Efficient preconditioning of the method of lines for solving nonlinear two-sided space-fractional diffusion equations

Timothy Moroney [*] **Qianqian Yang** [*]

[*] *School of Mathematical Sciences, Queensland University of Technology, GPO Box 2434, Brisbane, QLD 4001, Australia (e-mail: t.moroney@qut.edu.au, q.yang@qut.edu.au).*

Abstract: A standard method for the numerical solution of partial differential equations (PDEs) is the method of lines. In this approach the PDE is discretised in space using finite differences or similar techniques, and the resulting semidiscrete problem in time is integrated using an initial value problem solver.

A significant challenge when applying the method of lines to fractional PDEs is that the non-local nature of the fractional derivatives results in a discretised system where each equation involves contributions from many (possibly every) spatial node(s). This has important consequences for the efficiency of the numerical solver. First, since the cost of evaluating the discrete equations is high, it is essential to minimise the number of evaluations required to advance the solution in time. Second, since the Jacobian matrix of the system is dense (partially or fully), methods that avoid the need to form and factorise this matrix are preferred.

In this paper, we consider a nonlinear two-sided space-fractional diffusion equation in one spatial dimension. A key contribution of this paper is to demonstrate how an effective preconditioner is crucial for improving the efficiency of the method of lines for solving this equation. In particular, we show how to construct suitable banded approximations to the system Jacobian for preconditioning purposes that permit high orders and large stepsizes to be used in the temporal integration, without requiring dense matrices to be formed. The results of numerical experiments are presented that demonstrate the effectiveness of this approach.

Keywords: Nonlinear two-sided space-fractional diffusion equation; method of lines; Jacobian-free Newton-Krylov; banded preconditioning; finite differences

1. INTRODUCTION

In this paper we show how to construct an effective preconditioner for solving the nonlinear two-sided space-fractional diffusion equation

$$\frac{\partial u}{\partial t} = \kappa(u,x,t)\left[p\frac{\partial^\alpha u}{\partial x^\alpha} + (1-p)\frac{\partial^\alpha u}{\partial(-x)^\alpha}\right] + q(u,x,t) \quad (1)$$

using the method of lines, on the finite domain $0 < x < L$ with homogeneous Dirichlet boundary conditions and initial condition $u(x,0) = u_0(x)$. The fractional order α is assumed to satisfy $1 < \alpha \leq 2$. The function $u(x,t)$ can be interpreted as representing the concentration of a particle plume undergoing superdiffusion. The diffusion coefficient $\kappa(u,x,t)$ is assumed positive, and the forcing function $q(u,x,t)$ represents sources or sinks.

This equation has been discussed previously by Meerschaert and Tadjeran (2006), who considered the linear case. Here we consider the full nonlinear problem, where κ and q can be concentration-, space- and time-dependent.

Meerschaert and Tadjeran (2006) give the interpretation of the skewnesses $p \in [0,1]$ in terms of forward and backward jump probabilities in a stochastic model for anomalous diffusion. If $p = 0$ or $p = 1$ then (1) reduces to a one-sided space-fractional diffusion equation.

The left and right Riemann-Liouville space-fractional derivatives are defined by

$$\frac{\partial^\alpha u}{\partial x^\alpha} = \frac{1}{\Gamma(m-\alpha)}\frac{\mathrm{d}^m}{\mathrm{d}x^m}\int_0^x \frac{u(\xi,t)}{(x-\xi)^{\alpha-m+1}}\,\mathrm{d}\xi \quad (2)$$

and

$$\frac{\partial^\alpha u}{\partial(-x)^\alpha} = \frac{(-1)^m}{\Gamma(m-\alpha)}\frac{\mathrm{d}^m}{\mathrm{d}x^m}\int_x^L \frac{u(\xi,t)}{(\xi-x)^{\alpha-m+1}}\,\mathrm{d}\xi \quad (3)$$

with the integer m defined by $m-1 < \alpha \leq m$. The restriction on the fractional order discussed above means that $m = 2$ in this paper.

The method of lines for space-fractional PDEs has been used by a number of authors previously. Liu et al. (2004) used the method of lines to solve a fractional Fokker-Planck equation. Zhuang et al. (2009) used the method of lines to solve a variable-order fractional advection-diffusion equation. Yang et al. (2010) used the method of lines to solve a Reisz space fractional PDE.

The focus of this paper is to demonstrate how an effective preconditioner is crucial for improving the efficiency of the method of lines for solving the nonlinear two-sided space-

[1] The authors acknowledge the financial support provided by QUT's Science and Technology Research Career Development Programme and Early Career Academic Research Development Programme.

fractional diffusion equation (1). In particular, we show how to construct suitable banded approximations to the system Jacobian for preconditioning purposes that permit high orders and large stepsizes to be used in the temporal integration, without requiring dense matrices to be formed and factorised. This allows for the solution to be obtained using many more spatial nodes than would be possible with a method using direct Jacobian factorisation.

Krylov subspace iterative methods play a key role in our approach. Several authors have recently had success using Krylov subspace methods for solving space-fractional diffusion equations, including Ilić et al. (2008), Yang et al. (2011a,b) and Burrage et al. (2011). In these papers, Krylov subspace methods are used in the context of matrix function approximations for fractional Laplace equations. The present work is the only work we are aware of that uses preconditioned Krylov subspace methods to solve space-fractional differential equations using the method of lines.

The remainder of the paper is arranged as follows. In section 2 we present a finite difference spatial discretisation of (1) and show how it leads to a system of time ordinary differential equations. In section 3 we summarise the method of backward differentiation formulas for integrating initial value problems. The role of preconditioning is highlighted. In section 4 we derive a banded preconditioner that is suitable for the nonlinear two-sided space-fractional diffusion equation, and show how to construct it efficiently. In section 5 we present the results of numerical experiments that confirm that the preconditioner allows for the efficient solution of equation (1) using the method of lines. We draw our conclusions in section 6.

2. FINITE DIFFERENCE DISCRETISATION

To spatially discretise (1) using finite differences, we introduce a mesh with N uniform divisions of width $\Delta x = L/N$, and N+1 nodes $x_i,\ i = 0 \dots N$ where $x_i = i\Delta x$. Meerschaert and Tadjeran (2006) show how shifted Grünwald formulas may be used to approximate the space fractional derivatives $\partial^\alpha u(x_i,t)/\partial x^\alpha$ and $\partial^\alpha u(x_i,t)/\partial(-x)^\alpha$ by a weighted sum of neighbouring values $u(x_j,t)$. Using this approach, we obtain the spatial discretisation

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{\kappa_i}{\Delta x^\alpha}\left[p\sum_{j=0}^{i} g_j\, u_{i-j+1} + (1-p)\sum_{j=0}^{N-i} g_j\, u_{i+j-1}\right] + q_i \tag{4}$$

for $i = 1,\dots,N-1$, where $u_i(t)$ denotes the discrete approximation to $u(x_i,t)$, $\kappa_i(t)$ means $\kappa(u_i,x_i,t)$ (similarly q_i) and the normalised Grünwald weights g_j are given by

$$g_0 = 1\,,\quad g_j = (-1)^j\,\frac{\alpha(\alpha-1)\dots(\alpha-j+1)}{j!},\ j = 1,2,\dots. \tag{5}$$

When combined with the initial condition $u(x_i,0) = u_0(x_i),\ i = 1\dots N-1$, the system generated by imposing (4) at each internal node can be written as the initial value problem (IVP)

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(t,\mathbf{u}),\quad \mathbf{u}(t_0) = \mathbf{u}_0 \tag{6}$$

where the components of the vector $\mathbf{u}$ are the unknowns u_i, the components of the vector-valued function $\mathbf{f}$ come from the right hand side of (4) and the components of $\mathbf{u}_0$ are the initial values.

As is discussed in the next section, the Jacobian matrix $\mathbf{J} = \partial\mathbf{f}/\partial\mathbf{u}$ plays an important role in the numerical solution of (6). A distinguishing characteristic of space fractional PDEs is that finite difference discretisations of these problems give rise to Jacobian matrices which are fully or partially dense.

The Jacobian matrix for (4) will be fully dense if $0 < p < 1$, since together the two summations in (4) range over all u_j, implying that every component of $\mathbf{u}$ appears in every equation. For $p = 0$ or $p = 1$, only one sum remains, and the Jacobian matrix has a Hessenberg structure.

3. SOLUTION OF INITIAL VALUE PROBLEMS WITH BACKWARD DIFFERENTIATION FORMULAS AND JACOBIAN-FREE NEWTON-KRYLOV METHODS

3.1 Backward differentiation formulas

In this section, we discuss the use of backward differentiation formulas for solving (6), with special attention paid to the role of the Jacobian matrix, and the impact of its density on the efficiency of the approach.

The backward differentiation formulas (BDFs) comprise a family of implicit linear multistep methods for solving the initial value problem (6). Numerous IVP solvers utilising BDFs are available – the implementation used for this paper is the CVODE solver which is part of the SUNDIALS suite of nonlinear and differential/algebraic equation solvers (Hindmarsh et al., 2005).

We introduce BDFs by considering how to step from time $t = t_{n-1}$, at which point the numerical solution $\mathbf{u}_{n-1} \approx \mathbf{u}(t_{n-1})$ is known, to the next point in time $t_n = t_{n-1}+h_n$, where h_n is the stepsize. The derivation follows that of Hindmarsh et al. (2005).

The defining characteristic of BDFs is the approximation of the derivative $\mathrm{d}\mathbf{u}(t_n)/\mathrm{d}t$ in terms of present and past values of $\mathbf{u}$:

$$\mathrm{d}\mathbf{u}(t_n)/\mathrm{d}t \approx \frac{1}{h_n}\sum_{k=0}^{q}\alpha_{n,k}\mathbf{u}_{n-k} \tag{7}$$

where q is the order of the BDF, and the coefficients $\alpha_{n,k}$ depend on the recent stepsize and order history. The backward Euler method is the simplest and best-known BDF; it corresponds to $q = 1$ (first order), with coefficients $\alpha_{n,0} = 1$ and $\alpha_{n,1} = -1$.

Modern BDF-based IVP solvers use sophisticated algorithms to adaptively vary both the stepsize h_n and the order q in order to achieve a desired local error tolerance at each step, while keeping the stepsize as large as possible. This is a key advantage of such solvers over hand-coded backward Euler or similar methods; they offer high performance temporal integration using complex algorithms that have been subject to extensive testing and peer review, and which would be time-consuming to implement manually.

Evaluating (6) at $t = t_n$, substituting (7) and rearranging for the unknown $\mathbf{u}_n$ yields the nonlinear algebraic equation

$$\mathbf{g}_n(\mathbf{u}_n) := \mathbf{u}_n - \gamma_n\mathbf{f}(t_n,\mathbf{u}_n) + \mathbf{a}_n = \mathbf{0} \tag{8}$$

where $\gamma_n = h_n/\alpha_{n,0}$ and $\mathbf{a}_n = \sum_{k=1}^{q}(\alpha_{n,k}/\alpha_{n,0})\mathbf{u}_{n-k}$. This equation must be solved to advance the numerical solution in time.

Newton's method applied to (8) yields the iteration

$$\mathbf{u}_n^{k+1} = \mathbf{u}_n^k + \delta\mathbf{u}_n^k \tag{9}$$

where $\mathbf{u}_n^k$ is the kth iterate in the sequence $\{\mathbf{u}_n^k\}_{k=0}^{\infty} \to \mathbf{u}_n$ and the correction vector $\delta\mathbf{u}_n^k$ is found by solving

$$(\mathbf{I} - \gamma_n\mathbf{J}(\mathbf{u}_n^k))\delta\mathbf{u}_n^k = -\mathbf{g}_n(\mathbf{u}_n^k) \tag{10}$$

which is a linear system involving the Jacobian matrix $\mathbf{J} = \partial\mathbf{f}/\partial\mathbf{u}$.

3.2 Jacobian-free Newton-Krylov methods

Modern IVP solvers use Krylov subspace methods to solve the linear system (10). Briefly, a Krylov subspace method for the system $\mathbf{Ax} = \mathbf{b}$ seeks an approximate solution $\tilde{\mathbf{x}}$ by projecting onto the Krylov subspace

$$\mathcal{K}_m(\mathbf{A}, \mathbf{b}) = \text{span}\{\mathbf{b}, \mathbf{Ab}, \ldots, \mathbf{A}^{m-1}\mathbf{b}\} \tag{11}$$

(Saad, 1999). Such methods require the action of the matrix $\mathbf{A}$ only in the form of matrix-vector products on suitably-chosen vectors $\mathbf{v}$. In the context of the linear system (10), we have

$$\mathbf{A} = \mathbf{I} - \gamma_n\mathbf{J}(\mathbf{u}_n^k) \tag{12}$$

and a key observation is that the product $\mathbf{J}(\mathbf{u}_n^k)\mathbf{v}$ can be approximated by a first order forward difference

$$\mathbf{J}(\mathbf{u}_n^k)\mathbf{v} \approx \frac{\mathbf{f}(t_n, \mathbf{u}_n^k + \epsilon\mathbf{v}) - \mathbf{f}(t_n, \mathbf{u}_n^k)}{\epsilon} \tag{13}$$

with suitably-chosen shift value ϵ (Knoll and Keyes, 2004).

In this way, a solution to (10) may be found without ever forming the Jacobian matrix $\mathbf{J}$. This Jacobian-free approach, combined with Newton's method, leads to a class of Jacobian-free Newton-Krylov (JFNK) methods for solving (8) (Knoll and Keyes, 2004).

In the context of the nonlinear two-sided space-fractional diffusion equation (1), IVP solvers that utilise JFNK methods are particularly attractive. We saw in section 2 that the Jacobian matrix for this problem is dense, and hence the ability to solve it without needing to form and factorise this matrix represents a significant saving.

Furthermore, the adaptive order and stepsize selections made by the IVP solver are designed to reduce the number of steps required to advance the solution in time. This represents a further saving over non-adaptive methods, which may require additional steps to achieve the same level of accuracy for a given point in time.

In spite of these facts, it is well-known that BDF-based solvers can suffer from poor performance when applied to stiff problems (Hindmarsh et al., 2005; Knoll and Keyes, 2004). Krylov subspace methods that project onto the space (11) typically do not achieve an acceptable rate of convergence for these problems.

Preconditioning is the standard approach for overcoming this issue, so that rather than solve $\mathbf{Ax} = \mathbf{b}$ directly, the preconditioned system [2]

$$(\mathbf{AM}^{-1})(\mathbf{Mx}) = \mathbf{b},$$

[2] This is preconditioning on the right. Preconditioning on the left is also possible, but not considered in this paper.

is used to solve for $\mathbf{z} = \mathbf{Mx}$ by projecting onto the preconditioned Krylov subspace

$$\mathcal{K}_m(\mathbf{AM}^{-1}, \mathbf{b}) = \text{span}\{\mathbf{b}, \mathbf{AM}^{-1}\mathbf{b}, \ldots, (\mathbf{AM}^{-1})^{m-1}\mathbf{b}\}\,. \tag{14}$$

Here $\mathbf{M}$ is a preconditioner matrix which in some sense approximates $\mathbf{A}$, but whose inverse can be efficiently applied. A typical approach for forming such a preconditioner matrix is to form an (often quite crude) approximation to the Jacobian matrix $\mathbf{J}$ itself, and then construct $\mathbf{M}$ by way of equation (12) (Knoll and Keyes, 2004).

Furthermore, modern IVP solvers do not insist that the preconditioner always be kept up to date. Instead, they include logic to detect when an out-of-date preconditioner is hindering convergence, and only then is the user code to update the matrix invoked (Hindmarsh et al., 2005).

4. A BANDED PRECONDITIONER

In this section, we derive a banded preconditioner for the nonlinear two-sided space-fractional diffusion equation (1), which overcomes the problem of stiffness and allows efficient solution of the problem using BDF-based IVP solvers. We begin in section 4.1 by illustrating the process for a linear problem, before generalising the approach to the full nonlinear problem in section 4.2.

4.1 Linear case with κ constant and $q = q(x,t)$

We seek an approximation of the Jacobian matrix which is cheap to compute and invert, but which captures the dominant source of stiffness in the problem. To illustrate the idea, we first consider the simpler, linear problem of equation (1) with κ constant, and $q = q(x,t)$. By examining the right hand side of the spatially discrete form (4) we find that the Jacobian matrix, $\mathbf{J}$, for this problem has the form

$$(\mathbf{J})_{ij} = \frac{\kappa}{\Delta x^\alpha} \times \begin{cases} pg_{i-j+1}, & j < i-1 \\ pg_2 + (1-p)g_0, & j = i-1 \\ g_1, & j = i \\ pg_0 + (1-p)g_2, & j = i+1 \\ (1-p)g_{j-i+1}, & j > i+1 \end{cases}\,. \tag{15}$$

Recalling definition (5) of the normalised Grünwald weights, the key observation from (15) is that the magnitudes of the entries $(\mathbf{J})_{ij}$ become smaller as $|i-j|$ increases. That is, the entries become smaller, the further from the diagonal they are. This is consistent with the physical notion that the concentration at a given point is influenced more strongly by the concentration at points nearby, and less strongly by the concentration at points far away.

In Fig. 1 we plot the magnitude of the Jacobian's entries for the problem (4) with $\kappa = 1$, $p = 0.5$, $\alpha = 1.8$, $q = 0$ and $N = 4000$ to illustrate this point. The figure strongly suggests that the dominant behaviour of this problem is captured by the values within the dark diagonal strip of small bandwidth. This motivates the idea of retaining only the values within this small bandwidth and using the resulting matrix to form a preconditioner. Standard banded or sparse data structures and algorithms allow for the efficient storage and factorisation of this matrix, with substantial savings compared to the cost of forming and factorising the full Jacobian.

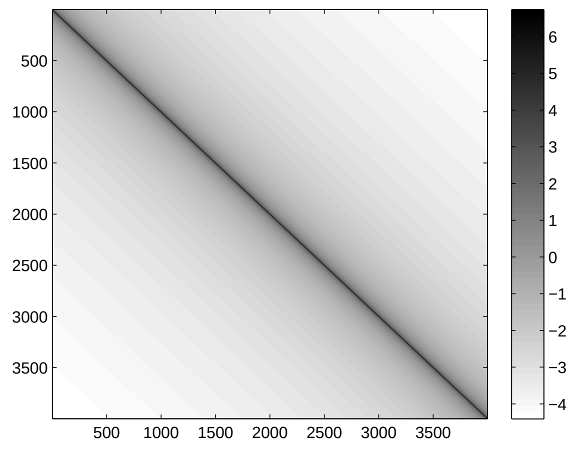

Fig. 1. Log to base 10 of the magnitude of the Jacobian's entries for problem (4) with $\kappa = 1$, $p = 0.5$, $\alpha = 1.8$, $q = 0$ and $N = 4000$. Black indicates largest magnitude, through grey and then white for smallest magnitude.

4.2 General, nonlinear case

While the preceding discussion motivates the use of a banded Jacobian approximation, in practice it can be inconvenient to form it directly by means of (15). Indeed, for problems with variable κ or a source term q dependent on u, equation (15) is no longer correct, and would need to be re-derived accordingly.

Fortunately there is a straightforward means of approximating the banded Jacobian which applies for general nonlinear problems. First, we observe that a single column of the Jacobian matrix can be approximated by a first order forward difference in the manner of (13):

$$\mathbf{J}_{\bullet j}(\mathbf{u}) = \mathbf{J}(\mathbf{u})\mathbf{e}_j \approx \frac{\mathbf{f}(t, \mathbf{u} + \epsilon\, \mathbf{e}_j) - \mathbf{f}(t, \mathbf{u})}{\epsilon} \qquad (16)$$

where $\mathbf{J}_{\bullet j}$ means the jth column of $\mathbf{J}$ and $\mathbf{e}_j$ is the jth coordinate vector (zero elements except for a one in position j). Equation (16) thus provides a means to build up a finite difference approximation of the Jacobian one column at a time, by shifting one component of $\mathbf{u}$ at a time.

We make several improvements to the efficiency of this basic approach. First, we exploit the (approximately) banded nature of the Jacobian and shift multiple components of $\mathbf{u}$ at a time, which are separated from one another by a distance of one bandwidth (Kelley, 2003). For example, if the bandwidth is 5, then in our first evaluation we shift u_1, u_6, u_{11} and so forth. For our second evaluation we shift u_2, u_7, u_{12} and so forth. In this way, a finite difference approximation to the Jacobian is constructed with just 5 (the bandwidth) additional evaluations of the function $\mathbf{f}$.

While this approach is the standard means of approximating banded Jacobians (Kelley, 2003), it should be noted that as the Jacobian in our problem is only approximately banded, this approach does introduce some small additional error into the computed values. In our numerical experiments this additional error was found to be of no consequence.

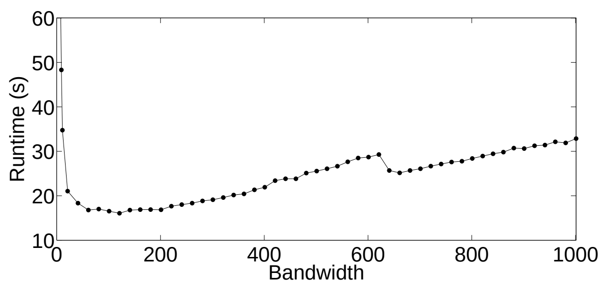

Fig. 2. Time taken to solve test problem 1 to time $t = 1$ versus the bandwidth of the preconditioner used.

The second efficiency improvement we make is to recognise that when evaluating $\mathbf{f}$ with shifted components (a "shifted evaluation"), we can recycle most of the information from the prior unshifted evaluation.

For the nonlinear two-sided space-fractional discretisation, the greatest expense in an evaluation of $\mathbf{f}$ is computing the two sums in equation (4). We exploit this by keeping auxiliary vectors of "unshifted sums"; two vectors of the same length as $\mathbf{u}$ are all that is required. These vectors are updated every time an unshifted evaluation is performed. Then, for all subsequent subsequent shifted evaluations, only small adjustments to these stored sums are necessary, corresponding to the terms which involve the shifted components.

In the next section we present the results of some numerical experiments that demonstrate the effectiveness of the overall method for solving equation (1).

5. NUMERICAL EXPERIMENTS

All numerical experiments were carried out in MATLAB version R2011a, 64-bit edition, using the CVODE IVP solver, part of the SUNDIALS suite of nonlinear and differential/algebraic equation solvers (Hindmarsh et al., 2005). Absolute and relative error tolerances were set to 10^{-6} and preconditioned GMRES (Saad and Schultz, 1986) was used for the Krylov subspace method. We used MATLAB's native sparse data structure and algorithms to store and factorise the banded Jacobian (MATLAB does not natively support band matrix storage – for a discussion of this design choice see Gilbert et al. (1992)). The test machine used an Intel Core i7 processor and 4 GB of RAM.

5.1 Test problem 1: linear

We consider equation (1) with parameters $\kappa = 1$, $p = 0.5$, $\alpha = 1.8$, $q = 0$, with $N = 4000$ spatial divisions and initial condition $u_0(x) = x(1 - x)$. The structure of the Jacobian matrix for this problem was illustrated in Fig. 1.

We begin by examining the effect that the bandwidth of the Jacobian matrix used to form the preconditioner has on the efficiency of the scheme. Fig. 2 plots the runtime taken to simulate to $t = 1$ against the bandwidth. As this is a linear problem, the Jacobian matrix needs to be formed just once, and we use the formulas (15) to compute all values within the chosen bandwidth.

From Fig. 2 we see that for bandwidths less than about 10, the BDF-based solver is not efficient and we conclude

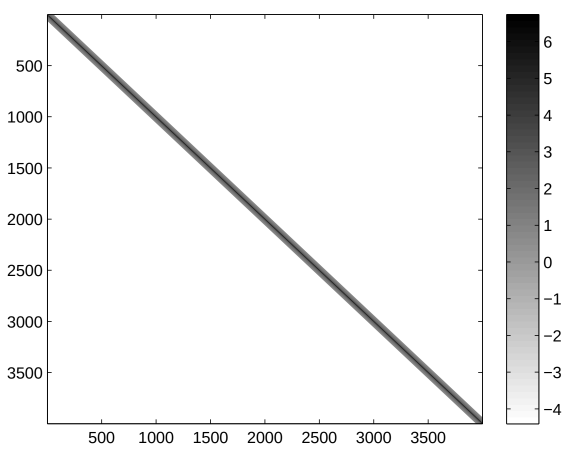

Fig. 3. Log to base 10 of the magnitude of the banded Jacobian approximation's entries for test problem 1. Black indicates largest magnitude, through grey for smaller magnitudes and white for zero. The colour scale is identical to that of Fig. 1 so that the two figures are directly comparable.

that the preconditioner is not adequately capturing the behaviour of the problem. We note that the vertical axis on this plot has been truncated to preserve space; the runtime for a bandwidth of 3 (tridiagonal preconditioner) is more than 140 seconds, and with no preconditioning at all it takes more than 400 seconds to solve.

As the bandwidth increases, the runtime improves, until it attains a minimum value for this problem of approximately 16 seconds with bandwidth 121. At this point, we conclude that all of the important behaviour in the problem is being captured by the banded Jacobian approximation. Increasing the bandwidth further from this point tends to increase the runtime, as the costs associated with forming and factorising the larger bandwidth matrix become more significant. The interesting (and reproducible) dip in runtime at bandwidth 641 is caused by a sudden improvement in the efficiency of MATLAB's sparse LU factorisation (by some 25%!) at this and higher bandwidths.

In Fig. 3 we plot the magnitude of the banded Jacobian approximation's entries when the bandwidth is 121. We observe that a bandwidth of 121 corresponds to the dark diagonal strip first exhibited in Fig. 1. Hence, we have further confirmation that this portion of the matrix is sufficient to capture the dominant problem behaviour in terms of constructing an effective preconditioner.

To further explore the effect of the preconditioner on the efficiency of the solver, we examine how the order of the BDF and the stepsize evolve over two runs of the solver: one using bandwidth 3, which we know from the previous discussion to be too small, and the other using bandwidth 121, which we identified as being optimal for this problem.

Fig. 4 plots the order and stepsize evolution for both runs; order is indicated by a solid line and stepsize by a dashed line. In Fig. 4(a), where a bandwidth of 3 was used, we see that the stepsizes increase rapidly at first, and the order of the BDF rises to 4. However, the order quickly falls back

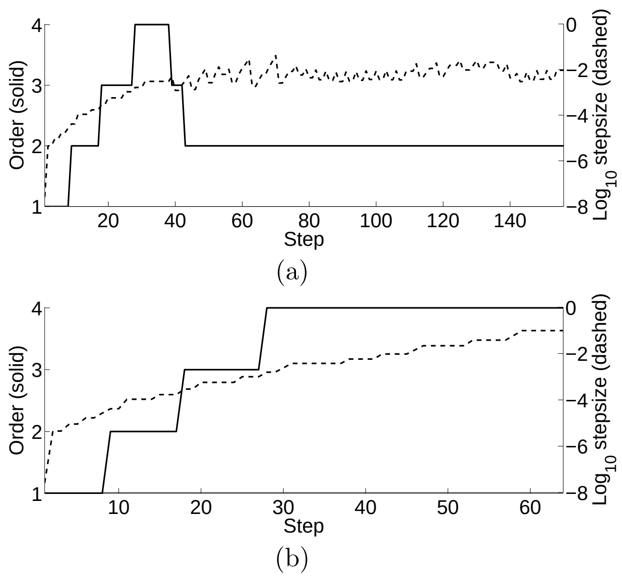

Fig. 4. Order and stepsize evolution for test problem 1 with: (a) preconditioner bandwidth of 3; (b) preconditioner bandwidth of 121.

to 2, and the stepsize evolution is erratic for the remainder of the simulation.

For a bandwidth of 121, Fig. 4(b) reveals a different picture. The order rises to 4 and remains there for the rest of the simulation, while the stepsizes increase monotonically throughout. As a result, the solver with bandwidth 121 is able to reach the final time $t = 1$ using just 64 steps and 151 function evaluations, compared with 156 steps and 1552 function evaluations required with bandwidth 3.

The large discrepancy in the number of function evaluations between the two runs is a result of the many additional iterations required to converge the JFNK method when the preconditioner is not performing adequately. This emphasises the fact that the preconditioner functions not only to reduce the number of steps required, but also to reduce the amount of work required per step.

5.2 Test problem 2: nonlinear

We now consider the nonlinear problem of equation (1) with $\kappa = 0.05u^{0.3}$, $p = 0.5$, $\alpha = 1.6$, $q = 2u^2(1-u)$ and initial condition $u_0(x) = x(1-x)$. We choose a mesh comprising $N = 16000$ divisions, as this represents a level of refinement for which standard direct factorisation methods using the full Jacobian are beyond the capacity of the test machine. To construct the preconditioner for this nonlinear problem we use the banded finite difference approximation to the Jacobian matrix discussed in section 4.2, with a bandwidth of 301.

Solving the problem to steady-state took 595 seconds using the test machine. The evolution of the order and stepsize is illustrated in Fig. 5. We see that for the bulk of the integration, the solver was operating at either fourth or fifth order, and the stepsizes increased monotonically throughout. As the solution approached its steady state, there was a rapid increase in the stepsizes, and the order of the method was able to drop back to first order, by which point the solution was essentially unchanging in time.

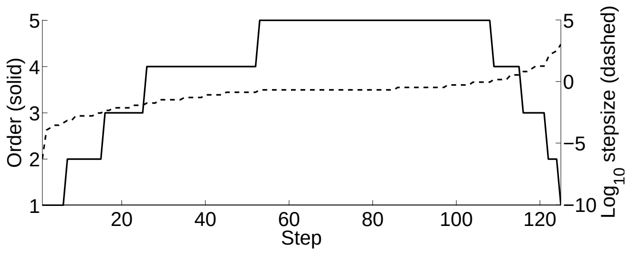

Fig. 5. Order and stepsize evolution for test problem 2 with preconditioner bandwidth of 301.

The figure suggests that the method performed close to optimally in terms of the size of the steps taken, confirming the effectiveness of the preconditioner. The relevant statistics for the simulation are that 125 steps were taken, the banded Jacobian approximation was formed 3 times and a total of 1322 function evaluations were required. Of those evaluations, 903 were shifted evaluations associated with forming the banded Jacobian approximation, and the remaining 419 were ordinary, nonshifted evaluations associated with Newton-Krylov iteration.

Despite the much larger number of shifted function evaluations, the total time spent in these evaluations was just 76 seconds (13% of total runtime), compared to 483 seconds (81% of total runtime) spent in nonshifted function evaluations. This confirms the significant efficiency improvements made in recycling information from the nonshifted evaluations when computing shifted evaluations.

The next most significant expense was the preconditioner factorisation. CVODE invokes the user code for factorisation whenever the stepsize (and hence the value of γ in equation (10)) changes. As a result, the factorisation routine was called 29 times, for a total cost of 27 seconds, or 4% of the total runtime. The only other significant expense was the application of the preconditioner, which occurs once for every Newton-Krylov iteration and cost 4 seconds (1% of total runtime) overall. The remaining 1% of runtime was associated with data structure and function call overheads. We note that all of these expenses were incurred in user code – there was no significant overhead associated with any CVODE code.

We emphasise that this problem could not be solved using a direct factorisation of the full Jacobian matrix on the test machine, since the memory required for the matrix and its factorisation exceeds the capacity of the machine. In comparison, using the preconditioned method of lines, the problem is comfortably solved in just a few minutes.

6. CONCLUSIONS

In this paper we have presented a banded preconditioner for the nonlinear two-sided space-fractional diffusion equation that allows for its efficient solution using the method of lines with backward differentiation formulas and Jacobian-free Newton-Krylov iteration. The advantage of our approach is that it avoids the need to factorise a dense Jacobian matrix that would otherwise be required using standard direct solution approaches. As a result, we are able to solve problems on computational domains involving many thousands of nodes, which would would be infeasible to do using direct factorisation methods. Numerical experiments illustrate that our preconditioner performs very well, allowing the initial value problem solver to use large stepsizes, and thereby integrate to the final time with minimal function evaluations.

In future work we will extend these ideas to problems in higher spatial dimensions, and to more complex problems.

REFERENCES

K. Burrage, N. Hale, and D. Kay. An efficient implementation of an implicit FEM scheme for fractional-in-space reaction-diffusion equations. *Numerical Analysis Technical Report, Oxford*, 2011.

J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM. J. Matrix Anal. & Appl.*, 13:333–356, 1992.

A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005.

M. Ilić, I. W. Turner, , and V. Anh. A numerical solution using an adaptively preconditioned Lanczos method for a class of linear systems related with the fractional Poisson equation. *Journal of Applied Mathematics and Stochastic Analysis*, Article ID 104525(26 pages), 2008.

C. T. Kelley. *Solving nonlinear equations with Newton's method*, page 30. SIAM, Philadelphia, 2003.

D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193:357–397, 2004.

F. Liu, V. Anh, and I. Turner. Numerical solution of the space fractional Fokker–Planck equation. *Journal of Computational and Applied Mathematics*, 166(1):209 – 219, 2004.

M. M. Meerschaert and C. Tadjeran. Finite difference approximations for two-sided space-fractional partial differential equations. *Appl. Numer. Math.*, 56:80–90, January 2006.

Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*, page 157. SIAM, Philadelphia, 1999.

Y. Saad and M. H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

Q. Yang, F. Liu, and I. Turner. Numerical methods for fractional partial differential equations with Riesz space fractional derivatives. *Applied Mathematical Modelling*, 34(1):200 – 218, 2010.

Q. Yang, T. Moroney, K. Burrage, I. Turner, and F. Liu. Novel numerical methods for time-space fractional reaction diffusion equations in two dimensions. *Australian and New Zealand Industrial and Applied Mathematics Journal*, 52:C395 – C409, 2011a.

Q. Yang, I. Turner, F. Liu, and M. Ilić. Novel numerical methods for solving the time-space fractional diffusion equation in 2d. *SIAM Journal on Scientific Computing*, 33(3):1159 – 1180, 2011b.

P. Zhuang, F. Liu, V. Anh, and I. Turner. Numerical methods for the variable order fractional advection-diffusion equation with a nonlinear source term. *SIAM J. Numer. Anal.*, 47:1760–1781, 2009.