ALTO DESEMPENHO E EFICIÊNCIA EM PROCESSAMENTO NUMÉRICO

Tiaraju Asmuz Diverio, Dr
Philippe Olivier A Navaux, Dr
Dalcidio Moraes Claudio, Dr

Instituto de Informática e PGCC da UFRGS
Caixa Postal 15064 91501-970 Porto Alegre RS
E-Mail: {diverio,navaux,dalcidio}@inf.ufrgs.br

Resumo:

Este trabalho trata das características: desempenho e eficiência em processamento numérico, através da definição de aritmética de alta exatidão e alto desempenho. Esta definição reune os avanços nas áreas da aritmética computacional, qualidade numérica e processamento de alto desempenho. No aspecto da matemática computacional, a aritmética de alta exatidão e desempenho inclui: padronização do formato dos números e das operações em ponto-flutuante pelo padrão IEEE 754; cálculo de operações com máxima exatidão; controle automático do erro nas operações aritméticas pelo uso de arredondamentos direcionados; produto escalar ótimo e matemática intervalar. No aspecto de alto desempenho, inclui rapidez no processamento obtida pelo uso da vetorização e verificação automática do resultado. Este trabalho, descreve ainda, os resultados obtidos com a biblioteca de rotinas intervalares, desenvolvida em Fortran 90, no ambiente do supercomputador Cray Y-MP, como protótipo desta aritmética de alto desempenho.

Abstract:

This paper treats about numerical processing characteristics of efficiency and performance, through the high performance and high accuracy arithmetic definition. This definition rejoins the progress in the computational arithmetic areas, numerical quality, and high performance processing. In the computational mathematics aspect, the high accuracy and performance arithmetic include numbers format standardization and operations in float-pointing for the IEEE 754; operations calculus with maximum accurancy; automatic control of errors in arithmetic operations by the use of direct rounding; optimal scalar product and interval mathematics. In the high performance aspect, the definition includes quickness in the processing through vectorization and result automatic verification. This work still describes the results obtained with the library of interval routines developed in FORTRAN 90, on environment of supercomputer Cray Y-MP, as prototype of this high performance arithmetic.

1 INTRODUÇÃO

A resolução de problemas numéricos em computadores ainda enfrenta pelo menos duas grandes barreiras decorrentes das limitações da máquina. A primeira se refere à qualidade do resultado, que depende de fatores que vão desde a forma como os números são representados e armazenados até a instabilidade do próprio problema. A segunda barreira diz respeito ao porte do problema, que é limitado pelo número de elementos ou parâmetros (dimensão do problema) que devem ser armazenados (espaço - memória) e manipulados - quantidade de operações que envolvem tempo. Portanto, o tempo de processamento e o espaço de armazenagem limitam a dimensão do problema a ser resolvido.

É marcante, entretanto, o avanço obtido na capacidade de processamento desde os primeiros computadores em relação aos supercomputadores atuais. Em poucos anos, passou-se de algumas dezenas de operações aritméticas em ponto-flutuante por segundo, para um desempenho da ordem de 10^{11} operações em ponto-flutuante por segundo. Com o desenvolvimento de máquinas que processam mais velozmente as operações aritméticas e, que tem uma capacidade de armazenamento muito grande, tem-se conseguido implementar métodos numéricos que há bem pouco tempo atrás não eram nem sonhados, como computações de larga escala.

Computações de larga escala são processos que exigem uma grande quantidade de cálculos, de onde vem o termo "larga escala". Elas ocorrem nas áreas científicas e militares, nas engenharias, no estudo e exploração de fontes de energia, na medicina, na inteligência artificial e, também, na pesquisa básica.

Para resolver problemas nestas áreas são necessários computadores de alto desempenho, que são muitas vezes chamados de supercomputadores. Nestas máquinas o processamento é vetorial e/ou paralelo. O processamento vetorial dispõe de um *hardware (pipeline)* que possibilita a troca de operações escalares por operações vetoriais, o que produz uma diminuição do tempo de processamento. O processamento paralelo deve dispor de duas ou mais unidades de processamento ou processadores (CPU's), onde tarefas são efetuadas ao mesmo tempo.

A *pipelinezação,* desde os anos 70, tornou-se um dos principais meios de aumentar a velocidade da computação. Em adição às quatro operações elementares, computadores vetoriais proporcionam um número de operações de composição tais como, "acumulação", "multiplica" e "multiplica e acumula". Isto aumentou a velocidade, mas demonstrou a existência de uma grande lacuna entre o avanço tecnológico no desenvolvimento de computadores cada vez mais rápidos e poderosos e a qualidade com que os cálculos são feitos. Evoluiu-se no sentido de resolver a segunda barreira, enquanto que no sentido de qualidade (exatidão) do resultado, só foi dado o passo inicial, que foi a padronização da aritmética de ponto-flutuante.

Como a definição da aritmética da máquina ficava ao encargo do fabricante, cada sistema tinha as suas próprias características e defeitos. Cálculos efetuados em diferentes máquinas raramente produziam resultados compatíveis. Então, em 1980, a IEEE adotou o padrão de aritmética binária de ponto-flutuante, conhecida como padrão IEEE 754[1]. Várias de suas características são necessárias para se ter resultados garantidos. Entre elas, pode-se destacar a forma de representação dos números de ponto-flutuante; os intervalos de representatividade, os símbolos especiais de mais e menos infinito e o símbolo "not-a-number"(NaN) utilizados no tratamento de *underflow* e *overflow* e de outras exceções e, por fim, as operações com máxima exatidão, pois são definidas de forma a só terem um arredondamento, garantindo que resultados difiram do valor exato apenas na última casa da mantissa. Isto também é garantido pelo uso dos arredondamentos direcionados e pelo controle automático do erro de arredondamento.

[1]ver [DIV94]

Isto foi um passo na direção certa no sentido de resolver a primeira barreira, mas este padrão não especificou tudo. Nada foi mencionado para aritmética complexa ou em dupla precisão, por exemplo. Uma falha grave deste padrão, é que com sua adoção os fabricantes criaram a idéia dele ser suficiente. Erros produzidos são muitas vezes considerados como pequenas falhas.

Simples exemplos encontrados na literatura, como em [DIV95a], mostram que é necessário e, até mesmo obrigatório, tornar os métodos numéricos e os computadores mais confiáveis do que o caso do uso exclusivo da aritmética ordinária de ponto-flutuante. Este desenvolvimento começou aproximadamente a 25 anos atrás e tem recebido mais e mais importância até os dias atuais. Entretanto, ainda permanece um problema de aceitação a ser considerado. Mesmo utilizando somente a aritmética ordinária, o computador é ainda uma poderosa ferramenta que difundiu fortemente o uso desta aritmética, pois para muitas aplicações, como na área econômica ou financeira, a aritmética de ponto-flutuante é suficiente.

Com a tendência de aumentar a velocidade de processamento de *softwares* numéricos, muitos programas estão sendo transferidos para computadores vetoriais, como o supercomputador Cray, onde operações aritméticas *pipelinizadas* podem ser utilizadas. O principal problema com estas operações é que elas não seguem a definição do padrão IEEE 754. Mas, mesmo se estas operações seguissem exatamente o padrão IEEE, o cálculo de somatórios através de *pipelines,* implica na troca de ordem das parcelas, o que pode produzir variações nos resultados, diferindo de resultados obtidos no modo escalar. Assim, tanto o resultado produzido no modo escalar quanto o produzido no modo vetorial pode estar completamente errado. Uma maneira de resolver estes problemas é realizar a acumulação de números em ponto-fixo, com registrador suficientemente grande ou produtos com somente um arredondamento final. Esta acumulação pode ser realizada por qualquer tipo de computador, tanto seqüênciais quanto vetoriais, garantindo que os resultados calculados sejam independentes da ordem do somatório. Esta idéia veio junto com uma Proposta[2] de Aritmética de ponto-flutuante vetorial exata aprovada pela GAMM em 1987 e pela IMACS em 1988.

Controlar todos os dados que entram como resultados intermediários em longas horas de computação, em estações de trabalho ou em supercomputadores, tem se tornado uma tarefa muito difícil ou mesmo impraticável. Em análise clássica de erros de algoritmos numéricos, o erro de cada operação em ponto-flutuante é estimado. Mas é evidente que para certos cálculos, como por exemplo, um algoritmo que realiza 10^{17} operações em ponto-flutuante em uma hora de processamento, uma análise de erros não é realizada. Assim, o fato de que o resultado calculado poderia estar errado, muitas vezes não é levado em consideração. Contudo, do ponto de vista matemático, o problema da correção dos resultados é de grande importância, ainda mais com a alta velocidade computacional atingida atualmente.

[2]ver [IMACS91]

As técnicas para o cálculo com verificação automática do resultado utilizam o produto escalar ótimo e a aritmética intervalar como ferramentas essenciais. Esta aritmética permite o cálculo de extremos seguros para as soluções de um problema. A alta exatidão é obtida por meio do produto escalar ótimo. A combinação de ambos é realmente um grande avanço na análise numérica.

2 CONCEITOS EM PROCESSAMENTO NUMÉRICO

Antes de prosseguir no detalhamento da definição, é importante formalizar as características de desempenho e eficiência neste contexto. Desempenho e eficiência são duas características importantes e necessárias tanto em Processamento Numérico como em Computação Científica. Suas definições evoluiram com o passar do tempo de acordo com o desenvolvimento tecnológico, mais especificamente com os computadores.

Eficiência diz respeito à qualidade do resultado numérico, referindo-se a exatidão e a forma como o resultado é calculado, envolvendo número de operações aritméticas necessárias (medida de complexidade de tempo) e estabilidade numérica. Por algum tempo, eficiência significou quantidade de memória utilizada, mais tarde, passou a significar velocidade de processamento, mas com a formalização de desempenho, eficiência, no contexto numérico, passou a significar a qualidade do resultado, independente da máquina utilizada e uma otimização dos cálculos.

Desempenho diz respeito à velocidade com que o resultado é calculado e os recursos utilizados para o cálculo (quantidade de memória, tempo de processamento, número de CPUs necessários, graus de vetorização do algoritmo). Não é fácil separar estas duas características, mas no contexto deste trabalho, eficiência está relacionado a qualidade numérica (exatidão) enquanto que desempenho está relacionado a otimização do tempo necessário para a solução do problema.

Cabe ainda, formalizar a idéia de validação e o que isto implica. Validação tem que se certificar quanto: ao modelo para que não seja incorreto; a aritmética, para que não cause surpresas de imprecisão e ao software para que não tenha erros. A incerteza do modelo diz respeito às teorias, dados e modelo incorretos, que não reflitam a realidade corretamente, devido ao problema ser mal condicionado, por exemplo. A incerteza numérica se refere à propagação do erro de arredondamento (devido a aritmética) e suas consequências no resultado por causa da instabilidade do algoritmo. A incerteza do software se refere à possibilidade de softwares ainda conterem defeitos do compilador refletidos numa execução incorreta do programa.

3 ARITMÉTICA DE ALTO DESEMPENHO

Para a caracterização de aritmética de alto desempenho foi desenvolvido um estudo sobre suas necessidades, as quais são brevemente relatadas aqui. Como protótipo desta aritmética de alto desempenho, foi desenvolvido um estudo, uma especificação e, posteriormente, implementado uma biblioteca de rotinas intervalares no supercomputador

Cray Y-MP2E, denominada *libavi.a*. O nome *libavi.a* significa biblioteca (*lib*) composta da aritmética vetorial intervalar (*avi*). O sufixo *.a* é o sufixo padrão de bibliotecas no Cray.

Para a especificação da *libavi.a*, foi necessário o estudo da aritmética computacional, especialmente do padrão de aritmética binária de ponto-flutuante denominado IEEE 754, o qual estabeleceu as operações aritméticas com máxima exatidão, onde o resultado das operações aritméticas diferem de no máximo meia unidade do último dígito da mantissa do valor exato. Para isto, são necessárias diferentes formas de controlar o erro de arredondamento. Entre estas formas, estão os arredondamentos direcionados que gozam de certas propriedades, que possibilitam a definição de operações pela regra geral[3], o que garante uma estrutura algébrica mínima das operações aritméticas definidas de acordo com esta regra sobre o conjunto de ponto-flutuante.

No ambiente vetorial, como do supercomputador Cray Y-MP2E com a linguagem de programação Fortran 90, a aritmética não segue o padrão da IEEE 754, na especificação do tamanho da palavra e nem na forma como os arredondamentos e operações aritméticas em ponto-flutuante são efetuadas, foi necessário desenvolver rotinas que simulassem tais operações.

Verificou-se, também, as limitações do processamento vetorial na realização do produto escalar e de somas acumuladas. As diferenças nos resultados foram identificadas como resultantes da forma como os cálculos são efetuados. As diferenças de resultados foram encontradas não só em relação a máquinas diferentes, mas também entre o processamento sequencial e vetorial. Para solucionar esta limitação seria necessário acumuladores longos, de precisão maior do que a disponível no Cray ou, então, a simulação por software da aritmética inteira de precisão infinita. Infelizmente isto ultrapassou os objetivos desta pesquisa, apesar de estabelecer certas limitações à biblioteca de rotinas intervalares quanto à aritmética de alta exatidão e ao desenvolvimento de algoritmos que verificam automaticamente o resultado.

Em relação ao tempo de execução, estes métodos podem ser mais demorados do que os métodos reais disponíveis nas bibliotecas matemáticas e científicas da Cray, mas se sabe que os resultados produzidos por ela são confiáveis, pois são verificados automaticamente, além do fato de que são utilizados as operações reais e as rotinas disponíveis no Cray, para se implementar as operações envolvendo intervalos.

A aritmética de alto desempenho, implementada na biblioteca de rotinas intervalares (*libavi.a*) é projetada para viabilizar o uso de supercomputadores na solução de problemas físicos, químicos, das engenharias entre outros problemas que necessitem alta exatidão. Para tanto, além da aritmética vetorial intervalar (operações, funções e avaliação de expressões), tem-se que providenciar bibliotecas que tornem disponíveis a estes usuários os métodos intervalares para solução de seus problemas.

---

[3]ver [KUL88]

Através destes estudos se concluiu que o avanço prático do uso da matemática intervalar na resolução de problemas reais está, em parte, limitado pela inexistência de ferramentas computacionais que possibilitem um uso efetivo de intervalos. Observou-se que algumas das ferramentas existentes são limitadas, pois o porte dos problemas que se pode resolver é pequeno. Isto dificulta a disseminação prática e o uso de intervalos nas engenharias e em outras áreas.

Portanto, para que se possa obter uma aritmética de alto desempenho, esta deve ser estendida ainda com outros elementos. Todas as operações com números de ponto-flutuante devem ser supridas de arredondamentos direcionados, isto é, arredondamentos para baixo (para o número de máquina anterior), para cima (para o número de máquina posterior) e simétrico (para o número de máquina mais próximo). Uma aritmética intervalar para números reais e complexos em ponto-flutuante pode ser construída com estas operações.

As operações sobre dois intervalos no computador resultam de operações sobre dois extremos apropriadamente escolhidos dos operandos intervalares, o cálculo do extremo inferior é arredondado para baixo e o cálculo do extremo superior para cima. Dessa forma, o resultado certamente contém todos os resultados da operação aplicada individualmente aos elementos do primeiro e do segundo operandos.

3.1 Aritmética Intervalar de Máquina

O uso da aritmética intervalar foi desenvolvido inicialmente por Moore ([MOO66]). Este ramo da matemática veio se desenvolvendo desde então. A denominação matemática intervalar foi proposta em 1974 por Leslie Fox, combinando diferentes áreas como: aritmética intervalar, análise intervalar, topologia intervalar, álgebra intervalar entre outras. Ela trata com dados na forma de intervalos numéricos. Ela surgiu com o objetivo de automatizar a análise do erro computacional e trouxe uma nova abordagem que permite um controle de erros com limites confiáveis, além de provas de existência e não existência da solução de diversas equações.

Intervalos podem ser aplicados para representar valores desconhecidos ou para representar valores contínuos que podem ser conhecidos ou não. No primeiro sentido, servem para controlar o erro de arredondamento, para representar dados inexatos, para representar aproximações e erros de truncamento de procedimentos, como consistência lógica de programas, como critério de parada de processos iterativos e como critério de parada de métodos de contração. No segundo sentido servem, por exemplo, para testes de verificação computacional para a existência e unicidade de soluções de sistemas não lineares. A condição necessária não é verificada manualmente, mas automaticamente pelo computador.

O uso da matemática intervalar sofre críticas em função de algumas vezes poder resultar intervalos pessimistas (demasiadamente grandes) e que se gasta muito tempo de máquina. Estas duas críticas são facilmente refutadas, uma vez que com a aritmética avançada os resultados são produzidos com máxima exatidão. Quanto ao tempo de

processamento, depende da forma como foi implementada. Existem várias formas, tanto via software quanto via hardware.

Uma abordagem detalhada sobre aritmética intervalar pode ser encontrada nos livros textos básicos da área, como os de Moore, os de Kulisch, o de Alefeld e Herzberger ou o de Neumaier, ou ainda, o de Claudio. Aqui só serão revistos os conceitos básicos. (Confira em [MOO66, MOO79], [KUL81, KUL83, KUL88], [ALE83], [NEU90] e [CLA89]).

Na matemática intervalar, em vez de se aproximar um valor real x para um número de máquina, ele é aproximado por um intervalo X, que possui, como limite inferior e superior, números de máquina, de forma que o intervalo contenha x. O tamanho deste intervalo pode ser usado como medida para avaliar a qualidade de aproximação. Os cálculos reais são substituídos por cálculos que usam a aritmética intervalar.

Pode-se justificar o uso de intervalos numéricos pelo fato da aritmética real não ser suficientemente confiável em muitos problemas relacionados à Física e a Química Experimental. Nestas áreas, nem sempre é possível efetuar medições de forma exata, mas sim dentro de uma certa faixa, que pode ser representada através de intervalos.

Para garantir a máxima exatidão em operações intervalares, é necessário definir as operações intervalares sobre o conjunto de intervalos de ponto-flutuante (∇^+, ∇^-, ∇^*, $\nabla^/$, Δ^+, Δ^-, Δ^* e $\Delta^/$). Estas operações seriam definidas com o auxilio dos arredondamentos monotônicos direcionados para baixo ∇ e para cima Δ. Elas são resumidas a seguir.

Pseudo Inverso aditivo: $-X = [-x2, -x1]$

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:
$$X*Y = [\ \min\{\ x1\nabla^* y1,\ x1\nabla^* y2,\ x2\nabla^* y1,\ x2\nabla^* y2\},\ \max\{\ x1\ \Delta^*y1,\ x1\ \Delta^*y2,\ x2\ \Delta^*y1,\ x2\ \Delta^*y2\}]$$

Divisão:
$$X/Y = [\ \min\{\ x1\nabla^/ y1,\ x1\nabla^/ y2,\ x2\nabla^/ y1,\ x2\nabla^/ y2\},\ \max\{\ x1\Delta^/ y1,\ x1\Delta^/ y2,\ x2\Delta^/ y1,\ x2\Delta^/ y2\}]$$

Intersecção: $X \cap Y = \begin{cases} [\max\{x1,y1\}, \min\{x2,y2\}], & \text{se } x1<y2 \text{ e } y1\le x2 \\ \varnothing & \text{caso contrário.} \end{cases}$

União: $X \cup Y = \begin{cases} [\min\{x1,y1\}, \max\{x2,y2\}], & \text{se } X\cap Y \neq \varnothing \\ \text{erro} & \text{caso contrário.} \end{cases}$

União convexa: $X \veebar Y = [\min\{x1,y1\}, \max\{x2,y2\}]$

Menor valor absoluto: $\langle X\rangle = \min\{|x| / x \in X\}$

Maior valor absoluto: $|X| = \max\{|x1|, |x2|\}$

Valor absoluto: $Abs(X) = [\langle X\rangle, |X|]$

Ponto médio: $m(X) = (x1+x2)/2$

Raio do Intervalo: $r(X) = (x2-x1)/2$

Distância entre dois intervalos: $q(X,Y) = \max\{|x1-y1|, |x2-y2|\} \ge 0$

Diâmetro: $d(X) = x2-x1$

Diâmetro Relativo: $d_{rel}(X) = d(X)/\langle X\rangle = (x2-x1)/\langle X\rangle$

Relação de igualdade: $X=Y \Leftrightarrow (x1=y1)$ e $(x2=y2)$

Desigualdade: $X\neq Y \Leftrightarrow (x1\neq y1)$ ou $(x2\neq y2)$

Demais relações:

X menor que Y,	$X < Y \Rightarrow$	$x2 < y1$	
X maior que Y,	$X > Y \Rightarrow$	$x1 > y2$	
X está contido em Y,	$X \subseteq Y \Rightarrow$		$y1 \leq x1$ e $x2 \leq y2$
X está contido propriamente em Y,	$X \subset Y \Rightarrow$		$y1 \leq x1$ e $x2 \leq y2$ e $(x1 \neq y1$ ou $x2 \neq y2)$
X é interior de Y,	$X \overset{\circ}{\subseteq} Y \Rightarrow$		$y1 < x1$ e $x2 < y2$
X contém Y,	$X \supseteq Y \Rightarrow$		$x1 \leq y1$ e $y2 \leq x2$
X contém propriamente Y,	$X \supset Y \Rightarrow$		$x1 \leq y1$ e $y2 \leq x2$ e $(x1 \neq y1$ ou $x2 \neq y2)$
x pertence a X,	$x \in X \Rightarrow$	$x1 \leq x \leq x2$	
X intersecção Y é vazia	$X \cap Y = \emptyset \Rightarrow$	$x2 < y1$ ou $y2 < x1$	

A principal vantagem da utilização da aritmética intervalar de máquina é a determinação de um intervalo que contém a solução do problema que se pretende resolver.

3.2 Processamento Vetorial

Processamento vetorial ou vetorização tem diferentes significados para diferentes tipos de pessoas. Para usuários, vetorização significa a introdução de instruções vetoriais de *hardware* em seus programas, para que a alta velocidade destas instruções possam ser utilizadas de forma a melhorar o desempenho de seus programas. Adotando este ponto de vista, mostra-se como se pode obter a vetorização e uma caracterização mais formal das instruções vetoriais.

O objetivo primário da vetorização é achar operações seqüenciais que podem ser convertidas em operações vetoriais semanticamente equivalentes, fazendo uso de vantagens do *hardware* vetorial. Em geral, um comando pode ser vetorizado se ele não requiser entrada na iteração de uma variável do *loop*, a qual é computada anteriormente no *loop* (dependência verdadeira).

O objetivo final da vetorização de um *loop* é produzir instruções de máquina que executem grupos de elementos de dados em registradores vetoriais. Na forma simples, um *do-loop* é transformado de uma iteração sobre elementos simples para uma iteração sobre grupos ou seções de elementos.

Pode-se obter a vetorização recompilando os códigos escalares em um compilador que os vetorize automaticamente; pelo reestruturar do código fonte, auxiliando o compilador de forma a se obter um maior grau de vetorização e pelo desenvolvimento de um novo algoritmo para colher o maior benefício das características vetoriais da máquina.

O processamento vetorial elimina a maioria dos testes e controle de operações, principalmente as associadas a *loops*. Ele minimiza o tempo dispendido na espera da carga dos operandos e de armazenagem do resultado, ao referenciar grupos de locação de memória e pelo uso de registradores de múltiplos elementos. Outra vantagem do processamento vetorial é que ele reduz o número de instruções, em linguagem de máquina, que precisa ser carregado, decodificado e executado.

Este estudo foi desenvolvido, para auxiliar o projeto e implementação da ferramenta *libavi.a*, que reune as características da matemática intervalar e do processamento vetorial num conjunto de rotinas intervalares. A seguir esta ferramenta será descrita em mais detalhes.

4 BIBLIOTECA DE ROTINAS INTERVALARES *LIBAVI.A*

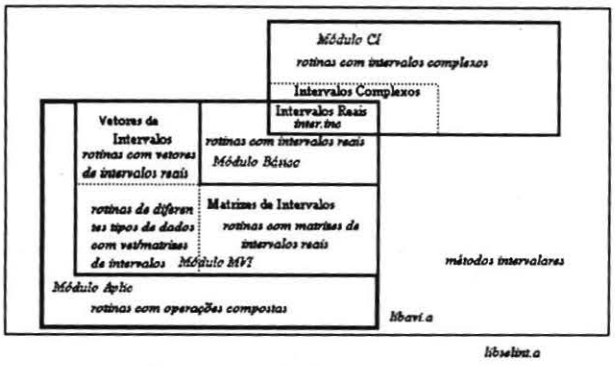

Fig.1 Hierarquia da *libavi.a*

A biblioteca *libavi.a* é um conjunto de rotinas intervalares que reúne as características da matemática intervalar no ambiente do supercomputador vetorial Cray Y-MP. A *libavi.a* foi desenvolvida em Fortran 90, o que possibilitou as características de modularidade, sobrecarga de operadores e funções, uso de *arrays* dinâmicos na definição de vetores e matrizes e a definição de novos tipos de dados próprios a análise matemática.

Como pode ser observado na figura 1, a biblioteca foi organizada em quatro módulos, estes são: *básico, mvi, aplic* e *ci*. O módulo *básico* contém a aritmética intervalar básica e, por isto é utilizado por todos os demais módulos. O módulo *aplic* contém os demais módulos, pois ele se utiliza deles. O módulo de intervalos complexos, contém o módulo *básico*. Os tipos vetores e matrizes de intervalos complexos foram definidos, mas não foram desenvolvidas as rotinas que os manipulam. Elas são a extensão do módulo *mvi* para complexos, o que na figura 1 representaria um retângulo incluindo *mvi* e *ci*. A seguir é descrito o conteúdo de cada um dos módulos:

- módulo ***básico*** - intervalos reais - 53 rotinas. Estas rotinas foram organizadas dentro deste módulo em seis conjuntos, de acordo com o tipo de operação que realiza. Esses conjuntos são: funções de transferência, operações relacionais, operações entre conjuntos, operações aritméticas, funções básicas e rotinas de entrada e saída.

- módulo ***mvi*** - matrizes e vetores de intervalos- 151 rotinas. Neste módulo são implementadas as operações entre vetores de intervalos, matrizes de intervalos e as rotinas que manipulam diferentes tipos de dados com intervalos, vetores e matrizes de intervalos, portanto

existem três seções neste módulo. Nas duas primeiras, as que implementam as operações com vetores e com matrizes de intervalos, são compostas, cada uma, por seis conjuntos de rotinas. Esses conjuntos são: funções de transferência, operações relacionais e entre conjuntos, operações aritméticas, funções básicas elementares, funções pré-definidas e rotinas de entrada e saída. A terceira seção, onde estão operações entre diferentes tipos de dados, existem sete conjuntos de operações, esses são: operações de reais com intervalos, operações de reais com vetores de intervalos, operações de reais com matrizes de intervalos, operações de intervalos com vetores de intervalos, operações de intervalos com matrizes de intervalos, operações entre vetores e matrizes de intervalos e operações entre vetor/matriz real com vetor/matriz de intervalos.

• módulo ***ci*** - intervalos complexos - 58 rotinas. Este módulo implementa intervalos complexos. Da mesma forma que o módulo básico, ele foi organizado em seis conjuntos de rotinas que são: funções de transferência, operações relacionais, operações entre conjuntos, operações aritméticas, funções elementares entre intervalos complexos e rotinas de entrada e saída.

• módulo ***aplic*** - aplicações da álgebra linear - 29 rotinas. Este módulo estende algumas das rotinas reais da biblioteca BLAS para intervalos. Este conjunto de rotinas incluem operações entre vetores, entre matriz e vetores e entre matrizes da álgebra linear. Estas rotinas incluem os três níveis da BLAS. Estão incluídas algumas rotinas que determinam normas de matrizes e vetores e condicionamento.

Além da aritmética vetorial intervalar (operações, funções e avaliação de expressões), sentiu-se a necessidade de providenciar bibliotecas que tornassem disponíveis a estes usuários os métodos intervalares para solução de seus problemas. Inicialmente, foi especificada a biblioteca científica aplicativa *libselint.a,* composta por algumas rotinas intervalares de resolução de equações algébricas e sistemas de equações lineares. O que na figura 1, representa o retângulo mais externo. Esta biblioteca intervalar aplicativa, *libselint.a* seria composta por quatro módulos, são estes:

• Módulo ***dirint***, incluir os métodos baseados em operações algébricas intervalares e propriedades intervalares, também são conhecidos como métodos diretos intervalares.

• Módulo ***refint***, incluir os métodos baseados em inclusões ou refinamentos intervalares da solução e do erro, também podem ser chamados de métodos híbridos, uma vez que se pode utilizar a solução inicial calculada por métodos pontuais ou a matriz inversa pontual.

• Módulo ***itrint***, incluir os métodos iterativos intervalares, conhecidos também como métodos de relaxação. Eles também se baseam em inclusões monotônicas.

• Módulo ***equalg***, deve conter rotinas do caso particular de sistema linear de ordem um, ou seja, resolução algébrica de equações por métodos intervalares, como as versões intervalares de Newton.

Observa-se que desta biblioteca aplicativa foram implementados apenas algumas rotinas visando verificar e validar o uso da biblioteca intervalar. Está sendo pesquisado sobre metodologias de desenvolvimento e métodos intervalares de resolução de equações e sistemas lineares. Os resultados desta pesquisa serão alvo de futuros trabalhos.

5 CARACTERÍSTICAS DA *LIBAVI.A*

Como o propósito do desenvolvimento desta biblioteca foi o de providenciar uma ferramenta útil para programação científica (em ambiente vetorial), as rotinas implementadas necessitavam de certas características como: exatidão, eficiência, confiabilidade, validação, facilidade de uso, boa documentação, modificabilidade, robustez e transportabilidade.

A exatidão destas rotinas foi obtida pelo uso das rotinas disponíveis no Fortran 90 para manipulação de números reais (números em ponto-flutuantes), especialmente as funções básicas elementares. Utilizou-se para melhorar a exatidão algumas rotinas em dupla precisão, especialmente para o cálculo do produto escalar mais próximo do ótimo. Utilizou-se, ainda, algumas das rotinas da biblioteca BLAS para a implementação das operações entre vetores e matrizes de intervalos. A extensão para rotinas intervalares se deu através da aplicação de princípios de conversão real-intervalo e das propriedades de semimorfismo. Manteve-se a exatidão disponível do supercomputador Cray para processamento científico, mas se acrescentou com intervalos a garantia dos resultados.

Em relação à eficiência, evoluiu-se mais neste conceito, pois incialmente eficiência significava um uso otimizado de memória, por esta ser muito cara e escassa. Com a evolução tecnológica, eficiência se vinculou a idéia de velocidade de processamento. Perdeu-se um pouco da idéia de qualidade para se ter velocidade. Com a matemática intervalar (acrescida de aritmética de alta exatidão) é resgatado o conceito de eficiência como qualidade e garantia do resultado calculado com rapidez. A biblioteca *libavi.a* produz resultados um pouco mais lentos do que a aritmética de ponto-flutuante ordinária, uma vez que os cálculos são efetuados para os extremos dos intervalos, mas ganha-se na garantia dos resultados, uma vez que o valor exato está contido no intervalo solução.

As qualidades de confiabilidade e validação, dizem respeito à biblioteca ser consistente com a documentação, ou seja, a execução do programa realiza exatamente que seu propósito descrito na documentação define, resolvendo a classe de problemas que se propõem.

Sobrecarga de funções e operadores é uma das facilidades da biblioteca de rotinas intervalares. A sobrecarga possibilita que o usuário utilize o nome genérico de uma função, independente do tipo de argumentos desta função. A tarefa de identificar a rotina correta que opera com tais argumentos é transferida ao compilador. Isto facilita a programação. Foram sobrecarregados os símbolos das operações aritméticas de soma, subtração, multiplicação, divisão e os símbolos relacionais existentes no Fortran 90. As demais funções matemáticas foram sobrecarregadas ao nome usual da notação matemática.

Os uso de *arrays* dinâmicos na definição de vetores e matrizes de intervalos contribui para a simplificação e facilidade de uso da biblioteca. *Arrays* dinâmicos permitem não só um uso otimizado da memória, através do uso de vetores e matrizes de tamanho

aleatório, garantindo alocação e liberação do espaço da memória, mas também facilitam a programação uma vez que o usuário apenas utiliza a sua definição segundo a compatibilidade do sistema.

A otimização e vetorização das rotinas da *libavi.a* se deram como que herança das características da máquina, do compilador e das rotinas que são utilizadas das bibliotecas, como as rotinas da BLAS. O uso destas rotinas, não só proporcionou esta vantagem, mas também proporcionou a mesma qualidade em termos de exatidão dos resultados, adicionado ao uso de intervalos, o que proporciona resultados contidos dentro do intervalo solução.

A modularidade é outra característica da biblioteca de rotinas intervalares. Em cada módulo são definidas as operações que manipulam com certo tipo de elementos ou dados. Isto facilita o entendimento, a manutenção e produz maior estruturação na biblioteca.

6 CONCLUSÕES

Este artigo apresentou a aritmética de alta exatidão e alto desempenho, onde são formalizados as características de eficiência e alto desempenho no contexto de processamento numérico. É demonstrado ainda a importância da qualidade com que as operações em ponto-flutuante são efetuadas, pois em computações de larga escala os erros se acumulam podendo gerar resultados errados.

Introduziu-se a biblioteca de rotinas intervalares *libavi.a*, a qual foi desenvolvida como protótipo desta aritmética de alto desempenho. Portanto, com a *libavi.a* se definiu a aritmética de alto desempenho, composta do processamento de alto desempenho (vetorial) e da matemática intervalar. Não se tem a aritmética de alta exatidão e alto desempenho pois no ambiente do supercomputador Cray Y-MP2E (com a linguagem de programação Fortran 90), a aritmética de ponto-flutuante não segue o padrão da IEEE 754 nem na especificação do tamanho da palavra e nem na forma como os arredondamentos e operações aritméticas são efetuadas. Foi necessário desenvolver rotinas que simulassem tais operações, como por exemplo, as rotinas *infla_down* e *infla_up* que proporcionam um controle do erro de arredondamento nas rotinas numéricas da biblioteca, proporcionando que os resultados das operações estejam entre limites confiáveis.

Outras providências que seriam necessárias para melhorar a qualidade do resultado são: aritmética de alta exatidão, garantida pelos arredondamentos direcionados para cima e para baixo, pela definição das operações aritméticas básicas segundo as regras de semimorfismo, pelo uso da matemática intervalar e do cálculo do produto escalar com máxima exatidão e o uso algoritmos com verificação automática do resultado. Tais melhorias estão sendo estudadas.

Está aberto um amplo campo de aplicações da matemática intervalar, pois com a biblioteca de rotinas intervalares *libavi.a* disponível aos usuários do supercomputador

Cray Y-MP2E do Centro de Supercomputação da UFRGS, os usuários tem a possibilidade de desenvolverem programas para resolver seus próprios problemas nas mais diversas áreas, tendo seus resultados entre limites confiáveis, dentro do intervalo solução.

7 Referências Bibliográficas

[ADA93] ADAMS, E.; KULISCH, U. (Eds.). **Scientific Computing with automatic result verification.** San Diego: Academic Press, 1993. 612p.

[ALE83] ALEFELD, G.; HERZBERGER, J. **Introduction to interval computation.** New York: Academic Press, 1983. 333p.

[CLA89] CLAUDIO, D. M.; MARINS, J. M. **Cálculo numérico computacional.** São Paulo: Atlas, 1989. 464p.

[CRA93] CRAY RESEARCH, Inc. CF90 Fortran language reference manual., n.sr.3902, versão 1.0. [S.l : s.n.], 1993.

[DIV94] DIVERIO, T. A.; HOLBIG, C. A.; NORONHA, C. R. **Sistema de Ponto-Flutuante e o Padrão IEEE 754.** Porto Alegre: PGCC da UFRGS, 1994. 58p. (RP, 225).

[DIV95a] DIVERIO, T. A.; FERNANDES, U. A. L.; DAHMER, A. **Limitações do Processamento vetorial no Cray Y-MP2E.** Porto Alegre: PGCC da UFRGS, 1995. 60p. (RP, 248).

[DIV95b] DIVERIO, T. A. ***LIBAVI.A* Manual de utilização.** Porto Alegre: PGCC da UFRGS, 1995. 350p.

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

[KUL81] KULISCH, U. W.; MIRANKER, W. L. **Computer arithmetic in theory and practice.** New York: Academic Press, 1981. 249p.

[KUL83] KULISCH, U. W.; MIRANKER, W. L. **A new approach to scientific computation.** New York: Academic Press, 1983. 384p.

[KUL87] KULISCH, U.; MIRANKER, W. R. **Computer arithmetic theory and pratice.** New York: Academic Press, 1987.

[KUL88] KULISCH, U.; STETTER, H. J. (Eds.). Scientific computation with automatic result verification. Wien: Springer Verlag, 1988. (COMPUTING SUPPLEMENTUM 6, Seminar held in Karlsruhe, Sept 30- Oct 2, 1987).

[MOO66] MOORE, R. E. **Interval Analysis.** Englewood Cliffs: Prentice Hall, 1966.

[MOO79] MOORE, R. E. **Methods and applications for interval analysis.** Philadelphia: SIAM, 1979. 190p.

[NEU90] NEUMAIER, A. **Interval methods for systems of equations.** Cambridge: Cambridge University Press, 1990. 260p.