551517

Comparison of Implicit Collocation Methods for the Heat Equation

Jules Kouatchou*
NASA Goddard Space Flight Center
Code 931
Greenbelt, MD 20771

Fabienne Jézéquel†
Université de Paris 6
Laboratoire d'Informatique
F-75015 Paris

Abstract

We combine a high-order compact finite difference scheme to approximate spatial derivatives and collocation techniques for the time component to numerically solve the two dimensional heat equation. We use two approaches to implement the collocation methods. The first one is based on an explicit computation of the coefficients of polynomials and the second one relies on differential quadrature. We compare them by studying their merits and analyzing their numerical performance. All our computations, based on parallel algorithms, are carried out on the CRAY SV1.

Key words: Collocation methods, differential quadrature, high-order compact scheme, iterative methods, parallel algorithm.

1 Introduction

In [11] Kouatchou presented an implicit technique to numerically solve the two-dimensional heat equation. The method, called the implicit collocation method (ICM), consists of first discretizing in space (using a fourth order compact scheme) the equation. The solution is approximated at each spatial grid point by a polynomial depending on time. The resulting derivation produces a linear system of equations. The order of the method is in space the order of the difference approximation and in time the degree of the polynomial [5, 8, 9, 10]. ICM when implemented on parallel computers, allows the parallelization across both time and space, i.e., at each iteration of the parallel algorithm, we can obtain the solution in the entire spatial domain at several consecutives time levels. Kouatchou proposed two parallel algorithms to compare ICM and the Crank-Nicolson method (known to be implicit) in [12]. His numerical experiments carried out on the SGI Origin 2000, showed that under some reasonable assumptions, ICM can be more cost effective than the Crank-Nicolson method. It was also observed in [13, 14] that higher-order algorithms in time (as ICM) can be made competitive with conventional time-marching algorithms, particularly if high accuracy is needed.

*E-mail: kouatchou@gsfc.nasa.gov. This author is also affiliated with Morgan State University and his research was supported by NASA under the grant No. NAGS-3508.

†E-mail: Fabienne.Jezequel@lip6.fr.

In [10, 11, 12], the authors used uniform time steps to derive the system of equations. In addition, regular polynomials were employed for the time discretization. It is an interesting problem to not only rely on non-uniform time steps but also to include orthogonal polynomials.

In this paper, we reformulate the implicit collocation method for the two dimensional heat equation. We present two methods to approximate the time components. The first method (ICM-EP), described above, is based on an explicit determination of polynomial of degree r that approximates at each spatial grid point the time components at r consecutive time levels [11, 12]. The second method (ICM-DQ) uses instead differential quadratures to find the solution at r prescribed time levels. The underlying approach in the second method still (indirectly) relies on polynomials of degree r that may be more "stable". In our new formulation we employ known results to show that the proposed methods have unique solutions, are implicit, are stable, and produce high accurate solutions. We perform basic comparisons of the two methods and discuss their merits. In our numerical experiments, we present the accuracy of the approximated solution and the parallel efficiency of each approach.

An outline of the paper is as follows. Section 2 presents the techniques used to discretize the equation in space and the concept of collocation for the time variable. Section 3 discusses the implementation strategies. Numerical experiments appear in Section 4. We formulate some conclusions in Section 5.

2 Derivation of the System of Equations

We consider the two dimensional heat equation:

$$\frac{\partial u}{\partial t}(x,y,t) = \alpha^2 \left(\frac{\partial^2 u}{\partial x^2}(x,y,t) + \frac{\partial^2 u}{\partial y^2}(x,y,t) \right), \qquad (x,y,t) \in \Omega \times [0,\infty) \qquad (1)$$

where $\Omega = (0,1) \times (0,1)$, and with the initial condition

$$u(x,y,0) = \psi(x,y), \quad (x,y) \in \Omega,$$

and with the assumption that $u(x,y,t)$ is known for any (x,y) in the boundary $\delta\Omega$ and for $t \geq 0$. We also assume that $u(x,y,t)$ is a smooth function on $\bar{\Omega}$.

The above equation models the flow of heat in the unit square domain that is insulated except on the boundary. α^2 is the thermal diffusivity.

2.1 Spatial Discretization

Let $h = 1/n$ be the uniform spatial mesh-width. We can subdivide the spatial domain as follows:

$$x_i = ih, \;\; y_j = jh, \qquad i,j = 0,1,\ldots,n.$$

For simplicity, we write the approximated solution of u and its time derivative at the spatial grid points (x_i, y_j) as:

$$U_{i,j}(t) = u(x_i,y_j,t), \;\; \text{and} \;\; U'_{i,j}(t) = \frac{\partial u}{\partial t}(x_i,y_j,t).$$

At any given time t, if we use the discretization of the steady state Poisson equation with a fourth-order scheme [7], we can approximate the spatial derivatives of (1). We obtain for any interior grid point (x_i, y_j):

$$\begin{aligned} & \tfrac{1}{2}\left[U'_{i+1,j}(t) + U'_{i,j+1}(t) + U'_{i-1,j}(t) + U'_{i,j-1}(t) + 8U'_{i,j}(t)\right] \\ = \; & \tfrac{\alpha^2}{h^2}[4\left(U_{i+1,j}(t) + U_{i,j+1}(t) + U_{i-1,j}(t) + U_{i,j-1}(t)\right) \\ & + U_{i+1,j+1}(t) + U_{i-1,j+1}(t) + U_{i-1,j-1}(t) + U_{i+1,j-1}(t) - 20U_{i,j}(t)], \end{aligned} \tag{2}$$

Eq. 2 is a system of $\tilde{n} = (n-1)^2$ ordinary differential equations and for any value of t, it is fourth-order in space. The system can be rewritten in a matrix-vector form as:

$$MU'(t) = WU(t) + c(t), \tag{3}$$

where $M = \frac{1}{2}tri[I_{n-1}, M_l, I_{n-1}]_{n-1}$, $W = \frac{\alpha^2}{h^2}tri[W_{l-1}, W_l, W_{l+1}]_{n-1}$ and $c(t)$ is a $\tilde{n}$ vector containing the values of u and its time partial derivative at the boundary grid points at time t. $M_l = tri[1, 8, 1]_{n-1}$, $W_{l-1} = tri[1, 4, 1]_{n-1}$, $W_l = [4, -20, 4]_{n-1}$, and $W_{l+1} = tri[1, 4, 1]_{n-1}$ are matrices all of order $n-1$. The initial condition $U(0)$ is given by the function $\psi(x, y)$. Here I_{n-1} is the identity matrix of order $n-1$ and $tri[a_{l-1}, a_l, a_{l+1}]_{n-1}$ denotes the tridiagonal matrix whose l^{th} row contains the values a_{l-1}, a_l and a_{l+1} on its subdiagonal, diagonal and superdiagonal respectively. The subdiagonal of the first row and the superdiagonal of the last row are not defined. The subscript $n-1$ determines the number of rows (or block-rows if the a_l are block-matrices).

Because M and W are constant nonsingular matrices and that the initial and boundary conditions are properly known, we can derive the following result:

Lemma 1 *The system (3) has a unique solution $U(t)$.*

The solution $U(t)$ is not only continuous in time but is also smooth (because of previous assumptions), i.e., at least the first few derivatives with respect to time exit and are continuous too. We can now introduce the principle of the collocation method to evaluate the time derivative.

2.2 Collocation Methods

Without lost of generality, we suppose that we are interested in finding the solution for the values of t in the interval $[0, \tau]$, where τ is any positive number. Consider r arbitrary mesh points given by:

$$0 = t_0 < t_1 < \cdots < t_{r-2} < t_{r-1} = \tau. \tag{4}$$

The r-stage collocation method is completely defined as a function of the above time mesh points by assuming that the approximated solution is reduced to polynomials of degree at most r. We then require that the approximated solution satisfies the initial and boundary conditions, and that it satisfies the differential equation (3) at the collocation points t_k, $k = 0, \cdots, r-1$. We propose two approaches.

Method 1: Explicit Polynomials (ICM-EP)
At any spatial grid point (x_i, y_j), we assume that the approximated solution at t_k ($k =$

$0, \ldots, r-1$) can be written as a polynomial of degree r:

$$U_{i,j}(t_k) \approx P_{i,j}(t_k) = a_{i,j,r} t_k^r + a_{i,j,r-1} t_k^{r-1} + \cdots + a_{i,j,1} t_k + a_{i,j,0}.$$

The polynomial expressions are substituted in (2) leading to a linear system of equations with $r\tilde{n}$ unknowns, $a_{i,j,1}, \ldots, a_{i,j,r}$, for $i, j = 1, \ldots, n-1$. We note that the $a_{i,j,0}$ are known and are equal to $u(x_i, y_j, 0) = \psi(x_i, y_j)$. After some algebraic manipulations we obtain the linear system of $r\tilde{n}$ equations [11, 12]:

$$AX = S, \tag{5}$$

where A is a block-tridiagonal matrix given by

$$A = tri\left[A_{l-1}, A_l, A_{l+1}\right]_{n-1}.$$

A_{l-1}, A_l and A_{l+1} are square matrices of order $r(n-1)$ defined as

$$\begin{aligned}
A_{l-1} &= tri\left[-E, \frac{1}{2}\frac{h^2}{\alpha^2}E' - 4E, -E\right]_{n-1}, \\
A_l &= tri\left[\frac{1}{2}\frac{h^2}{\alpha^2}E' - 4E, 4\frac{h^2}{\alpha^2}E' + 20E, \frac{1}{2}\frac{h^2}{\alpha^2}E' - 4E\right]_{n-1}, \\
A_{l+1} &= tri\left[-E, \frac{1}{2}\frac{h^2}{\alpha^2}E' - 4E, -E\right]_{n-1}.
\end{aligned}$$

E and E' are nonsymmetric dense matrices of order r. The vector X represents the $r\tilde{n}$ unknowns $a_{i,j,1}, a_{i,j,2}, \ldots, a_{i,j,r}$, and S is the right-hand-side containing values of the solution and its time derivative on the boundary of the domain.

The bandwidth of the matrix A is equal to $(2n+1)r$ and it has $r^2(3n-5)^2$ nonzero entries. After the $a_{i,j,k}$ are determined, it is easy to compute the solution at any point t in the interval $[t_0, t_{r-1}]$. More detailed information on this approach can be seen in [11, 12].

Method 2: Differential Quadrature (ICM-DQ)
In the differential quadrature method, the value of the time derivative of the solution at each collocation point is expressed as weighted linear sums of the function values at all the sampling points, i.e.,

$$U'(t_k) = \sum_{l=0}^{r-1} w_{k,l} U(t_l), \qquad k = 0, \ldots, r-1. \tag{6}$$

In particular, for any spatial grid point (x_i, y_j),

$$U'_{i,j}(t_k) = \sum_{l=0}^{r-1} w_{k,l} U_{i,j}(t_l).$$

The weighting coefficients $w_{k,l}$ $(k, l = 0, \ldots, r-1)$ can be determined such that Eq. 6 is satisfied exactly for r linearly independent test polynomials (such as Lagrange, Legendre,

Chebyshev, Lobatto polynomials). The $w_{k,l}$ are obtained solving a linear system of equations [15, 17].

With the knowledge of the weights, we can substitute equation (6) into the system (3) to obtain for a typical interior grid point at time level t_k:

$$\begin{aligned} & \sum_{l=1}^{r-1} \nu_{k,l}(U^l_{i+1,j} + U^l_{i,j+1} + U^l_{i-1,j} + U^l_{i,j-1}) - \sum_{l=1}^{r-1} \xi_{k,l} U^l_{i,j} \\ & +2\rho(U^k_{i+1,j+1} + U^k_{i-1,j+1} + U^k_{i-1,j-1} + U^k_{i+1,j-1}) \\ = \; & w_{k,0}(U^0_{i+1,j} + U^0_{i,j+1} + U^0_{i-1,j} + U^0_{i,j-1} + 8U^0_{i,j}), \end{aligned} \tag{7}$$

where $\rho = \alpha^2/h^2$, and

$$\nu_{k,l} = \begin{cases} 8\rho - w_{k,k}, & \text{if } k = l \\ -w_{k,l}, & \text{if } k \neq l \end{cases}, \qquad \xi_{k,l} = \begin{cases} 40\rho + 8w_{k,k}, & \text{if } k = l \\ 8w_{k,l}, & \text{if } k \neq l \end{cases}.$$

Eq. 7 leads to the following linear system of equations

$$BY = R, \tag{8}$$

where B is a block-tridiagonal matrix given by

$$B = tri\,[B_{l-1}, B_l, B_{l+1}]_{n-1}.$$

B_{l-1}, B_l and B_{l+1} are square matrices of order $(r-1)(n-1)$ defined as

$$\begin{aligned} B_{l-1} &= tri[G, F, G]_{n-1}, \\ B_l &= tri[F, D, F]_{n-1}, \\ B_{l+1} &= tri[G, F, G]_{n-1}. \end{aligned}$$

Here D, F and G are square matrices of order $r-1$ which entries are given by $D_{k,l} = -\xi_{k,l}$, $F_{k,l} = \nu_{k,l}$ and $G_{k,l} = 2\rho\delta_{k,l}$ (where $\delta_{k,l} = 1$ if $k = l$ and 0 otherwise). Y is the unknown vector representing the $(r-1)\tilde{n}$ values $U^k_{i,j}$ $(1 \leq i,j \leq n-1$ and $1 \leq k \leq r-1)$ at the collocation point. The matrix has $2n(r-1)$ as bandwidth and $(r-1)((5r-1)(n-3)^2 + (16r-8)(n-3) + 12r - 8)$ nonzero entries.

It is important to note that matrices A and B in (5) and (8) respectively, have the same block structure but B has a smaller order and fewer nonzero entries. Though both methods will give approximated solutions at the same collocation points, ICM-DQ requires less computations because Eq. 8 has fewer unknowns compared to Eq. 5.

Remark 1 *ICM-EP requires the explicit knowledge of the time derivative at the boundary grid points. It is not the case for ICM-DQ where we can use the quadrature weights to evaluate the time derivative at any grid point even on the boundary.*

The above collocation schemes are equivalent to certain Runge-Kutta methods [2, 4, 18, 19]. The two cases are A-stable methods [1, 16] and their accuracies are related to their growth functions [1, 5, 8, 9]. For ICM-EP, the accuracy in time is r regardless of the choice of the t_k. In ICM-DQ, we deal with polynomials of degree r. The accuracy of the derivatives in the differential quadrature is r. Then the overall accuracy of the method is r independent of the selection of the t_k and the test polynomials.

Lemma 2 *In the implicit collocation method for the heat equation described here, if the approximated solution is obtained at r consecutive time levels, the resulting method is fourth order in space and (at least) of order r in time.*

One of the advantages of ICM-EP is that we do not obtain the approximated solution at the levels t_k, $k = 0, \ldots, r-1$ only but also at any time t in the interval $[t_0, t_{r-1}]$. This is possible because of the explicit computation of the coefficients of the polynomials. For ICM-DQ, the solution is directly obtained at the collocation points t_k only. However, we can generate Lagrange polynomials with t_k as node points to find the solution at any points in $[t_0, t_{r-1}]$ too.

A question that may arise is whether r can be arbitrary large in order to obtain high accurate solution. For ICM-EP, Jézéquel numerically showed the existence of an optimal degree r beyond which the accuracy of the approximated solution will not improve because of round-off error propagation [10]. This optimal r depends on the number a spatial grid points n. The method of differential quadrature, if the collocation points are properly chosen, offers more "stable" polynomials. Because of the stability of its polynomials, ICM-DQ may require larger optimal degree r.

Remark 2 *There are other collocation approaches to discretize Eq. 1. We can mention the spectral method where spatial derivatives are discretized using differential quadratures and the time derivative by finite differences [6]. Here, the computation of spatial derivatives is global in the sense that all the nodes (in a given direction) are applied in the calculation. Compared to ICM, we may gain in accuracy but at the expense of memory requirement (dense matrix and large bandwidth), an increase of computational cost, and an additional computational complexity (due to the unstructure shape of the matrix). In addition, this method may introduce a stability requirement associated with the time step whereas for ICM there is no such requirement.*

3 Implementation

In this section, we want to describe how ICM with explicit polynomials (ICM-EP) and ICM with differential quadrature (ICM-DQ) are implemented. In fact, the implementation stategy for both methods is based on one algorithm presented in [12].

Given the number n for spatial grid points, an arbitrary time step Δt and the parameter r, we want to determine the approximated solution in the time interval $[0, T]$. We carry out η consecutive iterations of ICM with the assumption that $T = \eta(r-1)\Delta t$.

Let us focus on the first iteration of ICM algorithm i.e. the solution is to be found in the interval $[0, (r-1)\Delta t]$. The results at the end of $[0, (r-1)\Delta t]$ is used as initial conditions for the next time interval and the length of the interval for each subsequent iteration is still $(r-1)\Delta t$. Both end points of the interval are in the set of the collocation points. For ICM-EP, the r collocation points can be arbitrarily chosen in $[0, (r-1)\Delta t]$. The spatial and time discretizations give rise to a linear system of equations, Eq. 5, which solution is the coefficients of the polynomials. The approximated solution is then evaluated at any point in the interval $[0, (r-1)\Delta t]$.

In ICM-DQ, the collocation points are either taken arbitrary or are the r zeros (mapped into the interval $[0, (r-1)\Delta t]$) of a known polynomial of degree r. After the weights $w_{k,l}$

are determined, we also solve a linear system of equations, Eq. 8, which directly gives the approximated solution at the collocation points.

For the computation of the weighting coefficients $w_{k,l}$, we assume that the test polynomials (of degree at most $r-1$) under consideration, are defined in the interval $[-1,1]$ and therefore the r zeros belong to the same interval. We first calculate the weights by focusing on the interval $[-1,1]$ and the corresponding weights $w_{k,l}$ in $[0,(r-1)\Delta t]$ are just the product of the previous ones by a unique constant that is function of the the smallest and largest zeros in $[-1,1]$ and the two end points in $[0,(r-1)\Delta t]$. These weights are obtained by solving r linear system of equations with r unknowns each. All the systems have the same underlying matrix and the only change is the right-hand-side.

Remark 3 *There are more explicit formulas to determine the weighting coefficients [6]. They depend on the choice of the polynomials. For the sake of using a unique algorithm to find the weights for all the cases presented here, we use linear systems of equations. To avoid any ill conditioned problems related to those systems, we only consider small values of r (less than 10). Our numerical experiments suggest that the use of larger values of r do not improve the overall accuracy of ICM.*

The implementations of ICM-EP and ICM-DQ are presented in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1

1. Define the matrix A
2. Decompose the matrix A to obtain the matrix $\tilde{A} \approx A^{-1}$
3. For $l = 1 \rightarrow \eta$, do:
4. Define the right hand side S
5. Determine the coefficients $a_{i,j,k}$: $\tilde{A}AX = \tilde{A}S$
6. Find $U_{i,j}(t_k)$ at $(r-1)$ collocation points by computing $P_{i,j}(t_k)$
7. End do

Algorithm 2

1. Compute the weights $w_{k,l}$
2. Define the matrix B
3. Decompose the matrix B to obtain the matrix $\tilde{B} \approx B^{-1}$
4. For $l = 1 \rightarrow \eta$, do:
5. Define the right hand side R
6. Find $U_{i,j}(t_k)$ at $(r-1)$ collocation points: $\tilde{B}BY = \tilde{B}R$
7. End do

In ICM-EP, the order of the matrix is $r\tilde{n}$ whereas for ICM-DQ it is $(r-1)\tilde{n}$. The systems are solved by first preconditioning the matrices and then using the general minimal residual technique as the iterative accelerator [20]. There is no major difference between Algorithm 1 and Algorithm 2. Algorithm 2 differs from Algorithm 1 in two basic points: 1- It ends when the solution of the linear system is found (Step 6) and, 2- In its initialization phase we need to evaluate the weights $w_{k,l}$ (Step 1) which computational cost is negligible with respect to the one of the entire algorithm.

As n increases, most of the time in the two algorithms is devoted to finding the solution of the linear systems. Since ICM-DQ has fewer unknowns and non-zero entries in its matrix-equation, it uses less memory and will require less CPU time.

4 Numerical Experiments

In this section, we compare ICM-EP and ICM-DQ when they are both implemented on a shared memory vector computer, namely the CRAY SV1 (24 processors each running at 300 Mhz and 8 GB memory). The programs were coded in Fortran 90 programming language in double precision with 64-bit arithmetic. The parallel implementation of the two algorithms was achieved by introducing OpenMP directives in the code.

For all our simulations, we consider Eq. 1 with the exact solution given by

$$u(x,y,t) = e^{-\frac{t}{2}} \cos \frac{\pi}{2}(x+y) + e^{-2t} \sin \pi(x-y).$$

and $\alpha = 1/\pi$. Initial conditions and boundary conditions at any given time are taken from the above exact solution.

In all our numerical experiments, we assume that the target T where we want to get the approximated solution is equal to 10.0. The two algorithms are iterated until T is reached. We focus on the accuracy of the two methods and the computing time they require.

We first determine the elapsed times obtained with the two methods. It is important to note that for a given set the parameters (n and r), the computational cost of ICM-DQ remains the same regardless of the choice of the collocation points and the test polynomials. We consider $r = 4$, $n = 16$, and $\Delta t = 10^{-2}$ and record in Table 1 the computing times (in seconds) as function of the number of processors (#CPUs). We note that in both cases, as the number of processors increases (from 1 to 8), the computing time decreases. For a given number of processors, ICM-DQ requires the smallest time. This result was predicted in the previous section. In addition, ICM-DQ displays a slightly better speedup. It is worth mentioning that the Cray SV1 is not a dedicated computer, i.e., the requested processors are not reserved to a particular user during his run. The elapsed time measures the interval of time between the beginning of the run and its end. It includes any possible idle time.

#CPUs	ICM-EP	ICM-DQ
1	236.8	145.9
2	102.2	62.41
4	69.84	41.48
8	56.51	33.61

Table 1: Elapsed time (in seconds) as function of the number of processors when $n = 16$ and $r = 4$.

Now we want to test the accuracy of ICM-EP and ICM-DQ. We initially compute the maximum error (at $T = 10.0$) obtained with both methods when $\Delta t = 10^{-2}$, $r = 4$ and n varies. For ICM-DQ, we use as test functions regular polynomials (the set $\{1, t, \ldots t^{r-1}\}$),

the Lobatto, Legendre and Chebyshev polynomials. The results are presented in Table 2. We observe that in all the cases, the maximum errors decrease by a factor of 16 when n is doubled. Thus ICM solutions demonstrate fourth-order convergence in space and all of the methods offer comparable accuracies.

	ICM-EP	ICM-DQ			
n		Regular	Lobatto	Legendre	Chebyshev
4	4.13(-08)	4.17(-08)	4.18(-08)	4.07(-08)	4.11(-08)
8	2.74(-09)	2.61(-09)	2.62(-09)	2.55(-09)	2.58(-09)
16	1.62(-10)	1.54(-10)	1.58(-10)	1.53(-10)	1.54(-10)
32	1.00(-12)	1.24(-12)	4.92(-12)	3.51(-12)	4.59(-12)

Table 2: Maximum error obtained with ICM-EP and ICM-DQ when $\Delta t = 10^{-2}$, $r = 4$ and n varies.

We now study how the algorithms behave when n and r are fixed and Δt varies. It is important to mention that Δt does not necessarily measure the time step of each method. The actual time step Δt_a for a given method is obtained by $\Delta t_a = max(t_{k+1} - t_k)$ where the t_k belong to the time interval under consideration, say $[0, (r-1)\Delta t]$. For ICM-EP and ICM-DQ with regular test polynomials, we took $\Delta t_a = \Delta t$. For the remaining cases, Δt_a is larger than Δt and ICM-DQ with Lobatto polynomials has the largest value.

	ICM-EP	ICM-DQ			
Δt		Regular	Lobatto	Legendre	Chebyshev
16×10^{-2}	7.75(-10)	1.37(-08)	6.44(-09)	9.39(-09)	8.13(-09)
8×10^{-2}	1.74(-10)	5.39(-09)	3.75(-09)	2.22(-09)	1.93(-09)
4×10^{-2}	1.63(-10)	4.86(-10)	2.61(-10)	3.74(-10)	3.34(-10)
2×10^{-2}	1.61(-10)	9.05(-11)	1.21(-10)	1.06(-10)	1.11(-10)
1×10^{-2}	1.62(-10)	1.54(-10)	1.58(-10)	1.53(-10)	1.54(-10)
5×10^{-3}	1.61(-10)	1.60(-10)	1.61(-10)	1.60(-10)	1.61(-10)

Table 3: For $n = 16$ and $r = 4$, maximum error as function of Δt obtained with ICM-EP and ICM-DQ.

In Table 3, we record the maximum error as function of Δt when $n = 16$ and $r = 4$. As Δt decreases, the errors seem to level off to unique asymptotic value. ICM-DQ requires smaller Δt to reach that asymptotic value. When Δt is crude, ICM-DQ (as ICM-EP) needs few iterations to get to the target time $T = 10.0$ but needs more to adjust. We observe that if we allow the iterations to go well beyond $T = 10.0$, ICM-EP and ICM-DQ will give similar accuracies independent of Δt and r.

For the problem studied in this paper, the use of non-uniform time grids (such as zeros of Lobatto, Legendre, Chebyshev) does neither deteriorate nor enhance the accuracy of the quadrature solutions compared to the use of equally spaced sampling grid points. In other

types of problems; the utilization of non-uniform grids has resulted in better accuracies [3].

5 Conclusions

We compared two implicit methods to numerically solve the two dimensional heat equation. The two methods differ by the way that the time derivative is computed. One uses an explicit computation of coefficients of polynomials to approximate the time component and the other relies on differential quadratures. In the two cases, the spatial derivatives are discretized using a fourth-order finite difference scheme. Our numerical results obtained on the Cray SV1 showed that both methods produce high accurate solutions and that the one based on differential quadrature requires less memory and the smallest elapsed time.

Acknowledgment: The authors are grateful to the NASA Center for Computational Science, located at NASA Goddard Space Flight Center, for providing the computational resource to perform this work.

References

[1] U. Ascher and R. Weiss, Collocation for singular pertubation problems I: first order systems with constant coefficients, *SIAM J. Numer. Anal.*, **10**(3), p. 537-557 (1983).

[2] O. Axelsson, A class of A-stable methods, *BIT*, **9**, p. 185-199 (1969).

[3] C.W. Bert and M. Malik, Differential quadrature method in computational mechanics: a review, *Apllied Mechanics Reviews*, **49**(1), p. 1-28 (1996).

[4] J. C. Butcher, Implicit Runge-Kutta processes, *Math. Comp.*, **18**, p. 50-64 (1964).

[5] J. Douglas Jr. and T. Dupont, *Collocation Methods for Parabolic Equations in a Single Space Variable*, Lecture Notes in Mathematics, Vol. 385, Springer, Berlin, 1974.

[6] D. Funaro, *Polynomial Approximation of Differential Equations*, Lecture Notes in Physics, m8, Springer-Verlag, Berlin, 1992.

[7] M.M. Gupta, A fourth-order Poisson solver, *J. Comp. Phys.*, **55**, p. 166-172 (1984).

[8] E. Hairer, S.P. Norsett and G. Wanner, *Solving Ordinary Differential Equations I: Non-stiff Problems*, Springer Series in Computational Mathematics, Vol. 8, Springer, Berlin, 1987.

[9] B.L. Hulme, One-step piecewise polynomial Galerkin methods for initial values problems, *Math. Comp.*, **26**, p. 415-426 (1972).

[10] F. Jézéquel, A validated parallel across time and space solution of the heat transfer equation, *Applied Numerical Mathematics*, **31**, p. 65-79 (1999).

[11] J. Kouatchou, Finite differences and collocation methods for the solution of the two dimensional heat equation, *Num. Meth. for PDEs*, **17**, p. 54-63 (2001).

[12] J. Kouatchou, Parallel implementation of a high order implicit collocation method for the heat equation, *Math. Comp. in Simulation*, **54**, p. 509-519 (2001).

[13] M. Malik and F. Civan, Comparative study of differential quadrature and cubature method vis-a-vis some conventional techniques in the context of convection-diffusion-reaction problems, *Chemical Engineering Science*, **50**(3), p. 531-547 (1995).

[14] D.A. Peters and A.P. Izadpanah, hp-version finite element for space-time domain, *Computational Mechanics*, **3**, p. 73-88 (1988).

[15] J.R. Quan and C.T. Chang, New insights in solving distributed system equations by the quadrature method I. Analysis, *Computers and Chemical Engineering*, **13**, p. 779-788 (1989).

[16] E. B. Saff and R. S. Varga, On the zeros and poles of Padé approximants to e^z, *Numer. Math.*, **25**, p. 1-14 (1975).

[17] C. Shu and B.E. Richards, Application of generalized differential quadrature to solve two-dimensional incompressible Navier Stokes equations, *Int. J. for Num. Meth. in Fluids*, **15**(7), p. 791-798 (1992).

[18] R. Weiss, The application of implicit Runge-Kutta and collocation methods to boundary-value problems, *Math. Comp.*, **28**, p. 449-464 (1974).

[19] K. Wright, Some relationships between implicit Runge-Kutta, collocation and Lanczos τ methods, and their stability properties, *BIT*, **10**, p. 217-227 (1970).

[20] J. Zhang, A sparse approximate inverse preconditioner for parallel preconditioning of general sparse matrices, *Applied Mathematics and Computation*, to appear.