PAMIHR. A Parallel FORTRAN Program for Multidimensional Quadrature on Distributed Memory Architectures

G. Laccetti and M. Lapegna

Center for Research on Parallel Computing and Supercomputers - CNR
University of Naples "Federico II"
via Cintia - Monte S. Angelo. 80126 Napoli (Italy)
(laccetti,lapegna)@matna2.dma.unina.it

Abstract. PAMIHR: a parallel adaptive routine for the approximate computation of a multidimensional integral over a hyperrectangular region is described. The software is designed to efficiently run on a MIMD distributed memory environment, and it's based on the widely diffused communication system BLACS. PAMIHR, further, gives special attention to the problems of scalability and of load balancing among the processes.

1 Introduction and Software Description

PAMIHR (Parallel Adaptive Multidimensional Integration on Hyper-Rectangles) is a parallel FORTRAN package for the computation of:

$$I[f] = \int_{\Omega} f(t_1, ..., t_{IDIM})\, dt_1 \cdots dt_{IDIM} \tag{1.1}$$

on a MIMD distributed memory environment with P processes, where

$$\Omega = \texttt{[A(1),B(1)]} \times \cdots \times \texttt{[A(IDIM),B(IDIM)]}$$

is a IDIM-dimensional hyperrectangular region with $2 \leq \texttt{IDIM} \leq 10$.

The software we are describing is designed to return an approximation RESULT of $I[f]$ and an estimate ERROR of the absolute error $|I[f] - \texttt{RESULT}|$ such that:

$$|I[f] - \texttt{RESULT}| \leq \texttt{ERROR} < \max\{\texttt{ABSACC}, \texttt{RELACC} \times |\texttt{RESULT}|\}$$

where ABSACC and RELACC are absolute and relative input provided tolerances, respectively. In such a way the algorithm ends if the "fairest" accuracy is reached. PAMIHR is based on a typical global adaptive algorithm. For the computation of the integral and of the error estimates in each subdomain, PAMIHR uses an imbedded family of fully symmetric rules very similar to the well known integration rules developed by Genz and Malik in [7]. A common strategy to compute the error estimate in an automatic routine for the quadrature is based on the difference between two imbedded integration rules. However this procedure is very unreliable and PAMIHR uses the error estimate procedure developed in [1]. Is is well known that the main problem in the development of a parallel adaptive algorithm is the workload balancing, because, for this kind of algorithm, it is

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 1144–1148, 1999.

impossible to determine an *a priori* optimal decomposition of the integration region [3, 5, 10]. Therefore, in order to balance the workload, our algorithm configures the processes as a 2-dimensional periodic grid. In this topology each process p_i has 4 neighbours, 2 for each direction. In the generic direction dir, $(dir = 0, 1)$, we define $p_{i+}^{(dir)}$ and $p_{i-}^{(dir)}$, the next and the previous process of p_i, respectively. Therefore, at a generic iteration j, each process attempts to send to $p_{i+}^{(dir)}$ its subdomain $\hat{s}_i^{(j)}$ with largest error estimate, in order to share, with the other processes, its "hard" subdomains. More precisely, in a given direction $dir = mod(j, 2)$ let $\hat{e}_i^{(j)}$, $\hat{e}_{i+}^{(j)}$ and $\hat{e}_{i-}^{(j)}$ be respectively the largest error estimates in the current process p_i, in the process $p_{i+}^{(dir)}$ and in the process $p_{i-}^{(dir)}$; if $\hat{e}_i^{(j)} > \hat{e}_{i+}^{(j)}$ then the current process p_i sends the subdomain $\hat{s}_i^{(j)}$ with error estimate $\hat{e}_i^{(j)}$ to the process $p_{i+}^{(dir)}$, and $\hat{s}_i^{(j)}$ is removed from the memory of p_i. In the same way if $\hat{e}_{i-}^{(j)} > \hat{e}_i$ the current process receives the subdomain $\hat{s}_{i-}^{(j)}$ with error estimate $\hat{e}_{i-}^{(j)}$ from the process $p_{i-}^{(dir)}$. Then each process p_i can compute an approximation of the integral defined only over the subdomains located in its own local memory. Only one global sum among the processes is then required, at the end of the algorithm, for the computation of the final approximations `RESULT` and `ERROR`. Note that with this communication strategy, the scalability is improved by avoiding global communications and, at the same time, the workload is balanced by a distribution of the subdomains only among pairs of processes directly connected in a 2-dimensional mesh. For the communications among the processes, `PAMIHR` uses the `BLACS` (Basic Linear Algebra Communication Subprograms) [4] message-passing library. This library is the key to the portability of `PAMIHR`, because several versions of this library are available for different architectures (Intel Paragon, IBM SP2,...) as well for clusters of workstations and *pile of PC* (the so called Beowulf systems), so our software can run on all architectures where this communication system is available. However a prototype of the MPI version of `PAMIHR` is available too. The present version of the package is written in double precision ANSI Standard FORTRAN 77 except only for the calls to the BLACS routines written in non standard style. The `BLACS` communication system in not part of the package, because the efficiency depends strictly on the implementation of this library. So a vendor optimized version of `BLACS` should be used. However several versions of `BLACS` are available from *netlib*. The specification of `PAMIHR` is:

```
 SUBROUTINE PAMIHR (ICNTXT, IDIM, F, A, B, RELACC, ABSACC, MAXFUN,
*       ITEST, NSPLIT, RESULT, ERROR, FCOUNT, WS, LD, IFLAG)
 INTEGER ICNTXT, IDIM, MAXFUN, ITEST, NSPLIT(IDIM), FCOUNT, LD, IFLAG
 DOUBLE PRECISION A(IDIM), B(IDIM), RELACC, ABSACC, RESULT, ERROR,WS(LD,*)
 EXTERNAL  F
```

A detailed description of the input/output parameters can be found in the internal documentation of the routine[1]. In this contest we would like to emphasize

[1] refer to `http://pixel.dma.unina.it/RESEARCH/pamihr.html`

that, except for an initial explicit call to BLACS routines, the user calls PAMIHR in a simple and straightforward way, (in a sequential fashion, let say) without any explicit reference to the parallel environment and/or to communications among the processes.

2 Performance of PAMIHR

To test the performance of PAMIHR, we used the technique developed in [6] and a set of 8 functions families from the Genz package [6]. These families are characterized by some peculiarity (several kinds of peaks, oscillations, discontinuities, singularities,...). The families have been tested using several dimension up to 10 and different tolerances on the region $\Omega = [0,1]^{IDIM}$. Each family $f^{(j)}(\underline{x}), (j = 1, \ldots, 8)$ is composed by 20 different functions where some minor parameters change. The parameters determine the sharpness and the location of the difficulty. A complete and detailed report of such intensive test activity may be found in [9]. Here, just some results related to the function with corner singularity:

$$f^{(8)}(\underline{x}) = (\sum_{i=1}^{IDIM} \beta_i x_i)^{-IDIM/2.7}$$

are showed. All test are performed on the 512-node Intel Paragon XP/S operated by the Center for Advanced Computing Research. Access to this facility was provided by the California Institute of Technology.

The first set of experiments is aimed to measure reliability and accuracy of PAMIHR with just 1 process, by comparing the obtained results in 3 dimensions with the results from the sequential routine DCUHRE [2] that uses a similar computational kernel. Table 1 reports the required relative tolerance; then, for both PAMIHR and DCUHRE, there are: FCOUNT, the average, over the 20 functions of the family, of the number of function evaluations needed to reach the tolerance; OK, the number of cases where the tolerance is reached with FCOUNT$\leq$MAXFUN $=$ 100000$\times$IDIM (where MAXFUN is the maximum number of function evaluations); NU, the number of cases where the actual error is bigger than the estimated one; E1, the average, over the 20 functions of the family, of the actual relative errors. From this table we can say that 1-process PAMIHR execution, produces very accurate results with a moderate number of function evaluations as well as DCUHRE. Further PAMIHR gives about the same results, in terms of accuracy and efficiency, of DCUHRE. However, from the complete results in [9], PAMIHR seems to privilege the accuracy, with respect to the efficiency, in the sense that, in general, PAMIHR computes more exact digits than DCUHRE with few more function evaluations. In Table 2 are reported the results of the second set of experiments performed to measure the efficiency of the algorithm when the number of processes P increases and the workload (the number of function evaluations) is constant in the processes (the so called *scalability*) [8]. We note good value of scalbility for the $f^{(8)}(\underline{x})$ functions family. Further, in Table 2, we can see which average error is possible to obtain with PAMIHR in a fixed time (the time required for the execution with 1 process and a given tolerance) when the number of processes P grows. For all problems we note a significant error reduction. That means that

the "hard" subdomains are rapidly distributed across the processes so they do not waste time on unimportant subdomains.

	PAMIHR				DCUHRE			
RELACC	FCOUNT	OK	NU	E1	FCOUNT	OK	NU	E1
.10D+00	600	20	0	0.503D-02	277	20	0	0.111D-01
.10D-01	1771	20	0	0.336D-03	1840	20	0	0.319D-03
.10D-02	4335	20	0	0.208D-04	5782	20	0	0.131D-04
.10D-03	7053	20	0	0.246D-05	10826	20	0	0.124D-05

Table 1 - Accuracy test with 1 process for the $f^{(8)}$ functions family

RELACC	P=4	P=16	P=32	P=64
1.E-1	0.84	0.73	0.70	0.68
1.E-2	0.93	0.89	0.87	0.86
1.E-3	0.95	0.94	0.93	0.93
1.E-4	0.96	0.95	0.95	0.94

RELACC	P=4	P=16	P=32	P=64
1.E-1	2.E-3	1.E-3	8.E-4	7.E-4
1.E-2	1.E-4	7.E-5	5.E-5	4.E-5
1.E-3	3.E-7	1.E-7	8.E-7	7.E-7
1.E-4	1.E-8	1.E-9	5.E-10	2.E-10

Table 2 - Scaled efficiency and error reduction for the $f^{(8)}$ functions family

Finally some few words about a comparison with an early version of PAMIHR: the subroutine D01FAFP in the NAG Parallel Library. PAMIHR is a more robust software item than D01FAFP, in the sense that PAMIHR performs successfully the whole Genz test package, where D01FAFP shows some invalid result. For example in 2 dimensions with 1 process, for the *discontinuous function* of the Genz package:

$$f^{(6)}(\underline{x}) = \begin{cases} 0 & \text{if } x_1 > \beta_1 \text{ or } \quad x_2 > \beta_2 \\ exp(\sum_{i=1}^{IDIM} \alpha_i x_i) & \text{otherwise} \end{cases} \quad \text{with} \quad \begin{array}{|c|c|c|} \hline \alpha_i & 5.8213 & 19.178 \\ \hline \beta_i & 0.85403 & 0.62471 \\ \hline \end{array} \qquad (2.1)$$

(with such a values, the exact result, correct to 9 digits, is 204955.199) and RELACC = 0.10E-2 the two routines give the results reported in Table 3: the error estimate is correct for PAMIHR and not determined for D01FAFP. Such improvement is obtained in PAMIHR avoiding instructions leading to divisions by 0 and/or indeterminant forms in the error estimation procedure. Another interesting issue of PAMIHR is related to the maximum number of function evaluations MAXFUN. This number is an input parameter, and it refers to the "total" number of function evaluations. More precisely to each process is assigned the integer part of MAXFUN/P function evaluations. The node-algorithm is designed to avoid a next iteration if such (possible) iteration exceeds MAXFUN/P; but in general P does not divide MAXFUN, so each process still has available some function evaluations, Let M_i be the number of unused function evaluations in each process p_i. If $\sum_i M_i \geq 2 \times VALFUN$ (where $VALFUN$ is the number of function evaluation to compute the integration rule in a subdomain), some of the P processes can perform a further iteration. So, before returning to the calling program, PAMIHR distributes the function evaluations still available, assigning them to one or more processes, so that some of the processes can continue to work. Let us focus the attention, for example, on the *product peak* function of the Genz package in 10 dimensions with 8 processes:

$$f^{(2)}(\underline{x}) = \prod_{i=1}^{IDIM}(\alpha_i^{-2} + (x_i - \beta_i)^2)^{-1}$$

with `RELACC = 0.10E-2`, `MAXVAL=200000` and

α_i	.401	.408	.832	.339	1.33	1.21	$3.16E-3$	1.35	$3.38E-2$	$7.89E-2$
β_i	.910	.510	.150	.942	.503	.490	.275	.903	$4.71E-2$	.902

(2.2)

The routines give the results reported in Table 4, where we observe that `IFLAG=100` (maximum number of function evaluations exceeded) for `D01FAFP` with `FCOUNT` much smaller than `MAXFUN`, whereas `IFLAG=0` (successful exit) for `PAMIHR`.

	D01FAFP	PAMIHR
RESULT	2070E+02	2060E+02
ERROR	NaN	.4357E-03
FCOUNT	957	4257
IFLAG	0	0

Table 3: `PAMIHR` and `D01FAFP` results for $f^{(6)}(\underline{x})$ functions family. α_i and β_i as in (2.1)

	D01FAFP	PAMIHR
RESULT	.3567E-12	.3567E-12
ERROR	.6058E-04	.6292E-04
FCOUNT	104200	145880
IFLAG	100	0

Table 4: `PAMIHR` and `D01FAFP` results for $f^{(2)}(\underline{x})$ functions family. α_i and β_i as in (2.2)

References

[1] Berntsen J. - *Practical error estimation in adaptive multidimensional quadrature routines* - J. Comput. Appl. Math., vol. 25 (1989), pp. 327-340.

[2] Berntsen J., T.O. Espelid, A.C. Genz - *Algorithm 698: DCUHRE-An adaptive multidimensional integration routine for a vector of integrals* - ACM Trans. on Math. Software, vol. 17 (1991), pp. 452-456,

[3] de Doncker E., J. Kapenga - *Parallel Cubature on Loosely Coupled Systems* - in *Numerical Integration: Recent developments, software and Applications* (T. Espelid and A. Genz eds.), Kluwer, 1992, pp. 317-327.

[4] Dongarra J., R.C. Whaley - *A user's guide to the BLACS v1.0* - Tech. Rep. CS-95-281, LAPACK Working Note no. 94, Univ. of Tennessee, 1995.

[5] Genz A.C. - *The numerical evaluation of multiple integrals on parallel computers* - in *Numerical Integration* (P. Keast and G. Fairweather eds.), D. Reidel Publishing Co., 1987, pp. 219-230.

[6] Genz A.C. - *A Package for Testing Multiple Integration Subroutines* - in *Numerical Integration*, (P.Keast, G.Fairweather, eds.), D. Reidel Publishing Co., 1987, pp. 337-340.

[7] Genz A.C., A.A. Malik - *An imbedded family of fully symmetric numerical integration rules* - SIAM J. Num. Anal., vol. 20 (1983), pp. 580-588.

[8] Gustafson J., G. Montry and R. Benner - *Development of parallel methods for a 1024 processor hypercube* - SIAM J. on Scientific and Statistic Computing, Vol. 9 (1988), pp. 580-588.

[9] Laccetti G., M. Lapegna, A. Murli - *DSMINT. A Scalable Double Precision FORTRAN Program to Compute Multidimensional Integrals* - Tech. Rep. CPS-96-9 Center for Research on Parallel Computing and Supercomputers, 1996.

[10] Lapegna M. - *Global adaptive quadrature for the approximate computation of multidimensional integrals on a distributed memory multiprocessor* - Concurrency: Practice and Experiences, vol. 4 (1992), pp. 413-426