INFORMATICA, 1993, Vol.4, No.1–2, 227–249

NUMERICAL INVERSION OF MULTIVARIATE LAPLACE TRANSFORMS USING A PARALLEL SYSTEM

Peter ZINTERHOF and Markus SALCHEGGER

Research Institute for Software Technology
University of Salzburg
5020 Salzburg, Hellbrunner St. 34, Austria

Abstract. This work was stimulated by investigations on Markow Renewal Processes. For finding analytic solutions (to compute the probabilities of certain states of the system) multivariate Laplace transforms can be used. Tables with correspondences of function and their transforms very rarely help to solve such problems.

In Chapter I number theoretical numerics are applied to compute the original function of a multivariate Laplace transform given. Starting with the complex multivariate inversion theorem the domain of integration is mapped onto the s-dimensional unit cube G_s. Using a periodization of the integrand new results concerning the vanishing of the multivariate Laplace transform in regard of the modified numerical inversion formula are shown.

In Chapter II two implementations are discussed: A method to implement a Manager-Worker Process (MWP) to reduce the idle times of the processors is presented and the tasks of the Manager and the Workers are defined. The numerical inversion using this method with STRAND^{88} has been implemented on a heterogenous workstation net. The MWP provided a good load balancing. Another implementation with C-LINDA has been done on a Shared Memory MIMD system. We also implemented a kind of MWP. Numerical experiments have shown that the decomposition of the problem is sufficiently.

Key words: Laplace transform, numerical integration, good lattice points, parallel computation.

Chapter I

Introduction. This work was stimulated by investigations on Markow Renewal Processes (MRP). An MRP is described by a matrix-valued integral equation on several variables.

A possible question concerning such a process could be:

"What's the probability to rest t seconds in the state j if the system first was in the state i
($G(i,j,t) = P(T_{n+1} - T_n \leqslant t | X_n = i \wedge X_{n+1} = j)$) ?"

For finding analytic solutions we can use multivariate Laplace transforms. The inversion of such transforms can be difficult. Because tables with correspondences of functions and their transforms very rarely help to solve such problems.

The basic idea. Starting with the complex multivariate inversion theorem we map by means of conformal mappings the domain of integration onto the s-dimensional unit cube G_s. To meet the requirements for the utilization of numerical integration by number theoretical methods (e.g. good lattice points, optimal coefficiants) and to remove certain singularities of the integrand, we first have to apply either an algebraic periodization or a trigonometric Beta periodization. There are estimations concerning the deviation which consider either the differentiability and the vanishing of the Laplace transform as the method of good lattice points.

1. Numerical integration.

DEFINITION 1. Let G_s be the s-dimensional unit cube. A function $f \in E_\alpha^s$ if $f : G_s \to \mathbf{R}$ continuous, $f(X) = f(X+Z)$ $\forall Z \in \mathbf{Z}^s$ and

$$c(m_1,\dots,m_s) = O\left(\frac{1}{(\bar{m}_1 \cdots \bar{m}_s)^\alpha}\right). \tag{1}$$

REMARK 1.

$$c(m_1,\dots,m_s) = \int_{G_s} f(x_1,\dots,x_s)e^{-2\pi<M,X>}\, dx_1 \cdots dx_s \tag{2}$$

are the Fourier coefficients of $f(X)$, $\alpha > 1$ and $\bar{m} = \max(|m|,1)$.

Theorem 1. *$f \in E_\alpha^s$ and let $a_1,\dots,a_s \in \mathbf{Z}$ be optimal coeffi-*

cients modulo N then

$$R_N = \frac{1}{N}\sum_{k=1}^{N} f\left(\left\{\frac{a_1 k}{N}\right\},\dots,\left\{\frac{a_s k}{N}\right\}\right) - \int_{G_s} f(x_1,\dots,x_s)\,dx_1\cdots dx_s = O\left(\frac{\ln^{\alpha s} N}{N^\alpha}\right), \tag{3}$$

$\{x\} = x - [x]$ is the fractional part of x.

Korobov [5] proved the existence of optimal coefficients for all primes. This leads to the constuction of a so called optimal parallelepipedal lattice.

$$X_k = \left(\left\{\frac{a_1 k}{N}\right\},\dots,\left\{\frac{a_s k}{N}\right\}\right) \qquad k \in \{1,\dots,N\}.$$

H. Niederreiter [6] proofed the existence of good lattice points for all $N \geqslant 2$ and for all dimensions.

To meet the requirements of theorem 1 we must be sure that $f \in E_\alpha^s$.

In particular $f(X+Z) = f(X) \quad \forall Z \in \mathbf{Z}^s$ must be guaranteed.

DEFINITION 2. Let $\frac{\partial^{\alpha s} f}{\partial x_1^\alpha \cdots \partial x_s^\alpha}$ be continuous ($\leftrightarrow f \in H_\alpha^s$) and $\alpha \geqslant 2$.

The finding of a function $\Phi(x_1,\dots,x_s) \in H_\alpha^s$ which meets

$$\Phi(x_1,\dots,x_{\nu-1},0,x_{\nu+1},\dots,x_s) = \Phi(x_1,\dots,x_{\nu-1},1,x_{\nu+1},\dots,x_s), \tag{4}$$

$$\left(\frac{\partial^n \Phi}{\partial x_\nu^n}\right)_{x_\nu=0} = \left(\frac{\partial^n \Phi}{\partial x_\nu^n}\right)_{x_\nu=1} \qquad n = 1,\dots,\alpha-2, \tag{5}$$

$$\int_{G_s} f(x_1,\dots,x_s)\,dx_1\cdots dx_s = \int_{G_s} \Phi(x_1,\dots,x_s)\,dx_1\cdots dx_s \tag{6}$$

is called complete periodization.

DEFINITION 3. The function $\Psi : [0,1] \to [0,1]$, $\Psi(\tau) = x$ is called a periodization function of order α if

$$
\begin{aligned}
&(a)\ \Psi'(\tau) > 0 \qquad \forall \tau \in (0,1),\\
&(b)\ \bullet\ \Psi(0) = 0,\\
&\quad\ \ \bullet\ \Psi(1) = 1,\\
&(c)\ \Psi^{(n)}(0) = \Psi^{(n)}(1) = 0 \qquad \forall n \in \{1,\dots,\alpha\}.
\end{aligned} \tag{7}
$$

Theorem 2. *Let $\Psi(\tau_\nu) = x_\nu,\ 1 \leqslant \nu \leqslant s$ be a periodization function of order α and $D_\alpha(X) \overset{\text{def}}{=} \frac{\partial^{\alpha s} f}{\partial x_1^\alpha \cdots \partial x_s^\alpha}$ be continuous with $|D_\alpha(X)| \leqslant C \quad \forall X \in G_s$ then*

$$f(\Psi(\tau_1),\dots,\Psi(\tau_s))\Psi'(\tau_1)\cdots\Psi'(\tau_s) \in E_\alpha^s \tag{8}$$

and

$$
\begin{aligned}
&\int\limits_{G_s} f(x_1,\dots,x_s)\,dx_1\cdots dx_s\\
&\quad = \int\limits_{G_s} f(\Psi(\tau_1),\dots,\Psi(\tau_s))\Psi'(\tau_1)\cdots\Psi'(\tau_s)\,d\tau_1\cdots d\tau_s.
\end{aligned} \tag{9}
$$

Proof. Let $\Phi(X) = f(\Psi(\tau_1),\dots,\Psi(\tau_s))\Psi'(\tau_1)\cdots\Psi'(\tau_s)$.

We consider the Fourier series

$$\Phi(X) = \sum_{M \in \mathbb{Z}^s} c(M) e^{2\pi i <M,X>}$$

with the Fourier coefficients $\int\limits_{G_s} \Phi(X) e^{-2\pi i <M,X>}\,dx_1\cdots dx_s$.

Now we perform an α-times partial integration with respect to all s coordinates.

As $\Psi(\tau_\nu) = x_\nu$ are periodization functions of order α :

$$\big(\psi(\tau_\nu)\big)_{\tau_\nu=0} = \big(\psi(\tau_\nu)\big)_{\tau_\nu=1} = 0 \qquad \nu = 1,\dots,\alpha,$$

$$|c(M)| = \left| \frac{1}{(2\pi i)^{\alpha s}(m_1\cdots m_s)^\alpha} \int\limits_{G_s} \frac{\partial^{\alpha s}\Phi(X)}{\partial x_1^\alpha \cdots \partial x_s^\alpha} e^{-2\pi i <M,X>}\,dx_1\cdots dx_s \right.$$

$$|c(M)| < \frac{1}{(\bar{m}_1 \cdots \bar{m}_s)^{\alpha}} \int\limits_{G_s} |D_{\alpha}(X)|\, dx_1 \cdots dx_s,$$

$$|c(M)| \leqslant \frac{C}{(\bar{m}_1 \cdots \bar{m}_s)^{\alpha}} \rightarrow \Phi \in E_{\alpha}^{s}.$$

Definition 4.

$$\Psi(\tau) = B(\alpha, \beta, \tau) = \frac{\Gamma(\alpha + \beta + 2)}{\Gamma(\alpha + 1)\Gamma(\beta + 1)} \int\limits_{0}^{\tau} t^{\alpha}(1 - t)^{\beta}\, dt \tag{10}$$

is called algebraic Beta periodization of order (α, β) for $\alpha, \beta > -1$.

Definition 5.

$$\Psi(\tau) = B^{*}(\alpha, \beta, \tau) = \frac{1}{B(\frac{\alpha+1}{2}, \frac{\beta+1}{2})} \int\limits_{0}^{\tau} \sin^{\alpha} \frac{\pi t}{2} \cos^{\beta} \frac{\pi t}{2}\, dt \tag{11}$$

is called trigonometric Beta periodization of order (α, β) for $\alpha, \beta > -1$.

It is obvious that we can enforce the vanishing of the function to be periodized by choosing the parameters α and β.

In particular $B(k, k, \tau)$ for $k \in \mathbb{N}$ is a polynomial of degree $2k+1$.

Remark 2. Both periodizations are closely related. This can be seen if we make the following substitution in $B(\alpha, \beta, \tau)$:

$$\begin{aligned} t &= \cos^2 \frac{\pi u}{2}, \\ dt &= -\pi \cos \frac{\pi u}{2} \sin \frac{\pi u}{2}\, du. \end{aligned} \tag{12}$$

Although both periodizations are closely related, the trigonometric Beta periodization is in many cases preferable. It is sometimes possible to remove certain singularities of the integrand by means of the trigonometric Beta periodization.

2. Numerical inversion of $\mathcal{L}^s\{F\}$.

DEFINITION 6.

$$Q_s \overset{\text{def}}{=} \{T \in \mathbf{R}^s | 0 \leqslant t_k < \infty \wedge 1 \leqslant k \leqslant s\}. \tag{13}$$

DEFINITION 7.

$$\mathcal{L}^s\{F\} = f(z_1, \ldots, z_s) = \int\limits_{Q_s} e^{-<Z,T>} F(T)\, dt_1 \cdots dt_s \tag{14}$$

is called s-dimensional Laplace transform.

Theorem 3. *Let $e^{-<Z,T>}F(T)$ be integrable over Q_s in the sense of Lebesgue for a vector $Z_0 = (z_1^0, \ldots, z_s^0) \in \mathbf{C}^s$ and let $f(Z_0)$ converge absolutely then*

$$\mathcal{L}^s\{F\} = f(z_1, \ldots, z_s) = \int\limits_{Q_s} e^{-<Z,T>} F(T)\, dt_1 \cdots dt_s \tag{15}$$

converges in the complex half planes $\Re(z_k) \geqslant \Re(z_k^0), 1 \leqslant k \leqslant s$.

Theorem 4. *Suppose the multivariate Laplace transform $\mathcal{L}^s\{F\} = f(Z)$ exists for $\Re(z_k) > 0$ $1 \leqslant k \leqslant s$ and let $F(T)$ have continuous partial derivatives of order one at least then*

- *$f(Z)$ is a holomorphic function in the domain of convergence and*
- *$f(Z)$ can be inverted with the well known inversion formula.*

$$F(T) = \frac{1}{(2\pi i)^s} \int\limits_{z_1=x_1-i\infty}^{z_1=x_1+i\infty} \cdots \int\limits_{z_s=x_s-i\infty}^{z_s=x_s+i\infty} e^{<Z,T>} f(Z) dz_1 \cdots dz_s. \tag{16}$$

Now we want to apply the method of good latice points to the mulitivariate inversion formula. Let

$$z(w) = \frac{1+w}{1-w} \tag{17}$$

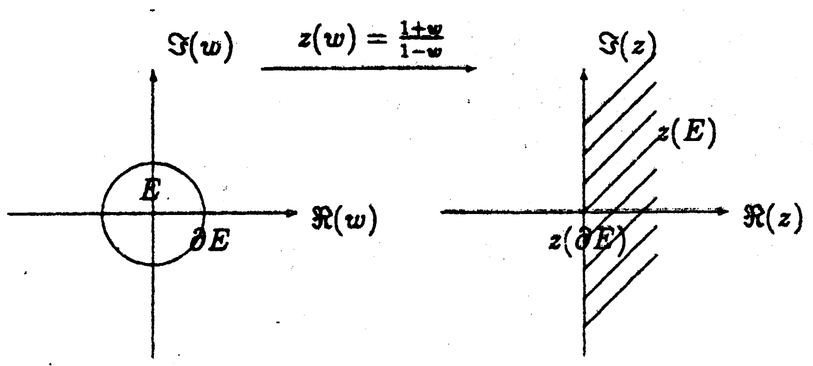

Fig. 1. Conformal mapping.

be the conformal mapping which transforms the unit disk $E = \{w \in \mathbf{C} : |w| \leqslant 1\}$ onto the right half plane $\Re(z) \geqslant 0$ (Fig. 1.).

The straight line $\Re(z) = 0$ corresponds to the circle

$$w = \frac{x}{1+x} + \frac{e^{-i\phi}}{1+x} \qquad 0 \leqslant \phi \leqslant 2\pi. \tag{18}$$

We compute straightforward

$$z(w) = x - i(1+x)\frac{\sin\phi}{1-\cos\phi} = x - i(1+x)\cot\frac{\phi}{2}. \tag{19}$$

As the Laplace transform is holomorphic in the right half plane, the integral does not depend on the path of integration. We just have to choose $x > 0$.

$$\frac{dz}{d\phi} = i\frac{1+x}{2\sin^2\frac{\phi}{2}}. \tag{20}$$

We substitute for each coordinate and get straightforward

$$F(T) = \frac{1}{(2\pi i)^s} \int\limits_{z_1=x_1-i\infty}^{z_1=x_1+i\infty} \cdots \int\limits_{z_s=x_s-i\infty}^{z_s=x_s+i\infty} e^{<Z,T>} f(Z) dz_1 \cdots dz_s,$$

$$F(T) = \frac{1}{(2\pi)^s} \int\limits_0^{2\pi} \cdots \int\limits_0^{2\pi} \left(\prod_{k=1}^{s} e^{t_k(x_k - i(1+x_k)\cot\frac{\phi_k}{2}} \frac{1+x_k}{2\sin^s\frac{\phi_k}{2}} \right) \times$$

$$\times f\Big(x_1 - i(1+x_1)\cot\frac{\phi_1}{2},\ldots,x_s$$
$$- i(1+x_s)\cot\frac{\phi_s}{2}\Big)\,d\phi_1\cdots d\phi_s. \qquad (21)$$

Let Ψ be a periodization of order α and let $0 < \epsilon < 1$. We define:

$$\begin{aligned}\phi_k &= 2\pi\Big(\frac{\epsilon}{2} + (1-\epsilon)\Psi(\tau_k)\Big),\\ d\phi_k &= 2\pi(1-\epsilon)\Psi'(\tau_k)d\tau_k,\end{aligned} \qquad (22)$$

e.g.,: $x = \frac{1}{2}$ and $0 < \epsilon < 1$ (Fig. 2.).

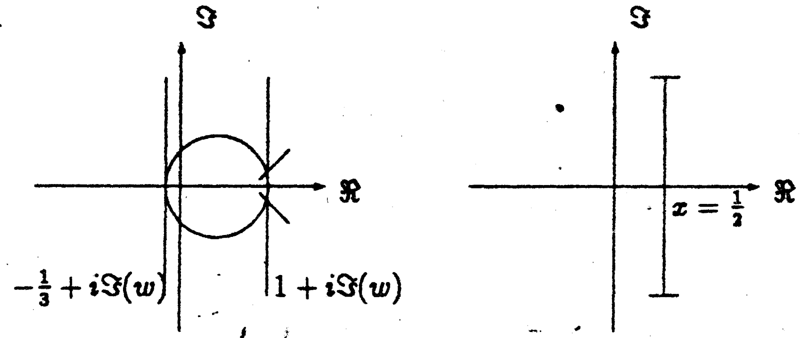

Fig. 2. Example.

Let $Z^* = (x_1 - i(1+x_1)\cot\frac{\phi_1}{2},\ldots,x_s - i(1+x_s)\cot\frac{\phi_s}{2})$ and $\phi_k = 2\pi(\frac{\epsilon}{2} + (1-\epsilon)\Psi(\tau_k))$.

If we apply that transformation we compute straightforward

$$F_\epsilon(T) = (1-\epsilon)^s \int_{G_s} e^{<T,Z^*>} f(Z^*)$$
$$\times \prod_{k=1}^{s} \frac{(1+x_k)\Psi'(\tau_k)}{2\sin^2\Big(\pi(\frac{\epsilon}{2} + (1-\epsilon)\Phi(\tau_k))\Big)} d\tau_k. \qquad (23)$$

Obviously holds

$$\lim_{\epsilon\to 0} F_\epsilon(T) = F_0(T) = F(T). \qquad (24)$$

Because of the sine in the denominator of (23) we got singularities of order 2 for $\tau_k = 0$ and $\tau_k = 1$ if $\epsilon = 0$.

These singularities can be removed by choosing a periodization of order > 2.

DEFINITION 8. We denote the integrand of the integral above:

$$H(\tau_1,\ldots,\tau_s) \overset{\text{def}}{=} e^{<T,x^*>} f(Z^*) \prod_{k=1}^{s} \frac{(1+x_k)\Psi'(\tau_k)}{2\sin^2\left(\pi\left(\frac{\epsilon}{2}+(1-\epsilon)\Phi(\tau_k)\right)\right)}. \qquad (25)$$

Theorem 5. *If $\Psi(\tau_k)$ are periodization functions of order $2\alpha+2$, $1 \leqslant k \leqslant s$ then $H(\tau_1,\ldots,\tau_s) \in E^s_\alpha$.*

Proof. Let

$$H(\tau_1,\ldots,\tau_s) = \sum_{m_1,\ldots,m_s=-\infty}^{\infty} c(m_1,\ldots,m_s) e^{2\pi i(m_1\tau_1+\cdots+m_s\tau_s)} \qquad (26)$$

be the Fourier series of $H(\tau_1,\ldots,\tau_s)$ with the Fourier coefficients

$$c(m_1,\ldots,m_s) = \int\limits_{G_s} H(\tau_1,\ldots,\tau_s) e^{-2\pi i(m_1\tau_1+\cdots+m_s\tau_s)}\, d\tau_1\cdots d\tau_s. \qquad (27)$$

Now we perform an α -times partial integration with respect to all s coordinates. (as shown in the proof of theorem 2) As $\Psi(\tau_k)$ are periodization functions we get straightforward

$$|c(m_1,\ldots,m_s)| = \left| \frac{1}{(2\pi i)^{\alpha s}(m_1\cdots m_s)^\alpha} \int\limits_{G_s} \frac{\partial^{\alpha s} H(\tau_1,\ldots,\tau_s)}{\partial^\alpha\tau_1\cdots\partial^\alpha\tau_s} \times e^{-2\pi i(m_1\tau_1+\cdots+m_s\tau_s)}\, d\tau_1\cdots d\tau_s \right|,$$

$$|c(m_1,\ldots,m_s)| < \frac{1}{(\bar{m}_1\cdots\bar{m}_s)^\alpha} \int\limits_{G_s} \left| \frac{\partial^{\alpha s} H(\tau_1,\ldots,\tau_s)}{\partial^\alpha\tau_1\cdots\partial^\alpha\tau_s} \right| d\tau_1\cdots d\tau_s. \qquad (28)$$

$f(Z^*)$ and all its partial derivatives are bounded by a suitable constant $K = K(f, x_1, \dots, x_s, \alpha)$.

$$\left|c(m_1, \dots, m_s)\right| \leqslant \frac{CK(f, x_1, \dots, x_s, \alpha)}{(\bar{m}_1 \cdots \bar{m}_s)^\alpha}. \tag{29}$$

The constant C does not depend on ϵ because the order of periodization we have chosen is $2\alpha + 2$.

DEFINITION 9. $F(T) \in D^s_\beta(C_1)$ if

- $\beta > 1$ and
- $$\left|\mathcal{L}^s\{F\}\right| = \left|f(Z)\right| \leqslant \frac{C_1}{\left(|z_1| \cdots |z_s|\right)^\beta}. \tag{30}$$

Theorem 6. *If $f(Z) \in D^s_\beta(C_1)$ then*

$$\left|F_\epsilon(T) - F(T)\right| = O\left(\epsilon^{\beta-1}\right). \tag{31}$$

Proof. We consider the following set:
$A = \{Z \in \mathbb{C}^s \,|\, \text{there is at least one coordinate with } |y_k| > \cot(\pi\epsilon)\}$.

$$\left|F_\epsilon(T) - F(T)\right| = \left|\frac{1}{(2\pi i)^s} \int_A e^{\langle Z,T \rangle} f(Z)\, dz_1 \cdots dz_s\right|. \tag{32}$$

As $f(Z) \in D^s_\beta(C_1)$ we get immediately the assertion of (31).

Now we actually want to concern with the numerical integration of the function $H(\tau_1, \dots, \tau_s)$.

Let

$$(\tau_1^k, \dots, \tau_s^k) \quad k \in \{1, \dots, N\} \tag{33}$$

be a set of nodes in the s-dimensional unit cube G_s. And let

$$R_N = \frac{1}{N} \sum_{k=1}^{N} H(\tau_1^k, \dots, \tau_s^k) - \int_{G_s} H(\tau_1, \dots, \tau_s)\, d\tau_1 \cdots d\tau_s,$$
$$R_N = I_N(H) - I(H). \tag{34}$$

Theorem 7 [5] [9b]. *Let $f(Z) \in D^s_\alpha(C_1)$. We compaire the classical method and the method of good lattice points:*

- *Using the method of good lattice points (optimal coefficients) the estimation*

$$F(T) - \frac{1}{N}\sum_{k=1}^{N} H(\tau_1^k,\dots,\tau_s^k) = O\left(\frac{\ln^{\alpha\beta} N}{N^\alpha}\right) \quad \beta \leqslant s \tag{35}$$

  holds.
- *Using the cartesian product rule the estimation*

$$F(T) - \frac{1}{N}\sum_{k=1}^{N} H(\tau_1^k,\dots,\tau_s^k) = O\left(N^{-\frac{\alpha}{s}}\right) \tag{36}$$

  holds.

Proof. Let $a_1,\dots,a_s$ be optimal coefficients modulo N. We first consider the optimal parallelepipedal lattice:

$$(\tau_1^k,\dots,\tau_s^k) = \left(\left\{\frac{a_1 k}{N}\right\},\dots,\left\{\frac{a_s k}{N}\right\}\right) \qquad k \in \{1,\dots,N\}. \tag{37}$$

As $H(\tau_1,\dots,\tau_s) \in E^s_\alpha(CK)$ (29), we take use of Theorem 1 and get straightforward

$$R_N(H) = O\left(\frac{\ln^{\alpha\beta} N}{N^\alpha}\right), \qquad \beta \leqslant s, \tag{38}$$

$$\begin{aligned} F(T) - \frac{1}{N}\sum_{k=1}^{N} H(\tau_1^k,\dots,\tau_s^k) &= F(T) - I_N(H) \\ &= F(T) - F_\epsilon(T) + I(H) - I_N(H). \end{aligned} \tag{39}$$

According to Theorem 6 ($f(Z) \in D^s_\alpha(C_1)$) we compute

$$F(T) - I_N(H) = O\left(\epsilon^{\beta-1} + \frac{\ln^{\alpha\beta} N}{N^\alpha}\right). \tag{40}$$

By choosing ϵ of the right size (e.g.: ϵ of the order $N^{-\frac{\alpha}{\beta-1}}$) the first assertion follows.

If we use the classical cartesian rule for periodic functions we need $N = n^s$ lattice points to perform the numerical integration.

$$(\tau_1^k, \dots, \tau_s^k) = \left(\frac{k_1}{n}, \dots, \frac{k_s}{n}\right) \qquad k_l \in \{1, \dots, n\}, \tag{41}$$

$$I(H) = \frac{1}{n^s} \sum_{k_1=1}^{n} \cdots \sum_{k_s=1}^{n} H\left(\frac{k_1}{n}, \dots, \frac{k_s}{n}\right) - R_N. \tag{42}$$

As $H \in E_\alpha^s$ the best possible result we can get (Korobov [5]) is

$$R_N(H) = I_N(H) - I(H) = O\left(\frac{1}{N^{\frac{\alpha}{s}}}\right). \tag{43}$$

Using the first assertion und choosing ϵ of the order $N^{-\frac{\alpha}{s(\beta-1)}}$ we can complete our proof.

Chapter II

Introduction. A numerical algorithm using all the ideas of Chapter I is very expensive in compution time. A high dimensional problem can increase the computation time enormously. It is quite impossible to get satisfying results at reasonable computing times using a usual single processor system. As the algorithm cannot be sufficiently vectorized (multiple exits, function interrupts a.s.o.) we want to implement it using a parallel environment. We want to discuss two different models:

- The STRAND^{88} model (on a virtual distributed system).
- The LINDA model (on a Shared Memory MIMD system).

1. Implementation with STRAND^{88}. We want to use a heterogenous network of workstations. STRAND^{88} is a programming language which supports the distribution of the several processes.

Something about STRAND^{88}. The programming language STRAND^{88} has been developped by the enterprise "Artificial Intelligence Ltd." with support of experts in logic and parallelism. STRAND^{88} is an abbrevation of STReaming AND parallelism. STRAND^{88} is a logical declarative programming language which supports two distinct program design methodologies. Modular decomposition and stepwise refinement. Large problems are initially decomposed into modules with carefully defined interfaces. Each module includes an interface definition that lists the functions it provides, or exports, to other modules in the system. Modules are sufficiently self-contained that they can be reused for other applications.
With the symbol '*:-*' we can seperate a problem from its decomposition into subproblems.
e.g.:

$$\begin{aligned}&big\_problem() :- \\ &\qquad subproblem_1(), \\ &\qquad \cdot \\ &\qquad \cdot \\ &\qquad \cdot \\ &\qquad subproblem_n(). \end{aligned} \tag{44}$$

Each of the subproblems in the above formalization may be considered the responsability of an independent entity that we shall refer to as a process. The processes can be executed in any order or in parallel. Synchronization can be done by adding a communication channel between the corresponding processes.
e.g.:

$$\begin{aligned}&big\_problem() :- \\ &\qquad subproblem_1(Done), \\ &\qquad subproblem_2(Done), \\ &\qquad \cdot \\ &\qquad \cdot \end{aligned}$$

$$subproblem_n(). \qquad (45)$$

The process $subproblem_2$ would wait to receive a message from $subproblem_1$ before performing its designated task.

By using the other statement '@' we can run processes on other computers.
e.g.:

$$\begin{aligned} &big\_problem() :- \\ &\qquad subproblem_1()@Node_i, \\ &\qquad subproblem_2()@Node_j. \qquad (46)\end{aligned}$$

The process $subproblem_1()$ will run on $Node_i$ and the process $subproblem_2()$ will run on $Node_j$. In this way we can spawn processes on our network.

As there are many possible ways to connect computers, STRAND^{88} provides a collection of virtual machines that are abstractions of the physical hardware. This provides the simulation of distributed systems. If the defined topology fits to the real available hardware the computations will be done in parallel.

As a high-level programming notation STRAND^{88} provides the ability to rapidly prototype programs. It is also possible to reuse existing code segments in C or FORTRAN. By defining an interface file which manages the communication between STRAND^{88} and the foreign code library whole processes can be written in C or FORTRAN (Fig. 3.).

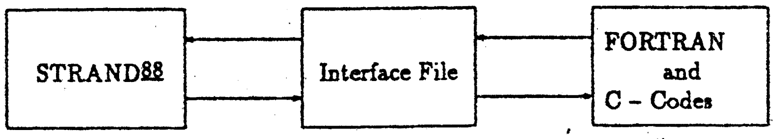

Fig. 3. Foreign Language Interface.

Computation in STRAND^{88} is performed by a set of interaction processes. Each process can be represented by a term of the form:

$$p(T_1,\ldots,T_n) \qquad n \geqslant 0, \tag{47}$$

where p/n identifies the program used to execute the process and $T_1,\ldots,T_n$ are the process arguments. The arguments are data structures (terms) that comprise the process state. A STRAND^{88} program describes the actions that processes may perform. There are just three types of actions:

- terminate,
- change state,
- fork.

Programs are composed of a set of rules. each of which describes a single action. Rules have the following general form:

$$H :- G_1,\ldots,G_m | B_1,\ldots,B_n. \quad m,n \geqslant 0, \tag{48}$$

where H is the rule head, $G_1,\ldots,G_s$ are the rule guards and $B_1,\ldots,$ B_n are the rule body.

More details on STRAND^{88} can be read in the book of Ian FOSTER [4].

Implementation. Suppose we had a heterogenous network system which consists of $n+1$ Nodes. (e.g. $n+1$ workstations).

Let $A = [a_1,b_1]$ X $\cdots$ X $[a_s,b_s]$ be the domain where we want to compute the original function $F(T) = F(t_1,\ldots,t_s)$. We suppose that we want to compute $F(T)$ for L points in A.

There exists an algorithm so that each number $k \in \{1,\ldots,L\}$ is unequivocal mapped on a tuple $(k_1,\ldots,k_s)$ representing the coordinates of the k-th point of inversion. This works if we use a presentation of the number k to the base number p, where p is the number of points in each dimension.

At each point $(k_1,\ldots,k_s)$ we perform a numerical integration (with all transforms and periodizations) using the method of good lattice points. The optimal coefficients, we need for that, have been obtained before and are stored in a file.

The inversion process itself is written in the C-programming language, and all other transforms, periodizations and procedures are available in user-defined C-libraries.

Now we want to start a process which is called **Manager-Worker Process (MWP).**

The basic idea (Fig. 4.). The manager is responsible for partitioning a problem into subproblems and allocating these to workers. The workers are responsible for solving a single subproblem and requesting additional work from the manager when they become idle. To achieve this balancing functionality, we define a balancing process that receives a list of subproblems and a stream of request from the workers. Let $Node_0$ be the node where we let the manager program run. Let $Node_1, \ldots, Node_n$ be the workers. Here actually the hard integration work will be done.

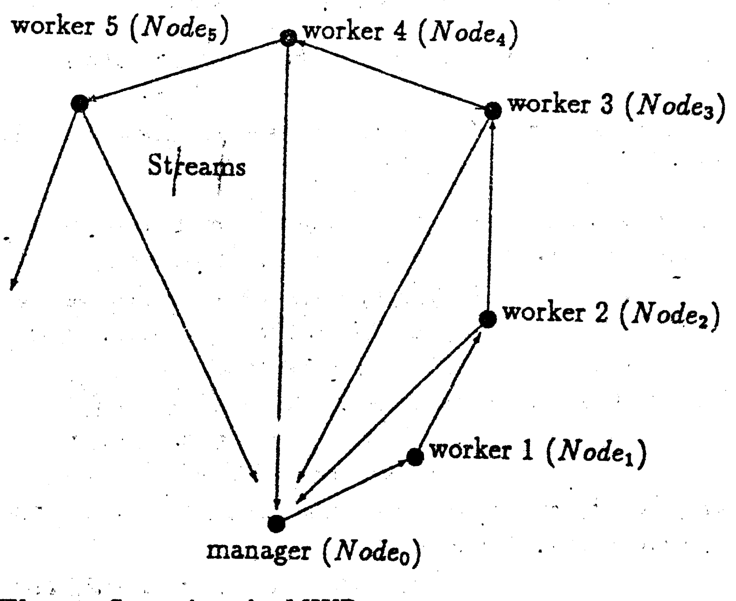

Fig. 4. Spawning the MWP structure.

```
laplace_inversion(ParameterList, DistributionInfo) :-
    manager(ParameterList, DistributionInfo, Requests),
    merger(Streams, Requests)
    init_workers(DistributionInfo, Streams).                (49)
```

```
manager(ParameterList, DistributionInfo, Requests) :-
    partition(ParameterList, DistributionInfo, Subproblems),
    balance(SubProblems, Requests).                         (50)
```

```
init_workers(DistributionInfo, [merge(Request)|Rs]) :-
    $n > 0$|
        $n = n - 1$,
        worker(Go, Request)@Node$_n$,
        init_workers(ModifiedDistInfo, Rs).
init_workers(DistributionInfo, []) :-
    $n = 0$|
        terminate.                                          (51)
```

The prococess *worker*() computes the original function $F(T)$ by changing into a C process or a FORTRAN process. The result is sent back to the manager.

The job of the manager:

- partition the problem,
- engage n workers,
- send them a task (e.g.: compute $F(T)$ for just a several number of points),
- receive the signal if a worker is ready,
- identify the worker,

- send a new task to the idle worker,
- send a stop signal to all workers if there are no more tasks,
- terminate when all workers are idle and all tasks are dispatched.

The job of the workers:

- request a task,
- do the work,
- send back the result to the manager,
- request a new task,
- terminate when the stop signal arrives.

If we use the MWP we should be able to minimize the idle times of the workers.

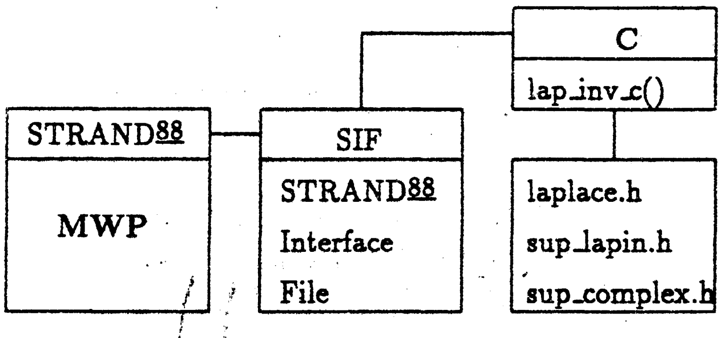

Fig. 5. Stucture of the implementation.

Experiments on a heterogenous network were very satisfactory. By the means of a profiling tool we could check the minimization of the idle times.

Furthermore we implemented the MWP on a homogenous system and could obtain a quite linear speed-up, which indicates a well parallelized algorithm.

2. Implementation with C-LINDA. C-LINDA is a realization of Linda that coordinates with the C programming language. We can write programs in C and make use of the operations in C-LINDA to create and coordinate multiple processes as required.

There are four basic tuple space (an allocated associative object memory) operations in C-LINDA:

- *out* (add a tuple to the tuple space (TS)),
- *in* (withdraw a tuple from the TS),
- *rd* (like *in* with the exception that the matched tuple remains in the TS),
- *eval.* (Similar to *out* except that a tuple is evaluated after it is placed in the TS, rather then before.)

In addition, there are two variant forms *inp* and *rdp*. These are predicate forms of *in* and *rd*. They do not cause process suspension if no matching tuple exists. Furthermore they return *true* if the matching does not fail; otherwise they return *false*. As the functions to be evaluated are placed in the TS, it is not predictable which processor will actually perform the computation. A detailed description of C-LINDA can be read in [7].

Implementation. We suppose we had shared memory system with P processors. Before we implement our algorithm we should consider the following facts:

- *Multiusersystem.* As there are many users logged in and let programs run, we do not know the system's performance at a certain time t_1 ,when we start our program at time t_0.
- *Processors* cannot be allocated before we start a program. Tuples and so called "evals" will be withdrawn from the TS without any determinism.

These are the reasons to partition the whole problem in small subproblems and to balance the load as dynamically as possible.

We suppose that there are L points for which we want to compute the original function. In the previous paragraph we could see that there is a bijective mapping of $\{1,\ldots,L\}$ onto the set of co-ordinates. So the inversion of $f(Z)$ for one point could be one subproblem. We write a program which contains the same idea that we used for the MWP in STRAND88.

"Manager Code":

```
real_main()
{
Place all shared variables in the TS (out);
for($i = 1; i \leq L; i++$)
    {
    Place $L$ different numbers in the TS (out);
    }
for($i = 1; i \leq P; i++$)
    {
    Place $P$ eval(laplace_inversion()) in the TS;
    /* These will be our workers. */
    }
for($i = 1; i \leq L; i++$)
    {
    Withdraw $L$ results from the TS (in);
    }
}
```

We placed the shared variables in the TS and we placed P worker processes in the TS. Each worker process is called *laplace_inversion*(). After that we start to withdraw the results from the TS until the last worker is idle and the last result has been placed in the TS.

"Worker Code":

```
laplace_inversion()
{
Read the shared variables from the the TS, without
removing them (rd);
while (inp(one of the L numbers))   /*If there is something to
do*/
    {
    compute the co-ordinates of the point;
```

```
        compute the original function $F(T)$;
        place the result together with the co-ordinates in the TS;
        }
}
```

Each worker tries to withdraw points from the TS (inp). Whenever a worker terminated the numerical inversion of $f(Z)$ for one point he requests more work immediately. A minimazation of the idle times should be guaranteed, too.

Examples.

- $$\mathcal{L}\left\{\frac{t^2e^{-t}}{2}\right\} = \frac{1}{(z+1)^3}.$$

Table 1. One dimensional example

A	$\#(N \in G_1)$	ϵ	α	error
(0,4)	10^2	10^{-6}	6	$4 \cdot 10^{-4}$
(0,4)	10^3	10^{-9}	6	$4 \cdot 10^{-6}$
(0,4)	10^4	10^{-12}	6	$4 \cdot 10^{-8}$

- $$\mathcal{L}^2\left\{\frac{e^{-2t_1-2t_2}t_1^3t_2^3}{36}\right\} = \frac{1}{(z_1+2)^4(z_2+2)^4}.$$

Table 2. Two dimensional example

A	$\#(N \in G_1)$	ϵ	α	error
$(0.5, 2.5) \times (0.5, 2.5)$	10009	10^{-9}	6	10^{-10}
$(0.5, 2.5) \times (0.5, 2.5)$	46368	10^{-10}	6	$8 \cdot 10^{-11}$
$(0.5, 2.5) \times (0.5, 2.5)$	196418	10^{-11}	6	10^{-12}
$(0.5, 2.5) \times (0.5, 2.5)$	317811	10^{-12}	6	$4 \cdot 10^{-13}$

We want to compute the speed-up values for that problem (Fig. 6.).

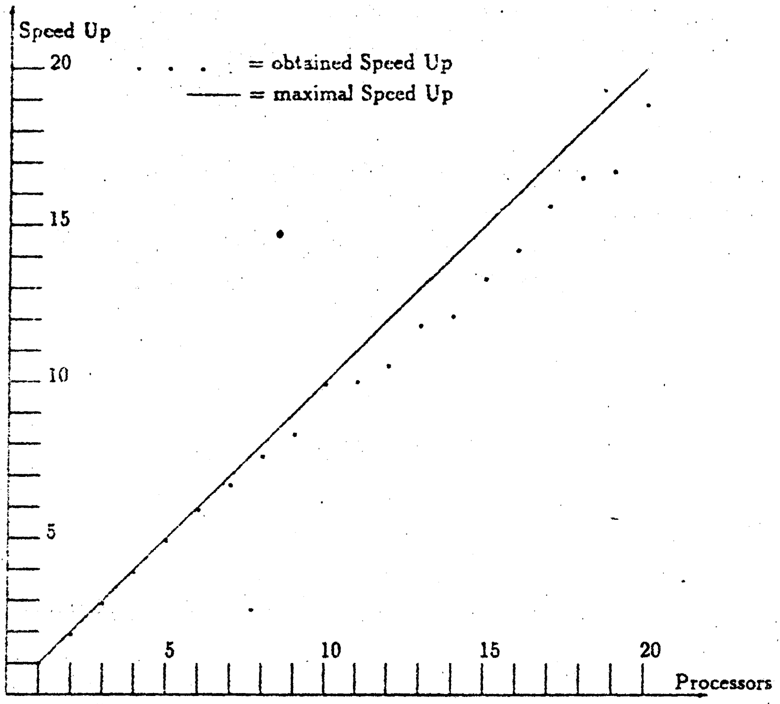

Fig. 6. Speed up for parallel processing.

Here the speed up is quite linear, too. The problem has been decomposed sufficiently and the idle times of the several workers have been minimized by a dynamic load balancing of the whole process.

REFERENCES

[1] Bertsekas, D., and J.Tsitsiklis (1989). *Parallel and Distributed Computation, Numerical Methods.* Massachusetts, Pentice-Hall.

[2] Davis, B. (1978). *Integral Transforms and Their Applications.* New York, Springer.

[3a] Doetsch, G. (1958) *Einf"uhrung in die Theorie und Anwendung der Laplace*

Transformation. Basel, Stuttgart, Birkh"auser.

[3b] Doetsch, G. (1971) *Handbuch der Laplace-Transformation Band 1.* Basel, Birkh"auser.

[3c] Doetsch, G. and D.Voelker (1950). *Dietrich: Die zweidimensionale Laplace-Transformation.* Basel, Birkh"auser.

[4] Foster, I. and S.Taylor (1990). *Steven: STRAND^{88} New Concepts in Parallel Programming.* New Jersey, Prentice Hall

[5] Korobov, N.M. (1963). *Number theoretic methods in approximate analysis.* Moscow. Fizmatigiz.

[6] Niederreiter, H. (1978). *Existence of Good Lattice Points in the Sense of Hlawka, Monatshefte f"ur Mathematik.* (86, 203 – 219), Springer Verlag.

[7] Sherman, A. (1990). *Original LINDA.* Scientific Computing Associates, Inc., New Haven, Connecticut

[8] Spiegel, M.(1977). *Theory and Problems of Laplace Transforms.* D"usseldorf, New York, Mc Graw Hill.

[9a] Zinterhof, P.(1987). Gratis lattice points for multidimensional integration, *Computing.* **38**, 347 – 353.

[9b] Zinterhof, P. (1992). Number theory and number crunching In F. Schweiger and E.Monstavičius (Eds.). *New Trends in Probability and Statistics,* Vol.2: Analytic and Probabilistic Methods in Number Theory, TEV, Vilnius – VSP, Utrecht. pp. 379–386.

Received February 1993

P. Zinterhof was born in Vienna in 1944. He received his PhD in Mathematics from the University of Vienna. Habilitation at the Technical University of Vienna in 1971. He is Professor for Mathematics and Systems Analysis at the University of Salzburg since 1972. He is founder and head of RIST++ (Research Institute for Software Technology + Paris Lodron Universit"at Salzburg) since 1986 and member of the Austrian Center for Parallel Computation (ACPC).

M. Salchegger was born in Salzburg in 1967. He received his Mag. Degree in Mathematics and Systems Analysis from the University of Salzburg. He is member of RIST++ since 1991 and he is member of the Austrian Center for Parallel Computation (ACPC). He is now working as an Assistant Prof. at RIST++ and the Dept. of Computer Sciences.