UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Biblioteca de Matemática Intervalar para Android

Thiago Davison Gonçalves Gonçalves

Pelotas, 2016

Thiago Davison Gonçalves Gonçalves

Biblioteca de Matemática Intervalar para Android

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Computação

Orientadora: Profª. Drª. Aline Brum Loreto

Pelotas, 2016

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

G635b Gonçalves, Thiago Davison

Biblioteca de matemática intervalar para Android / Thiago Davison Gonçalves ; Aline Brum Loreto, orientadora. — Pelotas, 2016.

80 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2016.

1. Matemática intervalar. 2. Android OS. 3. Plataforma Android. 4. Biblioteca intervalar. I. Loreto, Aline Brum, orient. II. Título.

CDD : 005

Elaborada por Aline Herbstrith Batista CRB: 10/1737

AGRADECIMENTOS

Primeiramente, quero agradecer a Deus, pois sei que sem ele nada poderia ter sido realizado.

Quero agradecer a minha família, meus pais principalmente minha mãe que sempre me deu todo o apoio para me dedicar aos estudos e suporte financeiro de forma a nunca deixar faltar nada, aos meus irmãos pelo carinho e por todas as coisas que compartilhamos ao longo dos anos. Um agradecimento especial para minha esposa Bruna, por todo o incentivo para me dedicar a este trabalho.

Agradeço a minha orientadora Profª. Drª. Aline Brum Loreto, pela dedicação, paciência e grande incentivo na realização deste trabalho, pois tenho certeza que o trabalho desenvolvido não teria sido o mesmo sem as suas participações.

Agradeço a todos os colegas do mestrado, especialmente ao Leonardo Soares, Mauricio Balboni e Lucas Tortelli, pois com certeza contribuíram para realização deste trabalho, pelas horas de estudos, conversas a respeito dos trabalhos e datas importantes a serem lembradas.

Talvez não tenha conseguido fazer o melhor,
mas lutei para que o melhor fosse feito.
Não sou o que deveria ser, mas
Graças a Deus, não sou o que era antes
— MARTHIN LUTHER KING

RESUMO

GONÇALVES, Thiago Davison Gonçalves. **Biblioteca de Matemática Intervalar para Android**. 2016. 80 f. Dissertação (Mestrado em Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2016.

As tecnologias móveis atualmente vem conquistando um espaço significativo e imprescindível no cotidiano das pessoas, das empresas e das indústrias. Mas quanto ao desenvolvimento de aplicações para dispositivos móveis a computação numérica ainda é muito pouco explorada, o que nos traz um problema: a qualidade dos resultados gerados. Sendo assim, o objetivo deste trabalho é o desenvolvimento de uma biblioteca contendo operações da matemática intervalar para Android, visando proporcionar um ambiente de Alta Exatidão no desenvolvimento de aplicativos móveis para esta plataforma. Este trabalho contempla o estudo e análise dos sistemas operacionais móveis líderes de mercado atualmente e suas ainda inexploradas relações com ambientes de Alta Exatidão, tendo em vista a pouca ou nenhuma utilização de técnicas de matemática intervalar para esses sistemas operacionais móveis e linguagens nativas de desenvolvimento móvel. O desenvolvimento da biblioteca IntDroid fornecerá um pacote de recursos com o tipo de dado Intervalo e operações intervalares no Ambiente de desenvolvimento Android, apresentando dados de exatidão de suas operações e utilização de recursos de memória e processamento consumidos por essas operações no ambiente móvel, sendo este prerrogativa de funcionamento de aplicações móveis. A biblioteca IntDroid possui como propósito garantir a qualidade dos dados gerados no desenvolvimento de aplicações desta plataforma, tendo como resultado o preenchimento de uma lacuna de ferramentas de computação científica para plataforma Android.

Palavras-chave: Matemática Intervalar, Android OS, Plataforma Android, Biblioteca Intervalar.

ABSTRACT

GONÇALVES, Thiago Davison Gonçalves. **Library with Interval Math for Android**. 2016. 80 f. Dissertação (Mestrado em Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2016.

The currently mobile technologies come conquering a significant and indispensable space in the day of people, enterprise and industry. But regarding the application development for mobile devices to numerical computing still little explored, what we bring a problem: the quality of results generated. So, the objective of this work it is the development of a library containing interval mathematics operations for Android, in order to provide a high accuracy of environment for application development mobile to this platform. This work include a study and analysis of Operating Systems Mobile market leaders today and he's still unexplored relations with high accuracy environments with a view of a little or no use of mathematical interval techniques for this Operating Systems Mobile and he's Languages Mobile Development. The Development of IntDroid library will provide the work with the data type interval and interval operations in the Android development environment, featuring accuracy data of this operations and the use of Memory and processing resources consumed by these operations in the mobile environment, that is a prerogative of Mobile Applications. The IntDroid library has to purpose guarantee the data quality generated in the Application Development from this platform, having with this results, filling a gap of scientific computing tools to Android platform.

Keywords: Interval Math, Android OS, Library of Interval Math, Android Platform.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
Wi-Fi	*Wireless Fidelity*
XML	*eXtensible Markup Language*
O.S.	*Operating System*
O.O.	*Object Oriented*
S.D.K	*Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO

Os dispositivos móveis com seus processadores de última geração, com muitos núcleos (do inglês *multicores*) receberam um alto poder computacional e se tornaram capazes de realizar muitas tarefas que em um passado recente só poderiam ser realizadas pelo hardware de computadores desktop ou servidores. Mas quanto ao desenvolvimento de aplicações para dispositivos móveis a computação numérica ainda é muito pouco explorada, o que nos traz um problema sobre a qualidade dos dados gerados (SANTIAGO; BEDREGAL; ACIÓLY, 2006).

Não se pode afirmar a exatidão dos resultados estimados sem auxílio de uma análise de erros e as técnicas intervalares surgem como solução, pois computam um intervalo, com a garantia de que a resposta pertence a este intervalo.

Utilizando-se intervalos para representação dos números reais, é possível controlar a propagação de erros de arredondamento ou truncamento, entre outros, em procedimentos numéricos computacionais (GARROZI, 2009). Portanto, resultados intervalares carregam consigo a segurança de sua qualidade.

Na medida em que surgem novas tecnologias e as antigas vão sendo aperfeiçoadas, chega-se a um ponto em que a velocidade do processamento das informações deixou de ser uma preocupação. A preocupação agora é representar essas informações de forma cada vez mais precisa.

Uma das propostas desenvolvidas para controlar os erros computacionais foi a matemática intervalar (MOORE, 1966). Esta aritmética tem por objetivo solucionar problemas que se concentram fundamentalmente em dois aspectos: na criação de um modelo computacional que reflita sobre o controle e análise dos erros que ocorrem no processo computacional; e, na escolha de técnicas de programação adequadas para desenvolvimento de softwares científicos buscando minimizar os erros nos resultados (GARROZI, 2009).

Sabe-se que quando se trabalha com números de ponto flutuante o resultado é apenas uma aproximação de um valor real e erros gerados por arredondamentos, por instabilidade dos algoritmos ou programas, podem levar a resultados com valores incorretos. O presente trabalho visa o desenvolvimento de uma biblioteca contendo

operações da matemática intervalar para Android, proporcionando um ambiente de Alta Exatidão no desenvolvimento de aplicativos móveis para esta plataforma.

Para fundamentar o desenvolvimento da biblioteca, foi necessário um estudo sobre a matemática intervalar, seus conceitos, suas operações e as soluções já construídas para vários ambientes e diversas linguagens de programação, entendendo o conceito do trabalho com o tipo de dado intervalo e suas abordagens.

A biblioteca IntDroid implementa o tipo de dado intervalo e as operações sobre o tipo. A versão inicial, executa operações entre intervalos com suporte aos arredondamentos direcionados para cima e para baixo, reconhecendo entrada de intervalos e realizando operações aritméticas e topológicas sobre os mesmos.

Através da execução das rotinas de cálculos da biblioteca IntDroid pode-se identificar a utilização de recursos e o desempenho das operações implementadas em IntDroid no Android OS, além dos consumos de recursos de memória e hardware.

A escolha de trabalhar com o Android OS deve-se ao fato do ambiente ser um sistema operacional de código aberto(do inglês, *Open Source*), amigável a comunidade de desenvolvedores e com grande aceitação de soluções à serem integradas ao Android SDK. As linguagens Java e XML são amplamente difundidas no meio acadêmico e científico, porém sendo o Sistema Operacional Android *Open Source* torna-o muito mais fácil o seu estudo, requisito muito importante para futuras implementações científicas deste trabalho. A fluidez do sistema Android com a evolução e constante atualização, sempre em busca da otimização de seus recursos em diferentes tipos de hardwares, também são fatores que justificam a escolha por este sistema. Outro ponto muito importante desta dissertação são as otimizações da máquina virtual criada e desenvolvida especificamente para o Sistema Operacional Android e o ambiente Linux Kernel, amplamente difundido na comunidade acadêmica.

O objetivo deste trabalho é o desenvolvimento de uma biblioteca, identificada por IntDroid, contendo operações da matemática intervalar para Android, visando proporcionar um ambiente de Alta Exatidão no desenvolvimento de aplicativos móveis para esta plataforma. O resultado e o desempenho das rotinas implementadas em IntDroid, são comparadas aos resultados das bibliotecas Java-XSC(FERREIRA; FERNANDES; CAMPOS, 2005), C-XSC(R. KLATTE U. KULISCH; RAUCH, 1993) e IntPy(VARJÃO, 2011). As informações obtidas dos resultados das operações e do processo de cálculos intervalares em hardware móvel são analisadas tendo em vista o alto controle de utilização dos recursos dentro do Sistema Operacional Android. O Android OS possui o controle sobre o comportamento da biblioteca permitindo que ela funcione em background quando os cálculos executados atingirem o limite de recurso de memória e processamento disponível pelo OS para o aplicativo. Este trabalho tem como pretensão fomentar a utilização de Ambientes da Alta Exatidão para dispositivos móveis, que ainda é muito pouco ou nada explorada.

A dissertação possui a seguinte estrutura:

- O Capítulo 2 descreve os princípios e fundamentos da Matemática Intervalar, apresenta uma solução a falta de precisão apresentada pela aritmética de ponto flutuante, utilizando-se da matemática intervalar. A realização de cálculos que demandem de grande precisão utilizando as propriedades da matemática intervalar, contém as vantagens quanto a consistência dos valores numéricos obtidos, como também a garantia de sua incerteza.
- No Capítulo 3 são apresentados os principais Sistemas Operacionais móveis, suas estruturas, ambientes de desenvolvimentos e sua relação com a matemática intervalar.
- O Capítulo 4 descreve o desenvolvimento da biblioteca IntDroid e suas operações.
- O Capítulo 5 apresenta os resultados obtidos pela biblioteca utilizando hardware móvel e Android, comparando seus resultados com das biblioteca intervalares IntPy, Java-XSC e C-XSC.
- Por fim as conclusões obtidas com este trabalho.

2 MATEMÁTICA INTERVALAR

Este capítulo apresenta o estudo da Matemática Intervalar. Para a implementação da biblioteca IntDroid utilizou-se como base bibliotecas que possibilitam o uso de intervalos, bem como a aplicação da Matemática Intervalar na computação científica.

A matemática intervalar busca resolver problemas que se concentram fundamentalmente em dois aspectos: na criação de um modelo computacional que reflita no controle e análise dos erros que ocorrem no processo computacional, e na escolha de técnicas de programação adequadas para desenvolvimento de softwares científicos buscando minimizar os erros nos resultados(MOORE, 1966).

O usuário não pode afirmar a exatidão da resposta estimada sem o auxílio de uma análise de erro, que é extensa, dispendiosa e nem sempre viável. No entanto, a matemática intervalar busca dar suporte a estes problemas. Infelizmente, muitos objetos fundamentais para a resolução de problemas do dia-a-dia não são finitamente representáveis em máquinas. A solução de problemas desse tipo utiliza as conhecidas aproximações que induzem a erros, como o erro de aproximação que é a distância entre o irracional e sua aproximação racional. O problema maior reside no fato de que o erro de aproximação não obedece à qualquer lei durante as computações, o que pode levar a distorções em sua computação. O controle desse erro de aproximação durante as computações é feito por uma computação em paralelo, requerendo esforços computacionais extras.

Existem três fontes de erros em computação numérica, são elas: a propagação de erros de dados e parâmetros iniciais, erro de arredondamento e erro de truncamento(GARROZI, 2009). Na primeira fonte de erro, ao se tentar representar um fenômeno do mundo físico por meio de um modelo matemático, raramente se tem uma descrição correta deste fenômeno. Normalmente, são necessárias várias simplificações do mundo físico para que se tenha um modelo matemático com o qual se possa trabalhar, tomando-se apenas algumas grandezas, como por exemplo: tempo, temperatura, distância, intensidade, etc. Estas por sua vez são obtidas de instrumentos que tem precisão limitada de modo que a incerteza destes parâmetros iniciais levará consequentemente a incertezas dos resultados.

Quanto ao erro de arredondamento, para a resolução de modelos matemáticos, muitas vezes torna-se necessária a utilização de ferramentas ou softwares de cálculo como computadores digitais. Sabe-se que estes instrumentos de cálculo trabalham com números arredondados, ou seja, representam os números em uma forma finita de dígitos, de acordo com o seu sistema interno de representação e que tais limitações geram erros durante as computações numéricas. Finalmente os erros de truncamento, que são provenientes da utilização de processos que deveriam ser infinitos, ou muito grandes, para determinação de um valor e que, por razões práticas, são truncados. Estes processos infinitos são muito utilizados na avaliação de funções matemáticas, tais como exponenciação, logaritmos, funções trigonométricas e várias outras que uma máquina pode ter.

A limitação dos dados de entrada e a acumulação do erro de arredondamento em qualquer sequência finita de operações aritméticas podem ser ambas rigorosamente controladas, simplesmente pela utilização de matemática intervalar.

As definições básicas da aritmética intervalar, as quais são necessárias para o desenvolvimento deste trabalho, estão baseadas nos trabalhos de (MOORE, 1966), (ACIOLY, 1991), (OLIVEIRA; DIVERIO; MORAES, 1997) e (LORETO, 2006).

A aritmética intervalar é uma ferramenta necessária para a solução de problemas relacionados com erros numéricos na computação científica (LORETO, 2006). Pode ser compreendida como um sistema algébrico consistindo do conjunto formado de todos intervalos fechados da reta real e com algumas operações definidas sobre ele. Em vez da utilização de algoritmos usuais para a resolução de certos problemas matemáticos, são utilizados algoritmos com operações intervalares, que são desenvolvidos para ter intervalos como resultados, no lugar de números reais. Resultado este que deve conter a solução exata do problema original e a amplitude do intervalo obtido como resultado deve ser a menor possível (CAMPOS, 1997).

As seguintes Seções apresentam as operações intervalares que serviram de base para as operações implementadas na biblioteca IntDroid, sendo fundamentação teórica necessária para o desenvolvimento da biblioteca.

2.1 Definições

As principais definições da matemática intervalar são apresentadas nas seguintes sessões, assim como as operações intervalares, sendo fundamentação teórica necessária para o desenvolvimento da biblioteca.

2.1.1 Intervalo Real

É constituído como intervalo um par ordenado $[\underline{x}; \overline{x}]$, com $\underline{x} \leq \overline{x}$, representando todos os números reais x, tal que $\underline{x} \leq x \leq \overline{x}$. Cada valor x é a representação de um

valor real, tendo como limites do intervalo $\underline{x}$ e $\overline{x}$. São exemplos de intervalos: [7;9], [-5,12], [2,2]. Este último também reconhecido como intervalo pontual, pois seus limites correspondem ao mesmo valor numérico.

Todo intervalo é representado através das letras do alfabeto da língua portuguesa em maiúscula (A,B,C,...,Z), e seus limites inferior e superior representados pela mesma letra do intervalo, porém minúscula. Desta maneira o intervalo A contém como par de limites $[\underline{a}; \overline{a}]$.

2.1.2 Conjunto dos IR

De modo semelhante ao que ocorre na aritmética real, a matemática intervalar contém um conjunto responsável por representar todos os intervalos reais existentes, isto é, IR = $\{[\underline{x}; \overline{x}] | \underline{x}; \overline{x} \in \mathbb{R}, \underline{x} \leq \overline{x}\}$. Assim, segue a seguinte cadeia de inclusões $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R} \subseteq IR$.

Na Figura 1 nota-se que cada intervalo $[\underline{x}; \overline{x}] \in IR$, um ponto $(\underline{x}; \overline{x}) \in \mathbb{R}^2$, obtém-se uma representação geométrica de IR.

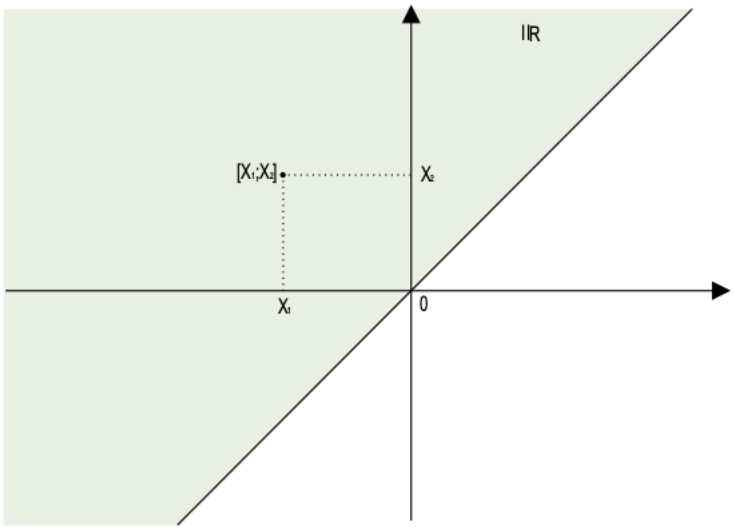

Figura 1: Representação geométrica de IR

2.1.3 Igualdade entre Intervalos

Dado dois intervalos X =$[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}] \in IR$ são ditos iguais, se e somente se $\underline{x} = \underline{y}$ e $\overline{x} = \overline{y}$.

2.1.4 Ordem da Informação

Sejam dois intervalos X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$, define que X $\subseteq$ Y $\iff$ $\underline{x} \leq \underline{y}$ e $\overline{y} \leq \overline{x}$. Dessa maneira, o intervalo não somente passa a representar o valor real x, mas contém toda informação sobre x.

2.1.5 Ordem de Inclusão

Sejam dois intervalos X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$, pela ordem de inclusão define-se que X $\subseteq$ Y $\iff$ $\underline{y} \leq \underline{x}$ e $\overline{x} \leq \overline{y}$.

2.2 Operações Aritméticas em IR

Sejam X,Y $\in IR$ dois intervalos de reais. As operações aritméticas básicas em IR são definidas através de X $*$ Y = $\{x * y | a \in X, y \in Y\}$, onde $* \in \{+, -, /, *\}$ é qualquer uma das quatro operações básicas da aritmética intervalar. Caso ω seja um operador unário aplicado ao intervalo X então: $\omega = \{\omega(x) | x \in X\} = [\omega(\underline{x}); \omega(\overline{x})]$.

Para realização da operação de divisão, toma-se como premissa básica que 0 $\notin$ Y, para correto funcionamento da operação.

2.2.1 Adição Intervalar

Sejam X, Y $\in$ IR dois intervalos reais com X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$, a adição intervalar de X + Y é dada por X + Y =$[(\underline{x} + \underline{y}); (\overline{x} + \overline{y})]$. Como exemplo, a Figura 2 demonstra graficamente a disposição dos intervalos X = [1; 2], Y = [3; 4] e a soma intervalar de X + Y = [1 + 3; 2 + 4] = [4; 6].

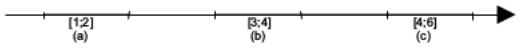

Figura 2: Soma intervalar entre intervalos

2.2.1.1 Propriedades Algébricas

Sejam X, Y e Z $\in$ IR. As seguintes propriedades algébricas se aplicam a adição intervalar em IR:

- **FECHAMENTO:** Se X $\in$ IR e Y $\in$ IR, então X + Y $\in$ IR;
- **ASSOCIATIVIDADE:** X + (Y + Z) = (X + Y) + Z;
- **COMUTATIVIDADE:** X + Y = Y + X;
- **ELEMENTO NEUTRO:** $\exists$0 = [0;0] $\in$ IR, tal que X + 0 = 0 + X = X;

Todas as propriedades acima, conferem a semântica da operação de soma sobre intervalos.

2.2.2 Pseudo Inverso Aditivo Intervalar

Seja X $\in$ IR um intervalo de reais, com X = $[\underline{x}; \overline{x}]$. Seja '$-$' operador do Pseudo Inverso Aditivo Intervalar e aplicando-se a propriedade de operador unário, têm-se:

$$\text{-X} = \text{-}[\underline{x}; \overline{x}] = [-\overline{x}; -\underline{x}]$$

Por exemplo, seja o intervalo X = [2;3], o intervalo solução para -X = [-3;-2].

2.2.3 Subtração Intervalar

Sejam X,Y $\in$ IR dois intervalos reais em que X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$. Define-se como subtração intervalar:

$$\text{X - Y} = [(\underline{x} - \overline{y}); (\overline{x} - \underline{y})]$$

Sejam X = [-1 ; 4] e Y = [3 ; 8], por exemplo. Substituindo X e Y na equação acima tem-se X $-$ Y = $[(-1) - 8; 4 - 3] = [-9; 1]$.

2.2.4 Multiplicação Intervalar

Sejam X,Y $\in$ IR dois intervalos reais em que X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$. Define-se a multiplicação entre dois intervalos como:

$$\text{X.Y} = [min\{\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\}; max\{\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\}]$$

As operações min e max servem para selecionar dentre o espaço de valores encontrados, respectivamente o menor e o maior valor obtido.

Sejam os intervalos X = [3,6] e Y = [-2,0], têm-se como solução: X.Y = [3,6].[-2,0] = [min(3.-2,3.0,6.-2,6.0) ;max(3.-2,3.0,6.-2,6.0)] = [-12; 0].

2.2.4.1 Propriedades Algébricas

Sejam X,Y e Z $\in$ IR. As seguintes propriedades algébricas podem ser conferidas à multiplicação de intervalos em IR:

- **FECHAMENTO:** Se X $\in$ IR e Y $\in$ IR, então X + Y $\in$ IR;
- **ASSOCIATIVIDADE:** X.(Y.Z) = (X.Y).Z;
- **COMUTATIVIDADE:** X.Y = Y.X;

- **ELEMENTO NEUTRO:** $\exists$1 = [1;1] $\in$ IR, tal que X.1 = 1.X = X;
- **SUBDISTRIBUTIVIDADE:** X.(Y + Z) $\subseteq$ (X.Y) + (X.Z);

Observações:

- O conjunto IR não contém um Inverso Multiplicativo, assim como ocorre com o Aditivo. Isto porque nem sempre pode-se encontrar um intervalo X^{-1} tal que $X.X^{-1}$ = 1;

2.2.5 Pseudo Inverso Multiplicativo Intervalar

Seja X $\in$ IR um intervalo de reais, em que X = $[\underline{x}; \overline{x}]$ e 0 $\notin$ X. Então:

$$X^{-1} = \frac{1}{X} = [\frac{1}{\overline{x}}; \frac{1}{\underline{x}}]$$

Exemplo: Seja X = [3;4], tem-se X^{-1} = $[\frac{1}{4} ; \frac{1}{3}]$.

2.2.6 Divisão Intervalar

Sejam X,Y $\in$ IR dois intervalos reais em que X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$ com 0 $\notin$ X. Define-se a divisão entre dois intervalos como:

$$\frac{X}{Y} = \frac{[\underline{x};\overline{x}]}{\underline{y};\overline{y}} = X.Y^{-1} =$$

$$[min(\frac{\underline{x}}{\overline{y}}, \frac{\underline{x}}{\underline{y}}, \frac{\overline{x}}{\overline{y}}, \frac{\overline{x}}{\underline{y}}); max(\frac{\underline{x}}{\overline{y}}, \frac{\underline{x}}{\underline{y}}, \frac{\overline{x}}{\overline{y}}, \frac{\overline{x}}{\underline{y}})]$$

As propriedades da divisão intervalar correspondem as mesmas da multiplicação, visto que essa é realizada através da multiplicação pelo inverso multiplicativo do intervalo Y.

Exemplo: Sejam X = [4,8] e Y = [2,4], tem-se que $\frac{X}{Y} = X.Y^{-1} = [min(\frac{4}{4}, \frac{4}{2}, \frac{8}{4}, \frac{8}{2}); max(\frac{4}{4}, \frac{4}{2}, \frac{8}{4}, \frac{8}{2})]$ = [1,4].

2.3 Definição Topológica em IR

Esta seção tem como objetivo apresentar os princípios básicos de topologia em intervalos.

2.3.1 Intervalo Simétrico

Seja X $\in$ IR um intervalo. X é um intervalo simétrico se -X = X.

Exemplo: Seja X = [-1;1], sendo X = -X $\Rightarrow$ [-1;1] = -[-1;1] $\Rightarrow$ [-1;1] = [-1 ; -(-1)] $\Rightarrow$ [1,-1].

2.3.2 Intersecção entre dois Intervalos

Sejam X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$ dois intervalos reais. Define-se como intersecção de dois intervalos X e Y como sendo X ∩ Y = $[max(\underline{x}, \underline{y}); min(\overline{x}, \overline{y})]$, sendo $max(\underline{x}, \underline{y}) \leq min(\overline{x}, \overline{y})$. Caso haja $min(\overline{x}, \overline{y}) \leq max(\underline{x}, \underline{y})$, então X∩Y = $\emptyset$

Através da Figura 3 é possível notar a intersecção entre os intervalos X e Y.

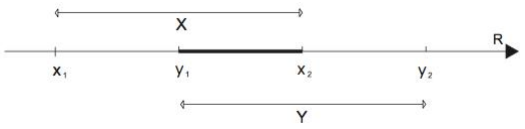

Figura 3: Representação da intersecção entre os intervalos X e Y em $\mathbb{R}$

2.3.3 União entre dois Intervalos

Sejam X = $[\underline{x}; \overline{x}]$ e Y = $[\underline{y}; \overline{y}]$ dois intervalos reais. Define-se a união de dois intervalos X e Y como sendo X∪Y = $[min(\underline{x}, \underline{y}); max(\overline{x}, \overline{y})]$.

Esta operação é possível a partir da $X \cup Y \neq \emptyset$.

Através da Figura 4 verifica-se a operação de união entre os intervalos X e Y.

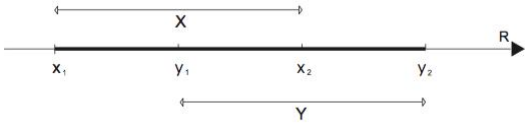

Figura 4: Representação da união entre os intervalos X e Y em $\mathbb{R}$

2.3.4 Distância entre dois Intervalos

Sejam X = $[\underline{x}; \overline{y}]$ e Y = $[\underline{y}; \overline{y}]$ dois intervalos reais. Define-se como a distância entre dois intervalos X e Y o número real não-negativo $\delta = [max\{|\underline{x} - \underline{y}|, |\overline{x} - \overline{y}|\}]$. A operação de distância é formalizada da seguinte forma:

$$dist(X, Y) = dist([\underline{x}; \overline{x}], [\underline{y}; \overline{y}]] = [max\{|\underline{x} - \underline{y}|, |\overline{x} - \overline{y}|\}] \leq 0$$

A distância entre dois intervalos está demonstrada graficamente pela Figura 5.

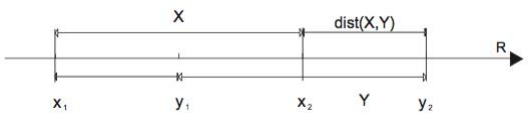

Figura 5: Representação da distância entre os intervalos X e Y em $\mathbb{R}$

2.3.5 Diâmetro do Intervalo

Seja X = $[\underline{x}; \overline{x}]$ um intervalo real. Define-se como diâmetro do intervalo X o número real não-negativo $d = \overline{x} - \underline{x}$. Assim tem-se:

$$diam(X) = diam([\underline{x}; \overline{x}]) = \overline{x} - \underline{x} \leq 0$$

Através da Figura 6 é possível observar a forma geométrica do diâmetro de um intervalo. O diâmetro contém uma propriedade de grande importância nas operações aritméticas intervalares, ele pode ser utilizado como medida de qualidade do intervalo solução(RATSCHEK; ROKNE, 1988).

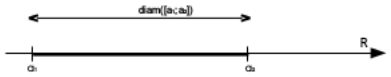

Figura 6: Diâmetro do intervalo

2.3.6 Ponto Médio de um Intervalo

Seja X = $[\underline{x}; \overline{x}]$ um intervalo real como representado na Figura 7. Define-se como ponto médio de um intervalo X:

$$\text{med(X)} = \text{med}([\underline{x}; \overline{x}]) = \frac{\underline{x}+\overline{x}}{2}$$

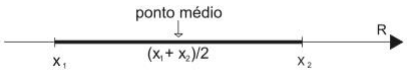

Figura 7: Ponto Médio do intervalo

2.3.7 Valor Absoluto de um Intervalo

Seja X = $[\underline{x}; \overline{x}]$ um intervalo real, o valor absoluto de X é definido por:

$$\mid X \mid = \text{max}[\mid \underline{x} \mid; \mid \overline{x} \mid]$$

As definições das operações intervalares acima são as operações básicas no desenvolvimento dos Ambientes Intervalares e Bibliotecas Intervalares, sendo as primeiras operações a serem implementadas, servindo de base para o contínuo desenvolvimento e ampliação dos recursos oferecidos por essas soluções.

2.4 Aritmética Intervalar

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 provar a existência ou não da solução de diversas equações.

Ela tem sido utilizada em diversas áreas, com a finalidade de resolver sistemas de equações lineares e não lineares, equações diferenciais ordinárias e parciais, equações integrais e problemas de otimização. Para cada uma dessas classes de problemas numéricos, o emprego da Aritmética Intervalar tem sido acompanhado pelo desenvolvimento de novas técnicas, as quais vão além da mera substituição dos coeficientes reais por intervalos de números.

A Aritmética Intervalar vem trazer uma técnica que permite um controle de erros com limites confiáveis, além de provar a existência e a unicidade da solução de muitos problemas numéricos. Surgiu com o objetivo de automatizar a análise do erro computacional. Várias das características do padrão IEEE são necessárias para a definição da aritmética intervalar. Entre estas, 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(FERREIRA; FERNANDES; CAMPOS, 2005).

O uso da aritmética intervalar foi desenvolvido inicialmente por (MOORE, 1966). Este ramo da matemática veio se desenvolvendo desde então. A matemática intervalar foi proposta em 1974 por Leslie Fox, combinando diferentes áreas como: aritmética intervalar, análise intervalar, topologia intervalar, algebra intervalar entre outras. 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 e para representar dados inexatos, aproximações e erros de truncamento de procedimentos (como consistência lógica de programas, critério de parada de processos iterativos). No segundo sentido servem, por exemplo, para testes de verificação computacional e 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 sofreu críticas em função de dois problemas principais, os quais se resumem em que, muitas vezes, podem resultar intervalos pessimistas, ou seja, demasiadamente grandes, que não possuem muita informação sobre o resultado, gastando-se muito tempo de máquina. Estas duas críticas foram 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.

2.5 Aritmética de Alta Exatidão

O computador foi inventado para, entre outras coisas, fazer o trabalho complicado e/ou repetitivo para o homem. Na questão de cálculos em ponto-flutuante, a evidente discrepância entre o poder computacional e o controle dos erros computacionais sugere que se coloque o processo de estimativa de erro dentro do próprio computador. Para fazer isso o computador tem que ser feito aritmeticamente mais poderoso que o comum. Na aritmética de ponto-flutuante ordinária (definida pelo padrão IEEE), a maioria dos erros ocorre em acumulações, isso é, pela execução de uma sequência de adições e subtrações. Na aritmética de ponto-fixo, contudo, a operação de acumulação é realizada sem erros. Assim, grande parte dos erros encontrados em ponto-flutuante podem ser evitados se a acumulação for realizada em ponto-fixo em um acumulador especial (KULISCH; MIRANKER, 1981).

Com a atual tecnologia pode-se realizar facilmente essa acumulação de ponto-fixo. Necessita-se, somente, providenciar um registrador de ponto-fixo na unidade de aritmética que cubra todo o domínio em ponto-flutuante. Se um registrador com esta característica não está disponível, então esse pode ser simulado na memória principal por software. Isso infelizmente pode resultar em perda de velocidade, que em muitos casos, é de maior importância do que o ganho em confiabilidade.

Por outro lado, se esse registrador tiver dupla precisão, então os produtos escalares de vetores de qualquer dimensão finita sempre podem ser calculados de forma exata. A possibilidade de calcular os produtos escalares de vetores em ponto-flutuante de qualquer dimensão, com exatidão total, abre uma nova dimensão na análise numérica. Em particular o produto escalar ótimo prova ser um instrumento essencial para alcançar maior exatidão computacional. No sentido de adaptar o computador para esta nova tarefa, o controle automático de erros, sua aritmética deve ser estendida ainda com um outro elemento.

Todas as operações com números de ponto-flutuante, que são a adição, a subtração, a divisão e a multiplicação, e o produto escalar ótimo de vetores em ponto-flutuante devem ser supridos com os arredondamentos direcionados, isso é, arredondamentos para o número de máquina anterior (para baixo), posterior (para cima)(VARJÃO, 2011).

A Aritmética de Alta Exatidão possibilita que os cálculos sejam efetuados com máxima exatidão, isto é, o resultado calculado difere do valor exato no máximo em um arredondamento. Para isto é necessário que o formato ou tipo de dado, as operações aritméticas suportadas pelo hardware ou pela linguagem de programação, satisfaçam as condições de um semimorfismo. Todas as operações definidas deste modo são então de máxima exatidão(GONçALVES, 2012).

Os requisitos para se ter Aritmética de Alta Exatidão são: o uso de arredondamen-

tos direcionados (para baixo *x e para cima *x), as quatro operações aritmética com máxima exatidão e o produto escalar ótimo(CALLEJAS; BEDREGAL; DUTRA, 2008). Resumindo, operações semimórficas para números reais e complexos, vetores e matrizes, assim como para intervalos reais e complexos, vetores e matrizes intervalares podem ser realizadas em termos de quinze (15) operações aritmética fundamentais em ponto- flutuante (+, -, * , /), cada uma munida de arredondamento (para cima, para baixo e para o mais próximo.

O uso da Aritmética de Alta Exatidão, aliado à matemática intervalar resulta em resultados confiáveis e com máxima exatidão, onde o resultado está contido em um intervalo cujos extremos diferem por apenas um arredondamento do valor real. Os cálculos intermediários são feitos em registradores especiais, de forma a simular a operação dos reais, sendo o arredondamento feito só no final, onde em cada extremo é aplicado o arredondamento direcionado para baixo e para cima.

2.6 AMBIENTES INTERVALARES

O uso de ambientes de programação que suportam representação intervalar para as operações de cálculo científico favorece o controle automático de erros através de métodos auto-validáveis, ou seja métodos que se encarregam de verificar e garantir a exatidão dos cálculos efetuados (GARROZI, 2009).

Esses ambientes intervalares são analisados segundo a qualidade de software, as características das linguagens de programação e critérios de qualidade do intervalo (M. D. C. BALBONI L. M. TORTELLI, 2014). Existem diversos ambientes computacionais para matemática intervalar: Maple (MONAGAN, 1989), MatLab IntLab (D. HANSELMAN, 1999), IntPy (VARJÃO, 2011), C-XSC (R. KLATTE U. KULISCH; RAUCH, 1993), Java-XSC (GONçALVES, 2012). Como forma de validar os resultados obtidos na biblioteca IntDroid, utilizam-se os resultados das bibliotecas Java-XSC, C-XSC e IntPy que são amplamente utilizadas na computação científica. A biblioteca C-XSC sendo a mais antiga e com o desenvolvimento consolidado em técnicas da matemática intervalar.

Nas seções seguintes serão descritos os softwares intervalares, o Maple INT e Matlab, assim como as linguagens de programação que possuem bibliotecas intervalares Java-XSC, IntPy e C-XSC, apresentando suas características, funcionalidades, operações, além de apresentar os resultados de operações dessas bibliotecas em conjunto com os resultados do IntDroid no Capítulo 5.

2.6.1 Maple INT

O software matemático Maple (MONAGAN, 1989) é um ambiente interativo, com uma interface amigável que, para muitas finalidades, dispensa a programação. Bibli-

otecas Maple, uma vez carregadas, disponibilizam os comandos e operadores necessários para cálculos específicos. Possui uma linguagem de programação fundamentada no conceito de linguagem interpretada e um mecanismo de construção e distribuição de pacotes de programas e funções. É largamente utilizado em computação científica, e também na implementação de protótipos de sistemas de grande porte que, após testados no Maple, são posteriormente implementados em outras linguagens, que proporcionem um processamento mais rápido do que sistemas de computação algébricos. Assim, a opção pelo software Maple em conjunto com a biblioteca INT com operações intervalares para estudo neste trabalho fundamenta-se na sua potencialidade de aplicações tanto na pesquisa quanto no ensino.

2.6.2 MatLab INTLAB

O IntLab (INTerval LABoratory) é um pacote desenvolvido para o software Matlab (D. HANSELMAN, 1999). Contém tipos de dados básicos e operadores para aritmética intervalar, bem como uma variedade de métodos numéricos usando intervalos. O foco principal é produzir resultados confiáveis. A filosofia do IntLab é que tudo deve ser escrito em código Matlab para garantir melhor portabilidade, usa extensivamente rotinas BLAS, o que garante tempos de computação rápidos, comparáveis à aritmética pura de ponto flutuante. Vetor intervalo e operações de matriz são muito rápidos em IntLab, no entanto, os cálculos não lineares e loops podem retardar o sistema de forma significativa, devido a sobrecarga de interpretação e uso extensivo do conceito de operador. O IntLab é um Software proprietário (D. HANSELMAN, 1999).

2.7 Bibliotecas Intervalares

As bibliotecas intervalares, como na Ciência da Computação em geral, são definidas por uma coleção de subprogramas utilizados no desenvolvimento de software (M. D. C. BALBONI L. M. TORTELLI, 2014). Bibliotecas contém código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma modular. Algumas são tanto programas independentes quanto bibliotecas. Sistemas Operacionais modernos como os sistemas operacionais móveis provêm bibliotecas que implementam a maioria dos serviços do sistema, que transformaram em comodidades os serviços que uma aplicação espera que sejam providos pelo sistema operacional. Assim sendo, a maior parte do código utilizado em aplicações modernas é fornecido por estas bibliotecas. As bibliotecas intervalares são bibliotecas que carregam os recursos necessários para a utilização de técnicas da matemática intervalar e de recursos necessários para sua utilização em uma determinada linguagem, plataforma ou aplicativos (M. D. C. BALBONI L. M. TORTELLI, 2014).

2.7.1 Biblioteca C-XSC

C-XSC(do inglês, *C++ for eXtended Scientific Computing*) é uma biblioteca desenvolvida na linguagem de programação C++ de fácil utilização, especialmente para aplicações científicas e de engenharia. Trata-se de uma biblioteca numérica para a Computação Científica baseada na linguagem C++. Possui alta geração de resultados com exatidão verificados automaticamente (R. KLATTE U. KULISCH; RAUCH, 1993). Fornece um grande número de tipos de dados numéricos e operadores predefinidos. Estes tipos são implementados como classes da linguagem C++. Assim, o C-XSC permite a programação de alto nível de aplicações numéricas em C++. Disponível para muitos ambientes computacionais que possuem um compilador C-XSC. Suas características mais importantes são (R. KLATTE U. KULISCH; RAUCH, 1993):

– Aritmética intervalar para números reais, complexos, intervalares e intervalares complexos com propriedades definidas matematicamente;

– Tipos de dados com alta exatidão;

– Operadores aritméticos predefinidos com alta exatidão;

– Aritmética de múltipla precisão dinâmica e funções padrão;

– Controle de arredondamento para os dados de entrada e saída;

– Bibliotecas de rotinas para resolução de problemas numéricos;

– Resultados numéricos com rigor matemático. C-XSC é particularmente adequado para o desenvolvimento de algoritmos numéricos que proporcionam resultados precisos verificados automaticamente. Todas as operações básicas sobre o intervalo são de exatidão máxima (R. KLATTE U. KULISCH; RAUCH, 1993).

2.7.2 Biblioteca Java-XSC

A Biblioteca Java-XSC foi implementada na linguagem Java, utilizando o ambiente de desenvolvimento Java SE para desenvolvimento Desktop integrado a IDE Eclipse. Java-XSC foi implementada para cálculos científicos com o objetivo de conter o erro criado pela máquina ao representar os dados numéricos em toda sua extensão, seu sistema de implementação é híbrido sendo compilado e interpretado. O interessante em Java é sua forte tipagem e declarações de operações, uma vez que a precisão é um fator importante da linguagem (FERREIRA; FERNANDES; CAMPOS, 2005). Suas características são:

– Extinta a existência de ponteiros, sua substituição é definida pela passagem por referência;

– Alta exatidão através da forte tipagem e pela forma que a linguagem é implementada;

– Grande número de operadores aritméticos;

– Altamente difundida entre os programadores;

– Alocação dinâmica da memória, aumentando assim a otimização e a eficiência dos algoritmos;

– *Garbage Collector*, elimina qualquer área não usada da memória;

– Intervalos representados no tipo *double.*

2.7.3 Biblioteca IntPy

É um pacote intervalar desenvolvido na linguagem de programação Python. Composto de 2 subpacotes, `support` e `irreal`, além de um módulo que encapsula as classes de exceção. O subpacote `support` agrupa toda a funcionalidade de suporte ao IntPy e é composto de três módulos: `roundingmodule.c, general.py, stdfunc.py`. O primeiro módulo é uma extensão C para Python que manipula os modos de arredondamento do processador. Este módulo é dependência para toda funcionalidade do IntPy. O segundo módulo organiza todas as peças de software pontuais. Uma dessas peças é a função `rational2fraction`(racional) responsável por converter uma representação em string de números racionais em outra representação mais facilmente manipulável por computador (VARJÃO, 2011).

O subpacote `irreal` compreende 2 módulos: `irreal.py` e `irmath.py`. O primeiro implementa a classe IReal que representa o tipo Intervalo Real. Todas as operações aritméticas e de conjuntos, relações de ordem e funções auxiliares são implementadas nessa classe. O segundo módulo implementa extensões intervalares das funções padrão. Todos são implementados na linguagem Python e reconhecidos como software livre GPL (VARJÃO, 2011).

As bibliotecas intervalares apresentadas mostram a utilização das técnicas intervalares em diversos aspectos e utilização, sendo estes em ambientes intervalares na forma de solução completa e integrada de um software ou em bibliotecas que extendem o potencial das linguagens de programação em seus ambientes de desenvolvimento.

O capítulo seguinte é parte fundamental deste trabalho, apresentando as principais diferenças dos Sistemas Operacionais(O.S.) utilizados em hardware Desktop e O.S. Móveis. Descreve-se diferenças estruturais, de hardware restritivo, com grande controle dos recursos de hardware e de memória em suas linguagens de programação, sendo modificadas para uso exclusivo do ambiente móvel através de seus Software

Development Kit(S.D.K.). Apresenta também as linguagens de programação utilizadas no desenvolvimento de suas aplicações móveis, abordando as linguagens nativas, ou seja, linguagens utilizadas dentro do O.S. e que trabalham sem nenhum tipo de tradução ou ponte entre o ambiente de desenvolvimento e o O.S.

3 SISTEMAS OPERACIONAIS MÓVEIS

Quando falamos de Sistemas Operacionais Móveis, primeiramente temos que falar do sistema líder em tecnologia e inovação e que vem dominando o mercado a alguns anos: o Android, atualizado e mantido pela Google, possui maior número de ativações em relação aos outros sistemas operacionais. A popularidade do Android se dá principalmente por ser *OpenSource* e permitir sua instalação em muitos modelos e aparelhos. O IOS, apesar de ser um sistema operacional proprietário vem concorrendo muito de perto com o Android, também é um sistema operacional baseado no Linux. Os sistemas operacionais baseado no Linux apresentam uma grande estabilidade e melhor consumo de memória como grande característica (SILBERSCHATZ A.; GALVIN, 2010). Por fim, cita-se o sistema operacional Windows Phone, que começou a ter relevância no mercado após ser adotado fortemente pela Nokia em seus aparelhos com o fim do Symbian OS, um sistema operacional que era mantido pela própria Nokia mas que não obteve sucesso e nem a aceitação dos seus concorrentes.

No presente capítulo descreve-se a composição desses sistemas operacionais e sua relação com seu ambiente de desenvolvimento, ferramentas oferecidas e ainda como esses sistemas operacionais e suas linguagens trabalham com a matemática em seus cálculos internos e dentro de seus aplicativos.

A Alta Exatidão é pouco ou nada explorada nos Sistemas Operacionais Móveis. Na maioria das vezes as soluções matemáticas são realizadas em particular para cada necessidade que aparece dentro do processo de desenvolvimento de uma aplicação ou software (GARROZI, 2009), o que não fornece solução para outros desenvolvedores que necessitam utilizar em suas aplicações a Alta Exatidão. A criação e a utilização de ferramentas na linguagem destes sistemas operacionais será abordado como possível solução, além do desenvolvimento da IntDroid como solução para a plataforma Android.

3.1 WINDOWS PHONE OS

O Windows Phone é um sistema operacional móvel, desenvolvido pela Microsoft, sucessor da plataforma Windows Mobile, que ao contrário do predecessor, é focado no mercado consumidor, em vez do mercado empresarial. Foi lançado na Europa, Austrália e Singapura no dia 21 de outubro de 2010, nos EUA e Canadá no dia 8 de novembro, no México no dia 24 do mesmo mês e, no início de 2011, na Ásia (LECHETA, 2013).

Entrando em completa ruptura com as antigas versões da sua plataforma móvel Windows Mobile, a Microsoft apresenta no Windows Phone uma nova interface gráfica, denominada Metro, não permitindo interfaces personalizáveis, como acontecia até então, controlando rigorosamente todo o hardware em que o sistema operacional funciona. Como resultado dessa ruptura nenhuma das aplicações que existiam para a plataforma anterior funcionam no novo sistema. O gerenciamento de memória no Windows Phone ocorre de forma semelhante a do seus principais concorrentes. O Windows Phone utiliza em média 200Mb desta memória para o sistema e os demais para uso dos aplicativos suportando multitarefa(do inglês, *Multitasking*).

O sistema mantém nos *Live Tiles*, que são os blocos do Windows Phone, as principais informações sobre o aplicativo. Para manter os aplicativos abertos sem que ocorra o uso excessivo da memória o Windows Phone criou um modo onde os aplicativos acessam apenas um pequeno número de APIs que podem ser executados em todos os momentos sem consumir a memória.

O sistema de arquivos utilizado pelo Windows Phone é o mesmo utilizado em sua plataforma anterior, que é a base para o Windows Phone. Utilizando o sistema exFAT: (*Extended File Allocation Table*), também conhecido como FAT64 é um formato de sistema de arquivos utilizado principalmente em discos de memória flash, introduzido com o Windows Embedded CE 6.0. A utilização do exFAT é uma alternativa para evitar o extensivo uso do sistema de arquivos NTFS o em sistemas que não suportam o NTFS.

O Windows Phone OS utiliza como linguagem de programação a linguagem C# e um framework de desenvolvimento de aplicações para Windows Phone que tem como base o Silverlight. A linguagem de interface XAML é uma linguagem declarativa baseada no XML utilizado comumente no desenvolvimento web. Quem já teve contato com desenvolvimento de aplicações em Silverlight tem uma familiaridade muito grande com o desenvolvimento para o sistema operacional Windows Phone.

Uma grande vantagem atualmente para os desenvolvedores desta plataforma é que a Microsoft disponibilizou gratuitamente a versão Visual Studio 2010 Express for Windows Phone seguindo exemplo de seus concorrentes que distribuem seus SDK de forma gratuita para familiarizar os desenvolvedores com a tecnologia antes mesmo de

estarem dispostos a desenvolver um projeto para a plataforma.

A linguagem C# faz parte do conjunto de ferramentas oferecidas na plataforma .NET e surge como uma linguagem simples, robusta, orientada a objetos, fortemente tipada e altamente escalável a fim de permitir que uma mesma aplicação possa ser executada em diversos dispositivos de hardware, independentemente destes serem PCs, handhelds ou qualquer outro dispositivo móvel.

O avanço das ferramentas de programação e dos dispositivos eletrônicos inteligentes, criou problemas e novas exigências. As novas versões de componentes compartilhados eram incompatíveis com o software antigo. Os desenvolvedores reconheceram a necessidade de software que fosse acessível para qualquer um e disponível por meio de praticamente qualquer tipo de dispositivo. Para tratar dessas necessidades, a Microsoft anunciou sua iniciativa .NET e a linguagem de programação C# (LECHETA, 2013).

Durante o desenvolvimento da plataforma .NET, as bibliotecas foram escritas originalmente numa linguagem chamada Simple Managed C (SMC), que tinha um compilador próprio. Mas, em Janeiro de 1999, uma equipe de desenvolvimento foi formada por Anders Hejlsberg, que fora escolhido pela Microsoft para desenvolver a linguagem. Dá-se início à criação da linguagem chamada Cool. Um pouco mais tarde, em 2000, o projeto .NET era apresentado ao público na Professional Developers Conference (PDC), e a linguagem Cool fora renomeada e apresentada como C#.

A criação da linguagem, embora tenha sido feita por vários programadores, é atribuída principalmente a Anders, hoje um Distinguished Engineer na Microsoft. Ele fora o arquiteto de alguns compiladores da Borland, e entre suas criações mais conhecidas estão o Turbo Pascal e o Delphi.

A Microsoft submeteu o C# à ECMA para uma padronização formal. Em Dezembro de 2001 a associação liberou a especificação ECMA-334 Especificação da Linguagem C#. Em 2003 tornou-se um padrão ISO (ISO/IEC 23270).

A Biblioteca C-XSC é uma provável candidata a solução para o sistema operacional Windows Phone no que se trata de máxima exatidão, podendo ser transportada da linguagem C++ para a linguagem C#, conforme necessidade do desenvolvedor para resolver a necessidade de utilização de técnicas de máxima exatidão, pois a biblioteca é uma das mais completas na área e com implementações diversas.

3.2 IOS

O sistema operacional iOS é o sistema utilizado pela Apple em seus dispositivos móveis equipados com telas sensíveis ao toque. É baseado no sistema operacional Mac OS X incluso em computadores Macintosh que, por sua vez, é baseado no sistema UNIX BSD.

Anteriormente ao lançamento da versão(4.0), era conhecido como iPhone OS, mas teve seu nome modificado para melhor refletir o fato de que ele também serve como base para outros produtos da empresa. O iOS roda atualmente em três linhas de produtos: iPhone, iPod Touch e iPad.

O desenvolvimento de aplicativos para iOS é feito com o uso do iPhone SDK (Software Development Kit), um conjunto de ferramentas de desenvolvimento acompanhadas de documentação. O kit é disponibilizado no site da empresa para desenvolvedores cadastrados e sua instalação requer um computador equipado com Mac OS X,10.6 ou superior (STEIL, 2012).

O cadastro no site e o download do SDK podem ser feitos gratuitamente e permite usar as ferramentas e testar aplicações em um simulador. Para realizar testes em dispositivos reais e submeter aplicativos para publicação na App Store, é preciso pagar uma taxa anual. O pagamento dessa taxa também traz outros benefícios, como suporte técnico e acesso a versões beta do iOS e do SDK. As principais ferramentas inclusas no SDK são:

- Xcode – ambiente integrado de desenvolvimento(IDE), usado para gerência, edição, compilação e depuração de código fonte e demais recursos associados a um projeto.

- Interface Builder – usado para criação de interfaces gráficas de forma visual.

- Instruments – ferramenta de depuração e análise de desempenho usada para identificar problemas como vazamentos de memória e gargalos de desempenho. Permite a visualização em tempo real de diversos aspectos de uma aplicação enquanto ela é executada no simulador ou em um dispositivo real conectado à máquina de desenvolvimento.

- Simulator – simula iPhones e iPads, executando aplicações em um ambiente semelhante ao encontrado em um dispositivo real.

O gerenciamento de memória precisa ser tratado com atenção, seus conceitos bem compreendidos, pois os aparelhos com iOS não possuem um volume muito grande de memória, então a aplicação precisa controlar seu uso. Uma framework importante dentro da camada Core Services é a Core Fundation framework, que habilita outras frameworks e bibliotecas de compartilhamento de dados e core.

A linguagem Swift foi anunciada pela Apple como sendo a nova linguagem a ser utilizada para desenvolvimento de aplicativos e softwares para os sistemas operacionais IOS e OS X. Essa notícia chegou na comunidade de desenvolvedores trazendo certa surpresa, a nova linguagem irá substituir completamente a linguagem Objective-c no desenvolvimento de aplicações para esses sistemas.

Swift se baseia no melhor de C e Objective-C, sem as restrições de compatibilidade C. Swift adota padrões de programação segura e adiciona características modernas para tornar a programação mais fácil, mais flexível e mais divertido. Swift, apoiado pelo Cocoa e Cocoa Touch, é uma oportunidade para repensar como desenvolvimento de software funciona (D. MARK, 2009). Foi resultado de anos no desenvolvimento interno. A Apple lançou as bases para Swift, avançando o compilador, depurador, e um Framework utilizando infraestrutura existente. Simplificou o gerenciamento de memória com contagem de referência automático (ARC). O Framework, construído com base sólida em colaboração com a Cocoa Foundation, foi modernizado e padronizado por toda parte. Se Objective-C evoluiu para apoiar blocos literais de coleta e módulos permitindo a adoção de tecnologias da linguagem moderna sem interrupções, graças a esta base, seremos introduzidos a uma nova linguagem para o futuro do desenvolvimento de software da Apple(G. SILVEIRA, 2014).

Swift elimina classes inteiras de código inseguro. As variáveis são sempre inicializadas antes do uso e a memória é gerenciada automaticamente. Sintaxe é ajustada para tornar mais fácil para definir a sua intenção - por exemplo, simples palavras-chave de três caracteres definir uma variável (var) ou constante (let).

Os padrões de segurança na Swift são ajustados para o Cocoa e para o Cocoa Touch APIs. Entender e lidar adequadamente casos em que objetos são nulos é fundamental para os framworks e códigos e isso Swift faz com muita facilidade. Isso tudo funciona em conjunto visando fazer o desenvolvimento para iOS e OS X mais fácil e seguro.

Analisando a utilização da matemática intervalar para a plataforma iOS, identifica-se a falta de uma solução final que possa ser distribuída entre os desenvolvedores da plataforma.

A Apple se mostra muito restrita a aceitação de módulos e bibliotecas desenvolvidos por terceiros, sendo dificilmente incluídos no SDK oficial de desenvolvimento de seus aplicativos. Tais barreiras também são mostradas ao longo do desenvolvimento de aplicações que necessitam instalar bibliotecas e scripts no seu sistema operacional IOS, não deixando aberturas para esses processos, dificultando tanto a prática científica de estudo do IOS quanto para o desenvolvimento de aplicações promovidas pelos desenvolvedores de aplicativos da plataforma. Sendo assim essa é uma constante que enquanto não for modificada pelas políticas da empresa, continuará a deixar o iOS um ambiente totalmente voltado para o comercial e muito pouco para o lado científico.

3.3 ANDROID OS

O Android é um sistema operacional baseado no kernel do Linux. Apesar de ter sido desenvolvido inicialmente para smartphones, hoje é usado em diversas outras aplicações como tablets, netbooks, relógios, etc.

Apesar de ser baseado no kernel do Linux, existe pouca coisa em comum com distribuições Linux convencionais (embarcadas ou não), lembrando que um sistema embarcado (ou sistema embutido) é um sistema micro processado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla (L. C. O. PEREIRA, 2009).

De certo modo, o Android é uma máquina virtual Java rodando sobre o kernel do Linux, dando suporte para o desenvolvimento de aplicações Java através de um conjunto de bibliotecas e serviços.

Muitas características presentes no Android, tais como gráficos 3D e suporte a banco de dados nativo, também estão disponíveis em outras plataformas móveis. Porém, apenas no Android há um componente que permite exibir e manipular um mapa do Google Maps, serviço de mapas do Google, dentro de uma aplicação. Somente no Android todos os aplicativos são criados igualmente. Ou seja, nele não há distinção entre aplicativos que são nativos e os demais. Isso possibilita uma grande customização do sistema operacional, permitindo a substituição completa de aplicativos nativos por outros, criados por terceiros. Além disto, todos os aplicativos têm acesso as mesmas funcionalidades.

Sua arquitetura possui basicamente 4 camadas(L. C. O. PEREIRA, 2009):

- Aplicações: A camada de aplicativos é a que está no topo da pirâmide da arquitetura do sistema operacional Android, composta pelo conjunto de aplicações nativas do mesmo. Dentre estes pode–se citar: cliente de e-mail, despertador, calendário, jogos, mapas, browser e internet, etc.

- Framework: A camada de framework nativo disponibiliza aos desenvolvedores as mesmas Applications Programming Interface (APIs) – Interface de Programação de Aplicativos utilizadas para a criação de aplicações originais do sistema operacional Android. Este framework permite que o programador tenha o mesmo acesso ao sistema que os aplicativos da camada de aplicativos possuem. Este framework foi criado para abstrair a complexidade e simplificar a reutilização de procedimentos. Essa camada funciona como um link com a camada de bibliotecas do sistema operacional que serão acessadas através de APIs contidas no framework.

- Bibliotecas e serviços: Essas bibliotecas são responsáveis por fornecer funcionalidades para manipular o áudio, vídeo, gráficos, banco de dados e browser.

Algumas bibliotecas são a Bionic, a OpenGL/ES para trabalhar com interface gráfica, e a SQLite para trabalhar com banco de dados. Aqui também estão os serviços usados em camadas superiores, como máquina virtual Java Dalvik. A maior parte destas bibliotecas e serviços estão desenvolvidos em C e C++;

- O Android Runtime: Permite que cada thread rode sua própria instância da MV (máquina virtual). Embora no desenvolvimento de aplicativos seja utilizada a linguagem Java, as aplicações não são executadas em uma máquina virtual Java tradicional, e sim em outra chamada de Dalvik. Essa máquina virtual é otimizada especialmente para dispositivos móveis. A plataforma Google Android permite o desenvolvimento de aplicativos na linguagem Java Android. Essa máquina virtual foi construída pelos engenheiros da Google, para obter um consumo mínimo de memória e isolamento de processos;

- Kernel Linux: A camada do kernel é baseada em um sistema do sistema operacional Linux. Esta camada atua também como responsável pela abstração entre o hardware e os aplicativos e é responsável pelos serviços principais do sistema operacional Android, como o gerenciamento de memória e de processos. Várias funções do kernel são utilizadas diretamente pelo Android, mas muitas modificações foram feitas para otimizar memória e tempo de processamento das aplicações. Essas modificações incluem novos dispositivos de drivers, adições no sistema de gerenciamento de energia e um sistema que possibilita terminar processos de maneira criteriosa quando há pouca memória disponível. O Linux foi escolhido por já conter uma grande quantidade de drivers de dispositivos sólidos e por ter um bom gerenciamento de memória e processos.

Se olharmos para a arquitetura interna do Android, veremos o nível de complexidade deste sistema Operacional na Figura 8 .

3.4 Ambiente de desenvolvimento para ANDROID

Existem diversas ferramentas para auxiliar no desenvolvimento de aplicativos para o sistema Android, a mais completa é o kit de desenvolvimento Android SDK, que contém um ambiente com as características e especificações oficiais do Android distribuído pelo Google, chamado Android Studio apresentado na Figura 9.

O Android SDK inclui uma variedade de ferramentas que ajudam a desenvolver aplicações móveis para a plataforma Android. As ferramentas são classificadas em dois grupos: ferramentas do SDK e ferramentas de plataforma. Ferramentas do SDK são independente de plataforma e são necessários, não importa qual plataforma Android que se está trabalhando. Ferramentas da plataforma são customizadas para apoiar as características da mais recente plataforma Android.

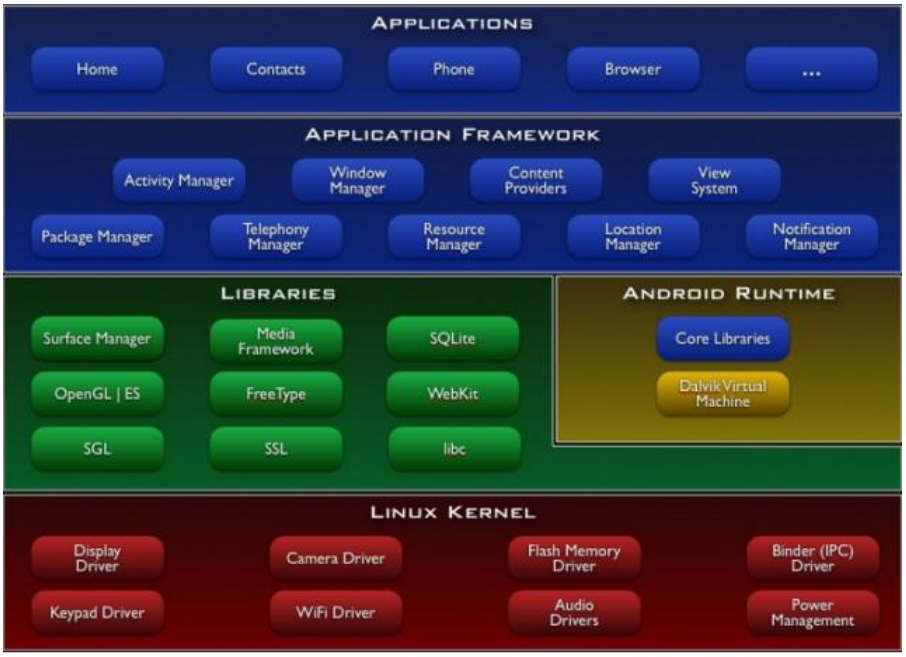

Figura 8: Arquitetura interna do Android OS

As ferramentas do SDK são instaladas com o pacote inicial SDK e são atualizadas periodicamente. São necessárias para o desenvolvendo aplicativos Android. As mais importantes incluem o Android SDK Manager, o AVD Manager, o Emulador e o Dalvik Debug Server Monitor.

-Virtual Device Tools

• Android Virtual Device Manager: O AVD Manager fornece uma interface gráfica do usuário no qual pode-se criar e gerenciar dispositivos virtuais Android(AVDs) que são executados no emulador Android.

• Emulador Android: é uma ferramenta de emulação de dispositivo baseado no QEMU que pode ser usada para depurar e testar seus aplicativos em um ambiente de tempo de execução Android.

• mksdcard: Ajuda a criar uma imagem de disco que pode-se usar com o emulador para simular a presença de um cartão de armazenamento externo (como um cartão SD).

• Development Tools: Permite gerenciar AVDs, projetos e os componentes instalados do SDK.

• Hierarquia Viewer (hierarchyviewer): Fornece uma representação visual do Vista hierarquia do layout com informações de desempenho para cada nó no layout, e uma visão ampliada do ecrã para examinar de perto os pixels em seu layout.

• Lint: A ferramenta lint Android é uma ferramenta de análise estática de código que verifica os arquivos de origem do projeto Android para possíveis bugs e melhorias de otimização.

• SDK Manager: Permite gerenciar pacotes do SDK, como plataformas instaladas e imagens do sistema.

• Sqlite3: Permite que se acesse os arquivos de dados SQLite criados e usados por aplicativos do Android.

-Debugging Tools

• ADB: Android Debug Bridge é uma ferramenta de linha de comando versátil que permite comunicar com uma instância emulador ou dispositivo conectado Android. Ele também oferece acesso ao shell do dispositivo.

• ADB Shell Commands: Aprender os comandos disponíveis para operações avançadas de linha de comando.

• Dalvik Debug Monitor Server: Permite o debug de Aplicações Android.

• Device Monitor: é uma ferramenta independente que fornece uma interface gráfica do usuário para várias ferramentas de depuração e análise de aplicativos Android.

• dmtracedump: Gera diagramas *call-stack* a partir de arquivos de log de rastreamento. A ferramenta usa o utilitário Graphviz para criar a saída gráfica, então é necessário instalar Graphviz antes de executar dmtracedump.

• hprof-conv: Converte o arquivo hprof que é gerado pelas ferramentas do SDK do Android para um formato padrão para que possa visualizar o arquivo em uma ferramenta de criação de perfil de sua escolha.

• Systrace: Permite analisar a execução de sua aplicação no contexto dos processos do sistema, para ajudar a diagnosticar problemas de exibição e desempenho.

• traceview: Fornece um visualizador gráfico para os logs de execução salvos pela sua aplicação.

-Build Tools

• JOBB: Permite a criação arquivos de expansão APK criptografados e não criptografados em formato opaco Binary Blob.

• ProGuard: Otimiza e ofusca o seu código através da remoção de código não utilizado e renomeando as classes, campos e métodos com nomes semanticamente obscuros.

• zipalign: Otimiza os arquivos do .apk para assegurar que todos os dados não comprimidos começam com um alinhamento particular, em relação ao início do processo. Esta deve ser sempre usado para alinhar os arquivos .apk depois de terem sido assinados.

-Image Tools

• Draw 9-patch: Permite criar facilmente um gráfico NinePatch usando um editor WYSIWYG. Ele também inspeciona as versões esticadas da imagem, e destaca a área em que o conteúdo é permitido.

• etc1tool: Um utilitário de linha de comando que permite codificar imagens PNG para o padrão de compressão ETC1 e decodificar ETC1 imagens comprimidas de volta para PNG.

• Tracer for OpenGL ES: Permite capturar comandos OpenGL ES e frame por imagens de quadros para ajudá-lo a entender como seus comandos gráficos estão sendo executados.

-Platform Tools

As ferramentas da plataforma são tipicamente atualizadas a cada vez que se instala uma nova plataforma SDK. Cada atualização das ferramentas da plataforma é compatível com plataformas mais antigas. Normalmente, utiliza-se apenas uma das ferramentas da plataforma, a Android Debug Bridge. Android Debug Bridge é uma ferramenta versátil que permite gerenciar o estado de uma instância emulador ou dispositivo Android. Também é possível utilizá-la para instalar um arquivo de aplicativo Android(.apk) em um dispositivo. As outras ferramentas da plataforma, são normalmente chamados pelo Android Build Tools ou Android Development Tools, assim raramente precisará chamar essas ferramentas diretamente. Como regra geral, deve-se contar com as ferramentas de compilação ou o plugin ADT para chamá-los quando necessário.

• bmgr: Uma ferramenta shell, pode-se utilizar para interagir com o gerenciador de backup em dispositivos Android que suportam API Nível 8 ou superior.

• logcat: Fornece um mecanismo de recolha de dados de saida e fornece uma visão da saída de depuração do sistema, muito utilizado para debug de saída.

3.5 Api's ANDROID

Dentre as vantagens no desenvolvimento Android, estão as APIs (Application Programming Interface) com diversas funcionalidades provendo os serviços do sistema (RETO, 2009).

As APIs estabelecidas para o Android permitem total modificação por meio de programação do seu conteúdo. Porém, programas que não precisam envolver-se em detalhes da implementação do software podem apenas utilizar os serviços, sem a preocupação de como funciona, utilizando apenas as características menos evidentes ao usuário padrão. Um ponto forte das APIs básicas do Android é a otimização que estas possuem, focando a utilidade dos pacotes, em conjunto com um bom aproveitamento, deixando de fora pacotes pesados e pouco evoluídos. Através destas, pode ser criada toda a interface com o usuário, permitindo a criação de telas, acessar arquivos, criptografar dados, ou seja, utilizar a funcionalidade definida pelo utilizador.

Entre as principais APIs pode-se destacar:

- Location Manager (android.maps): Usada para obter a posição geográfica do usuário. Como por exemplo, em aplicações que fazem uso de GPS;

- Telephony Manager (android.telephony): Informações sobre dispositivos como bateria e serviços de telefonia celular podem ser obtidos através dessa API;

- Window Manager (android.view): Responsável pelo gerenciamento de toda janela de uma aplicação, principais funções e componentes de inteface gráfica;

- Content Providers (android.provider): Responsável pela disponibilização dos dados através das aplicações tornando esses dados públicos. Quase todo tipo de dado é compartilhável, como áudio, vídeo, imagens e texto;

- Resource Manager (android.util): Todos os recursos que uma aplicação irá usar como áudio, vídeo, arquivos XML, são separados dela a fim de que sejam otimizados para ocupar menos espaço e demorar menos tempo para que sejam carregados. Essa API facilita o acesso a esses recursos;

- Notification Manager: Permite que uma aplicação exiba notificações, ative LEDs, luzes, sons ou vibração disponíveis no dispositivo;

- Activity Manager: Responsável pelo gerenciamento de cada atividade do sistema. No Android cada atividade é gerenciada através de uma pilha de atividades. Toda nova atividade criada vai para o topo de pilha de atividades e se torna uma *running activity*, significando que será executada;

- Webkit (android.webkit): Inclui APIs para conteúdo web, bem como um navegador embutido para utilização geral;

• android.app: APIs de Alto nível referentes ao modelo da aplicação;

• android.widget: Contém widgets prontos (botões, listas, grades, etc) para serem utilizados nas aplicações;

• android.database: Contém as APIs para comunicação com o banco de dados SQLite;

• android.os: Contém serviços referentes ao sistema operacional, passagem de parâmetros e comunicação entre processos.

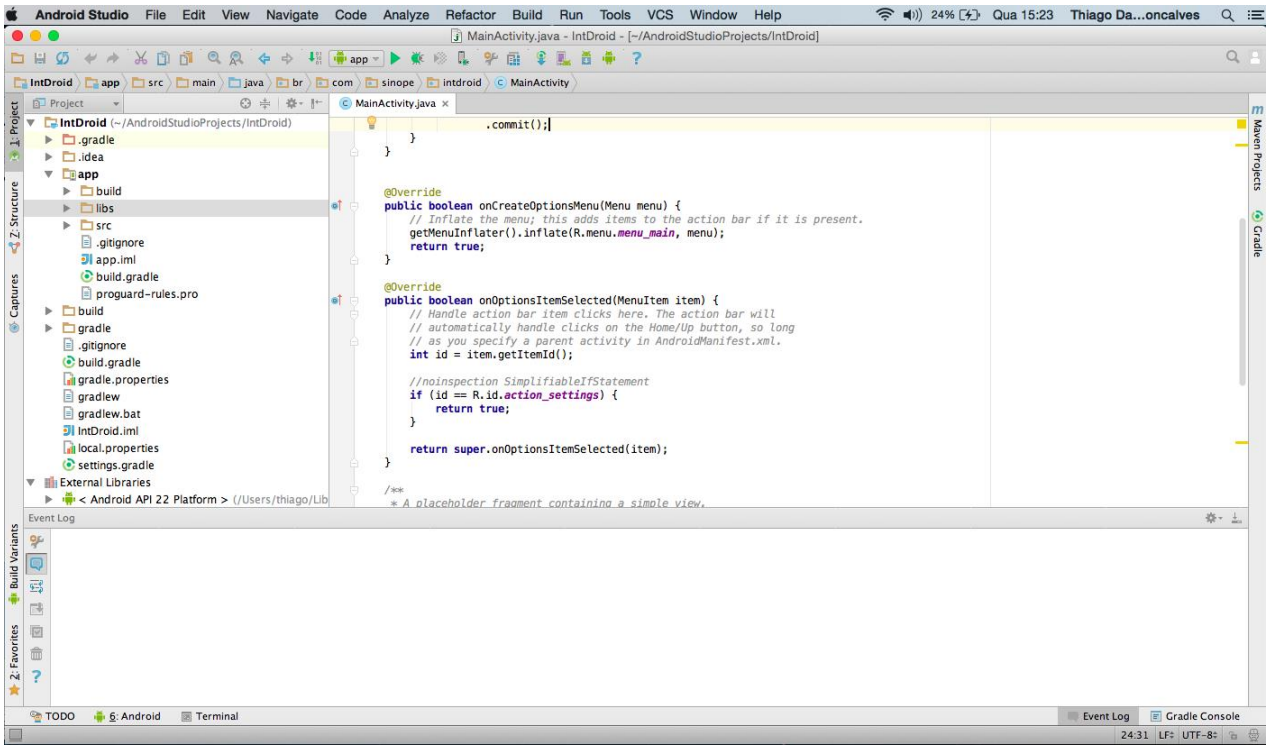

Figura 9: Ambiente de desenvolvimento Android

Este capítulo apresentou as características dos Sistemas Operacionais Móveis, suas ferramentas de desenvolvimento e seus SDK's, parte fundamental para o desenvolvimento de suas aplicações. Apresenta-se como principal sistema o Android, que foi adotado nesta dissertação por ser um Sistema Operacional de código aberto que mantém as portas abertas para os estudos científicos, aceitando colaborações no sentido de melhorar e evoluir sua plataforma. O capítulo 4 a seguir descreve o desenvolvimento da biblioteca IntDroid, suas classes e a estrutura utilizada no desenvolvimento de bibliotecas para a plataforma Android no formato AAR.

4 BIBLIOTECA INTERVALAR INTDROID

A medida que novas linguagens e plataformas de desenvolvimento foram sendo desenvolvidas, surgiu a necessidade de implementação de uma ferramenta de computação numérica específica para desenvolvimento móvel Android. Isto em conjunto com o sucesso das linguagens XSC motivaram o desenvolvimento das bibliotecas e a busca da Alta Exatidão nos ambientes e linguagens.

Esta dissertação visa o desenvolvimento da biblioteca IntDroid, por ter sido identificada a falta desse recurso nas ferramentas e soluções de desenvolvimento de aplicações Android (RETO, 2009). Foi implementada visando a necessidade de equilíbrio entre o poder de processamento do hardware do smartphone e o limite de resposta e utilização de recursos do sistema, por parte dos aplicativos Android, com o objetivo de evitar problemas comuns de aplicações móveis como o fechamento automático ou crash da aplicação pelo consumo excessivo de recursos de memória ou processamento durante a execução dos cálculos.

A base para o desenvolvimento dessa biblioteca foi a utilização de técnicas da matemática intervalar associadas aos modos de arredondamento direcionado. Foram implementados e desenvolvidos na biblioteca algoritmos de arredondamentos direcionados para cima e para baixo utilizados em conjunto das operações da matemática intervalar. Fazendo uma análise do funcionamento da biblioteca IntDroid através das ferramentas de debug e monitoramento utilizadas no Ambiente de desenvolvimento Android levando em consideração o trabalho com hardware móvel descrito na tabela 1. Por fim com objetivo de integrar da biblioteca com funcionalidades nativas do Android SDK sem nenhum tipo de acúmulo de utilização de recursos e fluidez do sistema.

4.1 Utilização e Aplicabilidade

Segundo Holbig (HOLBIG et al., 2005), o estudo das técnicas da aritmética intervalar é importante devido a sua aplicabilidade em uma grande variedade de problemas de engenharia e de outras ciências, tais como: a determinação de potenciais em redes elétricas, os cálculos do estresse em uma armação de construção ou de uma estru-

tura de ponte, o cálculo do padrão de escoamento num sistema hidráulico com ramos interconectados, o cálculo das estimativas de concentrações de reagentes sujeitos a simultâneas reações químicas entre outras.

A prevenção de erros causados pelo arredondamento ou truncamento de resultados possui grande interesse prático. Falhas de programas numéricos são causa de catástrofes e perda de vidas e recursos materiais. Na literatura, podem-se encontrar exemplos de diversos desastres causados por erros numéricos. Abaixo, são citadas algumas falhas catastróficas:

- Falha de míssil Patriot. Em fevereiro de 1991, durante a guerra do Golfo, um míssil americano Patriot falhou em interceptar um míssil Scud iraquiano. O incidente causou a morte de 28 soldados americanos. Um relatório chegou a conclusão que o tempo em décimos de segundo, como medido pelo relógio interno do sistema, era multiplicado por 1/10 para produzir o tempo em segundos. Este cálculo era realizado usando um registrador de ponto fixo de 24 bits. Em particular o valor 1/10, o qual é uma dízima periódica no formato binário, foi arredondado 24 bits após o ponto decimal. Esse pequeno erro acumulado em um grande número de iterações acarretou um erro final significante (ZHIVICH; CUNNINGHAM, 2009).

- Explosão do Ariane V. Em junho de 1996, um foguete lançado pela Agência Espacial Européia explodiu após quarenta segundos de seu lançamento. O foguete estava fazendo sua primeira viagem após uma década de desenvolvimento ao custo de sete bilhões de dólares. O foguete e sua carga estavam estimados em 500 milhões de dólares. Especificamente, um registro que armazenava um número de ponto flutuante de 64 bits relacionando a velocidade horizontal do foguete em relação à plataforma foi convertido em um inteiro de 16 bits sinalizado. Como o número era maior que 32767, o maior inteiro armazenável em um inteiro de 16 bits sinalizado, o sistema entrou em pane (ZHIVICH; CUNNINGHAM, 2009).

- Bolsa de valores de Vancouver. No ano de 1982, a bolsa de valores de Vancouver instituiu um novo índice inicializado para um valor de 1.000,000. O índice era atualizado após cada transação. Vinte e dois meses depois, repentinamente ele caiu para o valor de 520. A causa foi que o valor atualizado foi truncado em vez de arredondado. O valor arredondado deveria ter sido 1.098,892. Com o emprego crescente de computadores para fazer operações de *trading* automático baseado nas flutuações de índices, uma queda repentina no índice causou um falso *crash* na bolsa, gerando grandes perdas financeiras (ZHIVICH; CUNNINGHAM, 2009).

Basicamente, há três fontes de erros em computação numérica (M. LEONIDAS, 1987):

- Erros advindos dos dados e parâmetros iniciais. A modelagem de fenômenos do mundo físico, raramente possui uma descrição exata do evento. Normalmente, são necessárias simplificações do mundo físico para chegar a um modelo matemático. Por sua vez, as medidas são obtidas por instrumentos de exatidão limitada, de modo que a incerteza dos parâmetros iniciais acarreta incerteza de resultados.
- Erro de arredondamento. Esse erro é causado pelo modo como os cálculos são efetuados, seja manualmente, ou obtidos por computador, porque se utiliza uma aritmética de exatidão finita, ou seja, leva-se em consideração somente um número finito de dígitos. Como exemplo, o cálculo de 1/3 resultaria em 0,3333 se for usado 4 (quatro) casas decimais após a vírgula.
- Erro de truncamento. São erros advindos da utilização de processos, que a princípio deveriam ser infinitos ou que envolvam muitos passos, para a determinação de um valor e que por razões práticas, utiliza apenas uma parte finita dele. Um exemplo de processo infinito é a determinação da função trigonométrica seno que corresponde a uma seqüência infinita de soma de produtos.

Sendo assim a biblioteca IntDroid poderá ser utilizada por toda aplicação Android que necessitar de cálculos com alta exatidão, possibilitando ser integrada como biblioteca *Library Archieve Android*, formato padrão do Android SDK e fornecendo suas operações ao SDK e a aplicação.

4.2 Programando para Android

As Linguagens de programação utilizadas para desenvolver aplicações e recursos para a Plataforma Android são as mais variadas, passando por linguagens nativamente web como Html5 e Javascript além de Frameworks. A linguagem de programação nativa do Android é a Java Android, com uma série de modificações em relação a linguagem Java para desenvolvimento Desktop e Web. A Java para Android utilizada no desenvolvimento de aplicativos para Android OS utiliza a linguagem XML para trabalhar em conjunto. Uma das principais diferenças é a sua máquina virtual Android chamada Dalkin e a partir de 2016 a nova máquina virtual Android com o nome ART, o código compilado na ART gera arquivos opcode, diferente dos bytecodes gerados para Java Desktop e Web. Desta forma é implementada através de software, executando programas como se fosse um computador real.

Os aplicativos Android possuem uma função principal, as funções `onCrete`, `onResume`, `onPause` e `onDestroy` que devem ser substituídas.

Desta forma os programas/aplicativos são compilados, convertendo os opcodes de extensão .dex em código executável de máquina. Sendo assim, apesar de não haver em Java qualquer referência específica de hardware, estudos(VIANA, 2007) mostraram que os resultados de operações de ponto flutuante podem sofrer dependência de máquina(hardware), máquina virtual(ART ou Dalkin) e de sistema operacional neste caso o Android. O Hardware móvel se diferencia do desktop em diversos fatores principalmente em termos de estrutura e alimentação de energia. Também pode sofrer influências do Sistema Operacional, tendo em vista que a cada modificação Java Android se distancia dos padrões Java Oracle, sendo adaptado para as necessidades do Android, transformando a linguagem praticamente em uma nova ramificação, trazendo mudanças não somente para a linguagem como para o mundo exterior, onde a Oracle começa um processo judicial para impedir as modificações do Google em Java Android ou forçar o Google a instruduzir Java Android como uma linguagem totalmente nova desvinculando-se da Oracle.

As principais características da programação Android são:

- Thread - é uma linha de execução de código dentro de um aplicativo. Um aplicativo pode ter vários *threads* em execução num mesmo momento. Em outras palavras, *threads* permitem a um aplicativo ter comportamento multitarefa. A classe `Thread` é uma representação de um *thread* da Máquina Virtual Dalkin ou ART.

- Handlers - servem para entregar mensagens (uma mensagem ou Message é basicamente o encapsulamento de um *Runnable*, isto é, um trecho de código executável) para *threads* que ficam em loop aguardando a chegada desses trechos de código para executar.

  Para um thread ficar em loop, é preciso criar uma fila de mensagens chamando os métodos `Looper.prepare()` e `Looper.loop()` dentro do próprio *thread*. No caso do thread principal o próprio sistema já faz esses passos, de forma que já o encontramos em loop por padrão.

  Usamos um *Handler* quando queremos que um *thread* secundário execute muitas mensagens, ou quando queremos entregar uma mensagem ao *thread* principal. Neste último caso podemos evitar o uso do Handler através do método `Activity.runOnUiThread()` (caso estejamos em uma Activity), ou através de uma AsyncTask.

- AsyncTasks - Uma AsyncTask é uma classe que permite executar três trechos de código em sequência: o primeiro será executado pelo thread principal (thread de UI), o segundo por um thread secundário, e o terceiro de novo pelo thread principal. Isso então é implementado usando Threads e Handlers. AsyncTasks

têm a intenção de simplificar a implementação dessa sequência de passos, que é muito comum de acontecer no Android, como disparar uma animação de carregamento de dados, executar uma tarefa em segundo plano, e então interromper a animação. Essa sequência é feita assim, separada em threads e não tudo no mesmo thread, porque o thread principal é reservado para atualizar a tela e não pode executar tarefas em segundo plano sob pena de perder a responsividade.

- Orientada a objetos: Tudo em Java pode ser tratado como um objeto. Seus programas são organizados em classes, que ao serem instanciadas produzem estes objetos.
- Compilado e interpretado: Os aplicativos em Android Java são compilados e interpretados ao mesmo tempo pelo ambiente Android. Ao ser compilado é gerado um código binário, bytecode, com extensão .dex, sendo então executado, que difere da extensão .class utilizada em aplicações Java Desktop.
- Multi-tarefa: Android Java pode construir aplicações com múltiplos eventos. Desta forma é possível executar diversos segmentos de códigos simultaneamente.
- Dinâmica: A linguagem de programação Android Java foi projetada para se adaptar ao meio envolvido. É possível fazer uma conexão com bibliotecas não nativas de forma totalmente dinâmica.
- Robusta: Android Java escreve programas que são confiáveis em uma variedade de hipóteses. Ela coloca bastante ênfase no princípio de checar possíveis problemas e eliminar situações inclinadas para a ocorrência de erros.
- Segurança: Devido sua característica de ser distribuída, Android Java trata muito bem a questão da segurança. Ela possibilita o desenvolvimento de sistemas que estejam livres de adulterações e de vírus.
- Compacta: Por ocupar pouco espaço na memória, Android Java permite o desenvolvimento de aplicativos para palmtops e celulares sendo assim utilizada no desenvolvimento dos aplicativos Android. Android Java é fácil de ser manuseada e bastante intuitiva. Foi desenvolvida de forma a ficar semelhante a C++, com o intuito de ser bem aceita no mundo do desenvolvimento O.O, mas não herdando suas complexidades.

A seção 4.3 apresenta os passos de como construir uma biblioteca para ser utilizada no desenvolvimento de aplicações Android, fornecendo de forma similar as bibliotecas de tecnologia web, o fornecimento de um recurso para utilização na programação das aplicações e saídas ou respostas em formato de retorno da biblioteca para ser apresentado graficamente na aplicação.

4.3 Android Archieve Library

A biblioteca IntDroid conta com a tecnologia das bibliotecas *Android Archieve Library*, que tem suas especificidades mostradas nessa seção, assim como a maneira correta de sua utilização em um novo projeto Android. Com o propósito de reuso das bibliotecas em Android são utilizadas em extensão .aar de arquivos. Os arquivos AAR Android são construídos em cima deste conceito para permitir empacotar não apenas o código-fonte, mas também recursos auto contidos como saídas de métodos e interfaces de resposta. A biblioteca pode ter seu próprio manifesto, recursos e também pode fornecer opcionalmente configuração e até mesmo seus próprios filtros personalizados. Isso tudo é empacotado em um arquivo .aar que pode ser publicado em um repositório para obter acesso fácil a partir do Gradle(L. C. O. PEREIRA, 2009).

O formato de arquivo AAR File contém em seu pacote a estrutura abaixo composta de pastas e arquivos:

- AndroidManifest.xml (mandatory)
- classes (mandatory)
- res (mandatory)
- R.txt (mandatory)
- assets (optional)
- libs (optional)
- jni (optional)
- proguard.txt (optional)
- lint (optional)

Para criar o projeto de biblioteca AAR o Android Studio faz a maior parte do trabalho. Através do Android Studio - *create library module*, selecione um novo módulo do item de menu File, após selecione o Android Library como um módulo a ser criado, no próximo passo poderá usar um template para criar a sua biblioteca da mesma maneira que cria um novo projeto com uma diferença fundamental, será a utilização do include de `com.android.library` ao em vez do padrão `com.android.application`.

Quando construir o projeto de biblioteca AAR os arquivos irão automaticamente para os lugares abaixo em suas pastas.

- build
- outputs

- aar
- applib-debug.aar
- applib-release.aar

Para utilizar a biblioteca AAR será necessário escolher a maneira de fazer o *include* da biblioteca no seu projeto:

- Publicando em um repositório Maven externo
- Publicando em um repositório local
- Acessa-lo diretamente de uma pasta local

Incluindo a biblioteca AAR como dependência de uma pasta local:

1. Copie o arquivo AAR para a pasta libs no seu projeto. Crie a pasta caso ainda não exista. É necessário estar localizada na mesma pasta que o arquivo Gradle *build*, e não em um nível acima da pasta do projeto.
2. Declare a pasta libs como dependência no Gradle script de sua aplicação.

- repositories
- flatDir
- dirs 'libs'

A declaração ao Gradle de flatDir permite declarar o arquivo do sistema que é um repositório como uma dependência para seu projeto.

3. Declare sua biblioteca AAR como dependência:

- dependencies
- compile(name:'intdroid', ext:'aar')

O *AndroidManifest.xml* é um item compulsório da biblioteca AAR. Quando um aplicativo inclui uma biblioteca AAR no seu arquivo de manifesto ele precisa ser mesclado com o da aplicação, para existir apenas um arquivo manifesto no *build* final do aplicativo que utilizará a biblioteca.

A principal diferença entre o empacotamento em um AAR é que AAR incluem recursos como *layouts, drawables* etc. Isso torna muito mais fácil para criar componentes visuais auto suficientes. Por exemplo, se tiver vários aplicativos que usam a mesma tela de login, poderia-se compartilhar as classes, mas não o layout, estilos ou ainda teria que duplicá-los. Com AAR tudo é empacotado em um pacote puro. Tentativas semelhantes foram feitas com apk-libs, mas são agora obsoletos, tendo em vista que AAR são muito melhores (L. C. O. PEREIRA, 2009).

4.4 Classes IntDroid

O desenvolvimento do IntDroid foi realizado tendo por base a teoria da matemática intervalar e suas regras de funcionamento, incorporando suas funcionalidades ao Android SDK, utilizando da tecnologia *Android Archive Library*, descrito na seção 4.3. Conta com a classe principal chamada IR() na qual abriga as classes construtoras. A presente seção descreve seu comportamento e exemplos de utilização.

A classe principal possui os seguintes características:

- fornecer o tipo de dado intervalo a ser utilizado nas operações e passagem de dados da biblioteca
- passar por herança suas funcionalidades às operações através das entradas recebidas no método da operação
- utilizar a sobrecarga de método para criar o tipo intervalo
- receber através de passagem de parâmetros o valor real a ser devolvido como intervalo
- receber através de passagem de parâmetro valores definidos do extremo inferior e extremo superior do intervalo
- receber a precisão na qual o intervalo será criado e utilizado na operação.

Classe principal IR() recebe em sua inicialização os seguintes parâmetros:

- Construtor que recebe os valores do intervalo e constrói o objeto [inf, sup]:

  IR(inf, sup, precisão);

- Construtor do intervalo que recebe um valor real e gera o intervalo aplicando o arredondamento direcionado para criar o extremo:

  IR(valor, precisão);

- Construtor de intervalo vazio para testes de NaN:

  IR(precisão);

- Operação que retorna um intervalo que é a soma dos intervalos recebidos:

  IR add(Intervalo x, Intervalo y);

- Operação que retorna um intervalo que é a subtração dos intervalos recebidos:

  IR sub(Intervalo x, Intervalo y);

- Operação que retorna um intervalo que é a multiplicação dos intervalos recebidos:

  IR mult(Intervalo x, Intervalo y);

- Operação que retorna um intervalo que é a divisão dos intervalos recebidos:

  IR div(Intervalo x, Intervalo y);

- Operação que retorna a intersecção entre dois intervalos recebidos como parâmetro:

  IR intersection(Intervalo x, Intervalo y);

- Operação que retorna a união entre dois intervalos recebidos como parâmetro:

  IR union(Intervalo x, Intervalo y);

- Operação que retorna true se o intervalo x esta contido no intervalo y:

  isIn(Intervalo x, Intervalo y);

- Operação que retorna o valor absoluto de um intervalo:

  abs(Intervalo x);

- Operação que retorna a distância entre dos intervalos:

  distance(Intervalo x, Intervalo y);

- Operação que calcula o comprimento de um intervalo:

  width();

- Operação que calcula o ponto médio de um intervalo:

  middle();

- Operação que retorna o intervalo simétrico de um intervalo:

  symmetric();

- Operação que verifica se um número de ponto flutuante está contido no intervalo:

  pertains(x);

- Operação que retorna o intervalo recíproco:

  IR reciprocal()

Operações Auxiliares em IR:

- Operação que arredonda para baixo o número para um número decimal específico de dígitos:

  roundDown(x, dec);

- Operação que arredonda para cima o número para um número decimal específico de dígitos:

  roundUp(x, dec);

- Operação que retorna o valor máximo dentre os valores recebidos por parâmetro:

  max(x1, x2, x3, x4)

- Operação que retorna o valor mínimo dentre os valores recebidos por parâmetro:

  min(x1, x2, x3, x4)

- Operação que retorna se os intervalos x e y são iguais:

  equals(Intervalo x, Intervalo y)

- Operação que retorna se o intervalos x está vazio:

  isempty(Intervalo x)

  O presente capítulo descreveu o desenvolvimento da biblioteca IntDroid, frisando a importância da exatidão dos resultados, a utilização e aplicabilidade desta solução. A seção 4.2 descreveu as tecnologias utilizadas no desenvolvimento da biblioteca, as linguagens de desenvolvimento Android além das principais diferenças da linguagem Java Android, diferenças estruturais do hardware móvel e modificações do Google apresentando nova máquina virtual, novo compilador e envolvendo modificações da linguagem através do Android SDK. A seção 4.3 apresentou toda a estrutura e maneiras de integração de uma biblioteca AAR Android. A seção 4.4 especificou de maneira simplificada as classes contendo as operações de IntDroid que podem ser visualizadas com maior detalhes nos Anexos MainActivity e IR desta dissertação.

  O Capítulo 5 apresenta os resultados obtidos nas operações de IntDroid bem como os resultados das operações, de mesma entrada, para as bibliotecas IntPy, Java-XSC e C-XSC. Analisa-se os resultados obtidos realizando comparação dos resultados das operações nas demais bibliotecas, verificando que a IntDroid obteve resultados similares com as bibliotecas citadas, além de apresentar o tempo necessário para realizar as operações em cada biblioteca. A seção 5.1 apresenta resultados de utilização de recursos da biblioteca IntDroid no Android, utilização de memória e processamento, fundamentando a funcionalidade das operações intervalares no hardware móvel.

5 RESULTADOS DO INTDROID

Os resultados das operações do IntDroid podem ser observadas nas tabelas deste Capítulo, sendo informadas as entradas, operações, operadores indicados de cada biblioteca para a operação, os resultados obtidos e o tempo de processamento. Todas operações foram realizadas utilizando a precisão $\delta = 10^{-9}$ seguindo o padrão de exatidão encontrado nos testes das bibliotecas C-XSC, Java-XSC e IntPy. Os valores resultantes das operações são comparados com a biblioteca IntPy, Java-XSC e C-XSC a quesito de validação e comparação da exatidão dos resultados. Pode-se observar que as diferenças entre os valores dos resultados das bibliotecas tem como causa diferenças do trabalho com operações de ponto flutuante e principalmente de Hardware na medição dos tempos. No caso da IntDroid isto é acentuado por utilizar Hardware móvel, com diferenças de Sistema Operacional citadas no Capítulo 3, além das diferenças de compiladores. Todas operações seguiram os critérios de criação do intervalo de suas respectivas bibliotecas utilizando o δ juntamente com o arredondamento direcionado para obter a precisão, seguida da operação detalhada de acordo com cada tabela. É importante salientar que o IntPy apesar de ter a sua precisão definida como as demais bibliotecas, acaba permitindo uma maior amostra, apresentando mais dígitos do que o solicitado na operação. Isso pode ocorrer devido ao fato da biblioteca utilizar uma ponte para realizar o arredondamento direcionado através de um código na linguagem C em conjunto com o Python, sendo Python uma linguagem interpretada. Os valores da biblioteca Java-XSC apresentados foram obtidos através do trabalho desenvolvido por Ferreira (FERREIRA; FERNANDES; CAMPOS, 2005). Não foi possível reproduzir os resultados para medição dos tempos, por não ter sido possível reproduzir os resultados do artigo. Fato que pode ter ocorrido por algumas operações não terem fornecido o resultado junto a precisão definida na operação quanto os testes foram realizados.

A Tabela 1 apresenta o hardware utilizado nos testes, como processador e memória, além do Sistema Operacional utilizado e configurado no ambiente de testes para cada biblioteca.

A Tabela 2 apresenta os resultados da operação de adição de intervalos utilizando

Tabela 1: Hardware utilizados nos testes

Biblioteca	Dispositivo	Processador	Memória	O.S.
IntDroid	Samsung Tablet	1 GB Dual-Core	2 GB RAM	Android 3.1
C-XSC	PC IBM	1 GB Dual-Core	2 GB RAM	Linux Ubuntu 14.04
IntPy	PC IBM	1 GB Dual-Core	2 GB RAM	Linux Ubuntu 14.04

Tabela 2: Operação de soma, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Soma/[add]	[1.0,2.5]+[0.0,8.9]	[0.999999998, 11.400000002]	*0.0003s*
IntPy	Soma/[+]	[1.0,2.5]+[0.0,8.9]	[0.999999998, 11.400000002]	*0.0001s*
Java-XSC	Soma/[Add]	[1.0,2.5]+[0.0,8.9]	[0.999999999, 11.400000001]	-
C-XSC	Soma/[+]	[1.0,2.5]+[0.0,8.9]	[0.999999998, 11.400000001]	*0.000081s*

como dados de entrada o intervalo [1.0,2.5] somado ao intervalo [0.0, 8.9]. Com base na Tabela 2 podemos verificar que a biblioteca C-XSC obteve um intervalo solução mais preciso em comparativo ao IntPy, IntDroid e Java-XSC que obtiveram um intervalo solução similar. A biblioteca C-XSC obteve um tempo de processamento superior a IntPy e IntDroid, respectivamente. O tempo de processamento de Java-XSC não foi obtido.

Tabela 3: Operação de subtração, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Subtração / [sub]	[1.0,3.0] - [4.0,5.0]	[-4.000000002, -0.9999999979]	*0.0003s*
IntPy	Subtração / [-]	[1.0,3.0] - [4.0,5.0]	[-4.000000002, -0.999999997998]	*0.0001s*
Java-XSC	Subtração / [Sub]	[1.0,3.0] - [4.0,5.0]	[-4.000000001, -0.999999999]	-
C-XSC	Subtração / [-]	[1.0,3.0] - [4.0,5.0]	[-4.000000001, -0.999999999]	*0.000093s*

A Tabela 3 apresenta os resultados da operação de subtração de intervalos utilizando como dados de entrada o intervalo [1.0,3.0] subtraindo ao intervalo [4.0, 5.0]. A partir da tabela, analisamos que as bibliotecas C-XSC e Java-XSC apresentaram

um intervalo solução mais exato e também um tempo de processamento superior aos demais analisados, também vale salientar que o IntPy obteve um intervalo solução bastante similar em comparação ao IntDroid, mas com mais casas de precisão e um desempenho de tempo superior em comparação ao IntDroid.

Tabela 4: Operação de Multiplicação, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Multiplicação/ [mult]	[1.0,3.0] * [4.0,5.0]	[3.999999995, 15.000000008]	*0.0003s*
IntPy	Multiplicação / [*]	[1.0,3.0] * [4.0,5.0]	[3.999999998, 15.000000002]	*0.0001s*
Java-XSC	Multiplicação / [Mult]	[1.0,3.0] * [4.0,5.0]	[3.999999999, 15.000000001]	-
C-XSC	Multiplicação / [*]	[1.0,3.0] * [4.0,5.0]	[3.999999999, 15.000000001]	*0.000081s*

A Tabela 4 apresenta os resultados da operação de multiplicação de intervalos utilizando como dados de entrada o intervalo [1.0,3.0] multiplicando o intervalo [4.0, 5.0]. A partir da análise da Tabela 4 podemos verificar que a biblioteca C-XSC e Java-XSC obtiveram um intervalo solução mais preciso em comparação as demais, também obteve um tempo de processamento superior a IntPy e IntDroid, respectivamente.

Tabela 5: Operação de Divisão, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Divisão / [div]	[4.0, 5.0] / [1.0, 2.5]	[1.599999998, 5.000000002]	*0.0003s*
IntPy	Divisão/[/]	[4.0, 5.0] / [1.0, 2.5]	[1.599999998, 5.000000006]	*0.0001s*
Java-XSC	Divisão/[Div]	[4.0, 5.0] / [1.0, 2.5]	[1.599999999, 5.000000001]	-
C-XSC	Divisão/[/]	[4.0, 5.0] / [1.0, 2.5]	[1.599999999, 5.000000001]	*0.000081s*

A Tabela 5 apresenta os resultados da operação de divisão de intervalos utilizando como dados de entrada o intervalo [4.0 , 5.0] dividindo o intervalo [1.0 , 2.5]. Com base na análise da Tabela 5 podemos verificar que a biblioteca C-XSC obteve um intervalo solução mais preciso resultante da operação, seguido pelas bibliotecas IntPy e IntDroid, a qual obtiveram um intervalo igual. O Tempo de processamento da C-XSC foi superior aos demais, seguido por IntPy e por último IntDroid. O resultado obtido para Java-XSC conforme o artigo (FERREIRA; FERNANDES; CAMPOS, 2005), não

foi similar aos demais, podendo ter ocorrido algum engano no resultado obtido da tabela do artigo.

Tabela 6: Operação de Pertence, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Pertence / [pertains]	[0.0] pertains [-0.1, 0.1]	true	*0.0002s*
IntPy	Pertence / [pertains]	[0.0] pertains [-0.1, 0.1]	true	*0.0001s*
Java-XSC	Pertence / [pertains]	[0.0] pertains [-0.1, 0.1]	true	-
C-XSC	sem operação	-	-	-

A Tabela 6 apresenta os resultados da operação de pertence, na qual utiliza como dados de entrada o intervalo [-0.1 , 1.0] e retorna true se o valor [0.0] pertence ao intervalo. Com base na Tabela 6 verificamos que o retorno da função analisada é verdadeiro para IntPy, Java-XSC e IntDroid, essa operação é de grande importância para as bibliotecas, pois ela faz a verificação manual de que um valor está contido em um intervalo, assim validando o resultado obtido, a biblioteca C-XSC não possui a operação para testes, mas ele realiza a operação sempre que um novo intervalo é obtido fazendo a verificação do intervalo solução automaticamente. O tempo de processamento da IntPy foi superior que o tempo do IntDroid.

Tabela 7: Operação de Largura de um intervalo, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Largura / [width]	[1.0, 2.5]	1.5	*0.0002s*
IntPy	Largura / [width]	[1.0, 2.5]	1.5	*0.0001s*
Java-XSC	Largura / [width]	[1.0, 2.5]	1.5	-
C-XSC	Largura / [width]	[1.0, 2.5]	1.5	*0.000064s*

Tabela 7 apresenta os resultados da operação de largura de um intervalo, que utiliza como dados de entrada o intervalo [1.0 , 2.5] e retorna a largura do intervalo. Com base na Tabela 7 podemos verificar que a resposta de todas as biblioteca foram idênticas, mesmo se tratando de uma operação com intervalos, o resultado obtido é

um valor real, isso se dá pelo fato de que essa operação é utilizada para analisar a largura de um intervalo. Essa função é de suma importância para a matemática intervalar por ser bastante utilizada na verificação da qualidade do intervalo solução. O tempo de processamento da biblioteca C-XSC foi superior, seguido por IntPy e por último IntDroid.

Tabela 8: Operação de intervalo simétrico, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Simétrico / [symmetric]	[-1.0, 1.0]	[-1.0, 1.0]	*0.0002s*
IntPy	sem operação	-	-	-
Java-XSC	Simétrico / [symmetric]	[-1.0, 1.0]	[-1.0, 1.0]	-
C-XSC	Simétrico / [symmetric]	[-1.0, 1.0]	[-1.0, 1.0]	*0.000078s*

A Tabela 8 apresenta os resultados da operação de intervalo simétrico de um intervalo X, que utiliza como dados de entrada o intervalo [-1.0 , 1.0] e retorna o intervalo simétrico do intervalo recebido. Com base na Tabela 8 podemos verificar que a resposta da biblioteca IntDroid, Java-XSC e C-XSC foram similar com resultado de [1.0,-1.0] tornando o resultado final o intervalo [-1.0, 1.0] colocando de acordo com a prerrogativa da matematica intervalar na qual $\underline{x} \leq \overline{x}$. O tempo de processamento da biblioteca C-XSC foi superior seguida da IntDroid. Não foi obtido o resultado da operação em IntPy por não possuir a operação para teste.

Tabela 9: Operação do ponto Médio, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Ponto Médio / [middle]	[-10, 5]	-2.5	*0.0002s*
IntPy	Ponto Médio / [middle]	[-10, 5]	-2.5	*0.0001s*
Java-XSC	sem operação	-	-	-
C-XSC	Ponto Médio / [middle]	[-10, 5]	-2.5	*0.000084s*

A Tabela 9 apresenta os resultados da operação de ponto médio de um intervalo,

que utiliza como dados de entrada o intervalo [-10 , 5] e retorna o ponto médio do intervalo. Com base na Tabela 9 verificamos que todas as bibliotecas obtiveram o mesmo resultado com exceção de Java-XSC, que não apresentou resultado desta operação. Essa operação tem entrada um intervalo e como saída um valor real, por se tratar de uma operação dentro do intervalo, essa operação também é bastante utilizada para cálculos de qualidade de intervalo solução. O tempo de processamento foi superior pela biblioteca C-XSC se seguida de IntPy e IntDroid.

Tabela 10: Operação de Intervalo Recíproco, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Reciproco / [reciprocal]	[1.0, 4.0]	[0.2499999979, 1.000000003]	*0.0006s*
IntPy	Reciproco / [reciprocal]	[1.0, 4.0]	[0.249999999975, 1.000000001]	*0.0002s*
Java-XSC	Reciproco / [Reciprocal]	[1.0, 4.0]	[0.249999999, 1.000000001]	-
C-XSC	sem operação	-	-	-

A Tabela 10 apresenta os resultados da operação de intervalo recíproco, que utiliza como dados de entrada o intervalo [1.0 , 4.0]. Com base na Tabela, podemos verificar que a o intervalo solução gerado pelo IntPy foi mais preciso que o do IntDroid e Java-XSC, enquanto a biblioteca C-XSC não possui essa operação. O tempo de processando do IntPy também foi superior ao intDroid.

Tabela 11: Operação de Interseção, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Interseção / [intersection]	[1.23,1.89]∩[1.1,1.29]	[1.23, 1.29]	*0.0003s*
IntPy	Interseção / [&]	[1.23,1.89]∩[1.1,1.29]	[1.23, 1.29]	*0.0001s*
Java-XSC	Interseção / [Intersection]	[1.23,1.89]∩[1.1,1.29]	[1.23, 1.29]	-
C-XSC	Interseção / [&]	[1.23,1.89]∩[1.1,1.29]	[1.23, 1.29]	*0.00006s*

A Tabela 11 apresenta os resultados da operação de interseção de dois intervalos, que utiliza como dados de entrada os intervalos [1.23 , 1.89] e [1.1 , 1.29] e retorna

a interseção dos intervalos. Com base na Tabela 11 podemos verificar que ambas as bibliotecas tiveram o mesmo intervalo solução. O tempo de processamento da biblioteca C-XSC foi menor seguida da IntPy e por último a IntDroid.

Tabela 12: Operação de União, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	União / [Union]	[1.23,1.89]∪[1.1,1.29]	[1.1, 1.89]	*0.0004s*
IntPy	União / [Union]	[1.23,1.89]∪[1.1,1.29]	[1.1, 1.89]	*0.0002s*
Java-XSC	União / [Union]	[1.23,1.89]∪[1.1,1.29]	[1.1, 1.89]	-
C-XSC	União / [Union]	[1.23,1.89]∪[1.1,1.29]	[1.1, 1.89]	*0.000085s*

A Tabela 12 apresenta os resultados da operação de união de dois intervalos, que utiliza como dados de entrada os intervalos [1.23 , 1.89] e [1.1 , 1.29] e retorna a união dos intervalos. Com base na Tabela 12 podemos verificar que todas as bibliotecas analisadas obtiveram um resultado similar de intervalo [1.23 , 1.89]. O tempo de processamento da biblioteca C-XSC foi menor seguida da IntPy e por último a IntDroid.

Tabela 13: Operação que retorna se um intervalo X esta contido em um intervalo Y, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Contido / [isIn]	[1.23,1.89] $\supseteq$ [1.5, 1.6]	true	*0.0005s*
IntPy	Contido / [in]	[1.23,1.89] $\supseteq$ [1.5, 1.6]	yes	*0.0001s*
Java-XSC	Contido / [in]	[1.23,1.89] $\supseteq$ [1.5, 1.6]	true	-
C-XSC	sem operação	-	-	-

A Tabela 13 apresenta os resultados da operação de contido aonde se verifica se um intervalo esta contido em outro, essa operação utilizou como dados de entrada os intervalos [1.23 , 1.89] e [1.1 , 1.29] e retorna um valor boolean true ou false para IntDroid e Java-XSC e yes ou no para IntPy. Com base na Tabela 13 podemos verificar que a resposta nas bibliotecas IntDroid e IntPy foram as mesmas, ambas retornaram verdadeiro, sendo que a biblioteca C-XSC não tem a operação. O tempo de processamento da biblioteca IntPy foi menor e seguido da IntDroid.

Tabela 14: Operação de Distância, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Distância / [distance]	[1.23,1.89], [1.5,1.6]	0.27	*0.0004s*
IntPy	Distância / [distance]	[1.23,1.89], [1.5,1.6]	0.27	*0.0003s*
Java-XSC	Distância / [distance]	[1.23,1.89], [1.5,1.6]	0.27	-
C-XSC	sem operação	-	-	-

A Tabela 14 apresenta os resultados da operação de distância entre dois intervalos, essa operação utilizou como dados de entrada os intervalos [1.23 , 1.89] e [1.5 , 1.6] e retorna a distância entre os intervalos de entrada. Com base na Tabela 14 podemos verificar que o intervalo solução das bibliotecas IntDroid, IntPy e Java-XSC foi similar, sendo que a biblioteca C-XSC não tem a operação. O tempo de processamento da biblioteca IntPy foi melhor em relação ao do IntDroid.

Tabela 15: Operação do Absoluto do Intervalo, resultados e tempos de processamento

Biblioteca	Operação /Operador	Entrada	Saídas	Tempo/ Proc
IntDroid	Absoluto / [abs]	$\|[1.23, 1.89]\|$	[1.23, 1.89]	*0.0006s*
IntPy	Absoluto / [abs]	$\|[1.23, 1.89]\|$	[1.23, 1.89]	*0.0002s*
Java-XSC	Absoluto / [abs]	$\|[1.23, 1.89]\|$	[1.89]	-
C-XSC	Absoluto / [abs]	$\|[1.23, 1.89]\|$	[1.23, 1.89]	*0.000072s*

A Tabela 15 apresenta os resultados da operação de valor absoluto de um intervalo, essa operação utilizou como dados de entrada os intervalos $|[1.23, 1.89]|$ e retorna o valor absoluto do intervalo de entrada. Com base na Tabela 15 podemos verificar que o intervalo solução das bibliotecas analisadas foi similar, onde apenas Java-XSC apresentou resultado diferente, aparentemente o algoritmo da operação retorna somente o extremo superior do intervalo. Em relação ao tempo de processamento a biblioteca C-XSC obteve o melhor resultado, seguida da IntPy e por último a IntDroid.

Podemos através dos resultados das operações, verificar que as bibliotecas que utilizaram o Hardware Desktop, apresentaram tempos melhores, tanto pela estrutura do hardware quanto pela atenuação que o sistema operacional móvel agrega as

operações com o efetivo e constante controle dos recursos da biblioteca no Android OS.

5.1 Utilização de recursos da IntDroid no ANDROID OS

A utilização de recursos de hardware pelos aplicativos para plataformas móveis é uma preocupação constante, pois o controle do Sistema Operacional em relação aos recursos utilizados pelos aplicativos é muito rígido.

Tecnicamente, existem algumas limitações sobre o uso da memória Heap do Android. O problema é o agravante das próprias limitações dos dispositivos móveis, ao desenvolver uma aplicação você poderá encontrar problemas de *Memory Overflow* no Heap do Android. Para evitar esses problemas não deve-se trabalhar com um objeto muito grande de uma vez só na memória.

No início do desenvolvimento Android o hardware dos smartphones era mais limitado. Atualmente ainda existe a preocupação em relação ao consumo desses recursos, porém com os hardwares atuais muito mais potentes. Assim passa-se a pensar na utilização de técnicas de *background* e *release memory*, possibilitando um melhor controle desses recursos e permitindo que a biblioteca IntDroid seja utilizada em background liberando recursos do sistema para voltar a utilizar a capacidade total quando o sistema estiver com melhores condições de recursos de memória e processamento, sendo essa utilização da biblioteca IntDroid um recurso muito importante para que a aplicação que utilizar a biblioteca não apresente problemas de desempenho ou possa ser finalizada pelo Android. Na figura 10 é apresentada a utilização de recursos por parte da biblioteca IntDroid em estado de espera, onde é carregada pelo sistema operacional, mas não apresenta nenhuma operação sendo executada. Para realizar as medições foram utilizadas as ferramentas de debug e monitoramento do ambiente de desenvolvimento. A Figura 10 apresenta a utilização dos recursos por parte da biblioteca no estado de espera, aguardando ser acionada.

A figura 11 apresenta a biblioteca IntDroid utilizando recursos de memória ao executar uma série de 1000 operações, mostrando que neste cenário é utilizado 0.28MB de memória RAM dos recursos do Android OS.

A Figura 12 apresenta o gráfico de utilização do processamento em relação a execução de 1000 operações da biblioteca IntDroid, aumentando em torno de 9 % o consumo de recursos de processamento pelo Android OS.

5.2 Integração IntDroid e ANDROID SDK

Por padrão, o Android SDK inclui tudo que precisa para começar a desenvolver, mas não inclui todos recursos existentes para o desenvolvimento de aplicações An-

```
Applications Memory Usage (kB):
Uptime: 300870 Realtime: 300870

** MEMINFO in pid 2574 [br.com.sinope.intdroidapp] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     3209     3084        0        0    16384    14462     1921
  Dalvik Heap     1703     1648        0        0     1972     1414      558
 Dalvik Other      212      212        0        0
        Stack      172      172        0        0
   Other dev         4        0        4        0
    .so mmap      1005       96       68        0
   .apk mmap       151        0        8        0
   .ttf mmap        31        0       12        0
   .dex mmap      1884        0     1880        0
   .oat mmap      2215        0      696        0
   .art mmap      3890     3572        4        0
  Other mmap        21        4        0        0
     Unknown        98       96        0        0
       TOTAL     14595     8884     2672        0    18356    15876     2479

 Objects
               Views:       30         ViewRootImpl:        1
         AppContexts:        3           Activities:        1
```

Figura 10: Utilização de recursos de memória do IntDroid em estado de espera

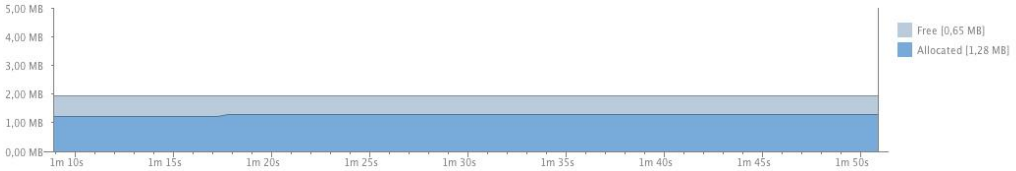

Figura 11: Utilização de recursos de memória do IntDroid em execução

droid. O SDK separa ferramentas, plataformas e outros componentes em pacotes que podem ser adquiridos conforme necessário usando o Android SDK Manager.

Para começar a adicionar pacotes personalizados ao Android SDK é preciso utilizar o Android SDK Manager.

Através do Android SDK Manager é possível gerenciar vários pacotes e incluir de forma local no seu Android SDK, pacotes para serem carregados adicionando assim novos recursos. Também é possível enviar pacotes ao Google Android, como forma de sugestão de melhorias ao Android SDK. Uma vez aceito o pacote torna-se público e disponível sendo listado como um recurso novo com opção de download pelo Android SDK Manager. Outra forma de realizar submissões de pacotes é enviando o pacote em formato Android Archieve Library diretamente na central de desenvolvimento Android `developer.android.com` que a sugestão de adesão ao Android SDK também será analisada pela equipe de desenvolvimento do Android. A importância de se integrar uma solução desenvolvida, às que já existem no Android SDK é que uma vez integrando o SDK, a atualização e o desenvolvimento contínuo da solução passa a ser realizada pela equipe da Plataforma Android.

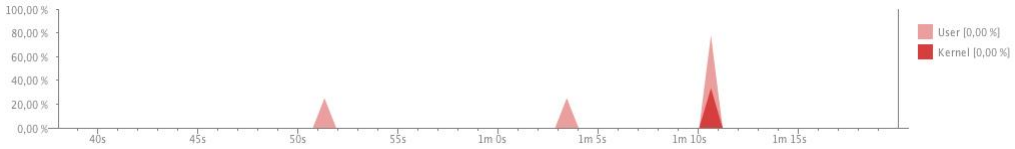

Figura 12: Utilização de recursos de processamento do IntDroid

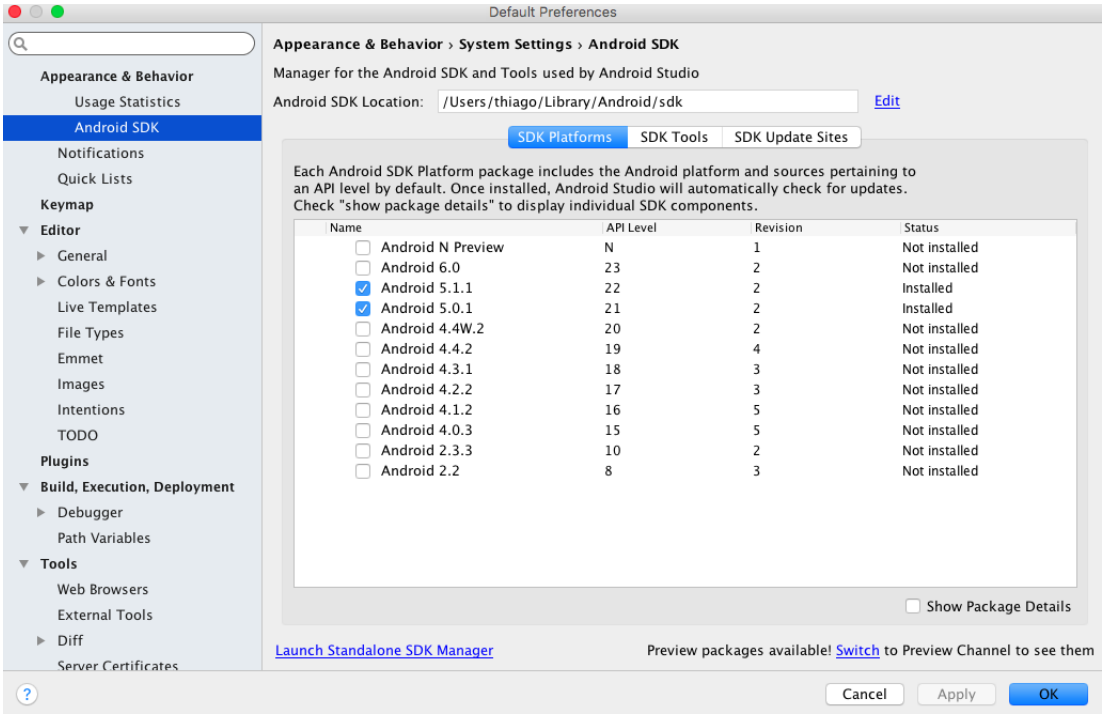

Figura 13: Android Manager SDK

6 CONCLUSÕES

O presente trabalho fornece solução de alta exatidão para plataforma Android, além de ser pioneiro em disponibilizar subsídios para o desenvolvimento de soluções de alta exatidão para os demais sistemas operacionais móveis. Tendo em vista que cada plataforma tem a necessidade de uma solução própria. Os sistemas operacionais móveis estão cada vez mais estáveis, confiáveis e com hardware compatível ao esforço computacional utilizados nos cálculos intervalares.

Objetivou-se com este trabalho implementar operações matemáticas em IntDroid e desenvolver algoritmos baseados na matemática intervalar.

Analisando-se os exemplos numéricos apresentados conclui-se que IntDroid obteve resultados satisfatórios quando comparado aos resultados encontrados no IntPy, Java-XSC e C-XSC, em precisão de máquina, tornando IntDroid uma ferramenta computacional confiável.

Esta biblioteca apresenta operações aritméticas e topológicas, mantendo toda portabilidade para possíveis incorporações a novos projetos.

IntDroid caracteriza-se como uma base para a realização de cálculos intervalares, assim como para a análise da eficiência imprescindível na prática da computação científica no ambiente móvel, onde a qualidade de um resultado depende do conhecimento e do controle que se possa ter sobre o erro.

Dentre os aspectos levantados para continuidade do trabalho destacam-se:

- Desenvolver o aplicativo no formato de uma calculadora intervalar para utilização da Biblioteca IntDroid;
- desenvolver operações de matrizes intervalares;
- desenvolver operações com funções intervalares, tais como, função exponencial, função logaritmo, função seno, função cosseno, função raiz quadrada, função potência intervalar;
- Intervalos Complexos;
- Intervalos Reais Vetoriais;

- Intervalos Complexos Vetoriais;
- Intervalos Complexos para Matrizes.

Os trabalhos futuros acima descritos são de suma importância para dar continuidade ao trabalho de desenvolvimento da biblioteca IntDroid, aumentando as funcionalidades da biblioteca, agregando novas operações à mesma e aumentando o seu poder de cálculo.

REFERÊNCIAS

ACIOLY, B. M. **Fundamentação Computacional da Matemática Intervalar**. 1991. Tese (Doutorado em Ciência da Computação) — UFRGS, Porto Alegre, Brasil.

CALLEJAS, B.; BEDREGAL, J.; DUTRA, E. M. JAVA-XSC: Estado da arte. , [S.l.], p.3, 2008.

CAMPOS, M. A. **Uma Extensão Intervalar para a Probabilidade Real**. 1997. 127p. Tese (Doutorado em Informática) — UFPE, Recife, Brasil.

D. HANSELMAN, B. L. **Matlab, Versao do Estudante, Guia do Usuario**. Brasil: Makron Books, 1999.

D. MARK, J. L. **Dominando o Desenvolvimento no iPhone**. São Paulo: Alta Books, 2009. v.1.

FERREIRA, R. V.; FERNANDES, B. J. T.; CAMPOS, M. A. Interval Computations with JAVA-XSC. **XXVIII Congresso Nacional de Matemática Aplicada e Computacional**, São Paulo,Brasil, 2005.

G. SILVEIRA, J. J. **Swift**: Programe para iPhone e iPad. 1.ed. São Paulo: Casa do Código, 2014.

GARROZI, A. A Aritmética Intervalar como Ferramenta para a solução de problemas de Otimização. , [S.l.], v.II, p.1, 2009.

GONçALVES, M. L. S. **JAVA-XSC**: Módulo Complexo Intervalar. 2012. Mestrado em Ciência da Computação — Universidade Federal do Rio Grande do Norte, Natal, Brasil.

HOLBIG, C. A.; Morandi, Jr., P. S.; CLAUDIO, D. M.; DIVERIO, T. A. Solving Real Life Applications With High Accuracy. In: JOUBERT, G. R.; NAGEL, W. E.; PETERS, F. J.; PLATA, O. G.; TIRADO, P.; ZAPATA, E. L. (Ed.). **Parallel Computing**: Current & Future Issues of High-End Computing. Malaga, Spain: Department of Computer Architecture, University of Malaga, Central Institute for Applied Mathematics, 2005. p.317–324. (John von Neumann Institute for Computing Series (Julich, Germany), v.33).

KULISCH, U.; MIRANKER, W. L. Computer arithmetic in theory and Practice. **Academic Press**, New York, 1981.

L. C. O. PEREIRA, M. L. S. **Android Para Desenvolvedores**. 1.ed. São Paulo: Brasport, 2009. v.1.

LECHETA, R. R. **Desenvolvendo para Windows 8**. 1.ed. São Paulo: Casa do Código, 2013.

LORETO, A. B. **Análise da complexidade computacional de problemas de estatísticas descritiva com entradas intervalares**. 2006. Ph.D. Dissertation — UFRGS, Porto Alegre, Brasil.

M. D. C. BALBONI L. M. TORTELLI, M. L. V. S. F. A. F. F. A. B. L. Critérios para Análise e Escolha de Ambientes Intervalares. **REIC, SBC**, [S.l.], v.II, p.1, 2014.

M. LEONIDAS, F. C. Cálculo numérico com aplicações. , [S.l.], p.1, 1987.

MONAGAN, M. B. The Collect Function in Maple. **Maple Newsletter**, [S.l.], 1989.

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

OLIVEIRA, P. W.; DIVERIO, T. A.; MORAES, C. D. Fundamentos de Matemática Intervalar. **Sagra-Luzzato**, Porto Alegre, Brasil, 1997.

R. KLATTE U. KULISCH, A. W. C. L.; RAUCH, M. **C-XSC - A C++ Class Lbrary for Extended Scientific Computing**. [S.l.]: Springer-Verlag, 1993.

RATSCHEK, H.; ROKNE, J. **New computer methods for global optimization**. Düsseldorf,Alemanha: Horwood Chichester, 1988.

RETO, M. **Professional Android Application Development**. Indianapolis: Wiley Publishing, 2009.

SANTIAGO, R. H. N.; BEDREGAL, B. R. C.; ACIÓLY, B. M. Formal Aspects of Correctness and Optimality of Interval Computations. **Formal Asp. Comput**, [S.l.], v.18, n.2, p.231–243, 2006.

SILBERSCHATZ A.; GALVIN, P. G. G. **Fundamento de Sistemas Operacionais**. 8ed.ed. [S.l.]: LTC, 2010.

STEIL, R. **IOS**: Programe para iPhone e iPad. 1.ed. São Paulo: Casa do Código, 2012. v.1.

VARJÃO, F. R. G. **IntPy**: Computação Científica Auto Validável em Python. 2011. Mestrado em Ciência da Computação — Universidade Federal de Pernambuco, Pernambuco, Brasil.

VIANA, J. F. **JFloat**: Uma biblioteca de ponto flutuante para a linguagem Java com suporte a arredondamento direcionado. 2007. Mestrado em Ciência da Computação — Universidade Federal do Rio Grande do Norte, Natal, Brasil.

ZHIVICH, M.; CUNNINGHAM, R. K. Secure Systems: The Real Cost of Software Errors. **IEEE Security & Privacy**, [S.l.], v.7, n.2, p.87–90, Mar./Apr. 2009.

ANEXO A MAINACTIVITY

```
package edu.br.ufpel.intdroidapp;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemSe
private EditText valor1EditText;
private EditText valor2EditText;
private Spinner operacoes;
private Button btncalcular;
private EditText resutadosEditText;
private int operacaoselecionada;
private double x1;
private double x2;
private double valorfinal;
private static final String TAG = "RESULTADOS";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_main);
operacoes = (Spinner) findViewById(R.id.operacoes);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
R.array.operacoes_array, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
operacoes.setAdapter(adapter);
operacoes.setOnItemSelectedListener(this);
btncalcular = (Button)findViewById(R.id.btncalcular);
resutadosEditText = (EditText) findViewById(R.id.resultadosEditText);
valor1EditText = (EditText) findViewById(R.id.valor1EditText);
valor1EditText.addTextChangedListener(valor1EditTextWatcher);
valor2EditText = (EditText) findViewById(R.id.valor2EditText);
valor2EditText.addTextChangedListener(valor2EditTextWatcher);
x1 = 0.0;
x2 = 0.0;
OnClickListener CalcularNovo = new OnClickListener() {
@Override
public void onClick(View v) {
IR nd = new IR(0, 0, 10);
IR ne = new IR(0.4, 0.4, 10);
for (int i = 0; i < 1000; i++) {
nd = IR.add(nd, ne);
}
resutadosEditText.setText(nd.toString());
Log.i(TAG, "Intervalo rotina 1000:" + nd);
}
};
btncalcular.setOnClickListener(CalcularNovo);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
int id = item.getItemId();
//noinspection SimplifiableIfStatement
```

```
if (id == R.id.action_settings) {
return true;
}
return super.onOptionsItemSelected(item);
}
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
operacaoselecionada = operacoes.getSelectedItemPosition();
//resutadosEditText.setText(Integer.toString(operacaoselecionada));
}
@Override
public void onNothingSelected(AdapterView<?> parent) {
}
private void updateStandard(){
View.OnClickListener CalcularNovo = new View.OnClickListener() {
@Override
public void onClick(View v) {
// change text of the EditText
// resutadosEditText.setText("Button Calcular clicado");
if (operacaoselecionada == 1){
//IR resp = IR.add(interval1, interval2);
//resutadosEditText.setText(String.format("%.02f", soma));
//resutadosEditText.setText(resp1.toString());
}
}
};
btncalcular.setOnClickListener(CalcularNovo);
}
private TextWatcher valor1EditTextWatcher = new TextWatcher() {
@Override
public void beforeTextChanged(CharSequence s, int start, int count, int after) {
}
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
try {
x1 = Double.parseDouble(s.toString());
} catch (NumberFormatException e) {
x1 = 0.0;
}
```

```
updateStandard();
}
@Override
public void afterTextChanged(Editable s) {
}
};
private TextWatcher valor2EditTextWatcher = new TextWatcher() {
@Override
public void beforeTextChanged(CharSequence s, int start, int count, int after) {
}
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
try {
x2 = Double.parseDouble(s.toString());
} catch (NumberFormatException e) {
x2 = 0.0;
}
updateStandard();
}
@Override
public void afterTextChanged(Editable s) {
}
};
}
```

ANEXO B IR

```
package edu.br.ufpel.intdroidapp;
public class IR {

public IR(double li, double ls, int precision) {
if (li < ls || Double.isNaN(li)
|| Double.isNaN(ls)) {
this.li = IR.roundDown(li, precision);
this.ls = IR.roundUp(ls, precision);
} else {
this.li = IR.roundUp(ls, precision);
this.ls = IR.roundDown(li, precision);
}
this.precision = precision;
}
public IR(double li, double ls) {
if (li < ls || Double.isNaN(li)
|| Double.isNaN(ls)) {
this.li = li;
this.ls = ls;
} else {
this.li = ls;
this.ls = li;
}
}

public IR(double x1, int precision) {
this.li = IR.roundDown(x1, precision);
this.ls = IR.roundUp(x1, precision);
```

```
this.precision = precision;
}

public IR(int precision) {
this.li = Double.NaN;
this.ls = Double.NaN;
this.precision = precision;
}

public static IR add(IR x1, IR x2) {
return new IR((x1.li + x2.li),
(x1.ls + x2.ls)
);
}

public static IR sub(IR x1, IR x2) {
return new IR((x1.li { x2.ls),
(x1.ls { x2.li)
);
}

public static IR mult(IR x1, IR x2) {
double p1 = x1.li * x2.li;
double p2 = x1.li * x2.ls;
double p3 = x1.ls * x2.li;
double p4 = x1.ls * x2.ls;
return new IR(IR.min(p1, p2, p3, p4),
(IR.max(p1, p2, p3, p4))
);
}

public static IR div(IR x1, IR x2) {
if (x2.pertains(0.0)) {
return new IR(Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY);
}
double p1 = x1.li * (1/x2.li);
double p2 = x1.li * (1/x2.ls);
double p3 = x1.ls * (1/x2.li);
```

```
double p4 = x1.ls * (1/x2.ls);
return new IR(IR.min(p1, p2, p3, p4),
(IR.max(p1, p2, p3, p4)));
}

public static IR sqrt(IR x1) {
IR returned;
if (x1.isempty()) {
returned = new IR(x1.precision());
} else {
returned = new IR(IR.roundDown(IR.sqrtinfinity(x1
.li()), x1.precision()), IR.roundUp(
IR.sqrtinfinity(x1.getls()), x1.precision()),
x1.precision());
}
return returned;
}

public static IR intersection(IR x1, IR x2) {
IR i;
if(x1.li > x2.ls || x1.ls < x2.li) {
i = null;
} else {
i = new IR(IR.max(x1.li, x2.li), IR.min(x1.ls, x2.ls));
}
return i;
}

public static IR union(IR x1, IR x2) {
IR i = new IR(IR.min(x1.li,x2.li), IR.max(x1.ls,x2.ls));
return i;
}

public IR symmetric() {
        return new IR(IR.roundDown(-this.ls, this.precision), IR.roundUp(-this.li,
}

public static boolean isIn(IR x1, IR x2) {
boolean retorno = false;
```

```
if(x2.li <= x1.li && x2.ls >= x1.ls) {
retorno = true;
}
return retorno;
}

public static double max(double x1, double x2, double x3, double x4) {
double max = x1;
if (max < x2) {
max = x2;
}
if (max < x3) {
max = x3;
}
if (max < x4) {
max = x4;
}
return max;
}



public static double max(double i1, double i2) {
double i = 0;
if(i1 > i2) {
i = i1;
} else {
i = i2;
}
return i;
}


 public static double min(double x1, double x2, double x3, double x4) {
        double min = x1;
        if (min > x2) {
            min = x2;
        }
        if (min > x3) {
```

```
            min = x3;
        }
        if (min > x4) {
            min = x4;
        }
        return min;
    }

public static double min(double i1, double i2) {
double i = 0;
if(i1 < i2) {
i = i1;
} else {
i = i2;
}
return i;
}


public double middle() {
        return (this.ls + this.li / 2);
    }

public static double abs(IR x) {
double i1 = x.li;
double i2 = x.ls;
if(i1 < 0) {
i1 = i1*(-1);
}
if(i2 < 0) {
i2 = i2*(-1);
}
double i = IR.max(i1,i2);
return i;
}

public static double distance(IR x1, IR x2) {
double i1 = x1.li - x2.li;
double i2 = x2.li - x2.ls;
```

```
if(i1 < 0) {
i1 = i1*(-1);
}
if(i2 < 0) {
i2 = i2*(-1);
}
double i = IR.max(i1,i2);
return i;
}

public boolean equals(Object x1) {
if (x1 instanceof IR) {
if (this.isempty() && ((IR) x1).isempty())
return true;
if (Math.abs(this.li - ((IR) x1).li) < IR.EPSILON
&& Math.abs(this.li - ((IR) x1).li) < IR.EPSILON) {
return true;
}
}
return false;
}

public String toString() {
String returned;
if (this.isEmpty())
returned = "[]";
else
returned = "[" + this.li + ", " + this.ls + "]";
return returned;
}

public double width() {
return (this.ls - this.li);
}

public boolean isEmpty() {
return (Double.isNaN(this.li) && Double.isNaN(this.ls));
}
```

```
public boolean pertains(double x1) {
return (x1 >= this.li && x1 <= this.ls);
}

public IR reciprocal() {
if (this.pertains(0.0)) {
return new IR(Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, this.precision);
}
return new IR(IR.roundDown((1.00 / this.ls),
this.precision), IR.roundUp((1.00 / this.li),
this.precision), this.precision);
}

public static IR generateIR(String x1, int precision) {
double d = Double.parseDouble(x1);
return new IR(IR.roundDown(d, precision), IR
.roundUp(d, precision), precision);
}
}
```

ANEXO C CÓDIGO FONTE DOWNLOAD

http://ufpel.sinope.com.br/download/intdroid

UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Biblioteca de Matemática Intervalar para Android

THIAGO DAVISON GONÇALVES GONÇALVES

Pelotas, 2016