Resultados confiáveis no Cray

Úrsula Adriane Lisbôa Fernandes[1]
Tiarajú Asmuz Diverio[2]

Instituto de Informática e CPGCC da UFRGS
Caixa Postal 15064 - 91501-970 Porto Alegre - Brasil
E-mail: ursula@pucrs.campus2.br
diverio@inf.ufrgs.br

Resumo

Este trabalho apresenta a biblioteca de alta exatidão desenvolvida para o ambiente do Supercomputador Cray Y-MP. Esta biblioteca se fez necessária para possibilitar que cálculos em ponto-flutuante sejam feitos com máxima exatidão, proporcionando resultados confiáveis. Foram desenvolvidos testes usando somatórios e produtos escalares, onde os resultados obtidos na aritmética de ponto-flutuante ordinária são incorretos. As falhas nos cálculos são oriundas da forma como as operações são efetuadas, da inexistência de registradores especiais para as operações de acúmulo e de produto escalar e da inexistência dos arredondamentos direcionados. Mostra-se a solução destes problemas com a biblioteca de alta exatidão, onde os resultados são corretos.

[1] Mestra em Ciência da Computação (CPGCC/UFRGS), professora da PUC - Campus II - Uruguaiana
[2] Professor Dr., orientador do CPGCC/UFRGS, financiado pelo CNPq/RS

1 Introdução

A Universidade Federal do Rio Grande do Sul foi contemplada em 1992, com o Supercomputador Cray Y-MP, resultando na criação do Centro Nacional de Supercomputação (CESUP). Com este equipamento, iniciou uma nova etapa no processamento de alto desempenho no Brasil, pois o Centro de Supercomputação é Nacional e permite o acesso ao Cray de qualquer Universidade via rede.

O Cray é um equipamento vetorial, onde as operações em ponto-flutuante são realizadas por unidades funcionais específicas, operando com um vetor de 64 registradores, cada elemento com 64 *bits*. Ele possui duas CPU's, que possibilitam o processamento paralelo e pico de processamento de 660 Mflops.

As aplicações iniciais no Cray estão concentradas nas áreas de: Geofísica; Química computacional; Dinâmica de Fluidos; Análise estrutural; Matemática/ algoritmos; Física do estado sólido; Física (campos/ partículas/ ótica/ astrofísica); Física de Plasma; Computação de Alto Desempenho; Redes Neurais; Engenharia Química e Ciências do Ambiente. Os serviços estão disponíveis, em princípio, a qualquer entidade privada ou pública, de ensino, pesquisa, industrial, agropecuária, comercial ou de consultoria técnica. Maiores informações podem ser obtidas pelo e-mail super@cesup.ufrgs.br.

Desde sua instalação, o Grupo de Matemática da Computação da UFRGS iniciou estudos sobre qualidade e confiabilidade dos resultados obtidos no Cray. Observou-se que:

- Resultados produzidos no modo escalar e no modo de vetorial podem ser diferentes, mesmo quando não haja dependência de dados;
- Resultados escalares de somatórios podem não ser confiáveis;
- Resultados de operações que envolvam produto escalar não são calculados de forma a minimizar o erro de arredondamento;
- A operação de divisão não é o inverso da multiplicação de ponto-flutuante.

Em virtude destas falhas, foi proposto um ambiente de alta exatidão e alto desempenho, em [DIV95].

2 Inexatidão dos resultados obtidos no Cray

Há uma tendência de se aumentar a velocidade de processamento de *software* numéricos, sendo assim, os programas foram transferidos para computadores vetoriais, onde operações aritméticas *pipelinizadas* podem ser utilizadas. Dependendo dos cálculos a serem realizados, não se pode chegar a nenhuma conclusão sobre o resultado obtido e, se o algoritmo for processado em modo vetorial, o usuário pode perder o controle da forma (ordem) como os cálculos serão executados.

No item anterior, foram citados algumas das falhas de cálculos observadas. Neste item, elas serão caracterizadas e exemplificadas. Objetiva-se com isto, justificar a necessidade de se ter cálculos mais exatos e confiáveis.

O exemplo 1 ilustra o fato de que resultados diferentes podem ser obtidos para uma mesma soma, no modo escalar e no modo vetorial. Observa-se que isto não se deve à dependência de dados e, sim à instabilidade do algoritmo.

Seja a soma de uma seqüência $\sum_{i=N}^{-N} 16^i - 16^i + \sum_{i=N}^{-N} 16^i - 16^i + 1$. A soma é igual a 1. Na fórmula, o termo 1 é adicionado no fim, mas em geral, ele pode ser adicionado em qualquer lugar do somatório; o índice i indica o termo depois do qual o valor 1 é adicionado. Por exemplo, S_0 significa que o termo 1 é adicionado depois do primeiro termo, etc. Sem 1, a soma possui 4 $(2N+1) = 8N+4$ termos, então i pode ter $8N+5$ diferentes valores de 0 até $8N+4$:

$$S_0 := 1 + 16^N - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N}$$
$$+ 16^N - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N}$$

$$S_1 := +16^N + 1 - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N}$$
$$+16^N - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N}$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$S_{8N+4} := +16^N - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N}$$
$$+16^N - 16^N + 16^{N-1} - + \ldots + 16^{-N} - 16^{-N} + 1$$

Até N=11, todos os valores calculados em modo escalar são iguais a 1. Para N=12, o resultado é igual a 1 a partir de S_{52}, para N=13 a partir de S_{58}, para N=14 a partir de S_{64}, e para N=15 a partir de S_{70}. Estes resultados podem ser explicados devido ao tamanho da mantissa do Cray, porque quando valores de magnitudes diferentes são somados, o termo menor é perdido. A tabela abaixo mostra os resultados para N=29. O resultado começa ser igual a 1 na tabela, a partir de S_{154}, porque o valor 1 não é mais cancelado por valores grandes.

No modo vetorial, até N=6 para todos S's, o valor é igual a 1. A partir de N=7 os valores são completamente imprevisíveis, como mostra a tabela N=29. A explicação para estes resultados errôneos é que a ordem das parcelas do somatório é completamente rearranjada.

N = 29		
Soma	Modo escalar	Modo Vetorial
S_0	0.000000E+00	0.2951479E+21
S_{22}	0.000000E+00	0.1152922E+19
S_{23}	0.000000E+00	0.1152922E+19
S_{119}	0.000000E+00	0.0000000E+00
S_{122}	0.000000E+00	-0.2951479E+21
S_{149}	0.000000E+00	0.2951479E+21
S_{150}	0.000000E+00	0.1000000E+01
S_{153}	0.000000E+00	0.1000000E+01
S_{154}	0.100000E+01	0.1000000E+01
S_{205}	0.100000E+01	0.2951479E+21
S_{206}	0.100000E+01	0.2951479E+21
S_{236}	0.100000E+01	0.2951479E+21

O exemplo 2 procura mostrar que uma mudança na ordem de como as parcelas são somadas pode influenciar o resultado. Um teste semelhante ao mencionado acima foi construído. Nele, os valores positivos e negativos são agrupados em blocos em cada uma das somas S_i. Por exemplo, S_0 foi reagrupado do seguinte modo:

$$S_0 := 1 + 16^N + 16^{N-1} + \ldots + 16^{-N}$$
$$- 16^N - 16^{N-1} - \ldots - 16^{-N}$$
$$+ 16^N + 16^{N-1} + \ldots + 16^{-N}$$
$$- 16^N - 16^{N-1} - \ldots - 16^{-N}$$

Para N=29, aplicou-se o modo escalar para a soma reagrupada, obteve-se S_0= -0.314824E+21, já no primeiro teste mostrado, obteve-se $S_0 = 0$ no modo escalar e S_0 = 0.2951E+21 no modo vetorial. Nenhuma das formas obteve o resultado exato 1.

O exemplo 3 envolve o cálculo do produto escalar. É utilizado em operações com vetores e matrizes. Mais especificamente na multiplicação de duas matrizes. Sejam v e w, dois vetores, o produto escalar é definido por: $v.w = \sum_{i=1}^{n} v_i . w_i$. O produto escalar de: v = (211.E10, 2.0, 23.0, -211.E10) por w = (2.E10, 510.0, 33.0, 2.E10) produz como resultado o valor 0, quando deveria produzir o valor 1779. Isto acontece porque os números somados são de diferentes ordens de magnitude, não sendo assim processados corretamente em aritmética de ponto-flutuante ordinária.

No Cray, as operações aritméticas básicas (adição, subtração, multiplicação e divisão) em ponto-flutuante são realizadas em unidades funcionais especiais. O Cray não possui uma unidade funcional dedicada à operação de divisão. Assim, o modo como realiza a operação muitas vezes gera erros.

O **exemplo 4** mostra a falha em relação à operação de divisão. No cálculo de 100.0/10.0, o resultado encontrado é 9.999999999999943. O Cray, primeiramente, acha recíproco de 10.0 que é 0.1 ou 1/10. O problema é que este valor pode não ser representável no sistema de representação binária, como no caso do exemplo. Então, o valor é truncado para um valor próximo. Quando multiplicado por 100.0, não reproduz o valor original, resultando o valor incorreto.

3 Estudo do problema

O tipos de erros mostrados acima ocorrem porque o Cray não possui implementado em seu *hardware* as características da aritmética de alta exatidão. Os requisitos para se ter aritmética de alta exatidão são: o uso de arredondamentos direcionados (para baixo ∇x e para cima Δx), as quatro operações aritméticas com máxima exatidão e o produto escalar ótimo.

A matemática intervalar vem trazer uma técnica que permite um controle de erros com limites confiáveis, pois o resultado é contido num intervalo. Este intervalo deve ser suficientemente pequeno para garantir a utilidade do resultado.

O uso da aritmética de alta exatidão, aliado à matemática intervalar, resulta que os intervalos sejam os menores possíveis, pois os extremos são calculados de forma exata, sendo somente no final arredondados (para baixo e para cima de acordo com o extremo). Portanto, os extremos diferem do valor real por apenas um arredondamento.

Padrão IEEE 754

O Padrão de aritmética de ponto-flutuante IEEE 754 surgiu com o objetivo de padronizar a aritmética de ponto-flutuante em todas as plataformas, sendo um passo inicial para a obtenção da aritmética de alta exatidão. Esse padrão não especificou os arredondamentos para variáveis complexas, operações entre vetores e matrizes e também não inclui o produto escalar ótimo, essencial para a garantia da alta exatidão nos cálculos e por causa disto a GAMM/IMACS propuseram outro padrão mais completo (ver [IMACS91]).

No Cray Y-MP2E, os números em ponto-flutuante não são representados segundo o padrão de aritmética binária de ponto-flutuante da IEEE (padrão IEEE 754), nem no que se refere ao tamanho da palavra (possui 64 *bits*), nem no modo de como as operações aritméticas são realizadas. O padrão IEEE 754 determina que uma operação aritmética deve ser efetuada como se o resultado exato fosse primeiramente calculado, e então, arredondado com o arredondamento selecionado. No Cray, as operações não são efetuadas dessa maneira, são feitas em unidades funcionais. Os valores são armazenados em registradores de ponto-flutuante (escalares ou vetoriais), sendo que o tamanho destes é de 64 *bits*, o mesmo das palavras da memória.

Aritmética de ponto-flutuante com operações em máxima exatidão

Seja R o espaço dos números reais e F o conjunto representável no computador. Se $\circ$ é uma operação aritmética em R, a correspondente operação no computador $\boxed{\circ}$ em F é dada pela fórmula abaixo, também chamada de Regra Geral (RG), onde [ ] é um arredondamento.

$$x \boxed{\circ} y := [](x \circ y) \qquad \text{para todos } x, y \in F$$

Isto é, a operação no computador deve ser efetuada como se o resultado exato fosse primeiro calculado e, então, aproximado com o arredondamento selecionado. O arredondamento [] unicamente determina a operação arredondada $\boxed{\circ}$ no computador. Mais detalhes em [KUL81].

No Cray as operações não são realizadas com máxima exatidão possível, pois para isto seriam necessários registradores de tamanho maior que o da palavra do Cray. Não há nas unidades funcionais e

nem dígitos de guarda associados aos registradores, que possibilitam a definição de diferentes tipos de arredondamentos e que se amplie a precisão.

Arredondamentos direcionados

Funções de mapeamento (ver em [KUL81]) são necessárias para representar os números reais em números de máquina. Há vários tipos de arredondamento, sendo que os mais importantes são o os arredondamentos direcionados (Δx, ∇x) e o arredondamento para o número mais próximo de máquina (Ox). Arredondamento para cima, Δx, é a função que aproxima o número real x para o menor número de máquina maior que o número real x. Arredondamento para baixo, ∇x, é a função que aproxima o número real x para o maior número de máquina, menor que o número real x.

Na implementação de uma aritmética de alta exatidão, deve-se ter os dois tipos de arredondamentos direcionados, para baixo ∇x e para cima Δx. mais o arredondamento para o número mais próximo de máquina Ox. Os arredondamentos direcionados são fundamentais para a implementação da aritmética intervalar de máquina, que será detalhada a seguir.

O Cray, conforme mencionado, não possui dígitos de guarda. A ausência de dígitos de guarda impede que os valores sejam arredondados de forma desejada. Um exemplo com números decimais ilustrará a importância destes dígitos extras. Na soma de 0.256×10^0 com 0.234×10^2, usando três dígitos de precisão (mantissa com 3 casa decimais) e arredondamento simétrico, obtém-se valores distintos no caso de arquiteturas que possuem dígitos de guarda daquelas que não possuem. A soma é feita por etapas; primeiramente, alinha-se os expoentes ao maior valor, no caso 2. Para alinhar o expoente, é feito um *shift* de duas casa à direita (dividido por 10^2), perdendo-se os dois dígitos menos significativos. No caso da soma sem os dígitos de guarda, ao se igualar os expoente, o valor 0.256×10^0 torna-se 0.002×10^2. A soma resulta em 0.236×10^2.

No caso da existência de dois dígitos de guarda, 0.256×10^0 torna-se 0.00256×10^2, já que os dígitos de guarda são suficientes para representar os dois dígitos menos significativos no momento do alinhamento dos expoentes. A soma resulta em 0.23656×10^2. Assim, depois do arredondamento, tem-se o valor 0.237×10^2. O resultado é diferente daquele obtido, quando a soma é calculada sem os dígitos extras.

Produto escalar ótimo

O produto escalar é definido como: dados dois vetores x e y. cada um (com n componentes) em ponto-flutuante cada e arredondamento [ ], o resultado s em ponto-flutuante da operação de produto escalar (aplicada para x e y) é definida pela fórmula abaixo:

$$s = [](\bar{s}) = [](x.y) = [] \left(\sum_{i=1}^{n} x_i * y_i \right)$$

Pela definição, todas as operações aritméticas são exatas. Em outras palavras, s será calculado como se um resultado intermediário $\bar{s} = x.y$ fosse correto com precisão infinita e com faixa de expoente ilimitada e, então, arredondado para o formato em ponto-flutuante desejado, de acordo com o arredondamento selecionado.

É utilizado na multiplicação de vetores, de matrizes e de vetores e matrizes nos diferentes tipos numéricos avançados[3] de dados. A questão gira a respeito do número de arredondamentos. O cálculo tradicional do produto escalar de dois vetores com n componentes cada, envolve $2n$-1 arredondamentos.

O Cray não possui um *hardware* especializado no cálculo do produto escalar de forma ótima (ver [IMAC91]), onde o resultado calculado somente difira de um arredondamento do valor exato. Além disto, dependendo do grau de otimização desejado pelo usuário, a ordem do somatório das parcelas no produto

[3] reais, complexos, intervalos de reais e intervalos de complexos.

escalar pode ser totalmente rearranjada. Isto é extremamente desagradável! E resultados diferentes são obtidos para uma mesma operação quando executados nos modos de processamento escalar e vetorial.

Matemática Intervalar e Aritmética Intervalar de Máquina

A matemática intervalar trata com dados na forma de intervalos numéricos e surgiu com o objetivo de automatizar a análise do erro computacional, trazendo uma nova abordagem que permite um controle de erros com limites confiáveis, além de provar a existência ou não da solução de diversas equações. Uma visão completa sobre ela pode ser encontrada em [OLP97].

A aritmética intervalar trata com dados na forma de intervalos numéricos e surgiu com o objetivo de automatizar a análise do erro computacional, trazendo uma nova abordagem que permite um controle de erros com limites confiáveis, além de provas da existência ou não da solução de diversas equações.

Ao invés de serem aproximados para números de ponto-flutuante mais próximos (por alguma das regras de aproximação ou dos tipos de arredondamento), os reais são representados por intervalos de números em ponto-flutuante, ou seja, um real x é representado por um intervalo X=[x1, x2], onde os limites inferior (x1) e superior (x2) são números de máquina, tais que $x1 \leq x \leq x2$. Todas as operações aritméticas, operadores relacionais, funções elementares, são definidas para argumentos intervalares. A partir da aritmética intervalar são desenvolvidos os conceitos que compõem a matemática intervalar e os métodos intervalares para resolução de problemas numéricos. A figura abaixo mostra algumas das operações existentes sobre o conjunto de intervalos de ponto-flutuante.

Inverso aditivo:	$-X = [-x1, -x2]$
Pseudo inverso multiplicativo:	$1/X = [1/x2, 1/x1]$ $0 \notin X$
Adição:	$X+Y = [x1 \nabla_{+} y1,\ x2 \Delta_{+} y2]$
Subtração	$X-Y = [x1 \nabla_{-} y2, x2 \Delta_{-} y1]$
Multiplicação:	$X*Y = [\min\{ x1 \nabla_{*} y1, x1 \nabla_{*} y2, x2 \nabla_{*} y1, x2 \nabla_{*} y2\}, \max\{x1 \Delta_{*} y1, x1 \Delta_{*} y2, x2 \Delta_{*} y2, x2 \Delta_{*} y2\}]$
Divisão:	$X/Y = [\min\{ x1 \nabla_{/} y1, x1 \nabla_{/} y2, x2 \nabla_{/} y1, x2 \nabla_{/} y2\}, \max\{x1 \Delta_{/} y1, x1 \Delta_{/} y2, x2 \Delta_{/} y2, x2 \Delta_{/} y2\}]$
Intersecção:	$X \cap Y = \{ [\max\{x1, y1\}, \min\{x2, y2\}]$, se $x1 < y2$ e $y1 \leq x2$, $\varnothing$ - caso contrário.
União:	$X \cup Y = \{ [\min\{x1, y1\}, \max\{x2, y2\}]$, se $X \cap Y \neq \varnothing$, *ERRO* - caso contrário.
Distância:	$d(X,Y) = \max\{ \lvert x1-y1 \rvert, \lvert x2-y2 \rvert\}$
Valor absoluto:	$\lvert X \rvert = d(X,[0,0]) = \max\{ \lvert x1 \rvert, \lvert x2 \rvert\}$
Diâmetro:	$D(X) = x2-x1$

Figura 1 - Operações Básicas Intervalares

4 Proposta de solução: *libavi.a*

No ambiente do Cray foi, então, desenvolvida uma biblioteca intervalar *libavi.a*, implementada em Fortran 90, com a finalidade de explorar o alto desempenho proporcionado pelo processamento vetorial (e paralelo) juntamente com o cálculo de operações intervalares. O objetivo final era ter um ambiente de alto desempenho e alta exatidão.

Na *libavi.a* estão disponíveis 320 rotinas, as quais implementam a aritmética intervalar básica, a aritmética de vetores e matrizes de intervalos e alguns métodos intervalares de solução de sistemas de equações lineares. A biblioteca é composta por quatro módulos. O primeiro é o *basico*. Neste módulo foram definidos intervalos reais e as operações que os manipulam. Ele é composto por seis conjuntos de rotinas, agrupadas de acordo com a natureza das operações. Estes conjuntos são: funções de transferências, operadores relacionais, operações entre conjuntos, operações aritméticas, funções básicas e rotinas de entrada e saída. O segundo módulo é o *mvi*. Ele é composto de três submódulos: o primeiro refere-se a definição e ao tratamento de vetores de intervalos; o segundo define matrizes de intervalos e suas operações e o terceiro trata de operações de diferentes tipos de dados com vetores e matrizes de intervalos reais. O terceiro módulo é o *ci*, onde são implementados os intervalos complexos, matrizes e vetores de intervalos complexos e as operações que os manipulam. Este módulo é uma extensão do módulo *basico* para complexos adicionado de algumas operações entre vetores e matrizes de intervalos complexos. O quarto

módulo é composto por 24 rotinas que implementam 18 métodos para a solução de sistemas de equações lineares intervalares e pontuais.

5 O Problema continua

Conforme já foi dito, objetivou-se ter um ambiente de alto desempenho e alta exatidão. Mas, devido às limitações da arquitetura do próprio Cray a biblioteca ficou limitada no que se refere à alta exatidão. Apesar dos resultados serem confiáveis, pois estão contidos nos intervalos, não são os melhores possíveis, pois não são feitos com máxima exatidão pelas impossibilidades do *hardware* do Cray. Um exemplo disso, é a questão dos arredondamentos. Na biblioteca intervalar *libavi.a*, estes não foram efetivamente implementados. Obteve-se, através de funções do Fortran, uma simulação satisfatória, que assegurava confiabilidade aos resultados, mas tinha como efeito "colateral", que o diâmetro geralmente era maior do que deveria. Essas rotinas não testavam se um determinado valor é representável no sistema de ponto-flutuante em questão, somando ou subtraindo 1.10^{-13+e} (no décimo terceiro dígito da mantissa), onde e é o expoente a ser inflado e sendo assim, o intervalo é sempre maior do que deveria ser.

Os testes realizados no item 2 serão apresentados aqui de forma intervalar utilizando-se a biblioteca *libavi.a*. Todos os valores são intervalos degenerados, isto é intervalos onde o extremo inferior é igual ao extremo superior.

Assim, no **exemplo 1**, todos os componentes do somatório foram transformados em intervalos degenerados. Na tabela são mostrados resultados para alguns N's. Os valores são sempre os mesmos para um determinado N, no modo escalar e no vetorial. Percebe-se que o valor correto 1 está sempre incluído em todos intervalos, apesar de que, para N's muito grandes o diâmetro do intervalo seja muito grande.

N	Modo escalar e vetorial
10	S0 até S84 [9.6770833333325E-01, 1.0161458333335E+00]
12	S0 até S100 [-3.1333333333341E+00, 5.1333333333341E+00]
13	S0 até 108 [-6.5133333333347E+01, 6.7133333333348E+01]
14	S0 até 116 [-1.0571333333336E+03, 1.0591333333336E+03]
15	S0 até 124 [-1.6929133333338E+04, 1.6931133333338E+04]
29	S0 até 236 [-1.2199446747421E+21, 1.2199446747421E+21]

O **exemplo 2** procura mostrar que uma mudança na ordem de como as parcelas são somadas pode influenciar o resultado. O intervalo obtido para N = 29 foi [-7.0009083108556E+22, 6.8789138433814E+22], sendo que o valor pontual encontrado no item 2, -0.314824E+21, se encontra dentro do intervalo.

O **exemplo 3** foi transformado no cálculo do produto escalar de dois vetores intervalares. Assim : v = ([211.E10, 211.E10], [2.0 2.0] , [23.0, 23.0] , [-211.E10-211.E10]) por w = ([2.E10,2.E10] , [510.0, 510.0] , [33.0, 33.0] , [2.E10,2.E10]) produz como resultado o intervalo [-1.073741824E+09, 1.073741824E+09].

Na *libavi.a.* foram implementadas duas rotinas que produzem um intervalo, contendo o valor do produto escalar. Nenhuma das duas rotinas é realizada de forma ótima, produzindo assim, geralmente, intervalos soluções com o diâmetro muito grande, contendo uma grande quantidade de informação desnecessária. Assim, todas as rotinas que realizam multiplicações de matrizes não possuem em sua implementação o produto escalar de forma ótima. A rotina *svmult* foi utilizada no teste do **exemplo 3**, ela calcula um intervalo que contém o produto escalar de dois vetores intervalares pela multiplicação das componentes dos vetores que são acumulados e são aplicadas às rotinas que tentam simular os arredondamentos.

O **exemplo 4** mostra a falha em relação à divisão. Ao invés de fazer a divisão de dois números reais, 100.0/10.0, estes foram colocados em variáveis intervalares. Assim é realizada a divisão do intervalo [100.0] pelo intervalo [10.0]. O resultado obtido foi o intervalo : [9.9999999999999E+00, 1.0000000000000E+01] sendo o valor do diâmetro: 1.13686837721616E-13.

Um outro exemplo para realçar mais a questão relativa aos arredondamentos direcionados é mostrado no exemplo da figura 1. Ela mostra o que acontece quando uma multiplicação de intervalos é

realizada sucessivamente. Multiplicando o intervalo [1.0, 1.0] por ele mesmo, 60 vezes, o resultado deveria ser sempre [1.0, 1.0] (se o ambiente possuísse as operações com máxima exatidão e os arredondamentos fossem efetivamente implementados), já que não deveria ocorrer nenhum arredondamento, visto que o produto é sempre o valor 1.0, representável no sistema (binário) de ponto-flutuante do Cray.

```
program arredondamento
use basico
type(interval) :: a
real :: diametro
a = sintpt(1.0)
do i=1,59
   a = a * a
   print *,"intervalo", i
   call swrite(6,a)
   print *,"diametro"
   diametro = sdia(a)
   write (6,210) diametro
   210 format (ES21.13)
end do
```

Iteração	Diâmetro
1	1.4210854715202E-14
10	1.0906830993918E-11
20	1.1175863789958E-08
30	1.1444113731329E-05
40	1.1741727852900E-02
50	2.9809396735744E+03
55	1.5114277002024+111
56	2.2844136929391+222
57	5.2185459204876+444
58	2.7233221524238+889
59	7.4164835458824+1778

Figura 1 - Multiplicações sucessivas sobre o intervalo [1, 1]

A figura 2 ilustra o crescimento do valor do diâmetro do intervalo a medida que multiplicações sucessivas são efetuadas. O resultado desta relação sobre a precisão de 14 dígitos, resulta no gráfico da perda de exatidão descrita acima.

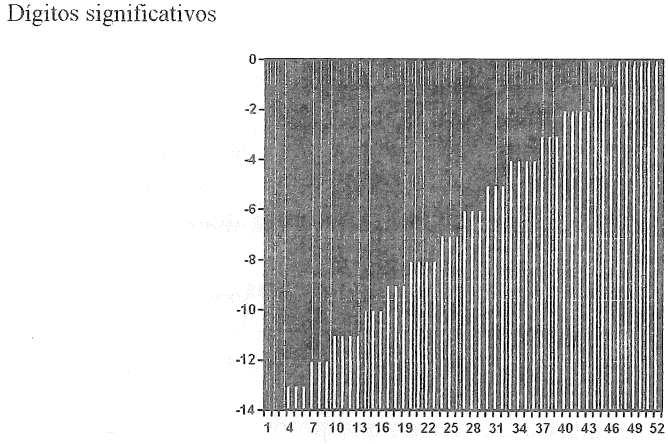

Figura 2 - Dígitos significativos x Multiplicações = Perda da exatidão

6 *Libavi.a* com núcleo de alta exatidão - solução definitiva

A *libavi.a* foi desenvolvida no Cray com a finalidade de proporcionar resultados confiáveis, uma vez que a aritmética computacional disponível é sensível a problemas de instabilidade e produz resultados diferentes, dependendo do modo de processamento. Visou à obtenção de uma aritmética de alta exatidão e alto desempenho, mas devido ao Cray não ter implementado em seu *hardware* as características do padrão IEEE e nem o produto escalar ótimo, a biblioteca necessitava da implementação de um núcleo de aritmética de alta exatidão, que implementasse e tornasse disponível estas características.

O núcleo de aritmética de alta exatidão deveria ser composto, então, pelas operações em ponto-flutuante com máxima exatidão dos arredondamentos direcionados e pelo cálculo do produto escalar. Em virtude da ausência dos dígitos de guarda para os arredondamentos, a maneira como o Cray realiza as

operações não permitiu que a biblioteca tivesse um melhor desempenho em termos de qualidade (exatidão). Tentou-se amenizar este problema utilizando as rotinas que sempre somam ou subtraem um determinado valor ao décimo terceiro dígito, o problema é que o intervalo era sempre inflacionado, mesmo quando não havia necessidade.

O cálculo do produto escalar, não é ótimo, produzindo intervalos de diâmetro tão grande que se tornam praticamente inúteis. Com a implementação do produto escalar ótimo, a exatidão máxima é garantida.

Conforme foi visto, havia possibilidade de fazer com que a *libavi.a* se tornasse uma poderosa ferramenta para ser utilizada em aplicações que exigissem rapidez e confiabilidade nos cálculos. Sendo assim, foi dada continuidade ao trabalho com a implementação do Núcleo de Aritmética de Alta sendo composto por rotinas escalares e vetoriais. As rotinas escalares são: soma, subtração, multiplicação e divisão de dois números em ponto-flutuante, arredondamento para cima, para baixo e para o número mais próximo.

As rotinas vetoriais implementadas são: produto escalar ótimo intervalar, produto escalar ótimo real, soma ótima dos elementos de um vetor intervalar e soma ótima dos elementos de um vetor real.

Várias modificações foram necessárias para a incorporação do núcleo à *libavi.a*. A incorporação do núcleo não alterou em nada a utilização da biblioteca. Os exemplos apresentados no item 5 foram testados em um segundo momento, após a incorporação do núcleo à biblioteca *libavi.a*.

No **exemplo 1**, o resultado para todos os valores N é o intervalo [1.0, 1.0], ou seja o melhor intervalo possível, já que o diâmetro é 0.

N	Modo escalar e vetorial
10	S0 até S84 [1.0, 1.0]
12	S0 até S100 [1.0, 1.0]
13	S0 até 108 [1.0, 1.0]
14	S0 até 116 [1.0, 1.0]
15	S0 até 124 [1.0, 1.0]
29	S0 até 236 [1.0, 1.0]

O **exemplo 2** procura mostrar que uma mudança na ordem de como as parcelas são somadas pode influenciar o resultado. O intervalo obtido para N = 29 foi [1.0, 1.0].

O **exemplo 3** apresentou como resultado o intervalo degenerado [1779, 1779], após a incorporação da rotina de produto escalar ótimo.

No **exemplo 4**, após a incorporação do núcleo, a divisão do intervalo [100.0, 100.0] pelo intervalo [10.0, 10.0] gera o intervalo correto [10.0, 10.0] sendo o valor do diâmetro igual a 0, ou seja, a solução com máxima exatidão possível.

O programa da figura 1 que multiplicava o intervalo pontual [1.0, 1.0] por ele mesmo repetidas vezes foi recompilado após a incorporação do núcleo. O resultado apresentado foi sempre o intervalo [1.0, 1.0].

7 Conclusões

Com a *libavi.a*, foi definida a aritmética de alto desempenho, composta pelo processamento de alto desempenho e da matemática intervalar. A biblioteca não possuía a aritmética de alta exatidão pelas próprias limitações impostas pelo ambiente em que foi desenvolvido o trabalho.

Após a incorporação do núcleo, foram feitos vários testes para avaliar o desempenho da biblioteca, tanto em relação à qualidade (exatidão) quanto ao desempenho propriamente dito (tempo de execução). Observou-se com os testes que o tempo de processamento foi penalizado cerca de três a dez vezes. Isto se deve ao fato de todas as operações terem sido simuladas por *software*, pelo uso de variáveis em dupla precisão, registradores longos em ponto-flutuante e pelo uso de acumuladores de ponto-fixo. Em compensação, obteve-se uma aritmética de alta exatidão no Cray, como foi verificado em vários exemplos. Em termos de otimização e vetorização, há ainda melhorias que podem ser realizadas.

Bibliografia

[BOH90] BOHLENDER, G. What do we need beyond IEEE arithmetic? In: ULLRICH, C. (Ed.). **Contributions to computer arithmetic and self-validating Numerical Methods.** Basel: J.C Baltzer, 1990. v.7. p.1-22.

[DIV95] DIVERIO, T. A. **Uso efetivo da matemática intervalar em supercomputadores vetoriais.** Porto Alegre: CPGCC da UFRGS, 1995. 290p. Tese de doutorado.

[DIV96] DIVERIO,T.A.; FERNANDES, U.A.L.; CLAUDIO, D.M. Erros in vector processing and the *libavi.a* library. **Reliable Computing**, Moscow, v.2, n.2, p.103-110, 1996 (Proceedings of SCAN-95, IMACS Annals on Computing and Applied Mathematics, Wuppertal, Set. 26-29, 1996).

[FER95] FERNANDES, U. A. L.; DIVERIO, T. A.; DAHMER, A. **Limitações do Processamento vetorial no Cray Y-MP2E.** Porto Alegre: CPGCC da UFRGS, 1995. 60p. (RP- 248).

[FER97] FERNANDES, U. A. L. **Núcleo de Aritmética de Alta Exatidão da Biblioteca Intervalar *libavi.a*.** Porto Alegre: CPGCC da UFRGS, 1997. 121p. Dissertação de mestrado.

[HOL96] HOLBIG,C.A. **Métodos intervalares para a resolução de sistemas de equações lineares.** Porto Alegre: CPGCC da UFRGS, 1995. 94p. Dissertação de mestrado.

[IMACS91] IMACS, GAMM. Resolution on computer arithmetic. In: KAUCHER, E.; MARKOV, S. M.; MAYER, G. (Eds.). **Computer Arithmetic, Scientific Computation and Mathematical Modelling.** Basel: J.C.Baltzer, 1991. p.477-479. (Proceedings of SCAN'90, IMACS Annals on Computing and Applied Mathematics, Albena, Sept 24-28, 1990, 12).

[KUL81] KULISCH, U.; MIRANKER, W. R. **Computer arithmetic theory and pratice.** New York: Academic Press, 1981.

[OLP97] OLIVEIRA, P.W.; DIVERIO,T.A;CLAUDIO,D.M. **Fundamentos da Matemática Intervalar.** Porto Alegre: Sagra-Luzzatto, 1997. 93p. (Série Matemática da Computação e Processamento Paralelo, v.1, Instituto de Informática da UFRGS, Projeto ArInPar-ProTeM-CNPq).