1977

Comparison of Fast Direct Methods for Elliptic Problems

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

T. S. Papatheordorou

Report Number:
77-218

COMPARISON OF FAST DIRECT METHODS FOR ELLIPTIC PROBLEMS

by

E. N. Houstis
Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

T. S. Papatheodorou
Department of Mathematics
Calrkson College of Technology
Potsdam, NY 13676

CSD-TR 218

January 1977

Comparison of Fast Direct Methods for Elliptic Problems

E. N. Houstis* and T. S. Papatheodorou**

Abstract. We compare and evaluate some efficient algorithms to solve linear elliptic partial differential equations with constant coefficients and Dirichlet boundary conditions. A new fourth order Fourier Series method is presented and comparisons have been made with some well known fast direct methods which indicate the superiority of the new method.

1. INTRODUCTION

Our goal in this paper is to confirm experimentally our belief that the superior methods are higher order. In [6], [7] Houstis et al outline a framework in which to compare computational methods for elliptic partial differential equations. The basic model for selection and evaluation of algorithms is described in Section 2 and it is used here.

In Section 3 we present the problem subset. The fourth section describes the algorithms to be compared and evaluated. Finally, in Section 5 we present the results of some numerical experiments and an asymptotic operation count that strongly support our belief.

2. METHODOLOGY FOR COMPARING METHODS

The basic model for selection and evaluation of algorithms to be used in this comparison consists of a problem space, a subset of algorithms from which the selection is to be made and measures of the performance of algorithms. This framework of comparing methods for solving elliptic partial differential

---

* Department of Computer Science, Purdue University, West Lafayette, IN 47907
Work supported in part by NSF grant MCS 76-10225.
** Department of Mathematics, Clarkson College of Technology, Potsdam, NY 13676

equations was employed by Houstis et al [6], [7] to compare finite element and finite difference methods.

An arbitrary element of the problem space to consider is described as follows. Let u satisfy the equation

$$\alpha\, u_{xx} + \beta\, u_{yy} + \lambda u = f \quad \text{on} \quad \Omega = [0,1] \times [0,1], \tag{2.1}$$

subject to Dirichlet boundary conditions

$$u = g \quad \text{on} \quad \partial\Omega \tag{2.2}$$

with α, β and λ constants. Estimate u to a given tolerance ϵ.

The subset of algorithms consists of seven specific methods.

As measurements of performance of an algorithm we use execution time of the components of each algorithm.

3. THE PROBLEM SUBSET

We consider eight equations, previously used by Houstis et al [7] to compare finite difference and finite element methods, which are tabulated in Table I. All equations are defined on unit square. The function f is determined by applying symbolically the differential operator to the true solution.

The solution $\phi(x)\phi(y)$ of problem 6 has a steep slope along a right angle at the center of the domain and

$$\phi(x) = U(.35) + (U(.35) - U(.65))P(x)$$

where P(x) is a quintic polynomial determined so that $\phi(x)$ has two continuous derivatives and U(x) is unit step function.

Table I. The problem subset

No.	Partial Differential Equation Operator True Solution	Boundary Conditions
1.	$u_{xx} + u_{yy} = -6xye^x e^y\ (xy + x + y - 3)$ $u = 3e^x e^y (x - x^2)(y - y^2)$	$u = 0$
2.	$u_{xx} + u_{yy} = f$ $u = x^{5/2}y^{5/2} - xy^{5/2} - x^{5/2}y + xy,$	$u = 0$
3.	$4u_{xx} + u_{yy} - 64\ u = f$ $u = 4(x^2-x)(\cos(2\pi y) - 1),$	$u = 0$
4/5	$u_{xx} + u_{yy} - 100u = (\mu^2-100)\cosh y/\cosh \mu$ with μ=10 or 20 $u = \cosh 10x/\cosh 10 + \cosh \mu y/\cosh \mu,$	$u = g$
6.	$u_{xx} + u_{yy} = f$ $u = \phi(x) * \phi(y)$, see text,	$u = g$
7.	$u_{xx} + u_{yy} = f$ $u = y[(x-2)^2+y^2-1]e^{-.0625x(x-4)(y-2)}/[(3+(x-2)^2)(3+y^2)],$	$u = g$
8.	$u_{xx} + u_{yy} = f$ $u = 10\ \phi(x)*\phi(y),\ \phi(x)=e^{-100(x-.5)^2}(x^2-x),$	$u = 0$

4. THE ALGORITHM SUBSET

4.1. Nine point approximation to the Helmholtz equation

We introduce a fourth order nine-point difference scheme for the discretization of the equation (2.1). Let Δx, Δy be the mesh length on x,y - direction of uniform grid placed over Ω and define the variables

$$p = \Delta y/\Delta x,\ \gamma = (\Delta x)^2/12,\ q = \frac{\beta}{\alpha}\frac{1-\lambda(\Delta y)^2/12\beta}{1-\lambda\gamma/\alpha},$$

$$b = 12\,\frac{\beta}{\alpha}\frac{1}{p^2+q} - 2, \qquad a = \frac{12p^2q\alpha/\beta}{p^2+q} - 2,$$

$$c = (a+2)(1-\lambda\gamma/\alpha)\lambda(\Delta x)^2/\alpha - 2(a+b) - 4,$$

$$d = q/\beta \text{ and } e = 12\,(1-\lambda\gamma/\alpha)q/\lambda - 2d - 2$$

then a nine-point fourth order finite difference approximation to (2.1) is

$$\begin{array}{|ccc|} \hline 1 & b & 1 \\ \hline a & c & a \\ \hline 1 & b & 1 \\ \hline \end{array} \qquad u = \frac{p^2\Delta x^2}{p^2+q} \qquad \begin{array}{|ccc|} \hline 0 & 1 & 0 \\ \hline d & e & d \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad f$$

4.2 The Tensor Product Methods (TP)

A metalgorithm that represents the tensor product methods consists of a set of 4 components. A specific choice for the components is indicated.

(i) a grid of points over the domain Ω: uniform grid $N = 2^k$

(ii) a processor that generates a set of algebraic equations from the operator equations: 5-point and the 9-point finite difference approximation

(iii) Equation solver;

(α) A similarity transformation is applied in both directions to reduce system (ii) to a diagonal system.

(β) Explicit computation of the solution

(iv) Measures of performance: execution time.

For detailed information on the component (iii) see Lynch et al [8], [9].

4.3 The Tensor Product - Fast Fourier Transform Methods (TP-FFT)

A metalgorithm that represents the Tensor-FFT method consists of 4 components.

Our choice for the components is indicated.

(i) Grid: uniform grid, $N = 2^k$

(ii) Discretization processor: 5-point and the 9-point finite difference approximation, and Rietz-Galerkin with C^1 bicubic elements.

(iii) Equation solver:

(α) A sine and/or cosine transformation applied in one or both directions to reduce the system (ii) to a diagonal or block diagonal system. This step employs a fast sine and cosine transforms using standard FFT routines [5].

(β) A profile Gaussian elimination is used to solve the block diagonal system.

(vi) measurement of performance; execution time.

For more detailed information on the component (iii) in the case of finite element approximation see Bank [1].

4.4 The Fourier Series Methods (FFT)

Our new fourth order FFT method is described by the following metalgorithm.

(i) Grid: uniform grid, $N = 2^k$

(ii) Discretization processor: 9-point finite difference approximation

(iii) Equation solver:

(α) Odd/even reduction

(β) Fourier Analysis on even lines. This step is performed by a FFT routine [5]

(γ) Recursive Cyclic Reduction: Solve for harmonic amplitudes on the even lines

(δ) Use Fourier synthesis on the even lines. This step is performed by a FFT routine [5]

(ε) Solution on the odd lines

(iv) measures of performance: execution time.

For detailed information on the component (iii) in the case of 5-point difference approximation of Poisson equation see Hockney [5].

4.5 Cyclic Reduction Methods (NCAR)

A metalgorithm for this class of methods consists of four components. Our choice for the components is indicated

(i) Grid: uniform grid $N = 2^p 3^q 5^r$

(ii) Discretization processor: five-point finite difference approximation

(iii) Equation solver: Subroutines POIS in [10]. The system of equations is solved by the Buneman variant of cyclic reduction see [2,4]

(iv) measurement of performance: execution time.

For detailed informations on the implementation in FORTRAN and description of the above algorithm see [10].

5. Comparisons

5.1 Operations Counts

The asymptotic order of the total number of arithmetic operations (additions, multiplications, divisions) for each method is

Method:	TP	TP-FFT	FFT	NCAR
Operations	$8N^3$	$20N^2\log_2 N$	$5N^2\log_2 N$	$4.5N^2\log_2 N$

5.2 Numerical results

In this section, we present the results of some numerical experiments which confirm the contention that the fourth order fast direct method is uniformly superior to the second order methods compared. The rate of convergence is estimated by

$$\alpha_{m,n} = \frac{\ln \varepsilon_n / \varepsilon_m}{\ln n/m}$$

where ε_n is the estimated error for an nxn mesh.

TABLE 1 5-point difference approximation -- Problem 1

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	4.28E-03	.05	.05	.05	
16	1.09E-03	.31	.25	.26	2.0
32	2.72E-04	2.13	1.18	.90	2.0
64	6.81E-05	15.71	5.35	4.10	2.0
128	1.70E-05	120.07	24.04	18.69	2.0

TABLE 2 9-point difference approximation -- Problem 1

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	6.82E-05	.05	.06	.05	
16	4.27E-06	.33	.28	.19	4.0
32	2.68E-07	2.19	1.24	.76	4.0
64	1.68E-08	15.87	5.60	3.13	4.0
128	1.04E-09	121.08	24.45	13.03	4.0

TABLE 3 5-point difference approximation -- Problem 2

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	2.55E-04	.04	.05	.05	
16	7.08E-05	.29	.23	.27	1.9
32	1.90E-05	2.07	1.10	.90	1.9
64	4.97E-06	15.43	4.95	4.14	1.9
128	1.28E-06	119.70	22.28	18.72	2.0

TABLE 4 9-point difference approximation -- Problem 2

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	1.29E-04	.05	.06	.04	
16	2.53E-05	.31	.25	.17	2.4
32	4.80E-06	2.13	1.15	.67	2.4
64	8.80E-07	15.61	5.12	2.77	2.4
128	1.59E-07	119.62	22.76	11.59	2.5

TABLE 5 5-point difference approximation -- Problem 3

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	1.47E-02	.05	.05	.05	
16	3.67E-03	.30	.23	.19	2.0
32	9.17E-04	2.08	1.08	.87	2.0
64	2.29E-04	15.54	4.89	3.81	2.0
128	5.73E-05	126.00	21.57	17.70	2.0

TABLE 6 9-point difference approximation -- Problem 3

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	1.43E-03	.06	.06	.04	
16	9.27E-05	.32	.26	.17	3.9
32	5.85E-06	2.14	1.14	.69	4.0
64	3.66E-07	15.66	5.09	2.82	4.0
128	2.29E-08	126.50	22.21	11.53	4.0

TABLE 7 5-point difference approximation -- Problem 4

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	3.23E-02	.05	.05	.05	
16	9.08E-03	.27	.22	.26	1.8
32	2.35E-03	1.86	.99	.75	2.0
64	5.93E-04	13.97	4.32	3.32	2.0
128	1.49E-04	108.19	19.53	15.07	2.0

TABLE 8 9-point difference approximation -- Problem 4

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	1.61E-03	.05	.06	.04	
16	1.17E-04	.28	.23	.14	3.8
32	7.63E-06	1.89	1.00	.52	3.9
64	4.82E-07	14.12	4.43	2.06	4.0
128	3.02E-08	108.52	19.88	8.54	4.0

TABLE 9 5-point difference approximation -- Problem 5

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	1.08E-01	.06	.07	.06	
16	4.00E-02	.34	.27	.41	1.4
32	1.07E-02	2.20	1.19	1.22	1.9
64	2.74E-03	15.83	5.29	5.34	2.0
128	6.91E-04	121.02	23.54	22.95	2.0

TABLE 10 9-point difference approximation -- Problem 5

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	1.94E-02	.07	.08	.07	
16	1.81E-03	.36	.29	.22	3.4
32	1.20E-04	2.26	1.26	.84	3.9
64	7.63E-06	16.08	5.49	3.25	4.0
128	4.81E-07	121.50	24.18	13.00	4.0

TABLE 11 5-point difference approximation -- Problem 6

	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	1.90E-01	.07	.08	.08	
16	7.16E-02	.37	.32	.36	1.4
32	7.60E-03	2.28	1.40	1.15	3.2
64	4.09E-03	15.70	6.13	5.22	0.8
128	5.05E-04	114.77	27.07	22.41	3.0

TABLE 12 9-point difference approximation -- Problem 6

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	1.12E-01	.09	.09	.08	
16	5.17E-02	.41	.36	.27	1.1
32	4.09E-03	2.36	1.47	1.01	3.7
64	2.96E-03	15.90	6.30	4.07	0.5
128	5.70E-05	115.96	27.60	16.55	5.7

TABLE 13 5-point difference approximation -- Problem 7

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	4.97E-02	.11	.11	.11	
16	1.27E-02	.58	.52	.48	2.0
32	3.23E-03	3.29	2.26	2.01	2.0
64	8.09E-04	20.47	9.72	8.67	2.0
128	2.02E-04	138.85	41.30	36.95	2.0

TABLE 14 9-point difference approximation -- Problem 7

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	5.03E-03	.15	.15	.15	
16	2.88E-04	.67	.60	.53	4.1
32	1.77E-05	3.50	2.44	2.05	4.0
64	1.10E-06	21.03	10.11	8.05	4.0
128	6.89E-08	138.90	42.43	32.12	4.0

TABLE 15 5-point difference approximation -- Problem 8

N	Maximum Error	Execution Time (sec) TP	TP-FFT	NCAR	Convergence Rate
8	6.15E-01	.11	.11	.11	
16	7.27E-02	.57	.51	.62	3.1
32	1.64E-02	3.29	2.25	2.07	2.1
64	4.00E-03	20.52	9.78	8.93	2.0
128	9.23E-04	140.27	42.18	37.80	2.1

TABLE 16 9-point difference approximation -- Problem 8

N	Maximum Error	Execution Time (sec) TP	TP-FFT	FFT	Convergence Rate
8	3.55E-01	.15	.15	.14	
16	7.53E-03	.67	.60	.52	5.6
32	3.81E-04	3.48	2.46	2.03	4.3
64	2.27E-05	20.99	10.27	8.07	4.1
128	1.40E-06	140.65	43.31	32.46	4.0

TABLE 17 Data comparing 5-point cyclic reduction (NCAR) with the FFT 9-point for the same accuracy

	NCAR	5-point		FFT	9-point	
Pr. No.	N	Accuracy	Ex. Time	N	Accuracy	Ex. Time
1	128	1.70E-05	18.69	16	4.27E-06	.19
2	128	1.28E-06	18.72	64	8.80E-07	2.77
3	128	5.73E-05	17.70	32	5.85E-06	.69
4	128	1.49E-04	15.07	16	1.17E-04	.14
5	128	6.91E-04	22.95	32	1.20E-04	.84
6	128	5.05E-04	22.41	128	5.70E-05	16.55
7	128	2.02E-04	36.95	16	2.88E-04	.53
8	128	9.23E-04	37.80	32	3.81E-04	2.03

The results presented in Tables 1-16 confirm that 9-point difference method is of fourth order. The solution of problem 2 has singularity in the third derivative which explains the rate (2.4) of convergence computed. The rate of convergence for problem 6 is difficult to predict since the solution has a wave front and that shows in the rate estimates computed. All computations were performed on a CDC 6500 in single precision arithmetic.

Finally, in Table 17 we present data comparing the best from 5-point fast direct methods against the 9-point Fast Fourier method for the same or better accuracy. The data indicate that FFT 9-point method is on the average 51 times faster except for problem 6.

REFERENCES

1. R. E. Bank [1976], Efficient Algorithms for Solving Tensor Product Finite Element Equations, to appear.

2. J. Barkley Rosser [1975], Nine-point difference solutions for Poisson's equations, Comp. & Maths. with Appls. 1, pp. 351-360.

3. B. L. Buzbee, G. H. Golub, and C. W. Nielson [1971], On direct methods for solving Poisson's equations. SIAM J. Numer. Anal. 7, 627-656.

4. F. W. Dorr [1970], The direct solution of the discrete Poisson equation on a rectangle. SIAM Review 12, 248-263.

5. R. W. Hockney [1970], The potential calculation and some applications. Meth. in Comput. Phys. 9, pp. 135-211.

6. E. N. Houstis, R. E. Lynch, T. S. Papatheodorou and J. R. Rice [1975], Development, Evaluation and Selection of Methods for Elliptic Partial Differential Equations, Ann. Assoc. Inter. Calcul. Analog., 11, pp. 98-103.

7. E. N. Houstis, R. E. Lynch, T. S. Papatheodorou and J. R. Rice [1976], Evaluation of Numerical Methods for Elliptic Partial Differential Equations, to appear in the Journal of Computational Physics.

8. R. E. Lynch, J. R. Rice and D. H. Thomas [1964], Tensor product analysis of partial difference equations, Bull. Amer. Math. Soc. 70, pp. 378-384.

9. R. E. Lynch, J. R. Rice and D. H. Thomas [1964], Direct solution of partial difference equations by tensor product methods, Numer. Math., 6, pp. 185-199.

10. P. Swarztrauber and R. Sweet [1975], Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations, NCAR-TN/IA-109.