

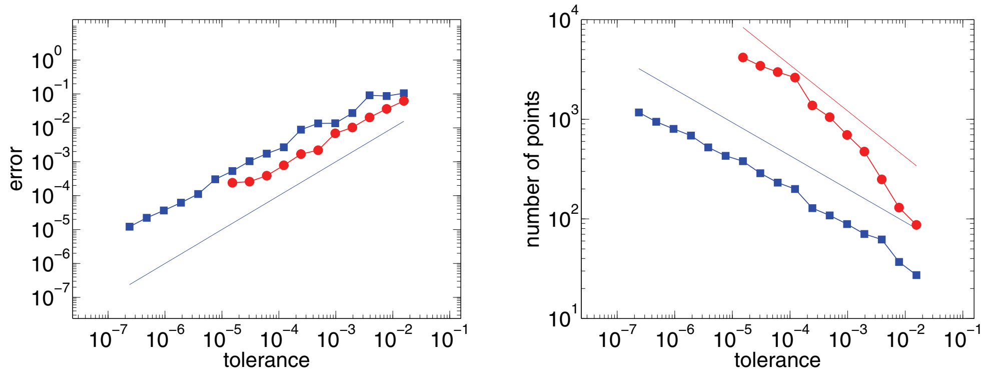

FIG. 2. *Error versus tolerance (left) and average number of interpolation points (right) at* $T = 1$ *for the meshfree exponential integrator, when applied to the Molenkamp–Crowley test. The blue squares mark the results for the* $2D$ *problem, the red circles for the* $3D$ *problem, obtained for the prescribed tolerances* $\mathrm{TOL} = 2^{-j}$*, for* $j = 6, \dots, 22$*, in two dimensions and* $j = 6, \dots, 16$ *in three dimensions. The blue line in the right figure has slope* $-1/3$ *and the red one has slope* $-3/6.5$.

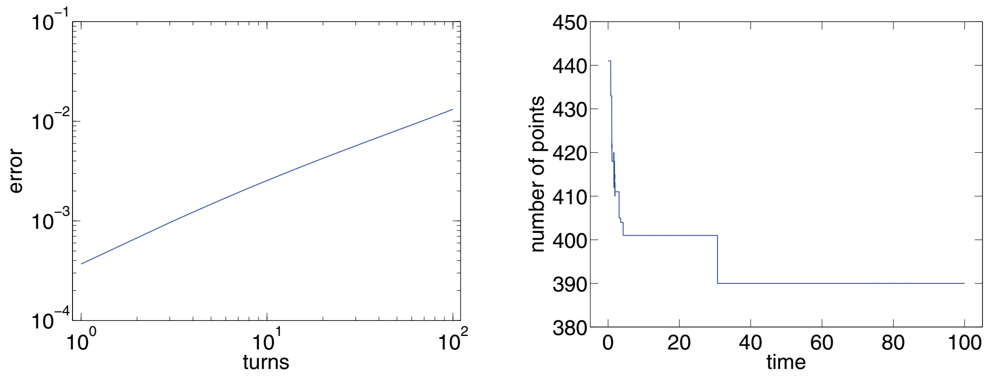

FIG. 3. *Interpolation error (left) and number of interpolation points (right) for the meshfree exponential integrator for the long-time integration of the Molenkamp–Crowley test. We computed* 100 *turns of the pulse around the origin with a prescribed interpolation tolerance of* 10^{-5} *for each step. The time step size was chosen to be* 10^{-2}.

error of size TOL, we have to expect a final error of $\mathrm{TOL} \cdot T/\Delta t = 10^{-1}$ in the worst case. It is important, however, that no oscillations occur during the time evolution.

In Figure 3 the results for the meshfree exponential integrator are displayed. As expected the error increases linearly with the final time T, as can be seen in the left figure. The right figure shows the number of interpolation points for every time step. The number of required interpolation points remains almost constant along the integration. This is also expected for this numerical experiment since the solution does not change its shape. Finally, the time evolution of the error is displayed in Figure 4 for different times. One can see that the essential support of the error remains small and that no oscillations occur.

We summarize that the meshfree exponential integrator behaves very satisfactorily for the Molenkamp–Crowley test. It meets all the requirements we are interested in.

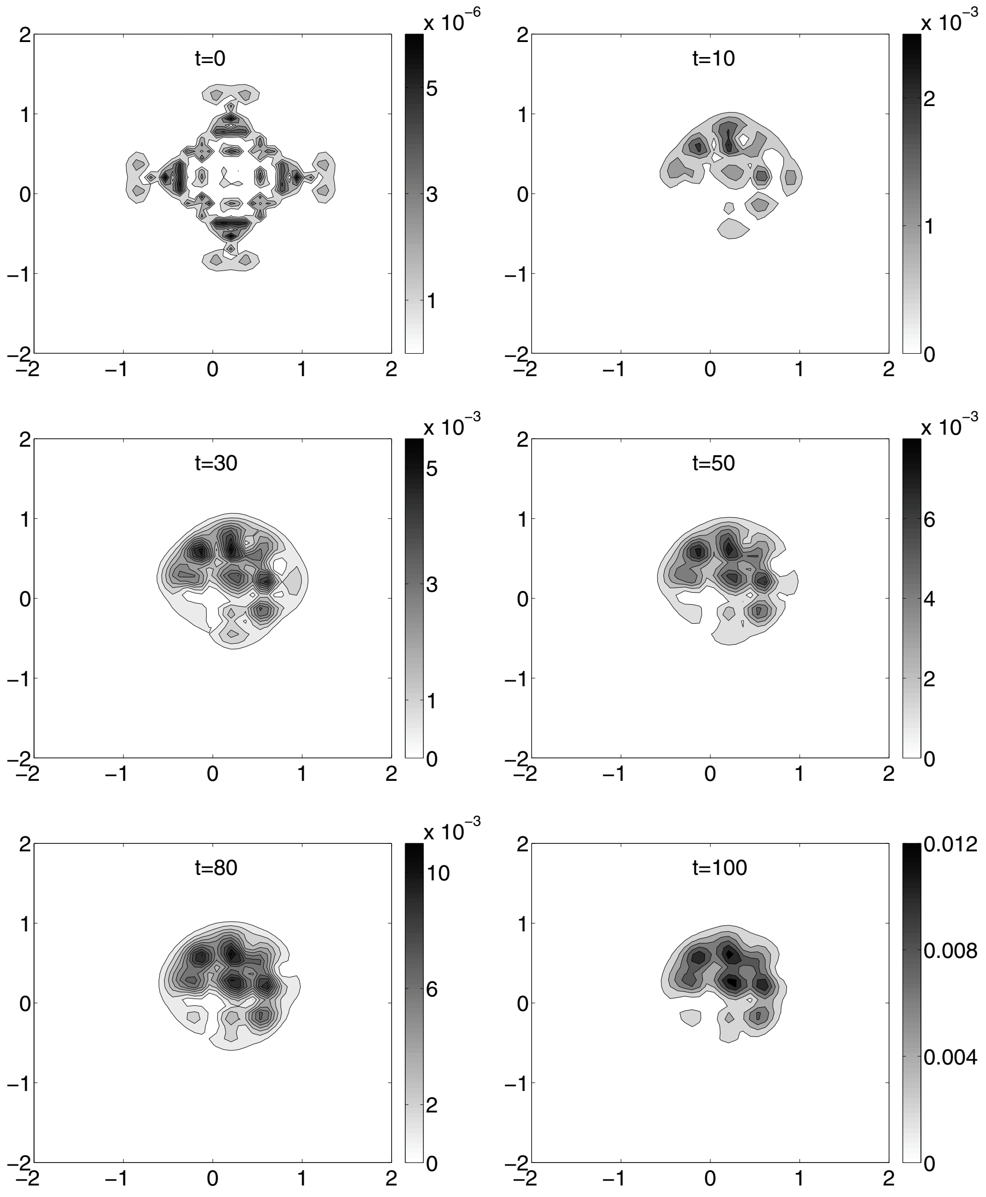

FIG. 4. *Time evolution of the error of the meshfree exponential integrator when applied to the Molenkamp–Crowley test. The figures show the error after* $0, 10, 30, 50, 80,$ *and* 100 *turns of the pulse around the origin for a prescribed interpolation tolerance of* 10^{-5}*. The time step size was chosen to be* 10^{-2}.

6.2. An advection-reaction-diffusion problem. In order to test the temporal error control of our meshfree integrator, we consider a semilinear advection-reaction-diffusion problem. The problem is given by the following differential equation:

$$\frac{\partial}{\partial t}u(t,x) = \epsilon \triangle u(t,x) - \alpha \nabla u(t,x) + \rho\, u(t,x)\big(u(t,x) - 1/2\big)\big(1 - u(t,x)\big) \tag{6.4}$$

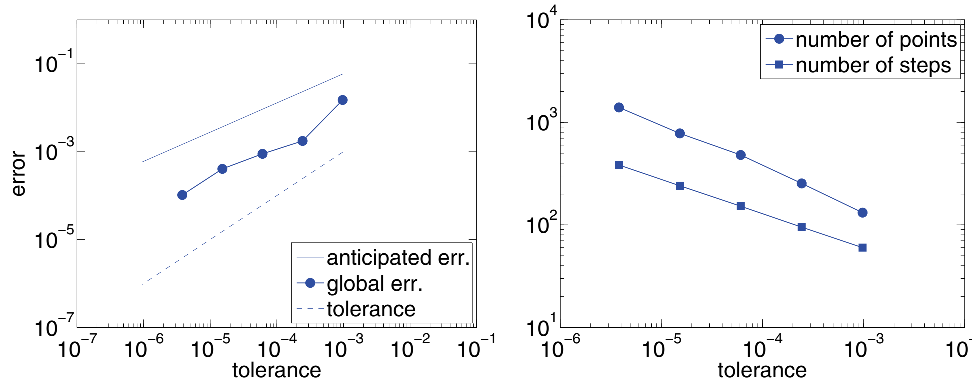

Fig. 5. *Error versus tolerance (left), number of interpolation points and number of time steps (right) at $T = 0.1$ for the meshfree exponential integrator when applied to* (6.4)*. The symbols mark the results, obtained for the prescribed tolerances* $\mathrm{TOL} = 4^{-j}$*,* $j = 5, \dots, 9$*.*

with $\epsilon = 5 \cdot 10^{-2}$, $\alpha = -6$, and $\rho = 70$. The computational domain and the initial solution are the same as in the previous subsection. The interpretation of the different terms of the equation is the following. The equation, supplied with homogeneous Neumann boundary conditions, has the three equilibria $u = 0$, $u = 1/2$, and $u = 1$. Both equilibria $u = 0$ and $u = 1$ are asymptotically stable, and the third one is unstable. The reaction term thus locally pulls values that are above $1/2$ to 1, and values below $1/2$ to 0. Considering just this term would eventually result in a discontinuous solution. The diffusion term now smooths the solution and the advection part transports it forward in time along the negative direction of the main diagonal. We perform three numerical experiments.

We start with an accuracy test. For prescribed tolerances TOL we integrate problem (6.4) up to $T = 0.1$ and determine the final error in the discrete maximum norm on an equidistant 50×50 grid. The threshold θ_r is chosen equal to TOL, the tolerance for the time integration θ_t is taken 10 times smaller. Since the problem contains a nonlinear part, the final error is composed of both errors. We also note that this time there is no need for an a priori step size restriction; we can simply use the built in step size selection of the embedded exponential Rosenbrock method `expr32`. The results are shown in Figure 5. The left figure displays the final error. One can see that the error is always below the anticipated error, which is θ_t times the number of required steps. In order to take larger time steps in this experiment one has to use higher order methods. The right figure contains the number of time steps and the number of interpolation points for a prescribed tolerance. The results show that the method is third-order in time, which we expect, since we use a Rosenbrock method of order three. The spatial order of convergence is better than predicted by (3.3), since the solution has enough regularity on the boundary; see [34, Thm. 7].

An important property of an adaptive integrator is that the error estimates in time and space do not influence each other. This means that if we take a small tolerance in time and a large one in space, we should see the spatial error and vice versa. Figure 6 displays the results of an experiment, where we integrate problem (6.4) up to $T = 0.1$ for different choices of the two tolerances. In the left figure we computed the average error per step for a prescribed tolerance θ_t and various spatial tolerances θ_r. In the right figure the opposite is shown: the spatial tolerance is kept fixed, whereas the integration tolerance is varied.

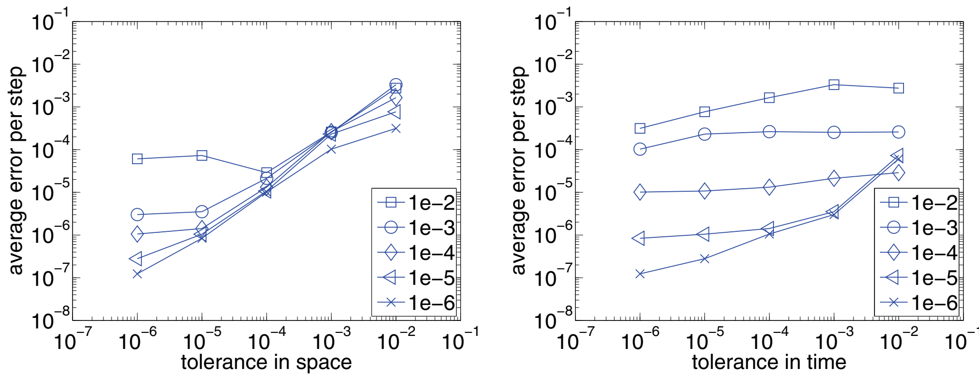

Fig. 6. *Average error per step for fixed choices of temporal tolerance and various choices of spatial tolerance (left), average error per step for fixed choices of spatial tolerance and various choices of temporal tolerance (right) for* (6.4) *when integrated up to* $T = 0.1$ *with our meshfree exponential integrator. The different symbols mark the results for the tolerances* 10^{-j}, $j = 2, \ldots, 6$, *in time and space, respectively.*

Both figures show that keeping one tolerance fixed and decreasing the other leads to a saturation of the error. We note that taking a prescribed tolerance in space gives a smaller average error per step than the desired tolerance. We also observed this in other experiments. The reason is that the adaptive error control in space is not a smooth process. Adding a point can decrease the error by orders of magnitudes, hence it is possible that adding one single point to the set of interpolation points leads to an interpolation error that is much smaller than the prescribed tolerance.

Finally, we plot a typical development of the set of interpolation points for (6.4). For this purpose we integrate (6.4) up to $T = 0.25$ with a prescribed tolerance of $5 \cdot 4^{-5}$. The results are displayed in Figure 7. In the beginning the convex hull of the interpolation points increases due to the diffusion term. Then, the point density is increased in the region where the slope of the function becomes steep. After some time an equilibrium between reaction and diffusion is reached. Therefore, the shape of the solution no longer changes and the number of interpolation points should remain constant. As the equation is nonlinear, the time integration error will influence the interpolation process, and the number of interpolation points will therefore slightly vary in time.

7. Summary.

In this paper we described the concept of a general meshfree exponential integrator. In particular we focused on advection dominated time-dependent PDEs, where the essential support of the solution moves in time. We used a residual subsampling procedure in order to find appropriate interpolation points for the spatial discretization. To keep the number of interpolation points small we used further information about the solution to predict the position of the essential support for the next time step. Time integration was performed with an exponential Rosenbrock method of order three. The integrator is particularly designed for situations where the computational domain is much larger than the essential support of the solution. In such situations a standard finite difference discretization would be prohibitively memory consuming. We showed that our integrator behaves very well with respect to stability and reliability. All numerical experiments were implemented in MATLAB. Note, however, that several parts of our integrator, e.g., the efficient implementation of the Leja points method on graphics processing units, are still a work in progress.

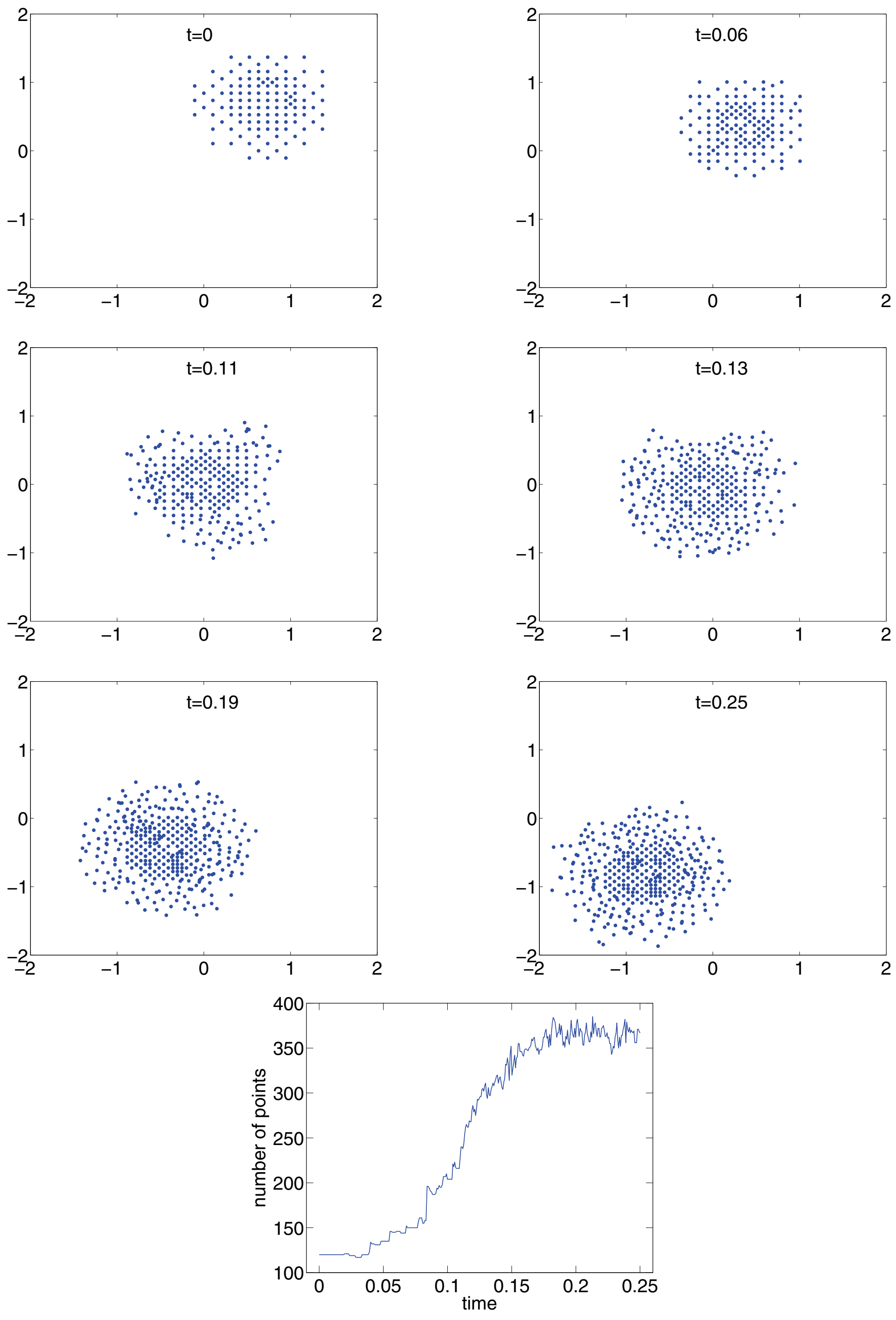

FIG. 7. *Evolution of the interpolation point set for the meshfree exponential integrator when applied to* (6.4). *The figures show the interpolation point set at times* $t = 0$, 0.06, 0.11, 0.13, 0.19, *and* 0.25, *respectively. The tolerance was chosen to be* $\text{TOL} = 4^{-5}$.

REFERENCES

[1] J. Behrens and A. Iske, *Grid-free adaptive semi-Lagrangian advection using radial basis functions*, Comput. Math. Appl., 43 (2002), pp. 319–327.

[2] J. Behrens, A. Iske, and M. Käser, *Adaptive meshfree method of backward characteristics for nonlinear transport equations*, in Meshfree Methods for Partial Differential Equations, M. Griebel and M. A. Schweitzer, eds., Springer, Berlin, 2003, pp. 21–36.

[3] L. Bergamaschi, M. Caliari, A. Martínez, and M. Vianello, *A parallel exponential integrator for large-scale discretizations of advection-diffusion models*, in Recent Advances in Parallel Virtual Machine and Message Passing Interface, B. Di Martino, D. Kranzlmüller, and J. Dongarra, eds., Springer, Berlin, Heidelberg, 2005, pp. 483–492.

[4] L. Bergamaschi, M. Caliari, A. Martínez, and M. Vianello, *Comparing Leja and Krylov approximations of large scale matrix exponentials*, in Computational Science—ICCS 2006, V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, eds., Springer, Berlin, 2006, pp. 685–692.

[5] M. D. Buhmann, *Radial basis functions*, Acta Numer., 9 (2000), pp. 1–38.

[6] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, Cambridge Monogr. Appl. Comput. Math. 12, Cambridge University Press, Cambridge, UK, 2003.

[7] M. Caliari, *Accurate evaluation of divided differences for polynomial interpolation of exponential propagators*, Computing, 80 (2007), pp. 189–201.

[8] M. Caliari and A. Ostermann, *Implementation of exponential Rosenbrock-type integrators*, Appl. Numer. Math., 59 (2009), pp. 568–581.

[9] M. Caliari, M. Vianello, and L. Bergamaschi, *Interpolating discrete advection-diffusion propagators at Leja sequences*, J. Comput. Appl. Math., 172 (2004), pp. 79–99.

[10] W. P. Crowley, *Numerical advection experiments*, Mon. Wea. Rev., 1 (1968), pp. 1–11.

[11] T. A. Driscoll and A. R. H. Heryudono, *Adaptive residual subsampling methods for radial basis function interpolation and collocation problems*, Comput. Math. Appl., 53 (2007), pp. 927–939.

[12] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific, Hackensack, NJ, 2007.

[13] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1236–1264.

[14] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations*. I. *Nonstiff Problems*, 2nd ed., Springer-Verlag, Berlin, 1993.

[15] M. Hochbruck and Ch. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[16] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.

[17] M. Hochbruck, A. Ostermann, and J. Schweitzer, *Exponential Rosenbrock-type methods*, SIAM J. Numer. Anal., 47 (2009), pp. 786–803.

[18] S. Hornus and J.-D. Boissonnat, *An Efficient Implementation of Delaunay Triangulations in Medium Dimensions*, Technical report, INRIA, Rennes, France, 2008.

[19] W. Hundsdorfer and J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer-Verlag, Berlin, 2003.

[20] E. J. Kansa, *Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics*. I. *Surface approximations and partial derivative estimates*, Comput. Math. Appl., 19 (1990), pp. 127–145.

[21] E. J. Kansa, *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), pp. 147–161.

[22] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998.

[23] A. Martínez, L. Bergamaschi, M. Caliari, and M. Vianello, *A massively parallel exponential integrator for advection-diffusion models*, J. Comput. Appl. Math., 231 (2009), pp. 82–91.

[24] C. R. Molenkamp, *Accuracy of finite-difference methods applied to the advection equation*, J. Appl. Meteor., 7 (1968), pp. 160–167.

[25] I. Moret and P. Novati, *RD-rational approximation of the matrix exponential operator*, BIT, 44 (2004), pp. 595–615.

[26] S. A. Sarra, *Adaptive radial basis function methods for time dependent partial differential equations*, Appl. Numer. Math., 54 (2005), pp. 79–94.

[27] R. Schaback, *Creating surfaces from scattered data using radial basis functions*, in Mathematical Methods in Computer Aided Geometric Design, M. Dæhlen, T. Lyche, and L. L. Schumaker, eds., Vanderbilt University Press, Nashville, TN, 1995, pp. 477–496.

[28] M. J. Schaefer, *A polynomial based iterative method for linear parabolic equations*, J. Comput. Appl. Math., 29 (1990), pp. 35–50.

[29] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals*, Electron. Trans. Numer. Anal., 29 (2007), pp. 1–18.

[30] H. Tal-Ezer, *Polynomial approximation of functions of matrices and application*, J. Sci. Comput., 4 (1989), pp. 25–60.

[31] A. Tambue, G. J. Lord, and S. Geiger, *An exponential integrator for advection-dominated reactive transport in heterogeneous porous media*, J. Comput. Phys., 229 (2010), pp. 3957–3969.

[32] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457.

[33] H. Wendland, *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, Adv. Comput. Math., 4 (1995), pp. 389–396.

[34] H. Wendland, *Sobolev-type error estimates for 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, pp. 337–344.

[35] H. Wendland, *Error estimates for interpolation by compactly supported radial basis functions of minimal degree*, J. Approx. Theory, 93 (1998), pp. 258–272.

[36] H. Wendland, *Scattered Data Approximation*, Cambridge University Press, Cambridge, UK, 2005.

[37] Z. Wu, *Compactly supported positive definite radial functions*, Adv. Comput. Math., 4 (1995), pp. 283–292.

[38] Z. Wu and R. Schaback, *Local error estimates for radial basis function interpolation of scattered data*, IMA J. Numer. Anal., 13 (1993), pp. 13–27.