Implementation Details: The Parallel Finite Element Method

David O'Neal and *Raghurama Reddy*, Pittsburgh Supercomputing Center, Carnegie Mellon University, Pittsburgh, PA 15213

ABSTRACT: *A remarkable new algorithm for solving problems arising from finite element discretizations has been in development at the University of Pittsburgh for about three years now. There are few papers on the method, but an upcoming article in Numerische Mathematik may change that.*

The Parallel Finite Element Method (PFEM) was first described in 1992 by Bill Layton and Patrick Rabier of the Mathematics Department at Pittsburgh. Since then, the basic algorithm has been implemented on a variety of platforms, demonstrating exceptional scalability and performance in every case.

A review of the essential features of the method will be presented in order to provide a context for discussion of data structures and implementation details. Excerpts from CRAFT, shared memory, and manually autotasked Fortran 90 sources will be used to illustrate the similarities and differences between the designs. We will conclude with a summary of our findings and a description of the ongoing work at Pittsburgh.

All known papers that incorporate aspects of the Layton and Rabier method are referenced within, hence the attached biblography represents a current survey of published material on this subject.

Keywords: finite element method, operator splitting, graph coloring, data parallel algorithms.

Overview

- Typical Finite Element Algorithms (FEM)
- Parallel Finite Element Algorithm (PFEM)
- Data Structures
- Implementation Details

f90
CRAFT
shared memory

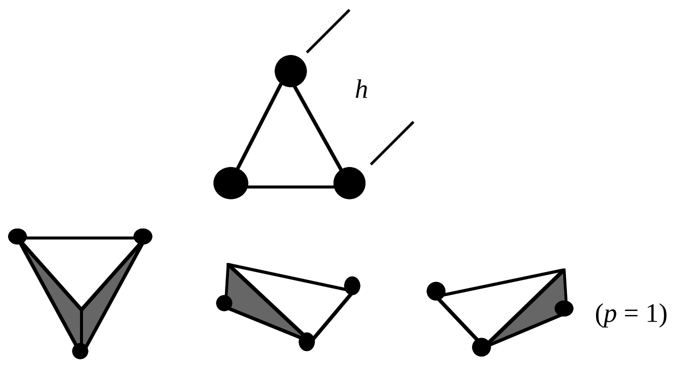
- Computational Experiments

PFEM
banded solver
conjugate gradient solvers

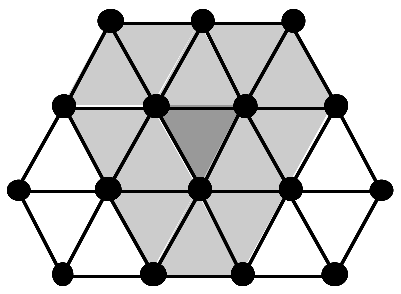
- Observations

Typical FEM Algorithms

- Conforming elements



- Mesh

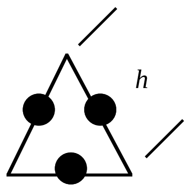


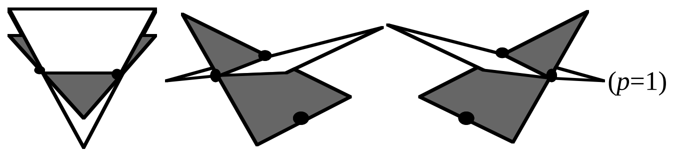
- Ritz-Galerkin methods

Ritz' method is based on the fundamental principle of minimum potential energy. The Galerkin method corresponds to the principle of virtual work.

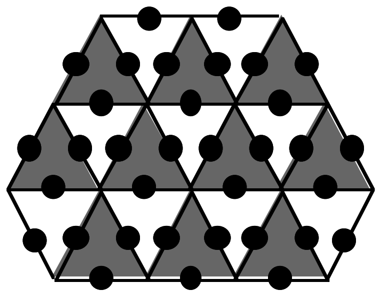
PFEM Algorithm

- Non-conforming elements





- Mesh with coloring scheme



- Peaceman-Rachford operator splitting

Basic theorem states that if an operator on a space can be written as A=A1+A2 where each Ai corresponds to a splitting associated with separation of variables, then a robust iterative method for solving the equation $A\zeta = b$ exists.

- Parameter

$$\rho_{optimal} \approx \sqrt{|\lambda_{\max}||\lambda_{\min}|}$$

A concise statement of the Peaceman-Rachford iterative method taken from Layton and Rabier (1995) reads as follows. Given a linear system

(1.1)$A\xi = b$

where $b \in R^N$ and $A \in R^{NxN}$ is an invertible matrix, the Peaceman-Rachford procedure for the solution of (1.1) consists of splitting A into the form

(1.2)$A = A_1 + A_2$

and, given $\xi_0 \in R^N$, calculating the sequence $(\xi_n, \zeta_n) \in R^N \times R^N$ defined inductively by

(1.3)$(\rho I + A_1)\xi_n \ = (\rho I - A2)\zeta_n + b,$

$(\rho I + A_2)\zeta_{n+1} = (\rho I - A_1)\xi_n + b$

Note that for the special case

$A \cong \partial^2/\partial x_1^2 + \partial^2/\partial x_2^2, A_1 \cong \partial^2/\partial x_1^2, A_2 \cong \partial^2/\partial x_2^2$, then (1.3) is better known as the "alternating direction implicit" (ADI) method.

The PFEM program is a research-level parallel finite element code suitable for solving Dirichlet-Poisson boundary value problems on a rectangular domain W in two dimensions:

$$\text{(2.1)} \quad \begin{cases} -\text{div}\,(a \text{ grad } u) + q\,u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

The solution surface is approximated by piecewise linear functions with inter-element discontinuities along edges.

The original version of this program was written in CM Fortran in July, 1992. A Fortran 90 port was produced in 1994 to support the Institute on Parallel Computing at PSC. In February, 1995, CRAFT and shared memory ports for the T3D were completed.

PFEM is based on material developed by William Layton and Patrick Rabier of the Department of Mathematics and Statistics at the University of Pittburgh.

Implementation Details - f90

f90 stack variable excerpts.

```
 integerxB2Rperm(LBLK,T), yB2Rperm(LBLK,T)

 real*8fBLK(LBLK,T), xBLK(LBLK,T), old_xBLK(LBLK,T),
     ^REDresid(0:LRED,T), BLKresid(0:LBLK,T),
     ^BLKmatrix(LBLK,T,T), norm_delta_xBLK
```

Compute the black residual.

```
!MIC$ DO ALL
!MIC$^SHARED (BLKmatrix, xBLK, BLKresid, fBLK, LBLK)
!MIC$^PRIVATE(k)
!MIC$^CHUNKSIZE(SIZE)

 do k=1,LBLK

            BLKresid(k,1) = fBLK(k,1)      - (BLKmatrix(k,1,1)*xBLK(k,1)
    ^                                      + BLKmatrix(k,1,2)*xBLK(k,2)
    ^                                      +  BLKmatrix(k,1,3)*xBLK(k,3))

            BLKresid(k,2) = BLK(k,2)       - (BLKmatrix(k,2,1)*xBLK(k,1)
    ^                                      + BLKmatrix(k,2,2)*xBLK(k,2)
    ^                                      + BLKmatrix(k,2,3)*xBLK(k,3))

            BLKresid(k,3) = fBLK(k,3)      - (BLKmatrix(k,3,1)*xBLK(k,1)
    ^                                      + BLKmatrix(k,3,2)*xBLK(k,2)
    ^                                      + BLKmatrix(k,3,3)*xBLK(k,3))
            end do
```

Map black residual into red ordering.

```
  do i=1,T

!MIC$ DO ALL
!MIC$^SHARED (REDresid, yB2Rperm, xB2Rperm, BLKresid, LBLK, i)
!MIC$^PRIVATE(k)
!MIC$^CHUNKSIZE(SIZE)

  do k=1,LBLK
    REDresid(yB2Rperm(k,i),xB2Rperm(k,i)) = BLKresid(k,i)
  end do
  end do
```

Compute global RMS-norm.

```
      norm = 0.0

!MIC$ PARALLEL
!MIC$^SHARED (perm, x, y, Lc, T, i, norm)
!MIC$^PRIVATE(k, local_norm)

      local_norm = 0.0
```

```
        do i=1,T
!MIC$ DO PARALLEL
!MIC$^CHUNKSIZE(SIZE)
        do k=1,Lc
          if (perm(k,i) .ne. 0) local_norm     = local_norm
     ^                                         + (x(k,i)-y(k,i))
     ^                                         * (x(k,i)-y(k,i))
        end do
        end do
!MIC$ GUARD
        norm = norm + local_norm
!MIC$ END GUARD
!MIC$ END DO
!MIC$ END PARALLEL

        norm = SQRT (norm/real(M))
```

Implementation Details - CRAFT

CRAFT stack variable excerpts.

```
CDIR$ GEOMETRY G1(:BLOCK,:)
CDIR$ GEOMETRY G2(:BLOCK,:,:)

  integer B2Rperm(LBLK,Tn), xB2Rperm(LBLK,Tn), yB2Rperm(LBLK,Tn)

  real*8norm_xBLK, norm_delta_xBLK,

     ^               fBLK(LBLK,Tn), xBLK(LBLK,Tn), old_xBLK(LBLK,Tn),
     ^               BLKresid(LBLK,Tn), REDresid(LRED,Tn),
     ^               BLKmatrix(LBLK,Tn,Tn)

CDIR$ SHARED norm_xBLK
CDIR$ SHARED (G1) :: B2Rperm, xB2Rperm, yB2Rperm
CDIR$ SHARED (G1) :: fBLK, xBLK, old_xBLK, REDresid, BLKresid
CDIR$ SHARED (G2) :: BLKmatrix
```

Compute black residual.

```
CDIR$ DOSHARED (k) on BLKresid(k,1)

  do k=1,LBLK

          BLKresid(k,1) = fBLK(k,1)     - (BLKmatrix(k,1,1)*xBLK(k,1)
     ^                                  +  BLKmatrix(k,2,1)*xBLK(k,2)
     ^                                  +  BLKmatrix(k,3,1)*xBLK(k,3))

          BLKresid(k,2) = fBLK(k,2)     - (BLKmatrix(k,1,2)*xBLK(k,1)
     ^                                  +  BLKmatrix(k,2,2)*xBLK(k,2)
     ^                                  +  BLKmatrix(k,3,2)*xBLK(k,3))

          BLKresid(k,3) = fBLK(k,3) - (BLKmatrix(k,1,3)*xBLK(k,1)
    ^                          +  BLKmatrix(k,2,3)*xBLK(k,2)
    ^                          +  BLKmatrix(k,3,3)*xBLK(k,3))
  end do

  Map black residual into red ordering.

 do i=1,T

CDIR$ DOSHARED (k) ON B2Rperm(k,1)

 do k=1,LBLK
   REDresid(yB2Rperm(k,i),xB2Rperm(k,i)) = BLKresid(k,i)
 end do
 end do

  Compute global RMS-norm.

 norm_delta_xBLK = 0.0

CDIR$ MASTER
 norm_xBLK = 0.0
CDIR$ END MASTER

CDIR$ DOSHARED (k) on xBLK(k,1)

 do k=1,LBLK
   norm_delta_xBLK = norm_delta_xBLK
    ^               + (xBLK(k,1)-old_xBLK(k,1))
    ^               * (xBLK(k,1)-old_xBLK(k,1))
    ^               + (xBLK(k,2)-old_xBLK(k,2))
    ^               * (xBLK(k,2)-old_xBLK(k,2))
    ^               + (xBLK(k,3)-old_xBLK(k,3))
    ^               * (xBLK(k,3)-old_xBLK(k,3))
 end do

CDIR$ CRITICAL
 norm_xBLK = norm_xBLK + norm_delta_xBLK
CDIR$ END CRITICAL

CDIR$ MASTER
 norm_xBLK = SQRT(norm_xBLK/real(M))
CDIR$ END MASTER
```

Implementation Details - shmem

SHMEM commons and stack variable excerpts.

```
      common / norm / local_norm_xRED, globl_norm_xRED,
    ^               local_norm_xBLK, globl_norm_xBLK

      real*8  local_norm_xRED(MSIZE), globl_norm_xRED,
    ^        local_norm_xBLK(MSIZE), globl_norm_xBLK,

      integer xB2Rperm(T,LBLK), ypeB2R(T,LBLK), yosB2R(T,LBLK)
      real*8  fBLK(T,LBLK), xBLK(T,LBLK), oldBLK(T,LBLK),
     ^         BLKmatrix(T,T,LBLK), BLKresid(T,0:N)
```

Compute black residual.

```
      do L1=1,LBLK,20
        L2 = MIN (L1+19,LBLK)
        call pref (15, xBLK(1,L1))
        call pref (45, BLKmatrix(1,1,L1))
          do k=L1,L2

            BLKresid(1,k) = fBLK(1,k)          - (BLKmatrix(1,1,k)*xBLK(1,k)
     ^                                         + BLKmatrix(1,2,k)*xBLK(2,k)
     ^                                         + BLKmatrix(1,3,k)*xBLK(3,k))
            BLKresid(2,k) = fBLK(2,k)          - (BLKmatrix(2,1,k)*xBLK(1,k)
     ^                                         + BLKmatrix(2,2,k)*xBLK(2,k)
     ^                                          + BLKmatrix(2,3,k)*xBLK(3,k))
            BLKresid(3,k) = fBLK(3,k)          - (BLKmatrix(3,1,k)*xBLK(1,k)
     ^                                         + BLKmatrix(3,2,k)*xBLK(2,k)
     ^                                         + BLKmatrix(3,3,k)*xBLK(3,k))
          end do
        end do
```

Map black residual into red ordering.

```
      do L1=1,LBLK,20
        L2 = MIN (L1+19,LBLK)
        call pref (15, xB2Rperm(1,L1), yosB2R(1,L1),
     ^                BLKresid(1,L1), ypeB2R(1,L1))
          do k=L1,L2
            do i=1,T

              if (ypeB2R(i,k) .eq. pe) then
                REDresid(xB2Rperm(i,k),yosB2R(i,k)) = BLKresid(i,k)
              else
                call SHMEM_PUT (REDresid(xB2Rperm(i,k),yosB2R(i,k)),
     ^                          BLKresid(i,k),      1, ypeB2R(i,k))
              endif

            end do
          end do
      end do

      call BARRIER( )
```

Compute global RMS-norm from the local vectors.

```
error_vec(1:T,1:Lc) = 0.0

do k=1,Lc
   do i=1,T
     if (perm(i,k) .ne. 0) error_vec(i,k) = (x(i,k)-y(i,k))**2
   end do
end do
```

```
local_norm(pe+1) = 0.0

do k=1,Lc
   do i=1,T
     local_norm(pe+1) = local_norm(pe+1) + error_vec(i,k)
   end do
end do

call SHMEM_PUT (local_norm(pe+1), local_norm(pe+1), 1, mgr)
call BARRIER( )

if (pe .eq. mgr) then

  globl_norm = local_norm(1)

  do i=2,npes
    globl_norm = globl_norm + local_norm(i)
  end do

  globl_norm = SQRT(globl_norm/real(M))

endif

call BARRIER( )
call SHMEM_GET (globl_norm, globl_norm, 1, mgr)
```

Computational Experiments

- Environment

The T3D at the Pittsburgh Supercomputing Center is currently configured with 512 PEs, each of which has 8MW of memory. Eight I/O gateways connect it to a Y-MP C90/16512 equipped with a 512MW SSD. Clock rates are 150 and 240 MHz respectively. The C90 runs Unicos 8.0 with AFS. The T3D is running the MAX 1.2.0.1 kernel.

The C90 executable was produced by the CF90 compiler (1.0.2.5) and SEGLDR (8.0g). T3D executables were produced by the CFT77_M compiler (6.2.0.5) and MPPLDR (10.u). MPP objects were linked to libsma (1.1.0.18).

All objects were produced using explicit compiler optimization control. The Fortran 90 multitasked and single-threaded objects specified “task1,scalar3,vector3" and “task0,scalar3,vector3” respectively. The CRAFT and shmem objects were built with the “aggress, scalar” compiler switches. The shmem executable was created using the "rdahead=on" loader switch.

- Model Problems

Problem 1

In the model equation (2.1), the value of the coefficient functions are defined as a = 1 and q = 0. The boundary conditions are given as g(x,y) = x + y, thus example problem 1 has the form:

$$(3.1) \quad \begin{cases} -\operatorname{div} a \operatorname{grad} u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

where the value of f is determined by substitution of the exact solution, $u(x,y) = x + y$, into (3.1).

Problem 2

The coefficient functions of (2.1) are assigned the values $a = 1$ and $q = 0$. Boundary conditions are given as $g(x,y) = \sin x \cos y$. The resulting form is as follows:

$$(3.2) \quad \begin{cases} -\operatorname{div} \operatorname{grad} u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

where the value of f is determined by substitution of the exact solution, $u(x,y) = \sin x \cos y$, into (3.2).

Problem 3

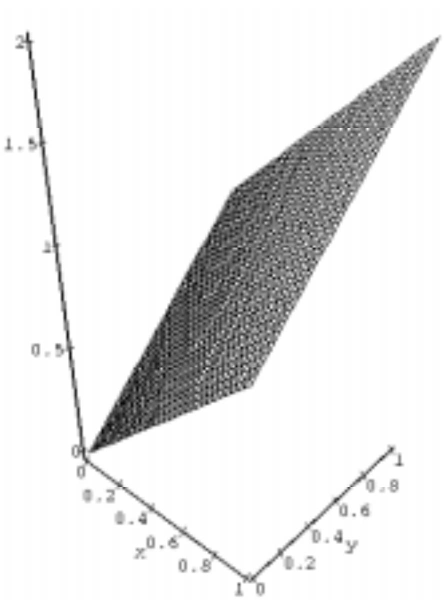
The value of the coefficient functions are are defined as $a = 1$ and $q = 2$ in the model equation (2.1). Boundary conditions are specified as $g(x,y) = (x^2 + y^2)\, e^{(x+y)}$ and so the problem statement takes the following form:

$$(3.3) \quad \begin{cases} -\operatorname{div} \operatorname{grad} u + 2u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}$$

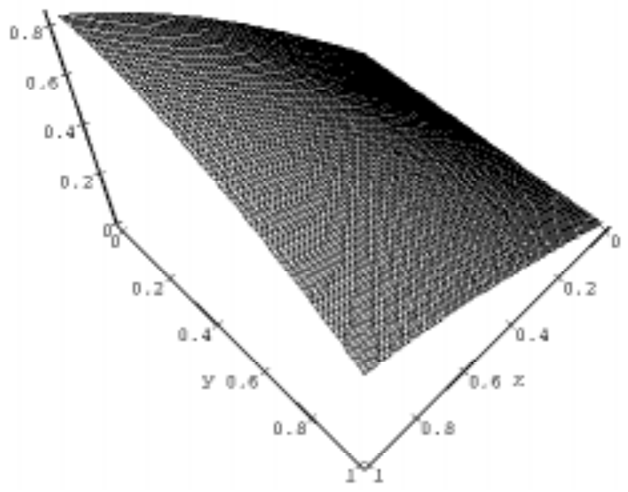
where the value of *f* is determined by substitution of the exact solution, $u(x,y) = (x^2 + y^2)\, e^{(x+y)}$, into (3.3).

- Data for each problem is presented in tabular form with respect to partition size, ie. each table represents a particular problem and partition size.
- Timing and scaling curves were produced from data associated with Problem 1. Curves for problems 2 and 3 were quite similar and hence were not included.
- Good starting values for the r-parameter were determined empirically for small problem sizes. Values of r for larger problems were set a priori by holding the ratio r/h approximately constant.
- For all cases, convergence was pushed to at least seven digits of precision.

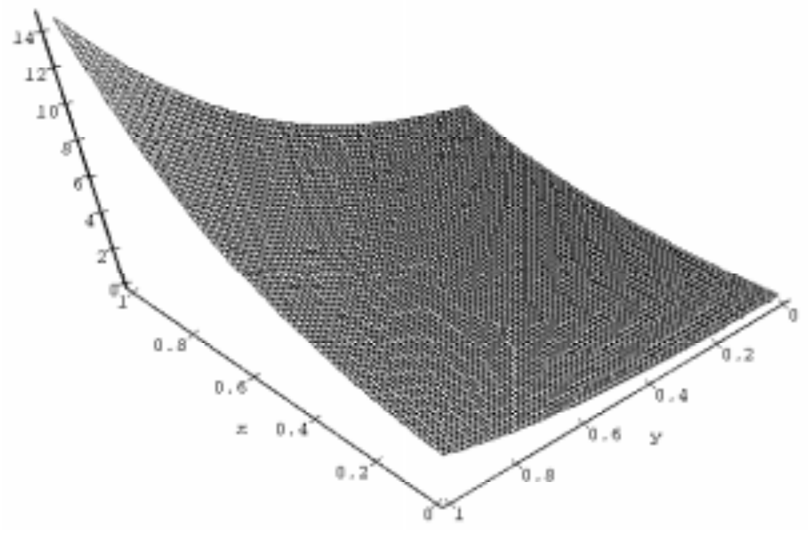
Solution Surfaces



Problem 1



Problem 2



Problem 3
(z-axis constrained)

Experimental Data - Problem 1

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0900	11.52	32768	49408	300	1.05	0.36	0.37	1.60	0.63
256	0.0457	11.70	131072	197120	600	8.10	1.19	1.86	16.73	4.63
512	0.0232	11.88	524288	787456	1200	64.98	8.81	7.98	138.21	36.14
1024	0.0130	13.31	2097152	3147776	2300	492.41	65.83	49.38	1078.89	268.75
1024	0.0100	10.24	4194304	6294528	3600	1477.33	212.33	150.58	3375.57	830.72

Table 1: Parameters and Timings
Problem 1, 64 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0900	11.52	32768	49408	300	1.05	0.36	0.37	0.72	0.37
256	0.0457	11.70	131072	197120	600	8.10	1.19	1.86	7.23	2.56
512	0.0232	11.88	524288	787456	1200	64.98	8.81	7.98	69.63	18.60
1024	0.0130	13.31	2097152	3147776	2300	492.41	65.83	49.38	541.31	137.62
1024	0.0100	10.24	4194304	6294528	3600	1477.33	212.33	150.58	1690.86	420.65

Table 2: Parameters and Timings
Problem 1, 128 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0900	11.52	32768	49408	300	1.05	0.36	0.37	0.38	0.21
256	0.0457	11.70	131072	197120	600	8.10	1.19	1.86	3.39	1.52
512	0.0232	11.86	524288	787456	1200	64.98	8.81	7.98	34.63	10.55
1024	0.0130	13.30	2097152	3147776	2300	492.41	65.83	49.38	272.40	74.69
1024	0.0100	10.24	4194304	6294528	3600	1477.33	212.33	150.58	861.69	224.77

Table 3: Parameters and Timings
Problem 1, 256 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0900	11.52	32768	49408	300	1.05	0.36	0.37	0.26	0.11
256	0.0457	11.70	131072	197120	600	8.10	1.19	1.86	1.62	0.83
512	0.0232	11.88	524288	787456	1200	64.98	8.81	7.98	15.24	6.21
1024	0.0130	13.30	2097152	3147776	2300	492.41	65.83	49.38	136.74	40.40
1024	0.0100	10.24	4194304	6294528	3600	1477.33	212.33	150.58	427.00	117.37
2048	0.0090	18.44	8388608	12587008	5900	---	---	---	---	383.18

Table 4: Parameters and Timings
Problem 1, 512 PEs

Experimental Data - Problem 2

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0590	7.55	32768	49408	500	1.71	0.51	0.52	2.67	1.05
256	0.0310	7.94	131072	197120	1000	13.57	2.11	3.32	27.87	7.72
512	0.0172	8.81	524288	787456	2100	113.29	15.94	14.95	241.86	63.25
1024	0.0100	10.24	2097152	3147776	4600	972.47	131.00	94.78	2157.86	542.32
1024	0.0080	8.19	4194304	6294528	6800	2864.52	390.97	272.82	6376.26	1569.28

Table 5: Parameters and Timings
Problem 2, 64 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0590	7.55	32768	49408	500	1.71	0.51	0.52	1.19	0.64
256	0.0310	7.94	131072	197120	1000	13.57	2.11	3.32	12.04	4.26
512	0.0172	8.81	524288	787456	2100	113.29	15.94	14.95	121.86	32.55
1024	0.0100	10.24	2097152	3147776	4600	972.47	131.00	94.78	1082.44	277.39
1024	0.0080	8.19	4194304	6294528	6800	2864.52	390.97	272.82	3194.54	799.34

Table 6: Parameters and Timings
Problem 2, 128 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0590	7.55	32768	49408	500	1.71	0.51	0.52	0.63	0.35
256	0.0310	7.94	131072	197120	1000	13.51	2.11	3.32	5.65	2.53
512	0.0172	8.81	524288	787456	2100	113.29	15.94	14.95	60.61	18.46
1024	0.0100	10.24	2097152	3147776	4600	972.47	131.00	94.78	545.07	150.19
1024	0.0080	8.19	4194304	6294528	6800	2864.52	390.97	272.82	1627.64	422.18

Table 7: Parameters and Timings
Problem 2, 256 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0590	7.55	32768	49408	500	1.71	0.51	0.52	0.43	0.18
256	0.0310	7.94	131072	197120	1000	13.51	2.11	3.32	2.69	1.39
512	0.0172	8.81	524288	787456	2100	113.29	15.94	14.95	26.68	10.88
1024	0.0100	10.24	2097152	3147776	4600	972.47	131.00	94.78	273.47	70.26
1024	0.0080	8.19	4194304	6294528	6800	2864.52	390.97	272.82	806.61	79.97
2048	0.0070	14.43	8388608	12587008	11600	---	---	---	---	755.93

Table 8: Parameters and Timings
Problem 2, 512 PEs

Experimental Data - Problem 3

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0900	11.52	32768	49408	550	1.86	0.55	0.57	2.93	1.15
256	0.0457	11.70	131072	197120	1150	15.66	2.48	3.74	32.05	8.87
512	0.0232	11.86	524288	787456	2300	123.99	17.44	16.41	264.89	69.27
1024	0.0120	12.29	2097152	3147776	4500	957.43	128.22	92.07	2110.81	530.54
1024	0.0100	12.24	4194304	6294528	7800	3158.09	443.78	310.80	7314.63	1800.06

Table 9: Parameters and Timings
Problem 3, 64 PEs

1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0640	8.19	32768	49408	550	1.86	0.55	0.57	1.33	0.70
256	0.0350	8.96	131072	197120	1150	15.66	2.48	3.74	13.84	4.89
512	0.0190	9.73	524288	787456	2300	123.99	17.44	16.41	133.41	35.65
1024	0.0090	10.24	2097152	3147776	4500	957.43	128.22	92.07	1057.69	271.35
1024	0.0080	9.21	4194304	6294528	5000	3158.09	443.78	310.80	3665.03	916.75

Table 10: Parameters and Timings
Problem 3, 128 PEs

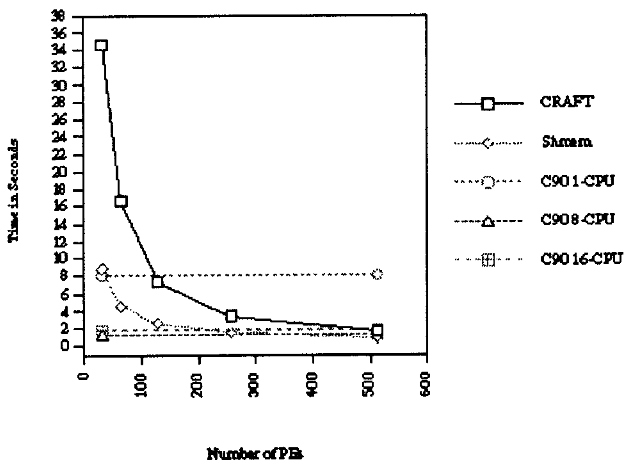
1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0640	8.19	32768	49408	550	1.86	0.55	0.57	0.69	0.38
256	0.0350	8.96	131072	197120	1150	15.66	2.48	3.74	6.51	2.92
512	0.0190	9.73	524288	787456	2300	123.99	17.44	16.41	66.37	20.21
1024	0.0100	10.24	2097152	3147776	4500	957.43	128.22	92.07	532.96	146.92
1024	0.0090	9.21	4194304	6294528	5000	3158.09	443.78	310.80	1867.00	484.26

Table 11: Parameters and Timings
Problem 3, 256 PEs

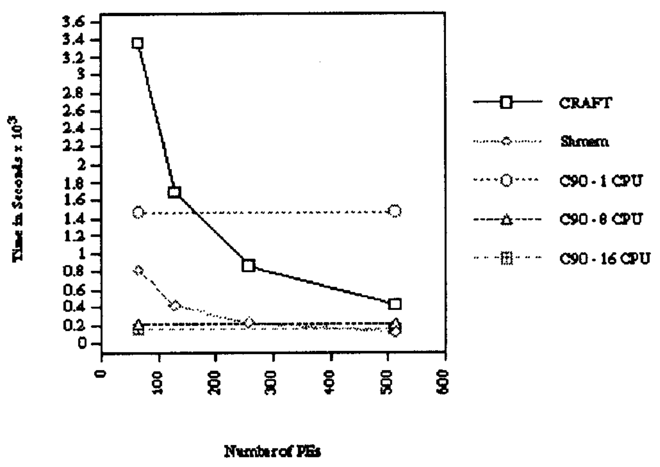
1/h	ρ	ρ*(1/h)	Elem	DOF	Iter	C90-1CPU	C90-8CPUs	C90-16CPUs	CRAFT	shmem
128	0.0640	8.19	32768	49408	550	1.86	0.55	0.57	0.48	0.20
256	0.0350	8.96	131072	197120	1150	15.66	2.48	3.74	3.10	1.60
512	0.0190	9.73	524288	787456	2300	123.99	17.44	16.41	29.23	11.91
1024	0.0100	10.24	2097152	3147776	4500	957.43	128.22	92.07	267.54	78.23
1024	0.0090	9.21	4194304	6294528	7800	3158.09	443.78	310.80	925.22	254.30
2048	0.0080	16.39	8388608	12587008	13000	---	---	---	---	847.16

Table 12: Parameters and Timings
Problem 3, 512 PEs

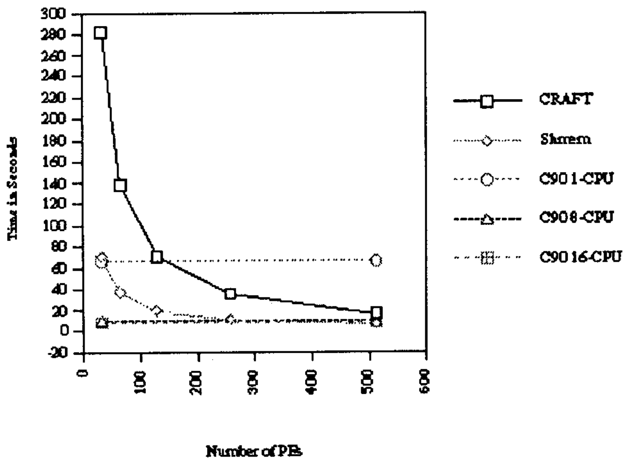
Prob 1: Time vs Number of PEs (0.2 M DOF)



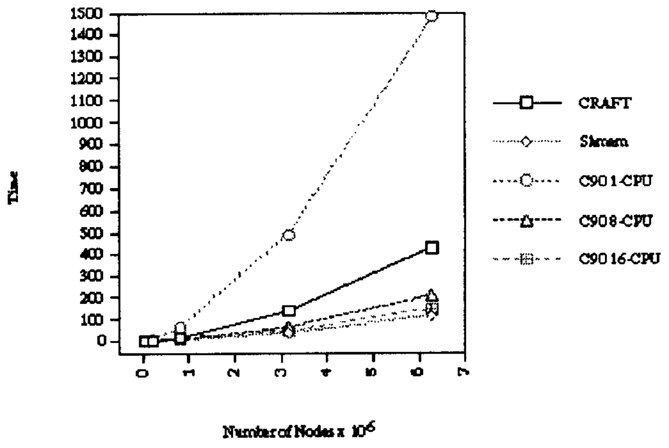
Time vs Number of PEs (6.3 M DOF)



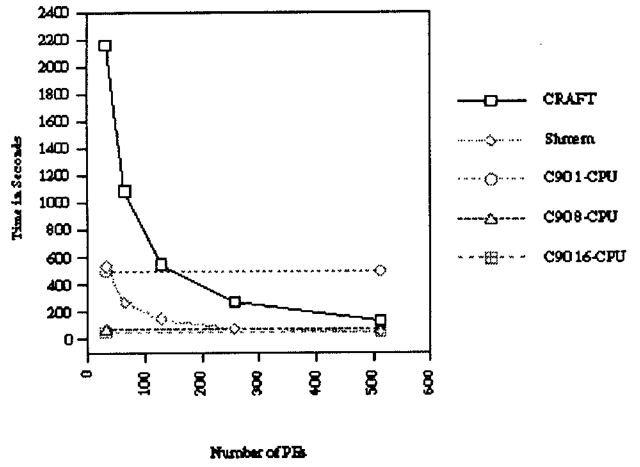
Prob 1: Time vs Number of PEs (0.8 M DOF)



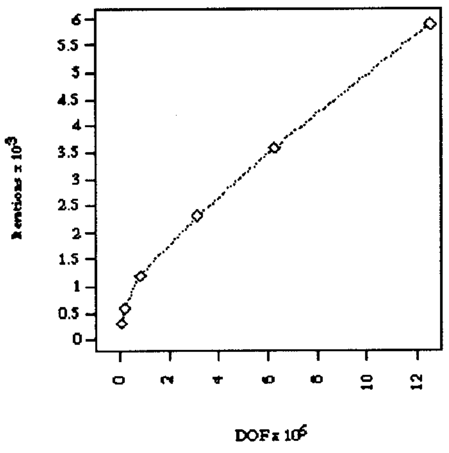
Time vs Degrees of Freedom on 512 PEs



Prob 1: Time vs. Number of Pes (3 M DOF)



DOF vs Number of Iterations



Observations

- The PFEM algorithm performs very well on a variety of architectures. In fact, it is competitive with contemporary preconditioned conjugate gradient solvers when implemented serially [Layton and Rabier, 1995].
- Speedups are near-linear throughout the range of available partition sizes. On fixed partitions, run times scale with the problem size like O(n3/2).
- The availability of such a method has significant implications. Memory requirements are minimized, hence larger problems become tractable. Its exceptional performance and scalability allows finer resolutions along spatial and time dimensions to be considered.
- Problems involving over 6 million degrees of freedom (DOF) were solved on the C90 in less than 3 minutes. Similar problems were solved on PSC's T3D in under 2 minutes. We were also able to solve problems comprised of over 12 million degrees of freedom in less than 7 minutes on the T3D.
- Use of the cache prefetching technique described by Brooks (1994) yielded better than a factor of two improvement in run times. The current communication to computation ratio is approximately 1:1.
- For comparison purposes, we also took a quick look at a LAPACK banded matrix solver. When operating on smaller systems constructed from similar meshes of conforming linear elements, the SGBTR_ routine was found to be an order of magnitude slower than our Fortran 90 PFEM code. For a discussion of PFEM performance with respect to conjugate gradient solvers, see Layton and Rabier (1995).
- Performance of the single-CPU Fortran 90 code was measured at 434 MFlop/s on the C90.
- The Fortran 90 code consistently required additional iterations to converge to the set tolerance (1E-7). We attribute this to the difference between the Cray and IEEE floating point systems.
- The Fortran 90 compiler at its maximum tasking level was not effective for our original source form. Directives were inserted manually. Measured speedup on the C90 indicates that the PFEM code is better than 99% parallel, which prompts us to make a point.

When citing cost-performance figures tied to C90 CPUs, examination of performance with respect to a single CPU as well as multiple CPUs is relevant. For our shared memory code, the ratio of PEs required to match the performance of one C90 CPU is around 36:1. However, the performance of 16 CPUs is surpassed by about 400 PEs for smaller problems and by 430 PEs for larger ones. Related ratios are about 25:1 and 27:1 respectively.

- The CRAFT model provides a very nice intermediate stage from which data parallel codes may be ported to the shared memory model. The ability to mix Fortran 90 array syntax with CRAFT directives and shared memory calls can be very handy. Performance is acceptable. For smaller problems (~200,000 DOF), the CRAFT code was able to match the timings of 16 C90 CPUs on a 512-PE partition.
- Current work by Layton and Rabier et al. includes a study of fluid flows characterized by high Reynolds numbers and implementation of full “h-p” adaptivity. Among other things, these works have produced new theorems in linear algebra and graph colorings and led to the formulation of higher degree nonconforming finite element spaces.
- Continuing work at PSC will center on issues related to the integration of such a method into a commercial FEM package.

Acknowledgements

First and foremost, we wish to thank Bill Layton and Patrick Rabier of the Mathematics Department at Pittsburgh for their guidance and support on this project.

We also wish to thank Chad Vizino (systems), Anjana Kar (graphics), and especially Bob Dixon (education) of the Pittsburgh Supercomputing Center for the invaluable contributions they made to the production of this presentation.

The colored grid that appears on the cover sheet was produced by Joseph Maubach of the University of Pittsburgh Mathematics Department. It illustrates an adaptive mesh of nonconforming finite elements clustering in the vicinity of a discontinuity.

The video clip shown during the latter part of this presentation was edited from material developed by members of the finite element group in attendance at PSC’s Institute on Parallel Computing, 1994.

Bibliography

Brooks, J., Single PE Optimization Techniques for the CRAY T3D System, CRAY Research, Inc., Benchmarking Group Publication dated September 27, 1994.

Crosbie, S. W., and Ervin, V. J., Implementation of the Parallel Algebraic Splitting Method, to appear in J. of Comp. Phy., 1995.

Crouzeix, M., and Raviart, P. A., Conforming and Nonconforming Finite Element Methods for Solving Stationary Stokes Equations, R.A.I.R.O. 3 (1973) 33-76.

Dyakonov, E. G., On a Method of Solving the Poisson Equation, Soviet Math. Doklady 3 (1962) 320-323.

Ervin, V. J., and Layton, W. J., Parallel Algebraic Splittings and the Peaceman Rachford Iteration, submitted to Linear Algebra and Its Applications, 1994.

Jeurissen, R. and Layton, W. J., Load Balancing by Graph Coloring, An Algorithm, Computers Math. Applic. Vol. 27, No. 3 (1994) 27-32.

Layton, W. J., Maubach, J., and Rabier, P. J., Uniform Convergence Estimates for an Element-wise Parallel Finite Element Method, I.C.M.A. Report, University of Pittsburgh (1992).

Layton, W. J., Maubach, J., and Rabier, P. J., Parallel Algorithms for Monotone Operators of Local Type, to appear in Numer. Math., 1995.

Layton, W. J., Maubach, J., Rabier, P. J., and Sunmonu, A., Parallel Finite Element Methods, In Proc. 5th Intl. Conf. on Parallel and Distributed Computing, (1992).

Layton, W. J., and Rabier, P. J., Peaceman-Rachford Procedure and Domain Decomposition for Finite Element Problems, to appear in Num. Lin. Alg. and Applic., 1995. I.C.M.A. Report, University of Pittsburgh (1991).

Layton, W. J., and Rabier, P. J., Domain Decomposition via Operator Splitting for Nonsymmetric Problems, Appl. Math. Letters 5 (1992) 67-70.

Marchuk, G. I., Splitting and Alternating Direction Methods, Handbook of Numerical Analysis (P. Ciarlet and J. Lions, ed.), North-Holland (1990) 197-464.

Maubach, J., Iterative Methods for Nonlinear Partial Differential Equations, C.W.I., Amsterdam (1993).

Peaceman, D. W., and Rachford, H. H. Jr., The Numerical Solution of Parabolic and Elliptic Differential Equations, J. Soc. Appl. Math. 3 (1955) 28-41.