Current status of the ParInt package for Parallel multivariate Integration

E. de Doncker, K. Kaugars, L. Cucos, R. Zanny
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008 USA
{elise,kkaugars,lcucos,rrzanny}@cs.wmich.edu

Abstract

The ParInt project focuses on the development of parallel algorithms and software for the computation of multivariate integrals. We will give an overview of the contents and capabilities of the package. Our objective has been to provide the end-user with state of the art problem solving power. This has required work in a number of areas, including the fundamental numerical techniques, strategies for parallelization, user interfaces for general use and specific applications, and visualization of computations to analyze the mutual influences of problem characteristics and algorithm behavior. Furthermore, the integration of all the above into a versatile set of tools is aimed toward an efficient use of the available parallel or distributed computer resources.

1 History and background of the PARINT project

PARINT is a package for parallel multivariate integration. The PARINT project started officially in 1994 with support from the National Science Foundation[1], as it culminated from the research of the investigators in the field of numerical integration and parallel integration in particular. A copyright was obtained from the U.S. Library of Congress in 1999 for PARINT1.0 [2]. Further NSF support[3] was secured in 1999-2000, which also resulted in the inception of our Athena cluster with 32 Athlon 800MHz nodes, each with 256MB of RAM and interconnected through a fast Ethernet switch. In 2001 our institution (WMU) financed an extension of our cluster with another 32 (1.2GHz) Athlon processor/512MB nodes; and we received an NSF Infrastructure grant[4] allowing us to acquire a fast (Myrinet) interconnect. Another new NSF grant for continued support of the project has recently been approved.

As a result, we were able to release PARINT1.1 and have PARINT1.2 in beta. The PARINT group currently consists of E. de Doncker, A. Gupta, K. Kaugars (WMU) A. Genz (WSU), R. Zanny, and L. Cucos (PhD students at WMU).

Over the years our interest in the project has been sparked further by applications in various fields, such as in Statistics (e.g., multivariate normal, t- distributions), Finite Elements (e.g., automotive simulations), Computational Finance (financial derivatives, e.g., Collateral Mortgage Obligations, Mortgage Backed Security problems), High Energy Physics (e.g., interaction cross sections), Computational Chemistry (e.g., Fock matrix representing the electronic structure of an atom or molecule), Computational Geometry (e.g., the expected volume of the convex hull of random points in a convex set), etc.

The PARINT code can be invoked by calls from the user's application, or via the command line or the Graphical User Interface (GUI). The PARINT1.2 release allows for the computation of multivariate integrals over hyper-rectangular or simplex regions. The user specifies the region $\mathcal{D}$ by the upper and lower integration limits in case of a hyper-rectangular region, or by the vertex coordinates of a simplex. The integrand f is supplied as a function written

[1] NSF Computer and Information Sciences and Engineering (CISE) grant CCR-9405377

[3] NSF CISE Advanced Computational Research (ACR) grant ACR-0000442
[4] NSF grant EIA-0130857

Evaluate initial region and update results
Initialize priority queue with initial region
while (evaluation limit not reached and estimated error too large)
    Retrieve region from priority queue
    Split region
    Evaluate new subregions and update results
    Insert new subregions into priority queue

Figure 1: Adaptive integration meta-algorithm

in C (or consisting of C callable code). The user's desired accuracy is determined by an absolute and a relative error tolerance, ε_a and ε_r, respectively, as well as a limit on the number of integrand evaluations.

Thus, an approximation $Q \approx I = \int_{\mathcal{D}} f(\mathbf{x})\ d\mathbf{x}$ is obtained and an absolute error estimate E, with the aim to satisfy the criterion $|Q - I| \leq E \leq \max\{\varepsilon_a, \varepsilon_r |I|\}$ within the allowed number of function evaluations, or indicate an error condition if the limit has been reached. Note that the integrand f represented here as a single valued function may in general be a vector function; E is then calculated in infinity norm.

PARINT is layered over MPI. As such, it is portable (on top of MPI) to powerful hardware resources, and can be executed using a user-specified number of processors of its host (distributed) system. The state of the art algorithms in the parallel integration engine include:

- An adaptive region subdivision algorithm for low to moderate (10, say) integral dimensions, equipped with a load balancing strategy to handle localized integrand difficulties such as peaks and singularities;
- A non-adaptive Quasi-Monte Carlo (QMC) method which can generally be used for high dimensions and fairly smooth integrands; load balancing is applied to deal with possible heterogeneous processors [3];
- Specialized applications based on QMC methods for the calculation of multivariate normal and Student-t distribution integrals.

PARINT is set up as a black box, to which the user specifies the integral and requested accuracy. The adaptive method inherently focuses on difficult integrand behavior, by concentrating subdivisions and thus integrand evaluation points in its vicinity. The more experienced user can adjust parameters to tweak, e.g., the subdivision and load balancing strategies.

For adaptive as well as non-adaptive methods, the parallel processes are organized as asynchronous workers, one of which may assume a controlling role (alternatively there is a dedicated controller) to update global quantities such as the total integral and error estimate and the current absolute error tolerance (which may be based on the current integral approximation via a relative error tolerance), and to check termination criteria. In the QMC methods, the worker processes sample the integrand and the controller collects results from the workers to produce the mean and standard error estimates.

In subsequent sections of this paper we will discuss the adaptive task partitioning paradigm with some parallel performance results in Section 2, the QMC based methods and performance results in Section 3, numerical experiments in Section 4, and future directions in Section 5.

2 Adaptive Methods

The adaptive algorithm adheres to the meta-algorithm which appears in Figure 1. The given domain is initially partitioned (statically) among the worker processes. Each worker then performs a sequence of subdivision steps consisting of the selection of the next region to subdivide (based on the local region error estimate), its subdivision and the evaluation of the subregions, and intermittent update messages to the controller to report changes in the worker's integral

```
(from Tobimatsu and Kawabata):
          RESULT        EST. ERR.      TIME      SYS.
DICE    2.8517e-02   0.0256e-02    44hrs56'  Fujitsu VPP500 (1 PE)
BASES   2.9203e-02   0.0016e-02    5'47''    HP9000/800

ParInt: 2.9140e-02   0.0291e-02    1hr59'    AMD Athlon 800MHz PC



RESULT: 0.0291396843583524
ESTABS: 0.000291396765466568
STATUS: Fcn count: 437730735; Region count: 6734319; Fcn count flag: 0
        Time: 7125.623
```

Figure 2: 4D integral results

and error estimate. We will refer to the global adaptive strategy where the region selection criterion is based on the local region error estimate as GA.

Most integrands will feature varying function behavior across the integration domain (e.g., singularities or peaks). Workers who initially receive a difficult portion of the integration domain will have to perform more work to complete their part of the problem than workers whose initial region is easy. These latter workers may become *idle*, correspondingly, dynamic load balancing is needed to maintain efficient use of the processors. Our current load balancing technique is *scheduler based* (SB): the controller maintains a list of idle and non-idle workers. Updates from workers contain information on the worker's idle status. When a worker is idle, the controller/scheduler directs that worker to request work from a non-idle worker. Our experiments with various load balancing strategies have indicated that random polling (RP) is a promising alternative. In RP, idle and near-idle workers select workers at random to request work from.

As an example we computed the cross-section integral from [11] for the $e^+ \, e^- \to \mu^+ \, \mu^- \, \gamma$ interaction, which gives rise to a 4-dimensional integration problem. The results of PARINT in the table of Figure 2 are compared those of to DICE and BASES [11] (note that BASES solves the problem after analytic simplifications).

Figure 3 furthermore compares the speedup obtained using RP vs. SB (vs. no load balancing) for the parallel computation. The speedup of RP is close to ideal on 32 nodes. In fact, the speedup curve shows a super-linear behavior (above ideal) for some numbers of processors. This type of behavior is often ascribed to effects of, e.g., different memory and cache usage when different numbers of processors are used. In our asynchronous distributed application, it must be pointed out that a problem is generally solved differently as the number of processors varies, for example different subdivisions may be done as a result of different communication delays, leading to different algorithmic decisions. Using random polling for load balancing introduces an additional (random) effect.

3 Quasi-Monte Carlo

3.1 Method

The Quasi-Monte Carlo (QMC) method used in PARINT generates a sequence of randomized (Korobov) lattice rule approximations

$$K_N(\boldsymbol{\beta}) = \frac{1}{N}\sum_{i=1}^{N} f(\{\frac{i}{N}\mathbf{v} + \boldsymbol{\beta}\}), \tag{1}$$

where $\mathbf{v}$ is the lattice generator vector, and $\boldsymbol{\beta}$ is a uniformly distributed random vector. Computing the rule for q random $\boldsymbol{\beta}_j$ vectors, the integral is approximated by

$$\bar{K}_N = \frac{1}{q}\sum_{j=1}^{q} K_N(\boldsymbol{\beta}_j), \tag{2}$$

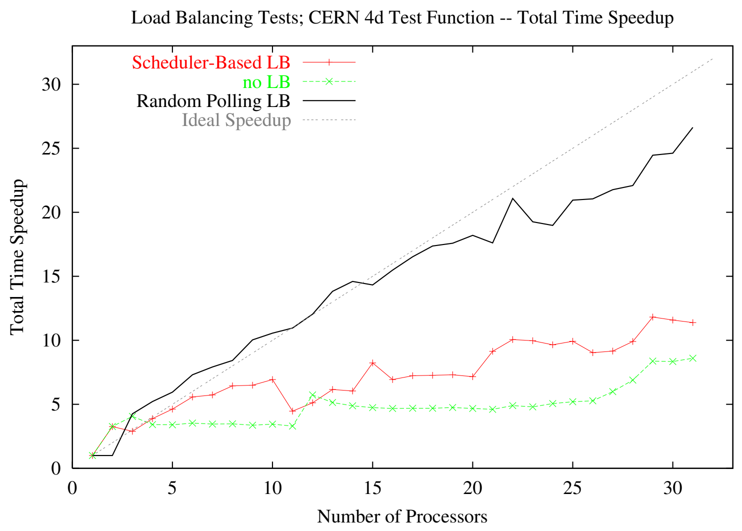


Figure 3: Speedup for cross section integral

which allows for a standard error estimation

$$E_N^2 = \frac{1}{q(q-1)} \sum_{j=1}^{q} (K_N(\beta_j) - \bar{K}_N)^2. \tag{3}$$

The algorithm calculates the $\bar{K}_N$ values of (2) for successively larger numbers N of integration points until either an answer is found to the user-specified accuracy or the function count limit is reached.

We denote by k_{ij} a cell of work $K_{N_i}(\beta_j)$, where $i = 1, \ldots, r_{\max}; j = 1, \ldots, q$, and q is the length of a row. The ith row of cells is $R_i = \{k_{ij} \mid j = 1, \ldots, q\}$. Currently we use $r_{\max} = 28$ rules from [4], with $N_1 = 31, \ldots, N_{28} = 2031713$, and generally $N_i >> N_j$, for $i > j$.

3.2 Asynchronous Distributed Algorithm

Consider using p processes for the computation, one of which is assigned a controlling role and the remaining ones are workers. The controller is responsible for maintaining the overall result and error estimate, while distributing work to the workers. Upon receiving a work result, the controller provides the contributing worker with a new task.

We investigated the following work distribution strategies:

- Single cell work unit. Each worker receives one k_{ij} to compute at a time. Figure 4 illustrates a case with less workers than cells in a row.
- Row slice work unit. Each worker computes a slice of a specific row i, i.e., a slice of each cell k_{ij}, for $j = 1, \ldots, q$ (cf. Figure 5).

It was found [1] that the second strategy is more scalable than the first for a (homogeneous) cluster of workstations. In particular, the first strategy may suffer from considerable breaking loss (which is loss in parallel performance caused by processes doing useless work at the end of the computation). This may occur as the controller flags termination for meeting the accuracy criterion, while some workers were still involved in the computation of larger rule cells; all the work on those cells then has to be deemed useless.

In a homogeneous system, the row slice work splitting strategy divides each row into as many slices as there are workers. An adaptation of this strategy (implemented at PARINT 1.2) for a heterogeneous environment allows splitting each row into considerably more slices than there are workers, so that the faster workers can obtain more work in the same row than the slower ones.

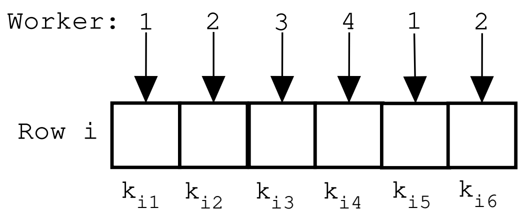


Figure 4: Single Cell work assignment for $p = 4, q = 6$

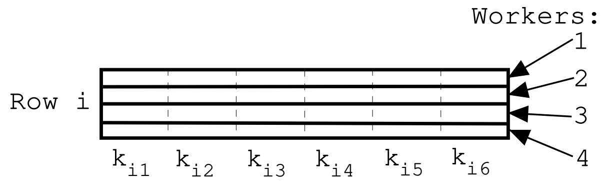


Figure 5: Row Distributed work split for $p = 4, q = 6$

3.3 Applications and performance results

Figure 6 shows the speedup obtained when solving the (360-dimensional) Mortgage Backed Security (MBS) problem from [6, 7] using a homogeneous cluster. The speedup of the row slicing algorithm appears nearly ideal, while the work unit distribution algorithm is inferior in performance. It should be pointed out that the total problem size should be large enough to warrant the parallel approach (e.g., with sequential time of at least a few seconds to a minute on a 1.2GHz processor).

This problem considers a pool of mortgages. With maturity of, say, 30 years, there are 360 cash flows since a cash flow occurs once every month. Each cash flow involves a monthly payment on the underlying pool of mortgages, a certain percentage of pre-paying and a certain interest rate. The amount of pre-paying and the interest rate are considered stochastic variables. The problem consists of evaluating the expected value of the total cash flow (over the 360 months). This leads to a 360-dimensional multivariate normal distribution integral.

Apart from providing parallel QMC as a general-purpose method, PARINT 1.2 also includes special-purpose software based on the QMC paradigm, for the (parallel) computation of multivariate normal and Student-t distribution integrals of the form

$$P(\mathbf{a},\mathbf{b},\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^s}} \int_{\mathbf{a}}^{\mathbf{b}} e^{-\frac{1}{2}\mathbf{x}^\tau \Sigma^{-1}\mathbf{x}} d\mathbf{x} \tag{4}$$

or

$$T(\mathbf{a},\mathbf{b},\Sigma,\nu) = \frac{\Gamma(\frac{\nu+s}{2})}{\Gamma(\frac{\nu}{2})\sqrt{|\Sigma|(\nu\pi)^s}} \int_{\prod_{i=1}^{s}[a_i,b_i]} (1 + \frac{1}{\nu}\mathbf{x}^\tau \Sigma^{-1}\mathbf{x})^{-\frac{\nu+s}{2}}\, d\mathbf{x}, \tag{5}$$

respectively, where Σ is a symmetric positive definite covariance matrix and the integration domain can be finite or infinite. T adheres to the definition of the standardized s-variate t-distribution function with ν degrees of freedom.

These problems are treated via a series of transformations after a Cholesky decomposition of the Σ matrix. The resulting integral is approximated using the sequence of Korobov lattice rules supplied by the QMC code.

4 Some numerical experiments

At the CPP'01 conference it was pointed out (in the talk delivered by F. Yuasa) that PARINT was not able to handle the most difficult cases (for small ε) of the test functions considered by Tobimatsu and Kawabata [11]. As a result, the authors spent considerable efforts to track down the precise nature of the problem. We will use this section to give a

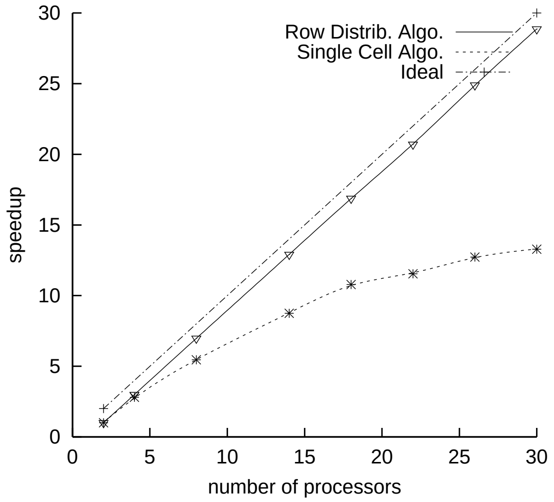


Figure 6: Speedup for Finance Problem

$$\text{DICE1} \qquad \int_0^1 \int_0^1 dx_1 dx_2 \frac{2\varepsilon x_2}{(x_1+x_2-1)^2+\varepsilon^2}$$

$$\text{DICE2} \qquad \int_{-1}^1 \int_{-1}^1 dx_1 dx_2 \frac{\varepsilon x_2^2 \theta(1-x_1^2-x_2^2)}{(x_1^2+x_2^2-a^2)^2+\varepsilon^2}$$

$$\text{DICE3} \qquad \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 dx_1 dx_2 dx_3 \frac{\varepsilon \sqrt{x_1^2+x_2^2+x_1^3}\,\theta(1-x_1^2-x_2^2-x_1^3)}{(x_1^2+x_2^2+x_1^3-a^2)^2+\varepsilon^2}$$

$$\text{DICE3B} \qquad \int_{-1}^1 \int_{-1}^1 dx_1 dx_2 \frac{\varepsilon \sqrt{x_1^2+x_2^2}\,\theta(1-x_1^2-x_2^2)}{(x_1^2+x_2^2-a^2)^2+\varepsilon^2}$$

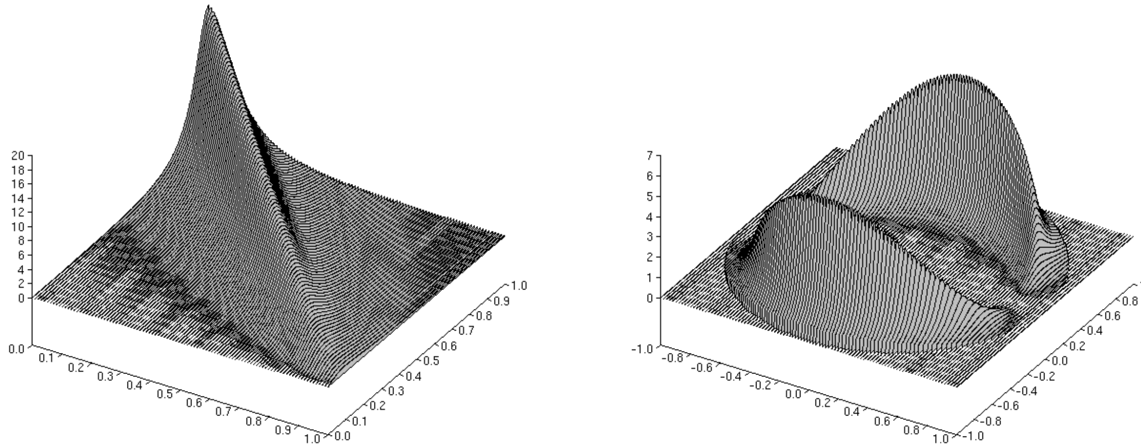
Figure 7: DICE functions

report on our findings, which is by no means complete. Among our group we have been referring to these functions as the "DICE functions", and we will do so here. The integrals listed in Figure 7 include the three problems from [11] (DICE1, DICE2 and DICE3). We have added DICE3B, a 2D analogue of DICE3, because we could visualize it.

Figures 8 and 9 give surface plots for $\varepsilon = 0.1$. DICE1 has a ridge at $x_1 + x_2 = 1$ the height of which is determined by $\frac{2x_2}{\varepsilon}$; the other functions have a ridged behavior along the radius = a value (set to $a = 0.8$), and a discontinuity at radius = 1 (outside of which the function is set to 0). For DICE1, for example, at $(x_1 + x_2 - 1)^2 = \varepsilon^2$, i.e., $x_2 = 1 - x_1 \pm \varepsilon$, the function value is $\frac{x_2}{\varepsilon}$ (half of the corresponding value at the top of the ridge), which gives an indication of the shape and narrowness of the ridge as a function of ε. In that sense the $\varepsilon = 0.1$ value used for the function plots makes the peaks only slightly pronounced.

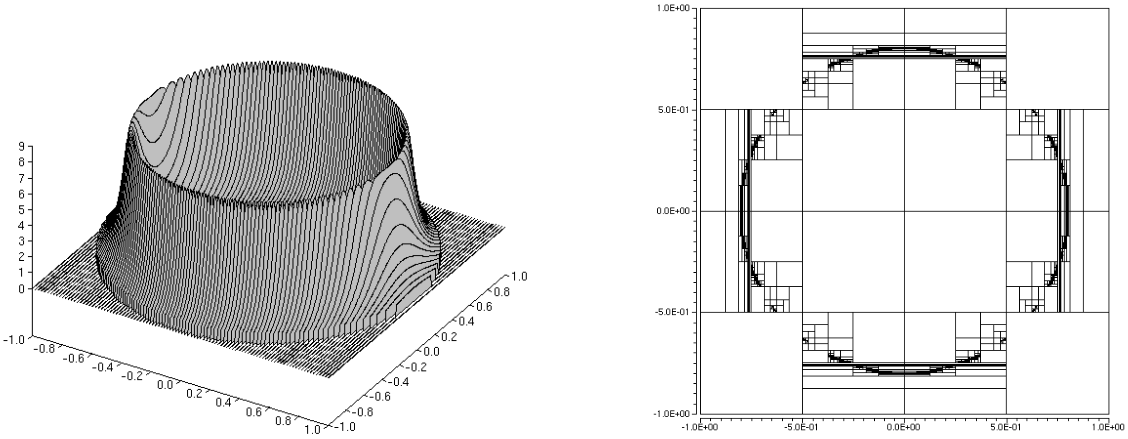
In Figure 10 we give numerical results for the 2D integrals (obtained with an earlier, sequential version of the GA method of PARINT), when requesting a relative accuracy of 1.e-5 within a function evaluation limit of 250M points. Where the error estimate is anotated with a #-sign mark, the algorithm indicated that the requested accuracy could not be reached.

For DICE3 we also list (cf. Figure 11) results of a modified adaptive strategy, RS, which differs from GA in that the subregions are selected according to the value of (their local absolute error estimate) $\times$ (their fraction of the original volume), whereas in GA the selection is based solely on the local absolute error estimate. This type of strategy was considered for 1D automatic quadrature in [9]. For comparison we also include results for DICE3 from a crude Monte-Carlo algorithm with simple antithetic variates (RCRUDE by A. Genz), run to a maximum of 100M $= 10^8$ evaluations.



DICE1 DICE2

Figure 8: Surface plots of the DICE1 and DICE2 functions with $\varepsilon = 0.1$



GA Evaluated Regions

Figure 9: The DICE3B function. Surface plot with $\varepsilon = 0.1$, GA evaluated regions with ε = 1.0e-5

dice1, max = 250M

eps	Exact	GAint	GAer.est
1.e-1	2.4807	2.4807	2.46e-5
1.e-2	3.0295	3.0295	3.03e-5
1.e-3	3.1258	3.1258	3.13e-5
1.e-4	3.1396	3.1396	3.14e-5
1.e-5	3.1413	3.1413	3.14e-5
1.e-6	3.1416	3.1078	2.68e-1#

dice2

eps	Exact	GAint	GAer.est
1.e-1	2.6436	2.6436	2.64e-5
1.e-2	3.1056	3.1056	3.11e-5
1.e-3	3.1530	3.1530	3.15e-5
1.e-4	3.1577	3.1577	3.16e-5
1.e-5	3.1582	3.1582	1.32e-4#
1.e-6	3.1531	3.1452	1.90e-1#

dice3b

eps	Exact	GAint	GAer.est
1.e-1	6.6387	6.6387	6.64e-5
1.e-2	7.7655	7.7655	7.77e-5
1.e-3	7.8826	7.8826	7.88e-5
1.e-4	7.8944	7.8944	7.89e-5
1.e-5	7.8956	7.8956	4.15e-4#
1.e-6	7.8957	7.8820	6.32e-0#

Figure 10: GA results for 2D functions

dice3, GA and RS max = 250M								MC max = 100M	
eps	Exact	GAint	GAer.est	GA#evals	RSint	RSer.est	RS#evals	MCint	MCer.est
1.e-1	10.575	10.574	1.28e-2#	max	10.574	1.42e-2#	max	10.577	9.06e-4
1.e-2	12.422	12.422	1.87e-3#	max	12.422	2.13e-3#	max	12.425	3.31e-3
1.e-3	12.612	12.635	1.78e-0#	max	12.638	2.75e-0#	max	12.616	1.07e-2
1.e-4	12.631	10.664	3.71e+1#	max	12.487	4.96e+1#	max	12.565	3.36e-2
1.e-5	12.633	3.1214	2.03e+1#	max	6.7499	5.49e+1#	max	12.474	1.05e-1
1.e-6	12.633	0.4298	3.65e-0#	max	1.2224	1.56e+1#	max	12.254	3.25e-1

Figure 11: GA, RS and MC results for DICE3

The GA results are poor for the smaller ε values. Detailed investigation involving many test runs and examination of region subdivision plots revealed that, as the adaptive method "discovers" the ridge or discontinuous behavior, and the estimated errors are large in the areas where the problems are hit, other regions which also need subdivision are not or barely handled. As one example of this behavior, Figure 9 shows the regions evaluated by GA for DICE3B with ε = 1.0e-5. Note that the corner regions are not subdivided. The RS strategy, which takes the volume of the subregions into account, produces somewhat better results for DICE3. For the other functions, the RS results generally do not improve on those of GA and use slightly more function evaluations.

The problem apparently benefits a great deal from a uniform distribution of the sample points, as is indicated by the results of RCRUDE for DICE3 (we note that, in comparison, the RCRUDE results are less promising for the other hard test cases at hand). In a future PARINT release we intend to include a simple parallel MC method for cases with an erratic integrand behavior and/or domain. For those cases we will be able to rely on MC, while for smoother cases we supply adaptive and QMC based methods with faster convergence than MC.

In view of the fact that parts of the domain are neglected by the adaptive strategies, we considered subdividing the domain into 4×4 subsquares of equal size at the start and performing the integration over each of those separately, to ensure that each of these subsquares would receive proper attention. Results of this method for are promising. For DICE3B, a result of 0.78957E+01 is obtained for ε = 1.e-6 using GA, with an error estimate of 4.55e-2, within around 300M evaluations; note that the result is accurate to 5 digits (in view of the symmetry, we actually performed the integration over $[0, 1]^2$ only, multiplying the integrand by 4). It appears that, pending some more research, we may be able to improve our strategies for these problems.

"Numerical Recipes" [8] alludes to how difficult these problems are to solve automatically. "If you don't know where the strongly peaked regions are, you might as well (at the level of sophistication of this book) quit: It is hopeless to expect an integration routine to search out unknown pockets of large contribution in a huge N-dimensional space". Although in higher dimensions the level of difficulty will increase enormously, for two and three dimensions we believe that we - as well as the DICE authors have demonstrated good progress for the class of problems at hand.

5 Future directions

We recently started a project which involves building PARINT into a self-contained Problem Solving Environment (PSE), which has the following characteristic properties [10, 12].

- **Problem domain**. PSEs offer an integrated environment for solving specialized problems, including means for composing, compiling and running applications, while handling security issues.
- **Parallel resources**. PSEs provide easy access to distributed computing resources, and state of the art problem solving power.
- **Black box configuration**. PSEs attempt to remove hardware and software complexity, in order to eliminate the need for the end user to be an expert in the solution methods used.
- **Grid distribution**. PSEs can be implemented on top of grids.

- **Visualization**. PSEs need to address the visualization of computations for steering and analysis, and the interpretation of resulting data.

We are currently extending the PARINT user interface to allow for remote specification, compilation and execution of integration problems via the Web. User-submitted integrand functions are checked for unsafe calls, compiled into a dynamic library and loaded by the PARINT executable at run-time.

The new GUI furthermore supports two types of visualization: *(i)* changes in the local error estimate at each processor can be monitored for display at run-ime; *(ii)* detailed information on the region subdivision structure and load balancing in the adaptive algorithm can be logged for post-mortem problem analysis and display via PARINT's companion post-mortem visualization tool, PARVIS [5].

A hierarchical implementation of PARINT, an early version of which was described in [2] and which is being improved for inclusion in a future release, allows grouping the available processes into MPI groups to which sequences of integration problems are assigned (dynamically). Currently each group executes an adaptive integration algorithm, but in general they could run any integration algorithm since the groups are independent.

This approach works well when large sets of (typically parametrized) integrals need to be computed. Furthermore, grid services such as those provided by Globus (http://www.globus.org) can be employed to assign problems to remote resources. Sets of compute-intensive problems are good candidates for this type of distributed computation.

6 Conclusions

The ParInt project focuses on the development of parallel algorithms and software for the *automatic* computation of multivariate integrals. We gave a brief overview of the current status and capabilities of the package. Our objective has been to provide the end-user with state of the art problem solving power. This has required work in a number of areas, including the fundamental numerical techniques, strategies for parallelization, user interfaces for general use and specific applications, and visualization of computations. We hope that our efforts will be beneficial, and that we may be able to fine-tune our software for success with your applications.

References

[1] CUCOS, L., AND DE DONCKER, E. Distributed qmc algorithms: New strategies and performance evaluation. In *Proceedings of the High Performance Computing Symposium (HPC'02)* (2002). To appear.

[2] DE DONCKER, E., GUPTA, A., AND ZANNY, R. Large scale parallel numerical integration. *Journal of Computational and Applied Mathematics 112* (1999), 29–44.

[3] DE DONCKER, E., ZANNY, R., CIOBANU, M., AND GUAN, Y. Distributed Quasi Monte-Carlo methods in a heterogeneous environment. In *Proceedings of the IPDPS Heterogeneous Computing Workshop* (2000), pp. 200–206.

[4] GENZ, A. MVNDST: Software for the numerical computation of multivariate normal probabilities, 1998. Available from web page at http: //www.sci.wsu.edu/math/faculty/genz/homepage.

[5] KAUGARS, K., DE DONCKER, E., AND ZANNY, R. PARVIS: Visualizing distributed dynamic partitioning algorithms. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00)* (2000), pp. 1215–1221.

[6] NINOMYA, S., AND TEZUKA, S. Toward real-time pricing of complex financial derivatives. *Applied Mathematical Finance 3* (1996), 1–20.

[7] PASKOV, S. H. Computing high dimensional integrals with applications to finance. Tech. rep., Columbia University, Department of Computer Science, 1994. CUCS-023-94.

[8] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 1991.

[9] SHAPIRO, H. D. Increasing robustness in global adaptive quadrature through interval selection heuristics. *ACM Transactions on Mathematical Software 10*, 2 (1984), 117–139.

[10] SHIELDS, M. S., RANA, O. F., WALKER, D. W., LI, M., AND GOLBY, D. A java/CORBA-based visual program composition environment for PSEs. *Concurrency: Practice and Experience 12* (2000), 687–704.

[11] TOBIMATSU, K., AND KAWABATA, S. Multi-dimensional integration routine DICE. Tech. Rep. 85, Kogakuin University, 1998.

[12] WALKER, D., RANA, O. F., LI, M., SHIELDS, M. S., AND HUANG, Y. The software architecture of a distributed problem-solving environment. Tech. rep., Computer Science and Mathematics Division, Oak Ridge National Laboratory, February 2000.