ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

"*Siscot* – Sistema Computacional para análise de Curvas de *Cot* no auxílio do estudo de genomas"

Trabalho de Conclusão de Curso

Engenharia da Computação

Aluna: Maíra Paschoalino Fernandes
Orientador: Prof. Fernando Buarque de Lima Neto

Recife, maio de 2005

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

"*Siscot* – Sistema Computacional para análise de Curvas de *Cot* no auxílio do estudo de genomas"

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Aluna: Maíra Paschoalino Fernandes
Orientador: Prof. Fernando Buarque de Lima Neto

Recife, maio de 2005

UNIVERSIDADE
DE PERNAMBUCO

Maíra Paschoalino Fernandes

"*Siscot* – Sistema Computacional para análise de Curvas de *Cot* no auxílio do estudo de genomas"

Resumo

Este trabalho de conclusão de curso apresenta a construção de um software melhor estruturado e adequado para o estudo de análise de genomas baseado na cinética de reassociação de DNA com base no programa original de Pearson, Davidson e Britten. O programa original foi desenvolvido no fim da década de 70 em linguagem Fortran e tinha como objetivo a análise de cinéticas de reassociação de ácidos nuclêicos para os primeiros estudos de organização genômica de vegetais

As adapatações feitas neste trabalho permitiram (i) a utilização do programa em terminais gráficos, (ii) a geração da curva de regressão exibida em modelos gráficos que identificam as diferentes curvas por cores distintas e com legendas facilmente lidas e entendidas (iii) a apresentação "*userfriendly*" de todos os resultados numéricos bem como a interação com o aplicativo.

Para incluir as facilidades acima descritas, o programa foi desenvolvido de maneira que sua interface de entrada e saída de dados fosse bastante intuitiva e permitisse ao usuário com conhecimento de cinética de reassociação a utilização do programa para diferentes fins como por exemplo idealizar o tamanho e estrutura do genoma, sem que todas as bases do mesmo sejam seqüenciadas.

Além do melhoramento visual em relação ao programa antigo, a nova aplicação incorporou maior confiabilidade nos resultados produzidos. Isto porque o programa de Pearson resultava direfentes valores quando executados em máquinas com configurações distintas. Para isto, foi feita uma busca detalhada em seu código para que fossem achados eventuais erros e pontos de possíveis falhas.

Uma outra novidade incorporada ao programa foi a possibilidade de sua utilização em aulas que permitam aos alunos entenderem, na prática, os passos envolvidos na determinação das constantes de reassociação.

Enfim, a disponibilização deste programa proposto para o trabalho de conclusão com uma interface mais próxima ao entendimento dos pesquisadores facilitará o seu trabalho, na área de genômica de plantas, uso no sistema de clonagem por fracionamento de *Cot* do DNA genômico contribuindo com os estudos e pesquisas na área da genética.

Abstract

This work, an obligatory requirement for the Degree of Engineering in Computing Systems, presents the construction of a piece of software on genetics. It is better structured and suited for users needs than its original version. The system aims at helping on the study of analysis of genomes based in the kinetic of reassociation of DNA in the original program of Pearson, Davidson and Britten. The original program was developed in the end of the 70 decade in computer language FORTRAN and had as objective the analysis of kinetic of reassociation of nucleic acids during the first studies of vegetable genomic organization.

The upgrade put forward in this work has allowed to (i) the use of the new system in graphic terminals, (ii) the generation of regression curves shown in graphical manner that is much better understandable than the previous version, and (iii) the "userfriendly" presentation of all the numerical results as well as the interaction with the applicatory one.

To include all benefits described above, the new system was developed in a way that its interface (data input/output) were intuitive and allowed an user with knowledge of kinetic of reassociation to use the program for different purposes. For example to infer the size and structure of the genoma, without having all the bases exactly sequenced.

Beyond the interface improvement, in relation to the old program, the new application incorporated greater precision in the produced results. This was obtained because the original program of Pearson resulted diferent values when processed in different machines with distinct configuration. For this, a detailed search in its code was made so that eventual errors and points of possible imperfections were found.

Another incorporated novelty to the program was the possibility of its use in classes, where pupils could better understand, within hands-on activities, all step involved in determining reassociation constants.

At last, the proposed system is thought to be a great help for practitioners not only because of the interface enhancement but also because of some add-ons, performance instrument on studies and researches in the genetic biology area, and potential use as a teaching tool in: plants genomic area and use in the system of cloning by fractionation of *Cot* of the DNA genomics.

Índice

Índice de Figuras

Índice de Tabelas

Agradecimentos

Primeiramente agradeço a Deus por essa vitória e por estar sempre guiando a minha vida.

Agradeço também aos meus pais, que muito me ajudaram a cumprir mais essa etapa da minha vida, deixando muitas vezes de realizar alguns de seus sonhos para estarem sempre me dando condições para que eu pudesse chegar nesta etapa tão sonhado de minha vida.

Agradeço aos meus amigos e colegas de turma, desde aqueles que mesmo de longe sempre me incentivaram até aqueles que batalharam, conviveram e festejaram comigo todo esse período de Universidade. Agradeço em especial ao aluno Rodrigo Brayner, que me auxiliou nos estudos preliminares deste projeto.

Enfim, agradeço aos meus orientadores Fernando Buarque e Carlos Winter que me ensinaram e torceram junto comigo para que este projeto pudesse ser realizado.

Capítulo 1

Introdução

1.1 Descrição da área deste trabalho

A publicação do artigo: "Molecular structure of the nucleid acids", de Francis Crick e James Wattson [7], publicado em 25 de abril de 1953 na revista Nature, impulsionou os estudos da ciência na busca pela origem da vida. Esta descoberta foi um ponto culminante nas pesquisas realizadas por inúmeros cientistas durantes anos e o trabalho destes dois pesquisadores permitiu que, teoricamente, se pudesse ler e interpretar a combinação genética de qualquer organismo.

Os estudos sobre estrutura e função do DNA durante os últimos cinquenta anos levaram a um conhecimento detalhado dos mecanismos envolvidos na expressão gênica. Os primeiros organismos a serem analisados foram aqueles com organização genômica mais simples, como vírus e bactérias. A medida que o conhecimento aumentava e os métodos progrediam, problemas mais complexos foram abordados. Atualmente pode-se conseguir a seqüência completa de um genoma com milhões de pares de bases.

Hoje em dia, o conhecimento acumulado sobre a estrutura do DNA muito contribuiu para a compreensão da função e evolução dos genes, auxílio no estudo de paternidade, determinação de doenças hereditárias e seus portadores, ajuda nas pesquisas criminalísticas, terapia gênica, entre outras diversas aplicações [6].

As descobertas nesta área permitiram aplicar os conhecimentos obtidos nos estudos sobre genomas vegetais, mais precisamente sobre as estrutura do DNA destes. Os resultados obtidos fornecem subsídios ao combate de pragas em lavouras, melhoria genética das espécies e, conseqüentemente, o aumento da sua produtividade. O estudo de genomas vegetais vem crescendo muito nos últimos anos e será abordado no decorrer deste trabalho [8].

Vários genomas já foram seqüenciados completa ou parcialmente: o genoma humano, de camundongo e rato, centenas de vírus e bactérias, fungos, alguns invertebrados, entre outros. Porém, a área de plantas ainda é muito pouco explorada, restrita apenas à *Arabidopsis thaliana*, o arroz (*Oryza sativa*) e à coleção de genes expressos da cana-de-açúcar (*Saccharum officinarum L.*) e do milho (*Zhea mays L.*) [8]. A dificuldade que os cientistas encontram para desenvolverem os estudos de genomas vegetais se deve a dois fatores fundamentais: (1) ao tamanho e à (2) repetição de bases nitrogenadas nos mesmos.

O tamanho dos genomas das plantas, se comparados com os genes humanos que possuem 2,8 bilhões de pares de bases, podem possuir até 100 bilhões de pares de bases. Desta maneira, o estudo de um genoma vegetal requer muito tempo e conseqüentemente grandes investimentos de

pesquisa. Assim, se fizermos um paralelo entre o tempo gasto para o seqüenciamento do genoma humano, que demorou cerca de três anos, poder-se-ia imaginar quanto o tamanho do genoma das plantas inviabilizaria o desenvolvimento de seus estudos e seqüenciamentos.

O segundo aspecto encontrado nos genomas das plantas, como citado anteriormente, é a repetição das suas sequências, que por sua vez, contribuem muito pouco a nível de expressão gênica. Eles codificam poucas informações para síntese de proteínas, podendo muitas vezes ser ignorados.

Baseado nestas informações, os cientistas descobriram que se estas bases repetitivas fossem filtradas, reduzir-se-ia drasticamente o tempo para os estudos dos genomas de plantas. Para que esta redução fosse feita, utilizaram duas técnicas já usadas entre os geneticistas: enzimas de restrição, onde os genes são cortados em locais específicos, para a seleção das seqüências menores – a tática "dividir para conquistar"; e uma outra técnica chamada *Cot Curves*, onde as moléculas são primeiramente desnaturadas (quando as fitas estão separadas) e posteriormente renaturadas (quando as fitas se unem). Especificamente, nesta segunda técnica, como as fitas repetitivas se unem com maior rapidez do que as fitas que contém seqüências não repetitivas, pode-se remover as primeiras e trabalhar apenas com as seqüências ricas em genes. Esta segunda técnica foi de fundamental importância neste trabalho [13].

Em 1960, o pesquisador Roy Britten e alguns colaboradores, começaram uma investigação sobre cinética de reassociação de DNA usando esta técnica que foi chamada de "plotagem de curvas de Cot" [4][5]. Através desta técnica foi possível calcular o tamanho do genoma; fração de seqüências simples do genoma; número, tamanho e complexidade das bases repetidas dos componentes do DNA, além de ajudar na compreensão da evolução dos genomas das espécies próximas às estudas. Vemos como grande vantagem este tipo de análise, a possibilidade de conhecer um genoma, tendo uma idéia de seu tamanho e estrutura, sem que todas as bases do genoma sejam seqüenciadas. Por isso, a cinética de reassociação de DNA foi proposta como uma técnica de baixo custo no estudo de genomas de plantas.

Hoje, diversos ramos e áreas de pesquisa como a biologia, por exemplo, se beneficiam dos avanços tecnológicos. Desta maneira, a informática se torna indispensável para o progresso científico nessas áreas. Uma grande contribuição da informática na biologia seria, por exemplo, seu uso no seqüênciamento e análise das bases nitrogenadas que compõe o DNA, gerados pelos projetos genoma. Este processo de sequenciamento seria de um trabalho exaustivo se não fossem as abordagens computacionais.

Assim, estas abordagens auxiliam os pesquisadores não só a conhecer as estrutura dos genes, analisar se esta seqüência tem confiabilidade e qualidade, montar e analisar seus mapas genéticos, mas principalmente, prover plataformas para a manipulação e análise de dados e desenvolver novos algoritmos que auxiliem e manipulem a mineração dos mesmos.

1.2 Descrição do problema

Os cientistas que trabalham com genômica possuem muita dificuldade para sequenciar e estudar os DNAs de origem vegetal, visto que estes genomas possuem um número muito alto de seqüências repetidas, tornando o seu estudo exaustivo e custoso.

A cinética de renaturação do DNA permite fracionar os genomas de acordo com as freqüências de repetição de suas seqüências. Assim, a fração que reassocia mais rapidamente possui uma sequência moderamente ou altamente repetitiviva carregando, muitas vezes, uma carga de informações genéticas irrelevantes.

Após a seleção da seqüência a ser estudada, pode-se construir a curva de *Cot* e obter as informções sobre o genoma que possam ser retiradas através de sua análise como por exemplo o tamanho do genoma; fração de seqüências únicas do genoma; número, tamanho e complexidade das bases repetidas dos componentes do DNA, como já citado anteriormente.

Para isso existe um programa escrito na linguagem Fortran, desenvolvido na década de 70 por Pearson, Davidson e Britten [12] para o cálculo dos componentes da curva de reassociação de DNA genômico de eucariotos.

Porém, este programa possui algumas limitações pois gera gráficos de forma não amigável ao usuário, escrito numericamente e no próprio *prompt* em que o programa é executado dificultando a manipulação dos dados obtidos pelo mesmo.

Além disto, só pode ser utilizado em terminais não gráficos, tendo sido originalmente desenvolvido para computadores da série PDP. Mesmo ainda tendo sido atualizado e adaptado para uso em máquinas com sistema operacional UNIX (Linux, Solaris, etc.), este roda em terminal não gráfico.

Um outro problema encontrado também neste programa de Pearson, foi a inconsitência dos seus resultados de acordo com a característica da máquina que ele era rodado, visto que ele gera difentes valores quando compilado em máquinas com diferentes configurações ou sistemas operacionais ocasionada, por exemplo, pela forma que os tipos de dados são interpretados em cada máquina.

1.3 Justificativa deste trabalho

Atualmente, existe um programa escrito na linguagem Fortran [12] desenvolvido na década de 70 pra o cálculo dos componentes da curva de renaturação de DNA genômico de eucariotos. Este programa está em uso no Instituto de Ciências Biomédicas do Departamento de Parasitologia Da Universidade de São Paulo – USP, segundo o parecer escrito pelo Professor Doutor *Carlos Eduardo Winter,* esse incluido no ANEXO-1.

No programa referido acima [12], foi usado um algoritmo destinado para regressão não-linear desenvolvido por um matemático chamado Levenverg-Marquardt [9][10]. Este algoritmo é uma técnica usada para a localização de mínimos locais de funções expressas pela soma dos quadrados de funções não-lineares [9][10].

O referido programa trabalha para o ajuste de segunda ordem do DNA. A ordem se refere as taxas observadas nas reações que dependem principalmente da concentração, temperatura e presença ou não de um catalisador para a indução da reação, que no caso específico do cálculo de renaturação do DNA, se refere à formação das ligações das pontes de hidrogêneo entre as duas fitas da molécula de DNA.

O programa em lide permite fazer a regressão de toda a curva de *Cot* extraindo seus componentes principais, ou seja, os fragmentos com freqüências parecidas – cópias únicas e repetitivas, que farão parte da parcela do genoma fundamental nos estudos do genoma.

Além do programa não ter sido testado para outras funções de ajuste, ele não possui uma interface amigável com o usuário e, em algumas ocasiões, produz erros em tempo de execução [12].

Então, mediante as descobertas e avanços tecnológicos, encontra-se no campo da engenharia da computação, a possibilidade de colaborar com o avanço das pesquisas em Genética vegetal, desenvolvendo-se assim, um software melhor estruturado e adequado para o estudo de análise de genomas baseado na cinética de reassociação de DNA.

1.4 Objetivos deste trabalho

Considerando a necessidade de melhorar o programa original de Pearson, Davidson e Britten para análise de genomas baseado na cinética de reassociação de DNA, surgiu a idéia de construir um novo sistema que pudesse contribuir nos estudos da genética vegetal. Os objetivos deste trabalho portanto são:

- migrar o programa escrito em linguagem procedural Fortran para a linguagem Java com paradigma orientado a objetos;
- fundir o programa de Pearson para análise de genomas baseado na cinética de reassociação de DNA, com o programa contido em seu pacote – *Cotfil* – para criação de arquivos de entrada de dados.
- revisar o código do programa existente;

1.5 Resultados Esperados

Este trabalho visa não somente a disponibilizar ao usuário uma interface mais amigável com relação a existente de Pearson [13], mas também aumentar sua eficiência e sua funcionalidade.

Ainda se espera também que a união do sistema incluso no programa de Pearson (*Cotfil*) ao sistema principal possa aumentar a produtividade dos pesquisadores, usuários do programa pois atualmente este programa é gerado separadamente, deixando o processo mais lento e passível a erros, além de ser extremamente limitado para com a entrada dos pares de dados no que diz respeito ao número de entradas e sequenciamento dos mesmos.

Como o programa será escrito em uma liguagem orientada a objetos – Java – o novo código estará estruturado de maneira modular, facilitando o incremento de novas funcionalidades em trabalhos futuros e facilitando o seu entendimento, já que o novo sistema se organizará em classes independentes ou parcialmente dependendentes.

Além disso, o sistema ficará portável, podendo este rodar em qualquer plataforma ou máquinas com diferentes configurações. Além destes ganhos descritos acima, a preocupação de revisar o código do programa original deve produzir resultados mais seguros e confiáveis para o usuário.

Enfim, com os ganhos em sua interface e facilidade de manipulação dos dados tanto de entrada como os de saída, propõem novos usos para a aplicação como por exemplo o exposição do sistema em aulas para permitir que os alunos entendam, na prática, os passos envolvidos na determinação das constantes de reassociação;

1.6 Sumário

Esta monografia é composta inicialmente de dois capítulos tratando de revisão bilbliográfica para as áreas de biologia e computação.

O segundo trás as explicações de como o processo de hibridização de ácidos nucléicos ocorre nos genomas vegetais explicando ainda como ocorrem as etapas de renaturação e desnaturação dos mesmos. Ainda é mostrado o processo de composição das curvas de *Cot* e suas características principais além de exemplificar a prática experimental em mapeamento genético em laboratório.

O terceiro aborda as características das linguagens Java e Fortran, suas vantagens e desvantagens, e a justificação do uso do paradigma orientado a objetos proposto para o desenvolvimento do novo sistema.

No quarto capítulo se mostra o processo de desenvolvimento do sistema proposto, detalhando cada etapa contida no mesmo desde a coleta de informações preliminares e construção de protótipos ao desenvolvimento propriamente dito.

O trabalho possui ainda um capítulo para resultados com ilustrações e exemplos do novo sistema, além de uma seção específica para comparações entre execuções dos programas testados/propostos.

Enfim, o trabalho se encerra com um capítulo de conclusão e algumas propostas de trabalhos futuros.

Capítulo 2

Conceitos de Biologia: Genética e cinética de reassociação

Os conceitos teóricos são de grande auxílio no desenvolvimento de um projeto, principalmente quando o projeto se destina a uma área fora do conhecimento do executor, como é o caso do sistema proposto neste trabalho de conclusão de curso, destinado ao estudo de análise de genomas baseado na cinética de reassociação de DNA – notadamente fora da área de Computação. A seguir serão descritas algumas definições que contribuirão para o entendimento e desenvolvimento do projeto como um todo.

2.1 Reassociação de Ácidos Nucleicos

O DNA dupla fita, encontrado normalmente nas células, é denominado nativo. Esta estrutura em dupla hélice é responsável pelas propriedades físico-químicas das soluções de DNA nativo, como exemplo a viscosidade. No DNA estas duplas fitas são mantidas unidas pela cooperatividade das pontes de hidrogêneo que as compõe. Apesar de poucas pontes de hidrogêneo não possuírem força pra manter as fitas unidas, um conjunto delas é capaz disto.

Assim, quando se aquece a fita de DNA ou ela é submetida a um pH alcalino, as pontes de hidrogêneo das extremidades se rompem tornando o rompimento das ligações seguintes mais fácil, ocasionando uma perda nas propriedades físico-químicas da solução (a perda da grande viscosidade por exemplo, observada no seu estado nativo, como dito acima). Por este motivo, a resposta ao aumento de temperatura é exponencial e não-linear.

A renaturação ou reassociação ocorre aos poucos numa temperatura dada, após as duas fitas terem sido separadas. Desta maneira, quando estas fitas estão separadas ou desnaturadas, a solução absorve uma maior quantidade de luz ultravioleta (260 nm) comparando quando estão unidas ou nativas. Na reassociação, a absorção de luz diminui à medida que a quantidade de DNA dupla fita aumenta. O processo de reassociação segue uma cinética própria que pode ser descrita matematicamente. A quantidade de DNA dupla fita aumenta com o tempo de incubação (t) e é dependente da concentração inicial de DNA fita dupla (Co) [6]. A concentração de DNA fita simples pode ser plotada em função do produto de Co e t (Cot), originando sempre uma curva sigmóide característica para cada DNA, dependendo da sua origem biológica, chamada curva de Cot.

2.2 Análise de Curvas de *Cot*

Desde 1970, os genomas vêm sendo investigados pelas ferramentas quantitativas de reassociação do DNA. Este processo é chamado de curvas de *Cot*.

A análise das curvas de *Cot* teve início com estudos de alguns pesquisadores como *Britten e Kohne* [4][5] anos antes, quando eles verificaram que a célula do rato continha múltiplas cópias de várias seqüências de DNA similares. Esta verificação se deu graças ao estudo do processo de reassociação. Este grupo de pesquisadores ainda introduziu no âmbito genético a possibilidade de representar as reações de renaturação e denaturação vistas nas curvas de *Cot* em um modo gráfico.

Eles foram ainda capaz de chegar a uma equação que mostrou que a quantidade de renaturação observada era proporcional ao produto da concentração inical pelo tempo da reação como mostra a Figura 1, onde C é a concentração de DNA de fita simples na solução, Co a concentração de DNA de fita dupla, K_2 é a constante de segunda ordem correspondendo à reação de renaturação e *Cot* é a concentração inicial versus o tempo da reação.

$$\frac{C}{Co} = \frac{1}{1 + k_2 Co t}$$

Figura 1. Equação mostra quantidade de renaturação é proporcional ao *Cot*

Estes cálculos ainda foram plotados em um gráfico *C/Co* versus *Cot*, de segunda ordem, que foram de grande utilidade na observação de alguns aspectos do DNA analisado como por exemplo as distintas fazes de renaturação e denaturação do mesmo como mostrado na Figura 2.

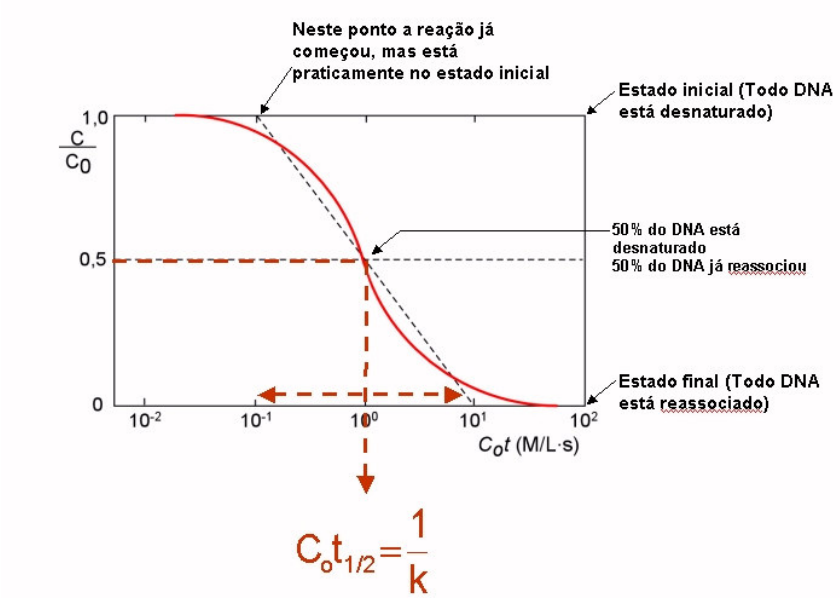

Figura 2. Cinética de renaturação ou denaturação (curva de *Cot*) de DNA de um procarioto hipotético

No estudo de genomas cuja complexidade analítica é muito grande, como nos eucariotos, observou-se a existência de três tipos de seqüências contidas no genoma: simples, contendo uma

fração de seqüências de complexidade relativamente pequena, repetição intermediária e seqüências únicas que possuem seqüências mais complexas, contendo a maioria dos genes fundamentais. Esta última normalmente representa a maior parte do genoma [6].

A complexidade de um genoma é dado não só pelo seu comprimento , mas também pela quantidade de informações que ele possui. No caso das plantas que possuem muitas sequências repetitivas, apesar de grande, seu genoma não é muito complexo.

A curva de *Cot* é capaz de mostrar exatamente estas características: complexidade e tamanho das seqüências. Assim, quanto mais complexa (possuir um número menor de repetições de pares de bases) for uma seqüência, mais tempo ela irá passar para encontrar a sua complementar se reassociando em *Cot´*s maiores.

A técnica de *Cot* se utiliza de um composto chamado *hydroxiapatita* que é usado para separar a dupla hélice em seqüências simples de DNA. Esse composto é um tipo de fosfato de cálcio que é absorvido pelo DNA dupla fita (dsDNA). O mesmo não acontece com o DNA fita simples, que passa direto pela coluna de hidroxiapatita, usada no experimento. Esta técnica permite verificar o quanto de dsDNA se tem após um determinado tempo de reação separando-se assim o DNA altamente repetitivo e que não codifica proteínas, em sua grande maioria, dos de cópia única que carregam a porção rica em informações genéticas.

As curvas de *Cot* usam a capacidade que o DNA tem de uma vez separadas as duplas fitas (desnaturação), elas voltarem a formar a dupla hélice. Então seguindo o princípio de Lê *Chatellier* [13] numa temperatura e concentrações fixas, as fitas complementares das seqüências mais freqüentes ou em maior concentração se encontram mais freqüentemente. Desta maneira, mede-se a quantidade de DNA dupla fita formada e obtém a curva de *Cot*. Assim, a medida que o tempo passa, cadas vez mais as seqüências menos freqüentes se encontram e viram dupla fita como é mostrado na Figura 3. Esta reação nunca chega ao fim, pois algumas sequências são tão pouco freqüentes que nunca se encontram.

De acordo com a Figura 3. é possível observar três regiões distintas. A região em vermelho representa o componente de cópia única, onde poucas repetições são verificadas no genoma. Na maioria das vezes, esta região representa a maior das porções e contém a maioria dos genes fundamentais A região em azul representa o componente altamente repetitivo, ou seja, porção que possui seqüências de complexidade relativamente pequena, as vezes repetidas milhares de vezes no genoma. E enfim, a região em verde é uma região intermediária onde estão incluídos certos tipos de genes como por exemplos os que codificam rRNA e alguns cujas funções são desconhecidas [21].

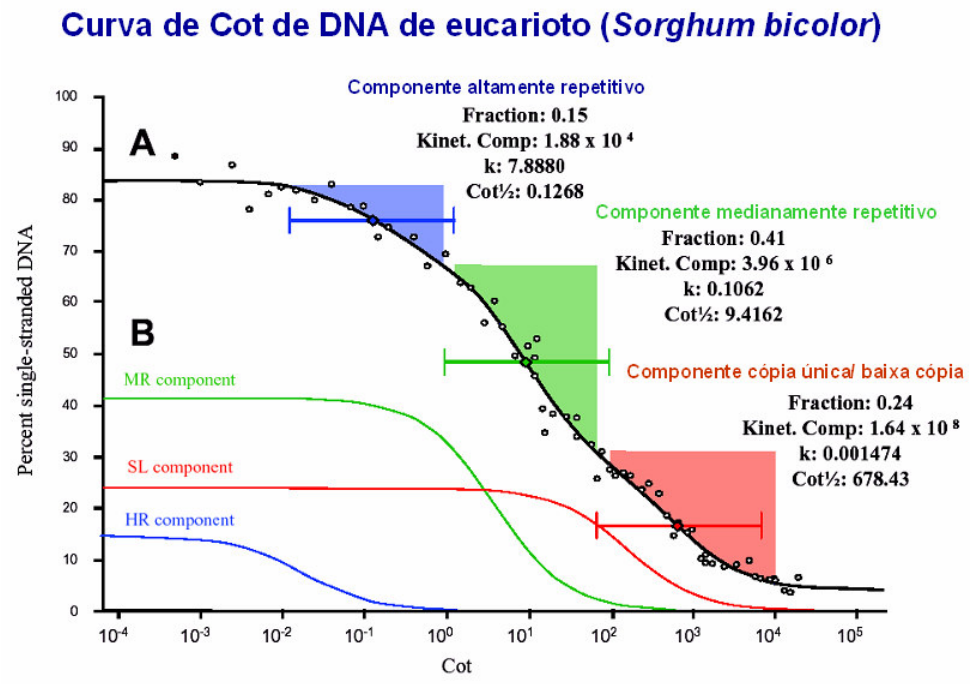

Figura 3. Exemplo de uma curva de *Cot* de eucarioto

Esta técnica de Cot auxiliou os geneticistas em uma série de descobertas do genoma, como, por exemplo, o seu tamanho, quantidade de DNA repetitivo, frações de seqüências simples, complexidade, entre outras já citadas do decorrer deste projeto, além de auxiliar os cientistas a compreenderem a evolução dos genomas de espécies próximas.

Enfim, o uso de um programa pra o cálculo dos componentes da curva de renaturação de DNA genômico de eucariotos é de extrema importância no meio da genética para auxiliar o profissional a entender cada vez mais estes genomas de maneira eficiente e automatizada.

2.3 A prática experimental em mapeamento genético

Esta seção explica como deve ser entendido o experimento [2] em bancada para a coleta dos dados usados como entrada no programa original de Pearson e também no proposto neste trabalho, será de grande interesse neste ponto da monografia para que fique claro a funcionalidade e contribuição dos estudos para cálculo dos componentes da curva de renaturação de DNA genômico de eucariotos.

Primeiramente, tem-se o objetivo de obter um DNA extremamente puro, sem contaminação de proteínas. Para isso extrai-se o DNA com fenol (o DNA é solúvel na presença de fenol, mas as proteínas não) e depois passa-se numa resina que retém os cátions. Tendo-se DNA purificado, este é cortado em pedaços de aproximadamente 500 pb[1].

Como o valor de Cot é a concentração versus tempo, pode-se ter o mesmo Cot com tempos diferentes, se variada a concentração inicial de DNA (Co). Assim Cot=1000 pode ser atingido com Co=10 e tempo=100 ou Co=100 e tempo=10, por exemplo. Por isso é importante saber a concentração exata de DNA que está sendo usanda.

Os fragmentos numa concentração conhecida e numa solução chamada SSC (standard saline citrate) ou outra qualquer (NaCl 0,5M por exemplo) são colocados num capilar de vidro (0,03 ml) cujas pontas são fundidas a fogo. Esses capilares (com os 0,03 ml de solução de DNA dentro) são então fervidos em água por 5 min. Isso é suficiente para desnaturar completamente os fragmentos de DNA que está presente na solução.

Esses capilares são então colocados num banho-maria de precisão (erro de + ou - 0,1ºC) ajustado para uma temperatura 25ºC abaixo do Tm[2]. Em tempos calculados para os valores de Cot desejados, vai-se tirando os capilares e jogando uma mistura de gelo seco e etanol (que está aproximadamente a -50ºC). Com isso a reação de renaturação estaciona imediatamente. Tudo que renaturou fica como dsDNA[3] e o que não renaturou permanece como fita simples.Cada um dos capilares, marcados para saber em que tempo foram retirados do banho, é quebrado e o conteúdo diluído (aproximadamente 200 vêzes) em uma solução tampão. Essa solução diluída é então digerida com uma enzima (*e.g.* DNAse S1) que reconhece só DNA fita simples. Destrói todo o DNA que não renaturou, sobrando apenas o DNA fita dupla. Enfim, dosa-se esse DNA fita dupla que sobrou com radioatividade, absorbancia a 260nm[4] ou fluorescência[5].

Os pontos obtidos são então plotados num gráfico em função do valor de Cot. Um grande problema observados neste exprimento é a falta de precisão quanto a manipulação do DNA podendo chegar o erro a 2 ou 3%.

Assim, para que a curva seja uma representação da realidade, os pontos experimentais devem ser repetidos para que a dispersão dos valores seja a menor possível. Existe um erro experimental na determinação da porcentagem de ssDNA (C/Co) [22][23]. Esta veriação se deve à variações da quantidade de DNA, por erros de pipetagem no momento da análise, por exemplo, e ao método de dosagem por flurescência. Esse erro não deve exceder os 10% e para isso devem

---

1 Pares de bases (pb) são unidades que caracterizam o comprimento de um fragmento de ácido nucléico de dupla fita. As duas fitas são mantidas por pontes de hidrogênio que unem uma purina a uma pirimidina. A complementaridade se faz pelo pareamento de G com C e de A com T no DNA (A com U no RNA).

2 A temperatura que correspondente ao ponto médio da transição é chama da Temperatura de fusão (Tm) e corresponde à situação em que a metade das moléculas da preparação está no estado nativo e a outra metade está desnaturada.

3 Fita dupla de DNA (dsDNA): polímero de nucleotídeos pareados dois, a dois formando a hélice de DNA

4 nm é abreviação de nanometro, que vale 10-9 metros

5 A Fluorescência é uma importante técnica analítica em que moléculas são excitadas por absorção de uma radiação eletromagnética

ser feitas pelo menos três determinações para cada ponto de modo a se calcular o erro ou desvio padrão. Para se evitar estes erros deve-se aumentar o número de pontos nos pontos de *Cot's* mais críticos, normalmente representado pelo região mais linear da curva.

É nesta última fase da prática experimental que os programas de análise de genomas baseado na cinética de reassociação de DNA são de grande importância e colaboração para os estudos citados pois eles recebem a média dos valores finais determinados para cada ponto e constróem as componentes de *Cot* correspondentes ao genoma

Capítulo 3

Aspectos Computacionais relevantes

O programa proposto nesta monografia foi desenvido na linguagem Java de programação, baseado no original proposto por William Pearson [12] escrito em linguagem Fortran na década de 70. Neste capítulo serão mencionadas algumas das justificativas da escolha da linguagem Java para realizar este projeto ao invés de se buscar melhorá-lo na linguagem de alto nível original.

3.1 Linguagem Java

O programa proposto neste trabalho de conclusão de curso foi desenvolvido na linguagem de programação Java, criado pela Sun Microsystems [15].

Esta linguagem foi escolhida, dentre outras, por possuir uma gama de características positivas para às necessidades identificadas no usuário: necessidade de execução em ambiente de rede – caso de aulas em laboratório e melhor apresentação gráfica (incluindo melhor interatividade com o usuário). Do ponto de vista técnico, podemos elicitar outras vantagens, igualmente importantes: simplicidade, orientação a objeto, robustez, segurança, de arquitetura neutra, portabilidade, entre outras características.

Primeiramente, a linguagem Java é simples porque é pequena (kernel) e de fácil entendimento; o seu interpretador ocupa aproximadamente 40kb de RAM, excluindo-se o suporte a multitarefa e as bibliotecas padrão, que ocupam pouco mais 175kb de espaço. E mesmo combinando a memória requerida com todos esses elementos, seu tamanho ainda permanece insignificante, se comparado a outras linguagens e ambientes de programação. Assim, um dos grandes objetivos da linguagem Java é habilitar a construção de softwares que possam rodar sozinhos em máquinas de pequeno porte. O que é de grande importância para o foco deste trabalho que é direcionado inicialmente aos pesquisadores e salas de aulas da área de biologia que, normalmente não possuem equipamentos de alta performance nos estabelecimentos de ensino.

Uma segunda característica positiva da linguagem Java é o seu paradigma orientado a objeto. As linguagens orientadas a objetos oferecem muitas vantagens sobre as linguagens procedurais tradicionais, como o Fortran. Este paradigma visa o encapsulamento de objetos e funções relacionados em unidades coesas, tornando-se fácil a localização de dependências de dados e realização de atividades de manutenção.

Sabe-se porém, que em suas versões mais modernas, a linguagem Fortran já pode suportar muitos conceitos de programação orientada a objetos como por exemplo o encapsulamento de dados [32], embora ainda não seja considerada uma linguagem orientada a objetos.

Uma outra propriedade favorável de Java é a sua reusabilidade, o que possibilita o programador reusar códigos e bibliotecas em seus projetos, diminuindo seu esforço e tempo na programação. Isto será útil para o desenvolvimento de novas versões deste projeto, ou utilização deste por outros desenvolvedores de softwares.

A linguagem Java também possui sua verificação de tipos de dados realizada em tempo de compilação, e não em tempo de execução, evitando muitos erros e condições aleatórias nos aplicativos. Além disso, a linguagem Java exige declarações explícitas de métodos, o que aumenta a confiabilidade de seus aplicativos.

Uma das características mais marcantes da linguagem Java é a sua independência de plataforma. O Java consegue essa independência pelo fato de seu compilador não gerar instruções específicas a uma plataforma, mas sim a um programa em um código intermediário, denominado bytecode, que pode ser descrito como uma linguagem de máquina destinada a um processador virtual que não existe fisicamente. Assim, o código Java compilado pode então ser executado por um interpretador de bytecodes, a JVM - Java Virtual Machine, que é um emulador de processador. Esta portabilidade da linguagem Java trouxe ao projeto uma grande contribuição, visto que, um dos problemas encontrados até os dias de hoje com os softwares voltados ao estudo de cinética de reassociação, foi a limitação de programas suportados apenas pela plataforma Unix e não em rede, como exemplo do programa de Pearson [2].

A linguagem de programação Java fornece a este trabalho uma facilidade de implementação, além de todos os aspectos citados anteriormente e a sua facilidade para a programação distribuída que será foco para trabalhos futuros, podendo assim auxiliar o desenvolvimento deste no intuito de construir um programa portável, facilmente implantado, e principalmente possuir ferramentas de suporte gráfico que auxiliem este projeto de conclusão de curso.

Atualmente, a linguagem Fortran já possui suporte gráfico com ferramentas visuais como exemplo o Visual Fortran [33], porém para a finalidade deste projeto (exposição do sistema em aulas), as considerações acimas colocadas com relação às vantagens vistas na linguagem Java prevaleceram na escolha desta linguagem para o desenvolvimento da ferramente proposta neste trabalho de conclusão de curso.

3.2 Fortran x Java visando trabalhos futuros

O programa proposto neste trabalho de conclusão de curso para o estudo de análise de genomas voltado para cinética de reassociação de DNA será baseado, como dito anteriormente, em um programa escrito na linguagem Fortran desenvolvido na década de 70 por William Pearson [12].

Como o próprio nome sugere – Fortran [30][31] – que vem do termo inglês *formula translation*, é uma linguagem de programação que permite a tradução quase que direta, através de simbologias de variáveis e operadores algébricos, de fórmulas matemáticas.

Desta maneira, a linguagem Fortran é, na maioria das vezes, aplicada aos problemas que possam ser formulados matematicamente, em particular nos campos da física, da engenharia, da estatística, da biologia e da própria matemática. Porém, apesar de Fortran ser uma linguagem ainda muito indicada para expressar algoritmos matemáticos como funções, outras linguagens estruturadas de programação [26][27], como exemplo a linguagem Java – utilizada no desenvolvimento do software proposto nesta monografia – conseguem fazer esta representação de forma tão adequada quanto a Fortran, além de incluir as vantagens mencionadas na seção anterior.

O avanço da tecnologia e a globalização fazem cada vez mais necessária uma forma de comunicação que seja democrática e de fácil acesso. Então, para corresponder a todas estas expectativas, a linguagem Java tem sido uma boa solução para esse problema por ter facilidades para desenvolvimento de sistemas distribuídos e também utilizar uma metodologia mais parecida com o mundo real que é a orientação a objetos.

Graças às facilidades para com a programação paralela e distribuída, conexão com a Internet e a sua portabilidade, a linguagem Java de programação vem conquistando uma crescente popularidade entre os programadores, engenheiros de software e usuários de aplicativos de rede [16], fazendo dessa linguagem uma excelente alternativa para o desenvolvimento dessa classe de sistemas. Além do suporte ao processamento distribuído, ainda existem ambientes que estendem as bibliotecas padrão facilitando e melhorando a programação distribuída.

Muitas razões foram consideradas para que Java fosse escolhida para o desenvolvimento do software deste projeto. Além de esta ser uma linguagem moderna, totalmente orientada a objetos e independente de plataforma, ela possui suporte à programação paralela e distribuída que será de fundamental importância na realização de trabalhos futuros.

De todas as linguagens que se tornaram populares atualmente, Java foi a primeira a ser projetada para interligar redes de computadores. Ela incorpora uma série de benefícios para o desenvolvimento de aplicações distribuídas como, por exemplo, independência de plataforma, segurança, possuir programas executados localmente, ter uma capacidade de atualização freqüente e de fácil expansão. Desta maneira, a rede torna-se o veículo de distribuição de suas aplicações.

Os aplicativos em Java podem se comunicar através da Internet apesar de estarem fisicamente espalhados pelo mundo e executando em arquiteturas de hardware e sistemas operacionais diversos. Esta deve-se, principalmente, pela sua independência de plataforma, como dito anteriormente. Assim, o código fonte é compilado em um bytecode, que passa a ser interpretado por uma implementação qualquer de JVM (Java Virtual Machine).

Além das vantagens ditas citadas acima, ela tem sido uma boa escolha quando se trata de desenvolvimento de sistemas distribuídos na Internet. Isto se deve à facilidade de criação e sincronização de threads, suporte à comunicação em rede, segurança e robustez.

No entanto, essas facilidades repercutem de forma negativa quanto ao seu desempenho quando comparadas a abordagens tradicionais como em Fortran e outras linguagens. Mesmo assim as desvantagem citadas tem sido amenizadas cada vez mais através da evolução dos compiladores e das máquinas virtuais java, que compilam bytecodes para formato binário em tempo de execução, otimizando o código a medida que o mesmo é executado. Hoje, já existem diversos projetos de pesquisa que buscam alternativas para aumentar a eficiência de Java em aplicações de alto desempenho, tendo os mesmos obtidos resultados promissores.

Desta maneira, com o poder de distribuição de aplicações do Java, o programa proposto nesta monografia poderá estar sendo disponibilizado futuramente em sites na web, atendendo possivelmente algumas centenas de pessoas instantaneamente e podendo realizar, por exemplo, a migração de dados de qualquer lugar do planeta para um sistema único. Isto eliminará a necessidade de instalação por parte dos usuários finais do sistema e, por fim diminuirá os investimentos com relação à distribuição do mesmo.

3.3 Computação de Regressão de Curvas

Para a construção da curva de Cot do exemplo proposto nesta monografia, teve-se que, com base em conjuntos de pares de entradas de dados – valores de *Cot* e *FractSS* – obter uma curva que melhor representasse esses pontos através de um método comum de ajuste de equações a estes dados experimentais.

Os dados cinéticos obtidos em laboratório ou em qualquer outra forma de estudo são apresentados na forma de uma tabela de valores "x" e "y" ou na forma de um gráfico no qual uma grandeza experimental medida dependente "y" é graficada contra uma variável independente "x" de acordo com uma função y = f(x).

Além de ser uma maneira mais econômica de apresentar os dados, o gráfico é também mais efetivo por proporcionar ao experimentador, uma visão mais intuitiva de como as variáveis do experimento estão relacionadas [19]. No caso das Curvas de Cot, por exemplo, o analista poderá verificar os pontos de desnaturação e renaturação do DNA, apenas analisando onde os eixos se cortam com relação a sua reta correlacionada além de outras informações também observadas no mesmo. O fato de existir um formalismo matemático para o ajuste das curvas, permite que as mesmas sejam calculadas por computador – acelerando e muito o processo experimental. Uma vez que os dados gerados são plotados em um gráfico, pode-se traçar uma curva suave que se aproxime ou passe pelos pontos que eles formam.

Ao fazer isso, assume-se que a grandeza física está variando suavemente ou continuamente. Traçando esta curva suave que passa pelos pontos permite-se distinguir entre a suposta variação suave da grandeza física e as supostas flutuações induzidas pelo erro experimental ou ruído. Também permite-se fazer uma interpolação ou extrapolação dos dados, isto é, prever valores de "x" ou "y" que não são parte dos dados medidos para que se construa uma curva contínua. Enfim, como dito acima, o gráfico possibilita a obtenção de características dos dados, como inclinações e integrais que podem possuir significado físico ou físico-químico da substância estudada.

O processo de encontrar uma curva suave que represente os dados físicos relacionadas a esta, é chamado de ajuste de curvas ou regressão como mostrado na Figura 4. A regressão é o processo de aprender uma função que mapeia um item de dados para uma variável de predição real estimada [28].

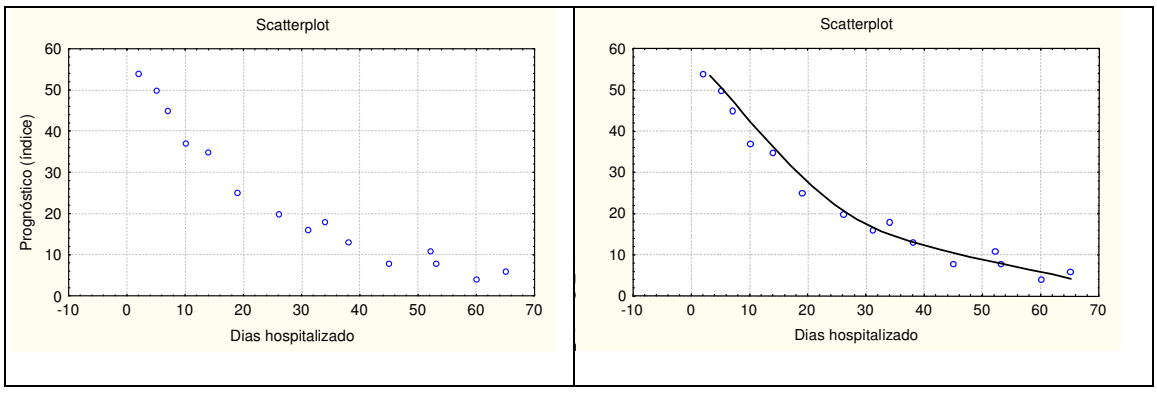

Figura 4. Exemplo de um ajuste de curva não-linear

Este também pode ser feito numericamente. Os métodos matemáticos para fazer isso já são conhecidos há séculos, mas apenas nas últimas décadas, com o desenvolvimento de calculadoras e computadores é que eles se tornaram práticos para uso rotineiro. Comparado com a análise visual do gráfico, o ajuste numérico de curvas têm a vantagem da exatidão e da imparcialidade cabendo ao usuário apenas a escolha da função a ser utilizada.

O ajuste de curva mais comum é o ajuste a uma linha reta. A análise neste caso é simples e direta e os parâmetros encontrados como inclinação e interceptações são facilmente calculados.

Existem várias formas de regressão: linear, múltipla, não-linear e os modelos lineares generalizados, por exemplo.

Na regressão linear, uma variável a ser predita é baseada em apenas uma outra variável numa relação de pares (x,y). Nestes modelos, o problema de estimação dos parâmetros resolve-se com um sistema de equações lineares com relação aos coeficientes de regressão desconhecidos. Existe uma solução única e, portanto, obtém-se uma forma analítica de estimação dos parâmetros

Para o caso das curvas serem lineares, usa-se, em sua maioria, o método de ajuste de curvas de regressão linear, que consiste em encontrar os coeficientes da função que minimizem a soma dos quadrados dos desvios padrão.

Em muitos casos porém, os dados obtidos não são lineares, como por exemplo a construção das curvas de Cot e variação de absorbância em função do tempo na reação de hidrólise do AAS. Porém, em alguns casos, os dados podem ser transformados para que se ajustem a um modelo linear. Os mesmos critérios acima citados para ajuste de curvas lineares podem ser usados quando o modelo é não-linear. Porém, neste caso, tem-se que escolher a função a ser utilizada para a regresão. Existem vários métodos de ajuste de curvas não-lineares: método dos Mínimos Quadrados, método de Gauss-Newton e o método de Levemberg Marquardt, por exemplo. Este último foi escolhido para o ajuste das curvas de *Cot* do programa de Pearson e do proposto nete trabalho de monografia.

O algoritmo de Levenberg Marquardt é uma técnica interativa que localiza o mínimo de uma função que expressa pela soma dos quadrados de funções não-lineares [9][10].

Este algoritmo se tranformou em uma técnica padrão para com problemas não-lineares sendo uma extensão do algoritmo de Gauss Newton, citado anteriormente. Desta forma, quando sua solução está distante da solução correta, o seu algoritmo é retardado garantindo a convergência da curva. Já quando sua solução se aproxima da correta, o mesmo se tranforma em um método de Gauss.

Capítulo 4

Desenvolvimento da Aplicação

4.1 Abordagem geral do projeto

Inicialmente, como o projeto possui natureza interdisciplinar, identificou-se a necessidade de um contato maior com profissionais da área de Biologia.

Após o contato inicial, deu-se continuidade aos estudos preliminares com a coleta de informações bibliográficas orientadas por pesquisadores da biologia genética. Este estudo preliminar foi de grande importância, pois ele viabilizou um entendimento claro do problema e a aquização dos conhecimentos necessários para a elaboração de soluções para o problema proposto neste trabalho de conclusão de curso. O projeto contou também com uma parceria forte com a Universidade de São Paulo – Prof. Carlos Winter do Departamento do Instituto de Ciências Biomédicas do departamento de Parasitologia Da Universidade de São Paulo - USP, que não somente orientou os aspectos de Genética deste trabalho bem como forneceu subsídios (dados e indicações bibliográficas) necessários para o desenvolvimento e embasamento teórico do projeto.

Após o estudo dos aspectos de Biologia do problema, iniciou-se uma outra fase também de estudos, na época: o estudo das técnicas matemáticas usadas para produzir a curva de cinética de reassociação de DNA [20][21]. Estas se baseiam em um algoritmo para regressão não linear e que calculam as componentes que correspondem às famílias de DNA repetitivo e de cópia única, vistas anteriormente como características próprias dos genomas.

Apesar de algoritmos já conhecidos, estes foram desenvolvidos no início dos anos setenta do século passado e já podem se encontrar ultrapassados do ponto de vista computacional. Assim, vê-se a grande necessidade da revisão destes algoritmos antigos e até da elaboração de algoritmos atualizados mais próximos dos modelos propostos para explicação da cinética de reassociação abordada nos dias de hoje, se necessário.

Desta maneira, será feito um estudo minucioso dos programas existentes, principalmente o desenvolvido pelo professor W.T. Pearson do Departamento de Bioquímica da Escola de Medicina da Universidade da Virgínia [21]. Este programa desenvolvido por W.T Pearson, possui uma série de limitações como: (i) a restrição do cálculo de função de segunda ordem do DNA e (ii) falhas de execução.

Além disso, uma grande dificuldade encontrada pelos biólogos que manuseiam estes programas feitos para a análise de cinética de reassociação de DNA, é a interface não amigável dos mesmos, visto que estes foram elaborados há bastante tempo atrás e em linguagem Fortran que não permitia a construção de interfaces de fácil usabilidade na época. Também, observa-se

que, a maioria dos softwares desenvolvidos, são voltados para fins acadêmicos, necessitando muitas vezes de um bom conhecimento sobre computador e sistema operacional por parte do pesquisador que o manuseia, pois, em geral, são feitos em sistemas Unix ou Linux [1]. Esta plataforma normalmente é escolhida por permitir atualizações constantes e aumentar sua capacidade de processamento, por serem de acesso livre ao proprietário do sistema.

4.2 Metodologia de Desenvolvimento

O software proposto nesta monografia não será desenvolvido de forma convencional baseado apenas em informações adquiridas através da necessidade do usuário para a identificação dos requisitos do sistema. Isto por que ele será construído, baseado em um software pré-existente, desenvolvido na década de 70 por Willian Pearson [21], como mencionado anteriormente. Assim, partir-se-á do programa original, (i) escrevendo o mesmo em uma nova linguagem – Java, (ii) construindo uma interface gráfica amigável, (iii) acrescentando novas funcionalidades e (iv) corrigindo os erros identificados pelo usuário.

Este programa existente servirá então como um protótipo para a construção do novo software. Este protótipo será usado para derivação dos requisitos do sistema, sendo um caso de prototipação descartável onde, após o desenvolvimento do projeto, este será descartado.

Na prototipação descartável utiliza-se o protótipo para retirar ou obter informações do sistema usado como modelo ou para validá-lo. Apesar do protótipo ser eliminado após o desenvolvimento do programa, este pode ser utilizado nas diversas etapas do ciclo de vida do sistema.

> "O objetivo principal desse modelo é entender os requisitos do sistema. Tem sido usado com sucesso para validar partes do sistema (Interface Gráfica e aspectos do sistema relacionados à arquitetura - ex: performance, portabilidade, etc.). Como na programação exploratória, a primeira etapa prevê o desenvolvimento de um programa (protótipo) para o usuário experimentar. No entanto, ao contrário da programação exploratória, o protótipo é então descartado e o software deve ser re-implantado na etapa seguinte, usando qualquer modelo de ciclo de vida (ex: cascata)" [18]

Antes da construção propriamente dita do protótipo, foi-se necessário a coleta e recolhimento de informações necessárias para e embasamento teórico do problema a ser solucionado com o desenvolvimento do software – cálculo dos componentes da curva de renaturação de DNA genômico de eucariotos.

Após estes estudos preliminares, construiu-se um protótipo baseado no sistema existente de Pearson [12]. Para a construção deste protótipo duas etapas foram de fundamental importância: a análise do sistema existente e a construção de um diagrama baseado no código-fonte do mesmo chamada de projeto Figura 5.

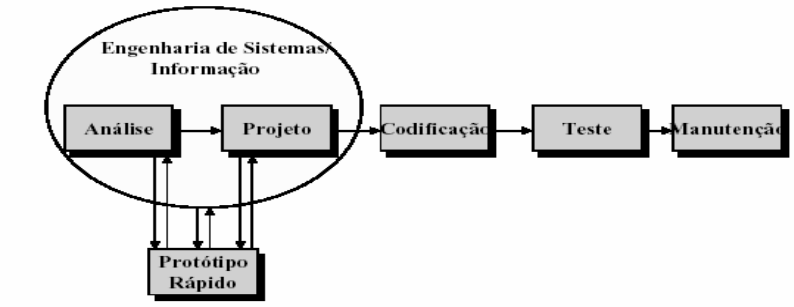

Figura 5. Modelo de Prototipação rápida do sistema

A união dessas duas fazes formaram o que se chamou de prototipação rápida para este sistema. Assim, baseado em informações obtidas através da análise do código fonte do programa de Pearson, pôde-se elaborar para cada rotina um protótipo descrevendo as suas etapas de execução. Esses protótipos serviram de base para a coleta de requisitos do sistema, já que este partiu de um programa já pronto, como citado anteriomente.

4.3 Análise do Sistema existente

O sistema criado por Pearson foi desenvolvido para plataforma Unix. A Figura 6 é um 'snapshot' da tela de interface atual do sistema original, como pode ser visto, o sistema possui uma interface pouco amigável com o usuário. Isto acontece, pois além de ser manipulado através de linhas de comando, este não possibilita o usuário voltar etapas após a inicialização do programa. Do ponto de vista de usabilidade (e estética), o programa não possui fácil interatividade com o usuário fazendo com que o trabalho fique demasiadamente exaustivo.

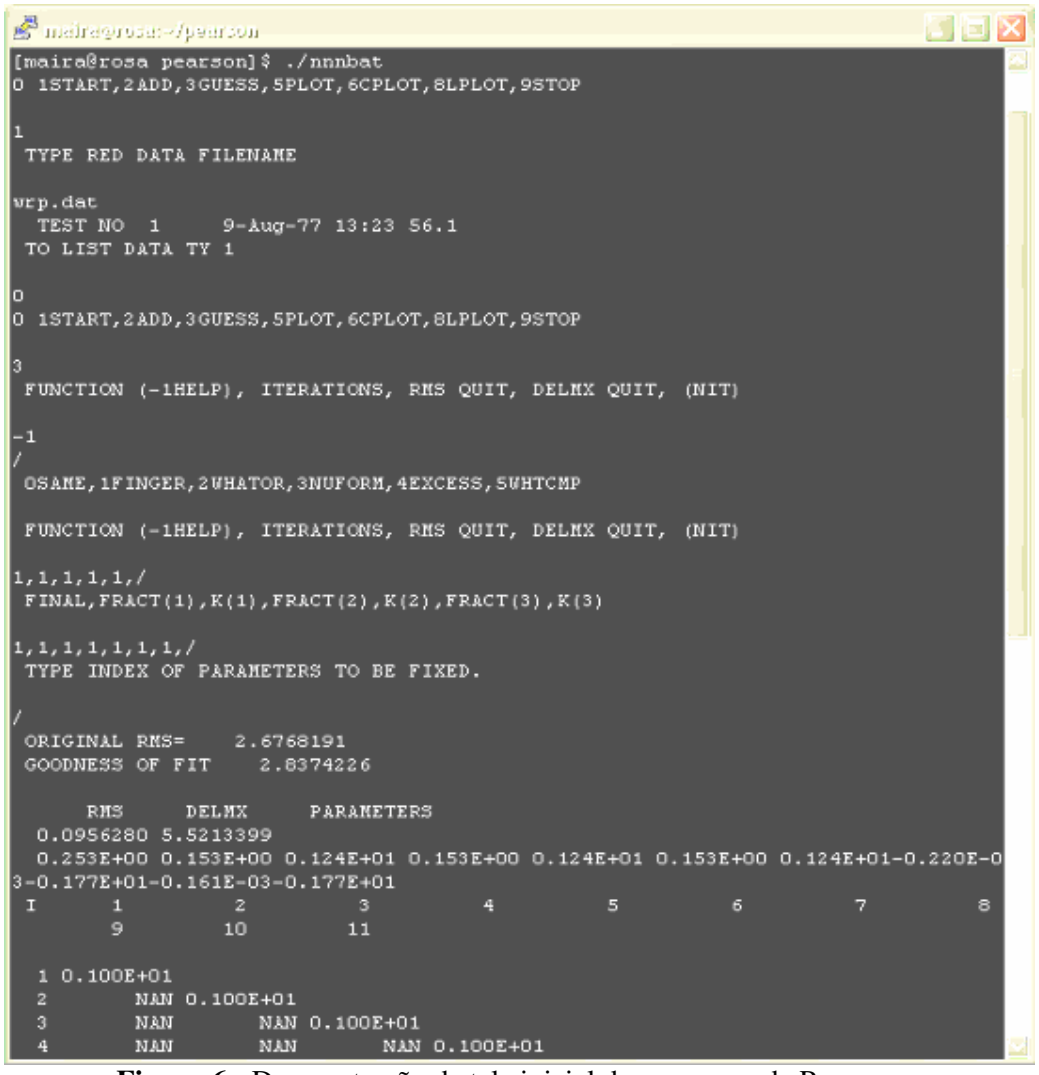

Figura 6. Demonstração da tela inicial do programa de Pearson

Para auxiliar o estudo e construção do protótipo, a análise de funcionamento da interface foi de crucial importância, visto que para se conhecer um sistema apenas pelo seu código-fonte,

este se tornaria um trabalho muito tedioso e lento. Assim, foram feitos diversos testes dentro do programa verificando-se quase todas as possibilidades de manipulação de dados dentro do mesmo. Usou-se, primeiramente, o arquivo que o programa disponibiliza em seu tutorial para a execução de testes Figura 7.

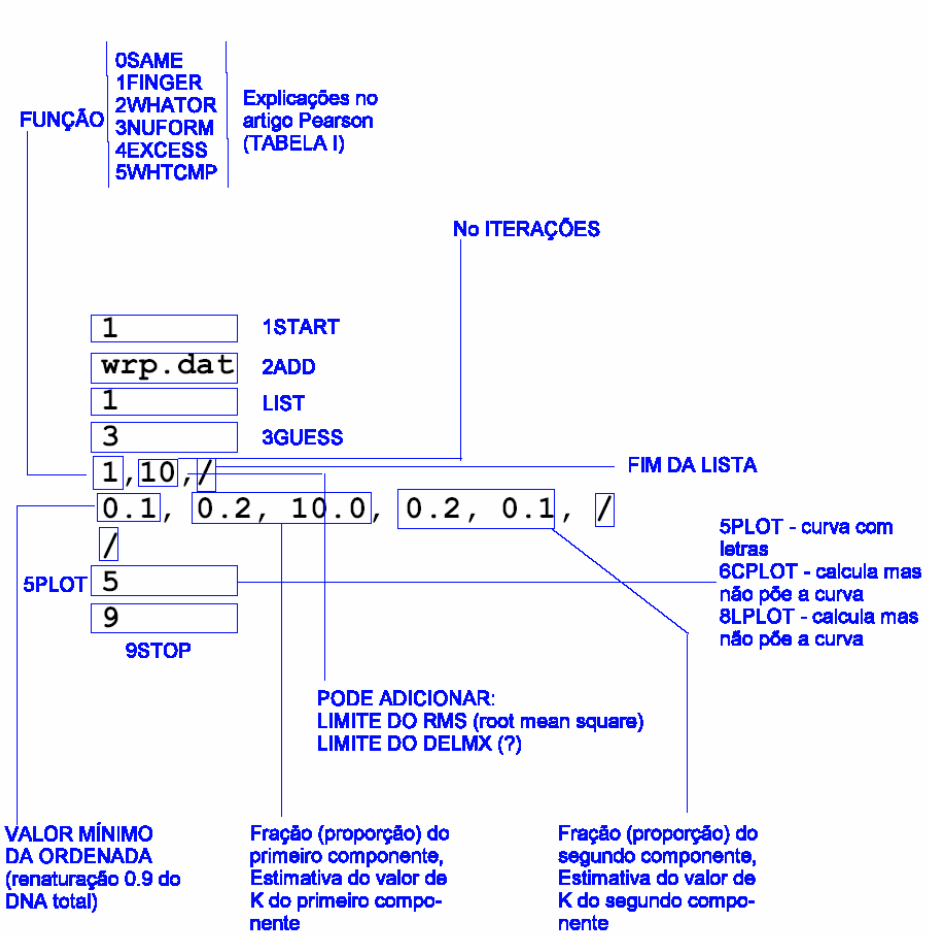

Figura 7. Exemplo auto-explicativo disponilibizado na breve documentação do programa de Pearson

O arquivo teste possui uma seqüência de comandos que foram executados separadamente para a verificação dos resultados obtidos. Como pode ser observado na Figura 6, quando o programa é executado, este disponibiliza ao usuário sete opções de escolha de comandos – 1START, 2ADD, 3GUESS, 5PLOT, 6CPLOT, 8LPLOT, 9STOP.

A primeira das opções – 1START – inicializa o sistema e recebe a entrada dos dados através de um arquivo existente. Este arquivo poder ser gerado por um programa chamado *cotfil* que vem no pacote do programa. O arquivo possui duas colunas: a primeira se refere à fração de DNA fita simples e a segunda ao valor de cot correspondente. Um dos problemas observados no programa auxilar de construção dos dados de entrada do programa de Pearson – *cotfil* – foi o exaustivo trabalho para inserção dos dados (pares de dados DNA fita simples e *Cot* correspondente) que devem ser inseridos um a um sem que haja nenhum erro. Sendo este um dos pontos de melhoria identificados. Caso ocorra um erro, da forma como o programa original

funciona, isto ocasionará o retorno ao início para a nova inclusão dos dados desde o primeiro par de entrada de dados. A Figura 8 mostra um exemplo de um arquivo de dados de entrada.

```
TEST NO  1      9-Aug-77 13:23 56.1
 0.77413 0.10000E-03
 0.82419 0.11498E-03
 0.78084 0.13219E-03
 0.80756 0.15199E-03
 0.76039 0.17475E-03
 0.75722 0.20092E-03
 0.78436 0.23101E-03
 0.83207 0.26561E-03
 0.75789 0.30539E-03
 0.79876 0.35112E-03
 0.78368 0.40370E-03
 ... //corte
 0.21051 0.57224E+02
 0.22066 0.65793E+02
 0.20952 0.75646E+02
 0.20544 0.86975E+02
 0.20038 0.10000E+03
 3.00000
```

Figura 8. Trecho do arquivo de entrada de dados do programa de Pearson

A funcionalidade do programa de Pearson é relativamente simples, porém para usá-lo é necessário o conhecimento de algumas convenções específicas do programa – o que ainda mais torna sua operação passível de erros. Para isso deve-se primeiro ler o seu arquivo onde se tem a explicação do programa. Neste arquivo alguns termos só são de fácil entendimento para profissionais da área de estudos de genomas vegetais. O mesmo ciclo ocorre para a opção 2ADD, que difere apenas em não inicializar o arquivo, concatenando o novo arquivo usado na execução anterior.

As opção – 3GUESS – fornece a opção de escolha da funções de regressão desejada. O programa de Pearson disponibiliza cinco funções diferentes para cálculo dos resultados do programa - FINGER , WHATOR, NOFORM, EXCESS e WHATCMP. Cada uns desses métodos se refere a uma condição experimental na dosagem do DNA dupla fita ou fita simples restante. Assim, para cada tipo de reação (DNA-DNA; DNA-RNA; RNA-DNA), escolhe-se uma equação para ajuste de curvas como explicado na Figura 9. Dependendo das condições de reação do DNA ou de seleção de DNA dupla fita (enzima S1 ou Hidroxiapatita), usa-se cinéticas diferentes. Assim, quando não se sabe qual função se adequa melhor ao cálculo das amostras, pode usar diferentes equações para o seu cálculo e compara-las no fim.

```
FINGER => mede o DNA dupla fita por afinidade a hidroxiapatita. Conta a
parte que sobra de fita simples na fita renaturada, mas é de segunda
ordem aparente.
WHATOR => não sabe a ordem e pode ajustar isso variando n.
NUFORM => usa um DNA marcado radioativamente (tracer) em concentração
baixa e DNA "frio" (não marcado)(driver) que dirige a reação. Dosa com
nuclease S1.
EXCESS => usa vasto excesso de RNA e DNA fita dupla desnaturado. O RNA
fita simples) dirige a reação e assim só existe uma fita. Logo, a
cinética é de primeira ordem.
WHTCMP => segunda ordem modificada, ou pseudo-segunda ordem. a parte que
sobra de fita simples na fita renaturada do DNA são eliminados pela
nuclease S1. O expoente (no caso -0,44) pode ser variado. A nuclease S1
nem sempre hidroliza só DNA fita simples. As vezes hidroliza DNA dupla
fita. Isso afeta a cinética. Esse ajuste é empírico, ou seja não é
baseado em dados teóricos
```

Figura 9. Explicação das funções de cálculo usadas no programa de Pearson

As opções 5, 6 e 8, imprimem o resultado obtido com o cálculo dos componentes da curva de renaturação de DNA esperado como mostra a Figura 10.

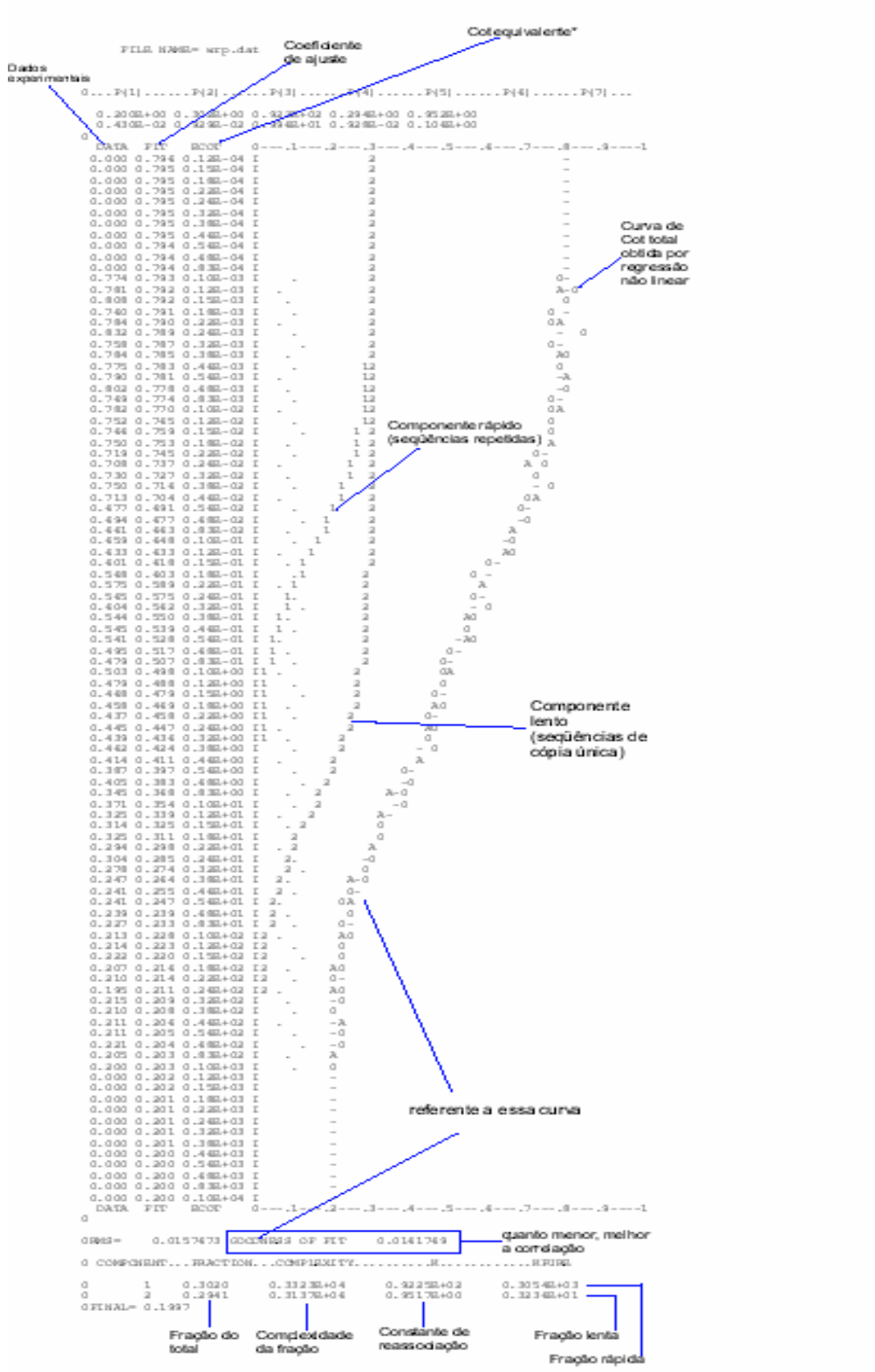

Figura 10. Arquivo de saída do programa de Pearson

Na Figura 10, a primeira linha do arquivo de saída – P(1), P(2)...P(7) – refere-se aos parâmetros relacionados à função escolhida no início do programa. Logo abaixo estão impressas duas linhas: a primeira corresponde ao valor atingido para cada parâmetro depois de *x* iterações antes defininas e a segunda ao erro da determinação. A coluna "data" refere-se aos dados dos valores de fração de DNA fita dupla da curva e a coluna "fit", ao coeficiente de ajuste relacionado.

O gráfico tem como abscissa os valores de *Cot* equivalente variado de acordo com o tamanho do genoma que são calculados pela multiplicação da concentração de DNA (em moles/L) versus tempo (em segundos). Estes valores são retirados do arquivo dado como entrada do programa mostrado na Figura 8. A ordenada corresponde à fração de DNA fita simples que permanece após cada *Cot* e vai de 0 a 1. Observa-se que o gráfico no arquivo de saída está invertido, ou seja, o eixo das abcissas está plotado na vertical, enquanto o eixo das ordenadas na horizontal, quando dereriam estar por convenção no sentido oposto ao citado. Enfim, o arquivo de saída fornece um valor de "goodness of fit" que mostra o coeficiente final do ajuste que deve ser o menor possível e serve como parâmetro no decorrer da análise dos dados no programa. O gráfico ainda dispõe valores da complexidade da fração, fração lenta, fração rápida e constante de reassociação que servem para a extração de informações quantitativas como número de seqüências, composição das seqüências, e outras informações adicionais, além das observadas pelos valores extrapolados na ordenada do gráfico.

Estas últimas informações servem como parâmetros para ajustar a curva até que ela se torne ideal. Não existe valores de parâmetros ideais fixos, pois esta análise é feita pelo pesquisador baeada em sua própria experiência profissional, no seu conhecimento sobre o genoma estudado e suas características. Quando se chega a valores "teoricamente" ajustados, analisa-se o gráfico verificando os pontos de declínio para coletar a amostra que representa as frações lentas ou rápidas, dependendo da análise.

Na Tabelas 1, 2 e 3, são mostrados alguns trechos do programa sendo executados em situações e momentos diferentes. O primeiro caso apenas inicializa o sistema com a primeira opção e exibe os dados do arquivo de dados de entrada. O segundo executa a opção de função escolhida com os dados de entrada para a realização dos cálculos desejados. Enfim, a terceira execução imprime na tela o resultado da curva de renaturação obtida com estes parâmetros de entrada. Vários testes foram executados, além dos relatados acima. Uma observação importante foi a interdependência de execução das opções do programa que não ficam claras para o usuário quando o mesmo manipula o programa.

No sistema proposto nesta monografia, teve-se a preocupação de guiar o usuário para que o mesmo não cometa nenhuma falha de seqüênciamento de operações. Isto evitará erros de execução e facilitará o uso do mesmo. Além disto o programa terá a preocupação de agilizar o manuseio do mesmo, tornando-o mais amigável e auto-explicativo.

Tabela 1 Trechos da execução – inicialização – do programa de Pearson

Passo 1:

```
root@voip2:~/pearson# ./nnnbat example.bat
0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP
1
 TYPE RED DATA FILENAME
wrp.dat
  TEST NO  1     9-Aug-77 13:23 56.1
1
      COT  FRACT SS
    0.100E-03  0.7741
    0.115E-03  0.8242
    0.132E-03  0.7808
    0.152E-03  0.8076...
```

Tabela 2 Trechos da execução – cálculo da curva – do programa de Pearson

Passo 1 e 3:

```
root@voip2:~/pearson# ./nnnbat example.bat
0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP

1
 TYPE RED DATA FILENAME

wrp.dat
  TEST NO  1     9-Aug-77 13:23 56.1
 TO LIST DATA TY 1

1
       COT  FRACT SS

    0.100E-03  0.7741
    0.115E-03  0.8242
    ...

0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP

3
 FUNCTION (-1HELP), ITERATIONS, RMS QUIT, DELMX QUIT, (NIT)

1,10,/
 FINAL,FRACT(1),K(1),FRACT(2),K(2),FRACT(3),K(3)

0.1, 0.2, 10.0, 0.2, 0.1, /
 TYPE INDEX OF PARAMETERS TO BE FIXED.

/
 ORIGINAL RMS=   43.2197685
 GOODNESS OF FIT   44.8168106

     RMS     DELMX     PARAMETERS
 13.2230082 3.9259841
-0.342E-01-0.230E+01 0.349E+03 0.359E+00-0.521E+00 0.892E+00-0.200E+01 0.202E+0
1 0.202E+01
 12.4430332**********
...
1 0.202E+01
I     1        2        3        4        5        6        7
  1 0.100E+01
  2-0.748E-01 0.100E+01
  3 0.660E-02-0.844E+00 0.100E+01
  4-0.344E+00-0.793E-02 0.278E-03 0.100E+01

 RMS=    0.5783510 GOODNESS OF FIT    0.5997220
     100 POINTS      93 DEG OF FREEDOM

 ...P(1).....P(2).....P(3).....P(4).....P(5).....P(6).....P(7)...

-0.653E+00-0.219E-01-0.256E+03-0.566E-01-0.525E+00-0.679E-01-0.123E+01 0.202E+0
1 0.202E+01
  0.697E-01 0.364E-01 0.178E+02 0.300E-01 0.180E-01 0.269E-01 0.375E-01 0.000E+0
0 0.000E+00
0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP
```

Tabela 3 Trechos da execução – impressão de resultados – do programa de Pearson

Passo 1/3/5

```
Obs: Copiado a partir do passo 5


0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP

5

obs: exibido apenas trechos do resultado
  DATA  FIT   ECOT    0---.1---.2---.3---.4---.5---.6---.7---.8---.9----1
 0.000 1.220 0.12E-04 I
 0.000 1.220 0.15E-04 I
 0.000 1.220 0.18E-04 I
 0.000 1.220 0.22E-04 I
 0.
.........
0.195-0.609 0.26E+02 I 4.      A
 0.215-0.617 0.32E+02 I 4 .
 0.210-0.623 0.38E+02 I4   .
 0.211-0.629 0.46E+02 I4 .      A
 0.211-0.633 0.56E+02 I4   .
 0.221-0.636 0.68E+02 I4    .
 0.205-0.639 0.83E+02 I4  .     A
 0.200-0.642 0.10E+03 I    .

  DATA  FIT   ECOT    0---.1---.2---.3---.4---.5---.6---.7---.8---.9----1
0

 RMS=    0.5783510 GOODNESS OF FIT    0.5997220

  COMPONENT...FRACTION...COMPLEXITY.....K.....KPURE

0       1   -0.0219     0.8699E+02    -0.2560E+03     0.1167E+05
0       2   -0.0566     0.1095E+06    -0.5247E+00     0.9271E+01
0       3   -0.0679     0.5621E+05    -0.1225E+01     0.1806E+02
0       4    2.0204     0.1015E+07     0.2020E+01     0.1000E+01
 FINAL=-0.6535
0 1START,2ADD,3GUESS,5PLOT,6CPLOT,8LPLOT,9STOP
```

A análise do sistema, neste caso, estabeleceu alguns dos requisitos necessários para que se pudesse coletar o maior número de informações como funções, desempenho exigido e interface gráfica esperada.

Enfim, esta etapa, juntamente com a construção dos diagramas baseados no código existente, auxiliará o desenvolvimento de um modelo de softawre que será implementado.

4.4 Diagrama de blocos funcionais do programa de Pearson

A segunda etapa, contida na elaboração do protótipo do sistema, consiste no estudo profundo do código-fonte do programa existente de Pearson. Esta etapa serviu para coletar a maioria das informações contidas no sistema.

Como o sistema existente foi desenvolvido numa linguagem pouco utilizada nos dias de hoje comparada à linguagens mais modernas, atualmente em uso, como a linguagem Java, precisou-se, além da semântica, de uma atenção especial com o estudo de sua sintaxe. Esta fase do estudo foi bastante facilitado, levando-se em conta que a linguagem Fortran (linguagem em que o programa de Pearson foi implementado) de programação é simples e de fácil entendimento.

Após este rápido estudo, iniciou-se a construção dos diagramas de blocos funcionais baseados no código fonte em Fortran. A construção deste diagrama possibilitou o entendimento profundo da lógica do programa como a seqüência de rotinas e principalmente o algoritmo utilizado para os cálculos propostos pelo sistema.

A construção do diagrama foi elaborada para todo o código do sistema de Pearson; a Figura 11 representa um trecho deste esboço onde o ciclo de acesso às opções 1, 2,4,7 e 9 é executado.

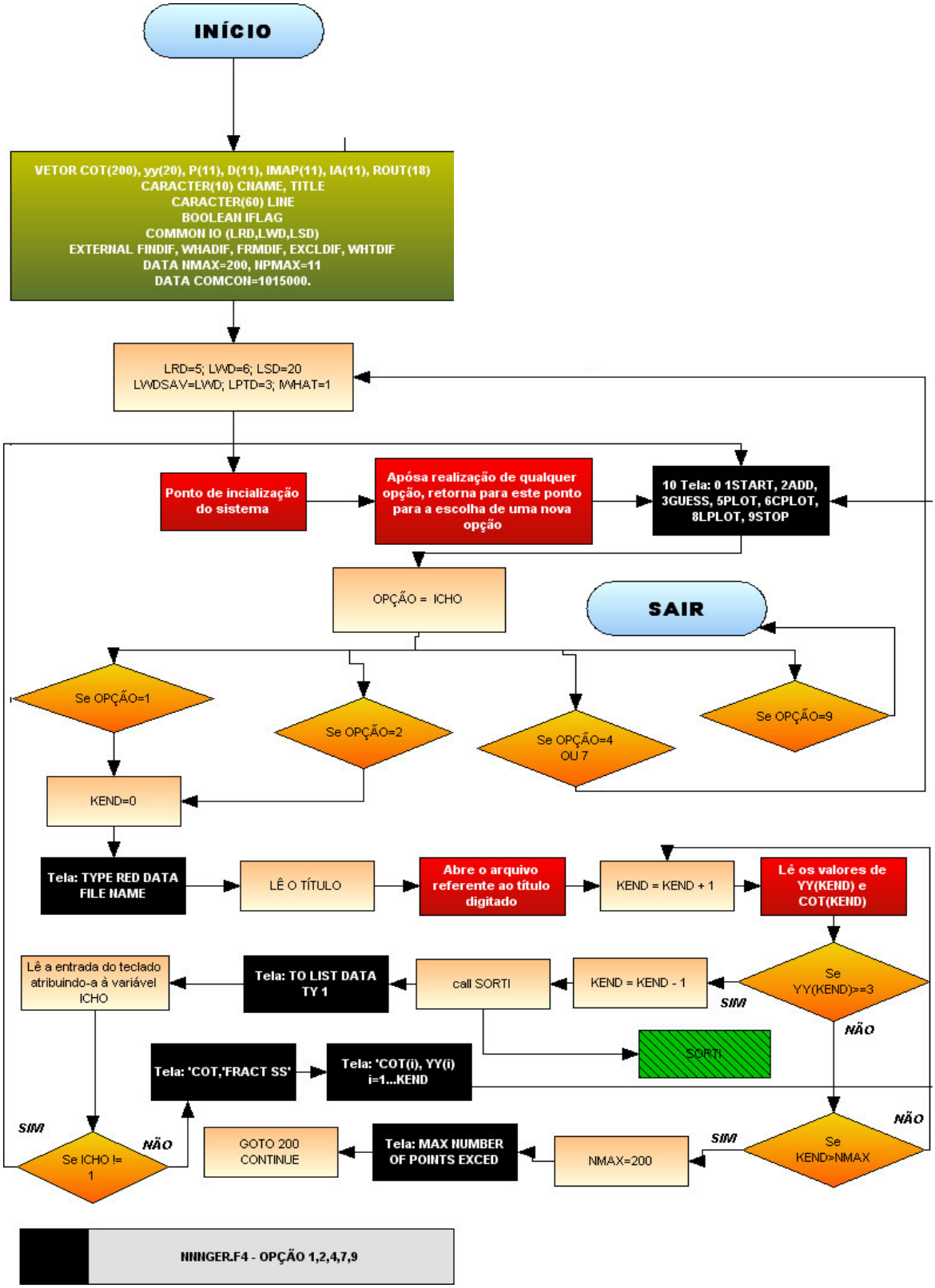

Figura 11. Diagrama de blocos para o programa de Pearson (opções de entrada das opções 1,2,4,7,9)

Neste diagrama, são chamadas algumas sub-rotinas do sistema que também foram esboçadas em forma de diagramas. O diagrama apresentado na Figura 11 representa o ciclo inicial do uso do programa de Pearson. O retângulos vermelhos representam comentários dentro do programa, os laranjas representam ações executadas pelo mesmo, os pretos são saídas de tela, os losangos laranjas representam operações condicionais dentro da rotina (exemplo: operação if) e os retângulos verdes hachurados representam sub-rotinas externas. A leitura do diagrama segue abaixo:

- Na lógica geral do programa, incialmente, são criadas as variáveis a serem usadas na aplicação. Depois, é exibida uma mensagem de escolha da operação a ser realizada pelo usuário. De acordo com a entrada no teclado pelo usuário, o programa se desvia para a rotina referente a esta função;
- Para as opções 1 e 2 – START e ADD – solicita-se o nome do arquivo de dados de entrada; todos estes valores são armazenados em arrays. Pearson, identificou o fim do arquivo por um número predefinido na contrução do mesmo – o número três. Por isso exite uma operação de comparação entre os valores de entrada e este número;
- Após o armazenamento dos dados de entrada em arrays, estes ainda passam pela sub-rotina “Sorti”(sic) que ajusta estes valores e os filtra para um outro array para que apenas os valores desejados sejam usados posteriormente para o cálculo da curva de cot;
- Feito isto, o usuário ainda tem a opção de listagem destes dados. Caso esta opção seja ignorada, o programa retorna ao passo 1 desta rotina;
- Ainda existe nesta rotina uma verificação do tamanho do arquivo de dados, que neste caso, pode receber apenas 200 entradas de valores. Para este passo, observou-se que para o programa proposto nesta monografia, poderia-se eliminar esta limitação, substituindo-se o uso de arrays fixos por arrays dinâmicos com o uso de vetores;
- Ainda para a operação condicional de escolha existem as opções 4,7 e 9. As primeiras apenas retornam ao passo 1 desta mesma rotina e a opção 9 é usada para sair do programa.

A partir da construção deste diagrama, que funcionou neste caso como um protótipo do sistema a ser desenvolvido, pôde-se então derivar os requisitos do sistema e seguir um processo de desenvolvimento convencional, que neste caso, foi escolhido o modelo em cascata incremental e iterativo.

4.5 Desenvolvimento do novo programa

Como afirmado na seção anterior e dadas as peculiaridades do sistema (usuário ignorante do modelo de computação a adotar e sistema atual com limitação de documentação), a elaboração dos requisitos do sistema seguiram um processo desenvolvido de forma interativo e incremental. Este modelo é uma interação de passos seguidos como no ciclo de vida clássico de desenvolvimento de sistemas, ou modelo em cascata [29], como ilustra a Figura 12.

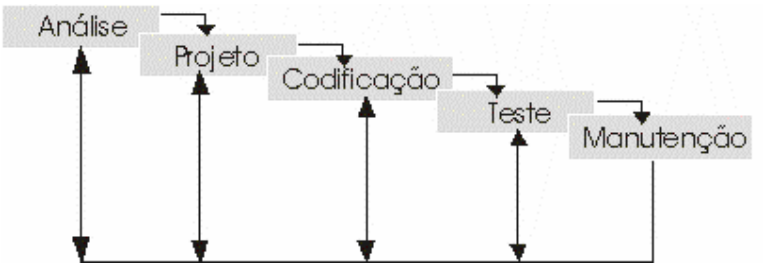

Figura 12. Modelo Cascata

O modelo cascata é um processo que requer uma abordagem sistemática, seqüencial ao desenvolvimento do software, que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção [29].

> "O objetivo principal desse modelo é entender o requisitos do sistema. Tem sido usado com sucesso para validar partes do sistema (Interface Gráfica e aspectos do sistema relacionados à arquitetura - ex: performance, portabilidade, etc.). Como na programação exploratória, a primeira etapa prevê o desenvolvimento de um programa (protótipo) para o usuário experimentar. No entanto, ao contrário da programação exploratória, o protótipo é então descartado e o software deve ser re-implantado na etapa seguinte, usando qualquer modelo de ciclo de vida (ex: cascata)" [18]

Apesar do modelo em cascata ser muito aceito pelos analistas, este nem sempre consegue ser executado seqüencialmente. Primeiramente, isto se deve pela falta de tempo para que cada etapa seja estudada e executada de forma completa e independente. Um outro problema acontece que, para entender e construir sistemas complexos, a estratégia seqüencial não é a mais indicada, e sim uma estratégia incremental e iterativa, entendendo-se suas partes gradativamente e voltando, muitas vezes, a cada uma destas. Além disso, o entendimento de uma das partes do sistema possibibilita o melhor entendimento das demais partes do sistema.

Desta maneira foi-se escolhido o modelo incremental e iterativo [29] proposto como uma resposta aos problemas encontrados no modelo em cascata, como citados acima. Um processo de desenvolvimento, segundo esse modelo, divide o desenvolvimento de software em iterações. Em cada iteração, são realizadas as atividades de análise, projeto, implementação e testes para uma parte do sistema, semelhantemente ao processo em cascata, se diferenciando apenas pela repetição destas fases do seu decorrer como mostra a Figura 13.

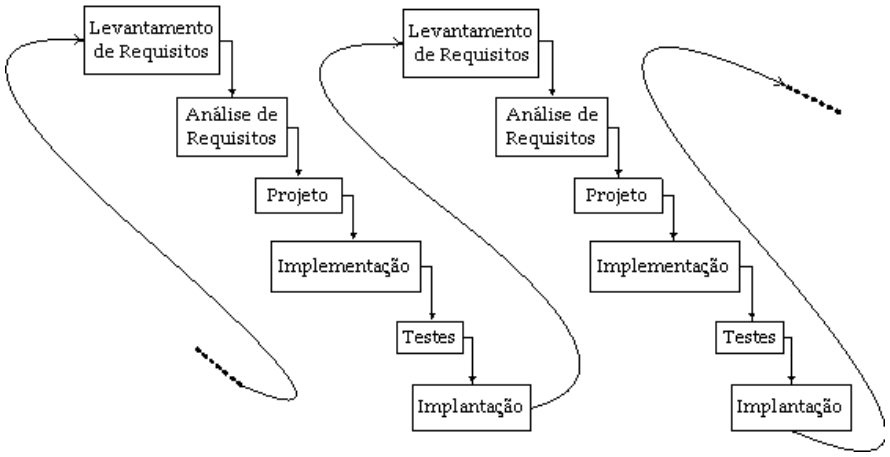

Figura 13. Modelo incremental e iterativo

O modelo incremental e iterativo incentiva a participação do usuário nas atividades de desenvolvimento. Isso ocorre, pois os usuários desde o início do projeto já possuem uma versão do sistema para utilizá-lo e analisá-lo. No caso deste sistema, as versões iniciais não continham a implementação de todas a funcionalidades necessárias. Mesmo assim envolveram-se o Prof. Winter e sua equipe no desenvolvimento das funções do novo Sistema. Destaco que ainda que essa situação (modelo incremental e interativo) é muito melhor do que a situação em que o usuário recebe o sistema todo de uma vez, somente no final do projeto, conforme ocorre no modelo em cascata e muitas vezes não consegue detectar falhas ou necessidade de melhorias em cada parte distinta do projeto.

Assim, baseado neste modelo, as etapas anteriores para o estabelecimento de requisitos para o sistema proposto, deram início à primeira fase do ciclo de desenvovimento do programa – o levantamento e análise dos requisitos. Nesta fase pôde se perceber algumas funções desejadas para o programa, desempenho e interfaces exigidos pelo mesmo.

Após a análise, deu-se início a mais uma outra etapa – o projeto. Nesta, foram (i) estabelecidas todas as estruturas de dados, (ii) enumerada as rotinas a serem codificadas, (iii) identificadas as funções atualmente implementadas, (iv) definida a arquitetura com as subdivisões das classes a serem criadas, (v) definida a comunicação e herança entre elas, (vi) caracterizada a interface e (vii) identificação de possíveis melhorias no programa. A estruturação foi baseada na sub-divisão já imposta no programa de Pearson escritas em forma de subrotinas. A construção do diagrama foi de fundamental importância nesta fase, pois facilitou o entendimento da da estrutura do referido programa.

O programa de Pearson – base para desenvolvimento do programa proposto nesta monografia – foi desenvolvido da linguagem Fortran, que possui um paradigma procedural de programação onde seqüências de passos são fornecidos para a realização das tarefas desejadas. Porém, ele inclui algumas sub-rotinas com funcionalidades distintas – característica frequentemente encontrada em programas na linguagem Fortran. Isto pertime a separação do programa como um todo em várias partes independentes executando tarefas diferentes. Adicionalmente, isto facilitou o entendimento do programa original e auxiliou indiretamente na tranformação destas subrotinas em classes Java no projeto desenvolvido nesta monografia, esta linguagem que segue um paradigma orientado a objetos.

O novo sistema foi dividido em três camadas. A primeira camada, de dados dos parâmetros e execução dos processos, uma outra camada destinada ao armazenamento e coleta de dados em arquivos e mais uma outra camada para a interface gráfica. Para esta estruturação adotada neste trabalho, foram criadas dezoito classes, de acordo com diagrama de classes mostrado na Figura 14.

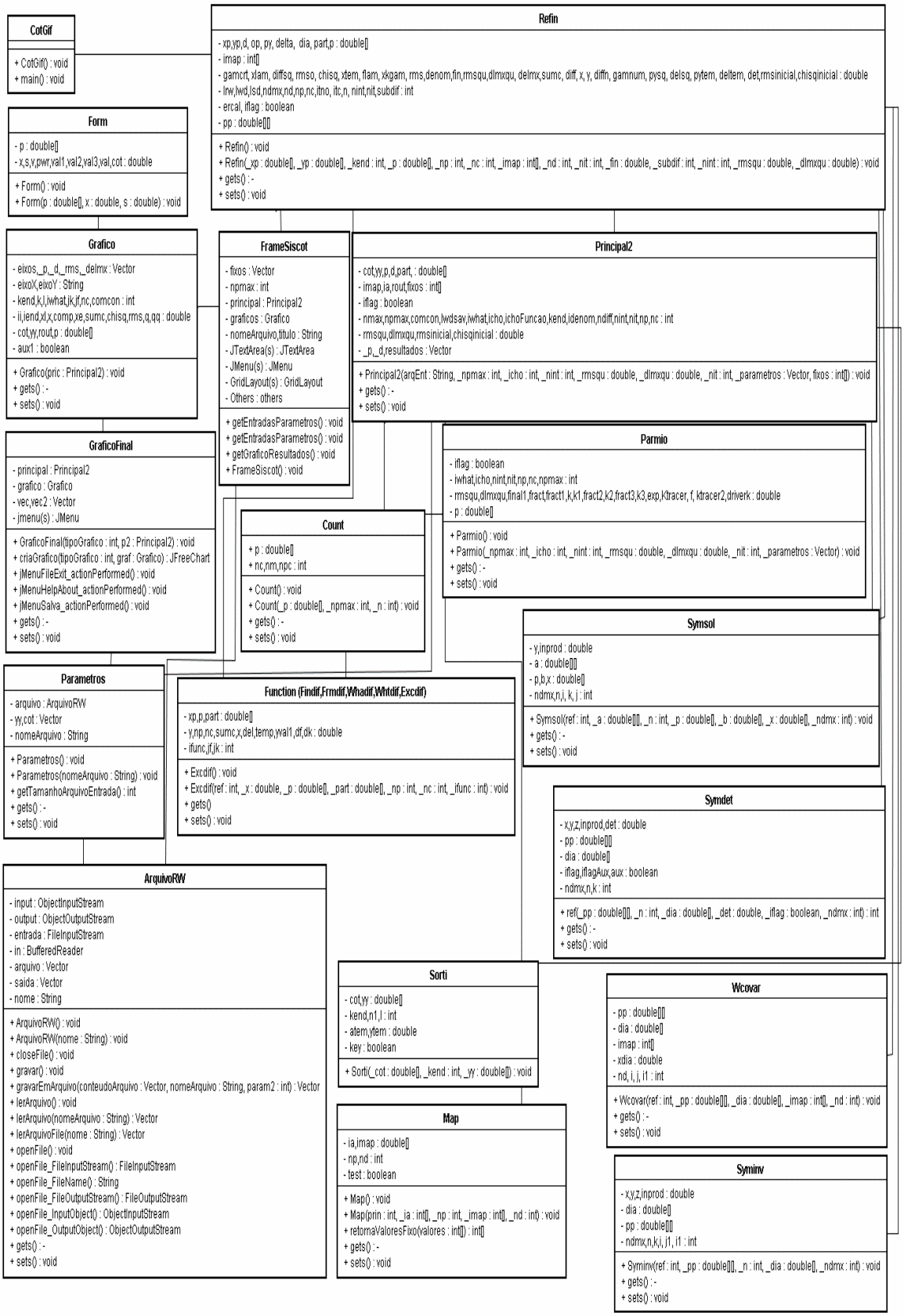

Figura 14. Diagrama de Classes do novo Programa

Uma breve descrição das principais classes do novo programa segue abaixo:

- A classe “CotGif”, inserida dentro da camada de interface gráfica, é a classe que possui o método main do sistema, ou seja, é através dele que o usuário executa o Siscot. Esta classe chama a interface gráfica gerada pela classe “FrameSiscot” que por sua vez, faz a conexão com as classes de dados do sistema.
- As classe “ArquivoRW” é usada pra leitura e escrita em arquivo. Ela faz a manipulação dos dados tanto na importação do arquivo de dados a ser usado como base para cálculo dos resultados e geração do gráfico, quanto na exportção dos resultados finais obtidos para gravação em qualquer dispositivo de armazenamento.
- Enfim, para a última camada – a de dados – foram construídas as demais classes. A classe “Principal2”, como o próprio nome sugere, é a classe principal dentro desta camada. Ela chama uma das principais classes dentro da camada de dados: “Parmio” e“Refin” que fazem a manipulação dos dados propriamente dita. A classe “Refin” é uma rotina que escala a derivada parcial da matriz e usa a correção pelo cosseno para o ajuste da curva, segundo dados obtidos pelo programa de Pearson. Como dito anteriormente, este ajuste usa um algoritmo escrito por Levenberg Marquardt [10]. A classe “Parmio” armazena um vetor para os valores de parâmetros da função escolhida pelo usuário usados no cálculo do ajuste e geração da curva de cot final.
- Ainda nesta camada, foram criadas cinco classes para a manipulação das funções disponibilizadas pelo programa encontradas nos experimentos de hibridização de DNA-DNA e DNA-RNA. Estas rotinas também manipulam matrizes simétricas, acham o determinante, resolvem equacoes lineares, invertem a matriz de cálculo entre outras funções exigidas para a manipulação dos dados para a geração da curva de cot.
- Além das classes citadas acima, o programa ainda usa algumas outras classes adicionais para cálculos intermediários dentro do processo, como por exemplo o “Count” que conta e armazena os parâmetros de entrada e a classe “Map” que recebe um array com os parâmetros referentes à função escolhida para a regressão e cria um segundo array com os parâmetros a serem fixados que também é escolhido pelo usuário na insersão dos dados de entrada.

Como pode ser observado, apesar do sistema proposto neste trabalho ser baseado na estruturação adotada no programa de Pearson, algumas modificações foram feitas para que se pudesse ter um sistema modularizado, de fácil extensibilidade e manutenibilidade, principalmente, de mais fácil usabilidade.

A terceira etapa do desenvolvimento do sistema consiste na codificação/implentação do programa. Assim, todo o projeto, juntamente com os estudos preliminares e protótipo construído, serviram como fonte para estas atividades. A tradução, como dito anteriomente foi feita para a linguagem Java de programação.

A codificação do programa é a etapa de maior complexidade, pois cada operação realizada pelo programa existente de Pearson, deve ser reformulada, sem perder a consistência dos dados e semântica de suas funcionalidades.

O desenvolvimento no modelo incremental e interativo ainda incluiu uma etapa com os testes para detecção de possíveis falhas no programa e gantaria de que as entradas definidas produzem os resultados desejados.

Esta etapa adicional foi de fundamental importância no projeto, pois foi aqui onde foram validados todos os resultados gerados pelo sistema obtido após o ciclo de desenvolvimento do

mesmo. Desta forma, os teste tem o objetivo de verificar se todos os requisitos do sistema foram corretamente implantados, assegurando a qualidade e corretude do software produzido, além de tentar reduzir custos de manutenção corretiva e retrabalho do mesmo. Os testes ainda tiveram o objetivo de assegurar a satisfação do usuário, no caso o Prof. Winter e seu grupo, com o produto desenvolvido.

Assim, foram comparados os resultados obtidos pelo *Siscot* (nome do novo Sistema) com os resultados esperados, que neste caso, deveriam resultar em valores iguais aos retornados pelo programa original de Pearson - usado como base no desenvolvimento deste sistema. Além disso, ainda se verificou se todos os componentes de softaware estavam corretamente integrados.

Para esta etapa foram usados dois tipos de testes: um feito no decorrer do desenvolvimento do programa, comparando alguns exemplos executados no programa de pearson e no Siscot; e um outro teste executado pelo Professor Doutor Carlos Eduardo Winter do Instituto de Ciências Biomédicas do Departamento de Parasitologia Da Universidade de São Paulo – USP. Esta segunda bateria de testes resultou em um parecer, elaborado pelo Doutor Carlos Winter de acordo com o APÊNDICE A que foi de grande importância para este projeto.

Enfim, o ciclo para a elaboração do programa finaliza-se com uma visita à USP para que o novo sistema – *Siscot* – fosse testado e remotamente implantado nos terminais do laboratório de Biologia. Esta visita foi de fundamental importância para o projeto, pois nela foi possível conhecer parte do processo de análise e coleta dos dados laboratoriais para uso no sistema e ainda identificar melhoras e sujestões para trabalhos futuros como mostra o esquema da Figura 15 .

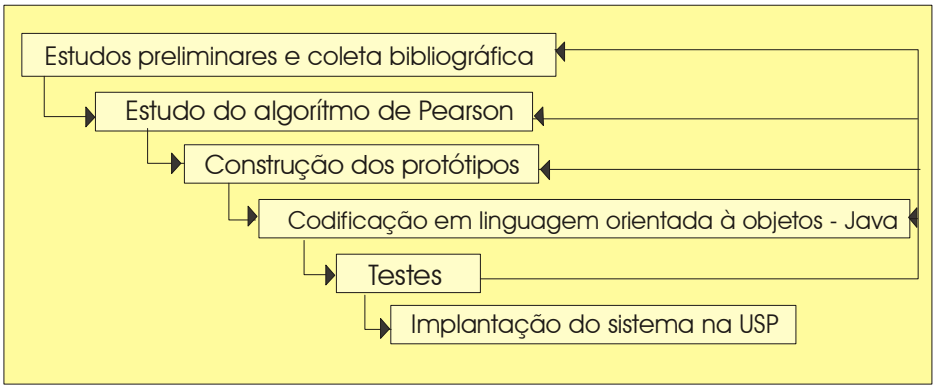

Figura 15. Esquema para a costrução do novo sistema – *Siscot*

Capítulo 5

Resultados

Este capítulo apresenta os resultados alcançados com o desenvolvimento do sistema proposto nesta monografia.

5.1 *Siscot* – Um novo programa para análise de cinética de reassociação de DNA

Como proposto inícialmente, o principal objetivo desta monografia era a construção e revisão de um sistema para análise de cinética de reassociação de DNA que, baseado no programa original de Pearson, Davidson e Britten desenvolvido no final da década de 1970, apresentasse uma interface mais atual e amigável com o usuário. O sistema, chamado de *Siscot*, foi desenvolvido com a finalidade de ser o mais intuitivo e de melhor usabilidade.

Além das duas características acima, houve a preocupação de construir o '*Siscot*' de maneira que fossem facilmente incorporadas novas funcionalidades. Para isto o programa foi dividido em classes distintas e independentes usando o paradigma orientado a objetos oferecido pela linguagem Java de programação.

Na versão do sistema construído, melhorias na interface e revisão de código foram conquistadas além da inclusão do pacote adicional do programa de Pearson – *Cotfil* – para a contrução do arquivo de dados de entrada.

Identificou-se como prioridade, a melhoria visual do programa de Pearson – base na construção do sistema – pois já seria de grande avanço e contribição para os usuários, visto que esses perdem muito tempo ao usar o sistema antigo. Esta perda de tempo deve-se por dois motivos. O primeiro, pelo fato de o programa de Pearson não permitir que, após entradas erradas, o usuário possa voltar etapas na longa manipulação dos dados, tendo que retornar obrigatoriamente ao inicio do programa. O segundo motivo foi a inconsistência dos dados, que resultam diferentes valores quando o programa original é executado em máquinas diferentes, ou até mesmo na mesma máquina.

5.2 O *Siscot* e sua interface

O sistema proposto – *Siscot* – foi desenvolvido, como afirmado acima, com o intuito de ser o mais amigável e interativo o possível. Assim, ele possui uma interface simples para a entrada de dados, manipulação de parâmetros e geração de resultados e gráficos.

O *Siscot* possui uma tela inicial com três módulos (ver Figura 16):

1- escolha do arquivo de dados de entrada;
2- escolha da função e entrada dos parâmetros; e
3- apresentação de resultados

Os dois primeiros módulos são independentes. Já o terceiro – resultados – depende dos dois anteriores para gerar seus valores de resposta. Cada módulo será explicado com maiores detalhes no decorrer deste capítulo.

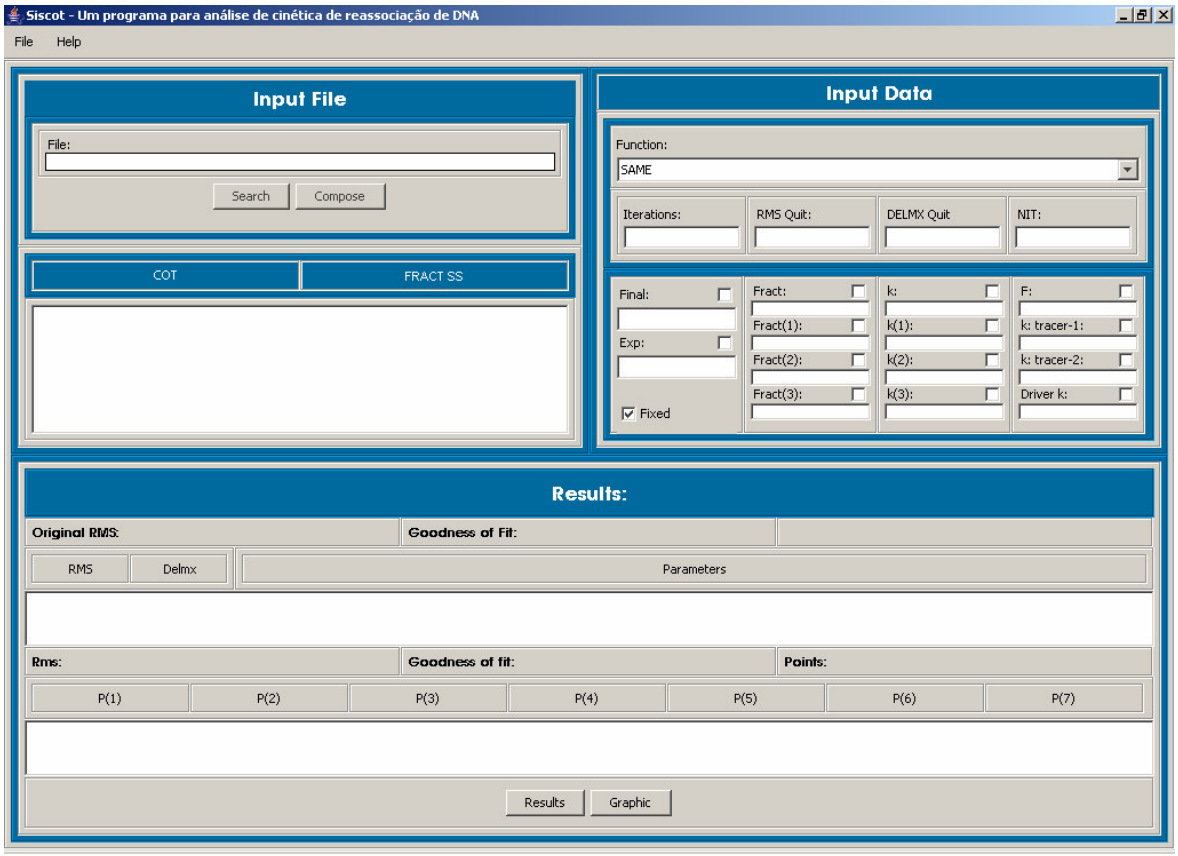

Figura 16. Tela inicial do *Siscot*, note-se a maior organização, clareza e intuitividade da nova interface

5.2.1 Módulo 1 - Entrada do arquivo de dados

O primeiro módulo mostrado na Figura 17 permite o usuário escolher um arquivo de dados de entrada previamente construído, baseado nos dados coletados nos experimentos em laboratório.

Este arquivo possui um formato semelhante ao desenvolvido pelo programa *Cotfil* do pacote do Pearson para que não seja preciso reformatar os arquivos de dados já existentes no usuário. Este é um arquivo de texto simples que carrega a listagem dos valores de *Cot* e fração de DNA correspondente (*Fract ss*).

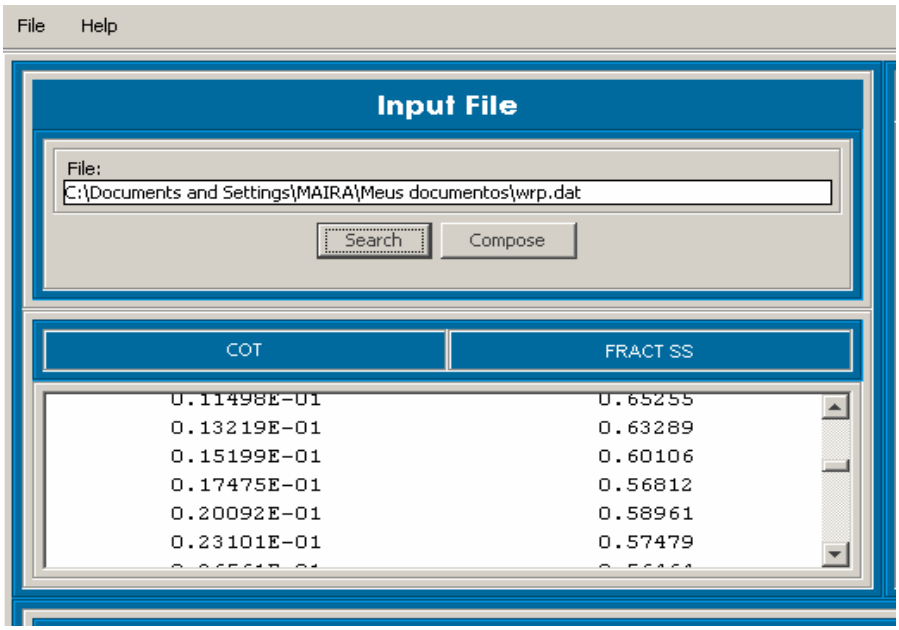

Figura 17. Tela para entrada de Dados

Após a escolha do arquivo ou construção manual do mesmo, os dados são listados na tela, permitindo o usuário ter a visão dos dados utilizados na construção da curva de cot resultante.

5.2.2 Módulo 2 - Entrada dos dados de entrada

O segundo módulo mostrado na Figura 18 permite a escolha da função (note as diferentes opções de escolha de função na tela) que melhor se adequa à base de dados coletados no experimento como explicado anteriomente no decorrer deste trabalho. Quando escolhida a função, os parâmetros referentes a esta serão habilitados para que o usuário entre com os dados de entrada.

Ainda existe uma opção de fixar parâmetros para que eles não sejam modificados no decorrer do cálculo da regresão. Esta opção pode ser executada marcando ou desmarcando a caixa ao lado de cada parâmetro.

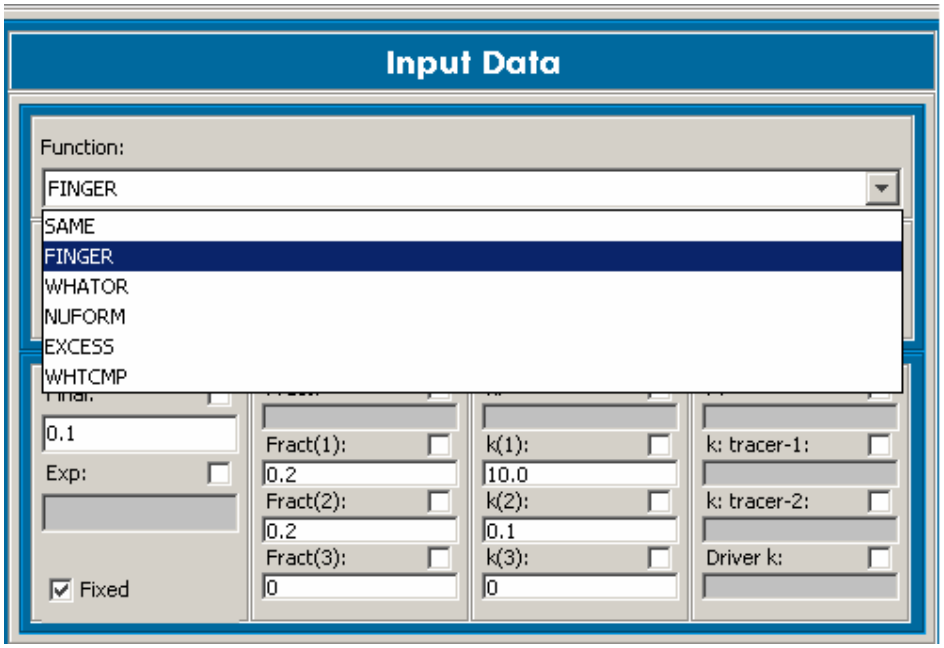

Figura 18. Tela para entrada de parâmetros e escolha da função de regressão

Estes campos se referem aos parâmetros das funções que elas representam. Para cada tipo de função existe um conjunto de parâmetros, assim, ajustando estes parâmetros ou os fixando tenta-se chegar ao menor ajuste da curva (*goodness of fit*).

5.2.3 Módulo 3 - Geração de resultados

Este terceiro módulo ilutrado na Figura 19 depende dos módulos 1 e 2 – que aqueles estejam corretamentes prrenchidos. Caso contrário, aparecerão janelas de aviso, orientando o usuário a concertar o erro cometido. Esta uma das muitas melhorias incluidas no novo sistema.

Quando os campos são corretamente preechidos, basta que o usuário clique no botão "result", que os resultados dos cálculos da regressão aparecerão em seguida. O usuário pode fazer inúmeras interações para achar o menor valor de *rms* usado como parâmetro para saber se a curva construída foi a melhor. Este valor, *rms*, representa a taxa de erros encontradas na construção da curva.

O primeiro campo mostra os cálculos dos parâmetros. Estes executam até o número de ciclos determinado ou até o valor do *rms* também exibido ao lado esquerdo chegar ao mínimo também pré determinado pelo usuário antes do cálculo.

Ao final do cálculo da regressão, são exibidos os valores dos parâmetros finais encontrados para o ciclo executado. Estes valores servirão como parâmetros para a construção final do gráfico gerado pelo siscot. Ainda é exibido o valor final do *rms* e o valor do ajuste da curva (*goodness of fit*) que servirá como limiar de parada para os cálculos finais. Se este ajuste for "bom" deve-se plotar o gráfico final, caso contrário, ajusta-se novamente os parâmetros até que *goodness of fit* fique o menor possível.

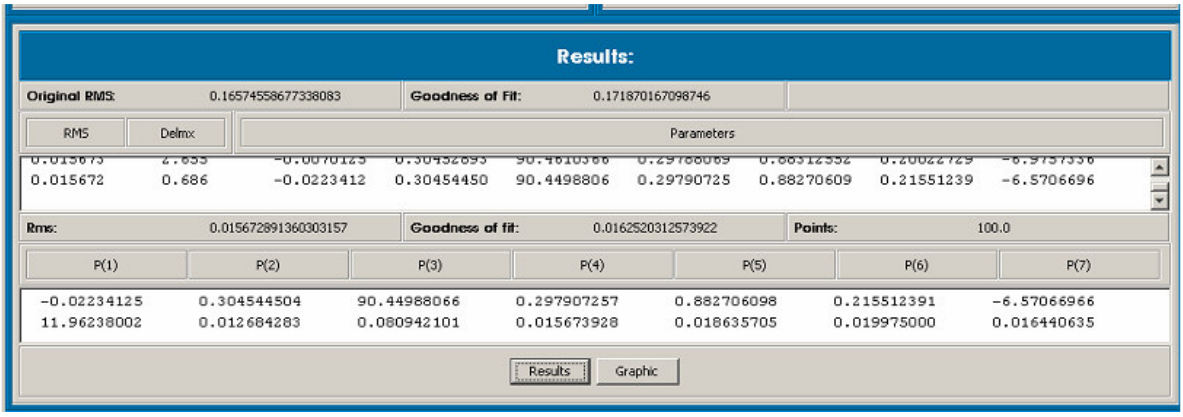

Figura 19. Tela com os resultados

5.2.4 Módulo 4 - Gráfico

Este terceiro e último módulo mostrado na Figura 20 representa a geração da curva de *Cot* para o material em análise. Atualmente ele é representado em forma de linha, mas poderá ser apresentado apenas por pontos, dado que seus valores não são contínuos.

Além do gráfico, os valores de entrada de dados, coeficientes de ajuste da curva e cot equivalentes são listados ao lado do gráfico correspondente.

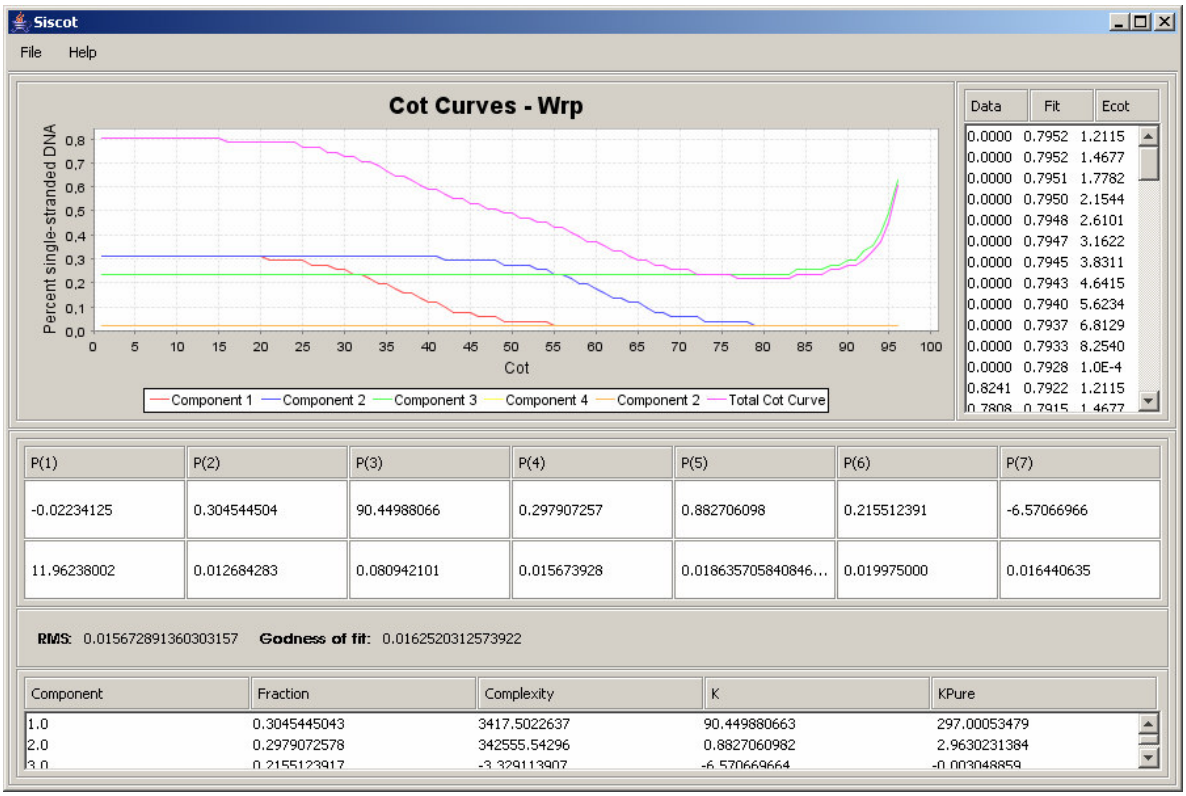

Figura 20. Tela com resultados e gráfico de Cot

Abaixo no gráfico, ainda são listados os parâmetros finais encontrados pelo cálculo da curva, juntamente com o *rms* final e o menor valor de ajuste (*goodness of fit*), usados como

critério de parada para o cálculo da curva. Ao final do cálculo da regressão, são exibidos os valores dos parâmetros finais encontrados para o ciclo executado. Estes valores servirão como parâtros para a construção final do gráfico gerado pelo siscot.

Enfim, são gerados para cada componente: valores da fração, complexidade, k, e kpure, que são valores usados, juntamente com o formato e as características do gráfico, na análise dos resultados e na captura das características obtidas através do experimento e cálculo da curva de *Cot*.

5.3 Erros corrigidos do programa de Pearson

Além de melhorias na interface do sistema de Pearson, no decorrer do desenvolvimento do sistema, foi feita uma análise minuciosa no código do programa. Já nesta primeira versão do novo sistema foi corrigido o erro existente no programa de Peason quanto a geração dos resultados que se dava de forma inesperada. Isto ocorria quando se entrava com um número *x* de parâmetros e o resultados exibia uma quantidade de parâmetros diferentes dos de entrada Figura 21. Por exemplo, para a função FINGER, existem sete entradas de parâmetros – Final, Fract1, k1, Fract2, k2, Fract3, k3. O resultados obtidos pelas diferentes compilações e execuções do programa de pearson, retornaram cinco(5) ou onze(11) saídas como ilustrado abaixo.

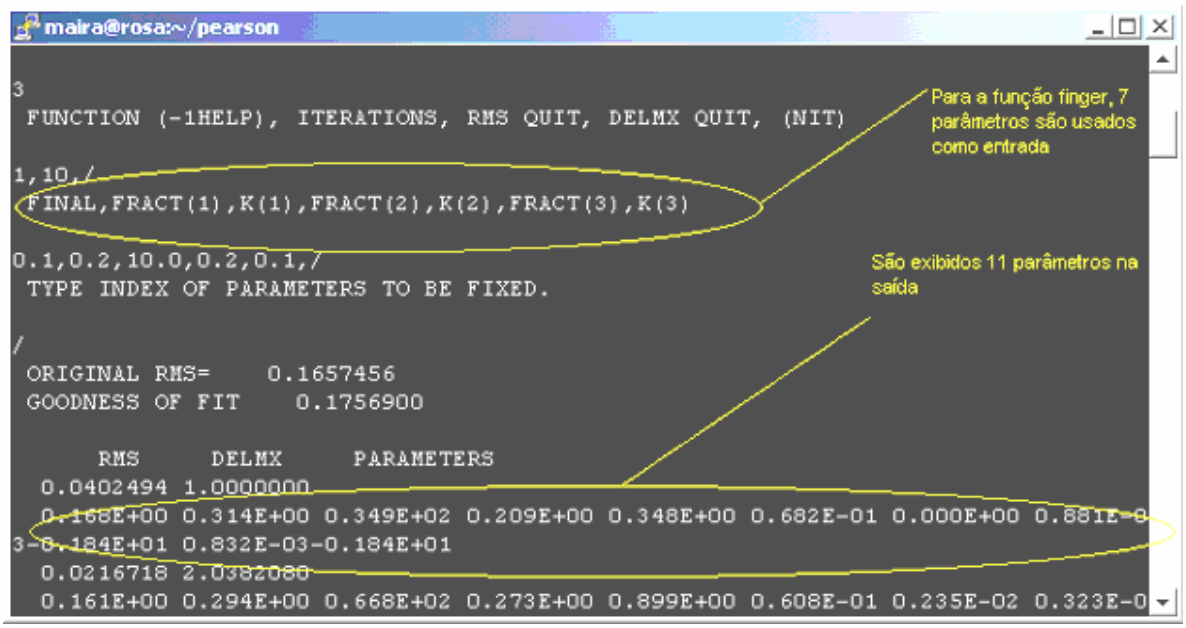

Figura 21. Tela do programa de Pearson com os onze parâmetros sendo impressos errados

Este erro repercute na geração dos resultados, pois algumas rotinas que utililizam iterações sucessivas (laços *for*, por exemplo) são baseadas no número de entradas de parâmetros. Assim, se este número estiver errado, estruturas que utilizam recursão são afetadas resultando em valores completamente diversos dos esperados. Além disto, este erro ainda pode ser refletido em outras rotinas que utilizam os seus resultados, causando uma divergência de dados ainda maior.

Um outro aspecto notado, como já citado anteriormente no decorrer deste trabalho, foi a inconsistência dos dados vista com execuções diversas dos programa de Pearson. Em vários casos, para um mesmo exemplo com entradas de dados e funções idênticas, valores diferentes são resultados quando rodados em máquinas com diferentes configurações como mostra a Figura 22 com exemplos de gráficos gerados pelo programa. Isto se devia à maneira que estas máquinas interpretavam os tipos de dados, ou seja, a quantidade de bytes, por exemplo, que um tipo de dados era armazenado. Como o programa de Pearson era sempre recompilado nestas diferentes máquinas, ele trazia resultados distintos de leituras destes dados, fazendo com que os cálculos retornassem valores diferentes.

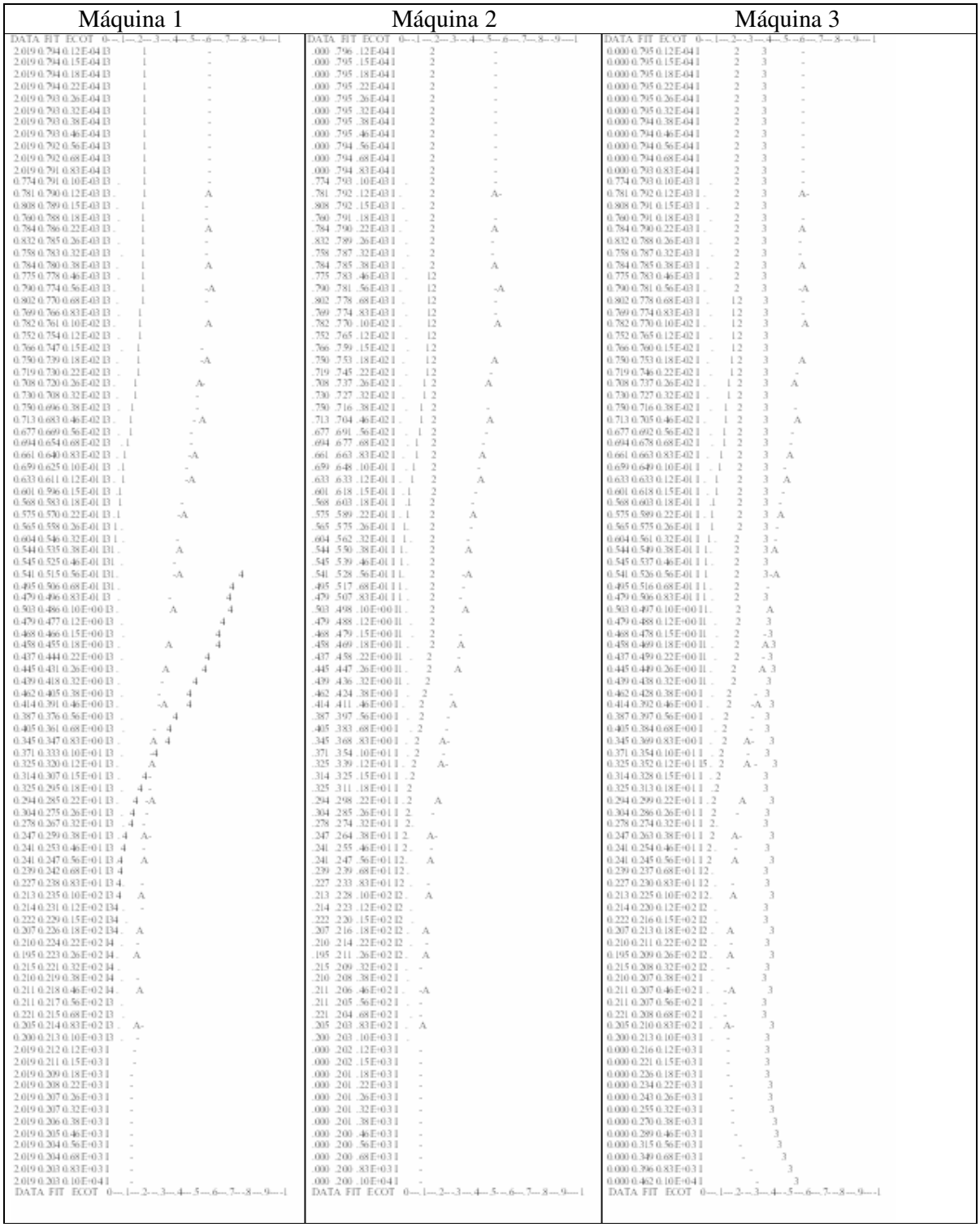

Figura 22. Exemplo rodado no Pearson em três máquinas com configurações diferentes

5.4 Análise Comparativa entre o novo sistema e programa de Pearson

Na etapa de testes incluída no desenvolvimento deste sistema, foram propostas duas análises comparativas entre o sistema antigo de Pearson e o atual.

Para a primeira análise comparativa, o Professor Winter disponibilizou alguns testes executados, utilizando o Programa de Pearson, no Instituto de Ciências Biomédicas do Departamento de Parasitologia da Universidade de São Paulo – USP. Em seguida, estes mesmos testes ainda foram executados numa segunda máquina com outra configuração. Para a execução em qualquer uma destes máquinas o novo sistema resultou os mesmos valores.

Observou-se que para as duas compilações do programa de Pearson foram obtidos diferentes resultados para a mesma entrada de dados e parâmetros. Isto já havia sido verificado pelo próprio professor e outros usuários do programa que apontavam esta inconsistência de dados no programa de Pearson. Este problema, relativamente grave, foi um dos objetivos propostos para serem resolvidos no sistema ora proposto.

Com os dados utilizados nos testes acima em mãos, fez-se comparações entre os resultados do programa de Pearson processado nas duas máquinas e o sistema novo. A primeira versão do novo sistema apresentou resultados compatíveis com os obtidos na segunda máquina; ver Figura 23.

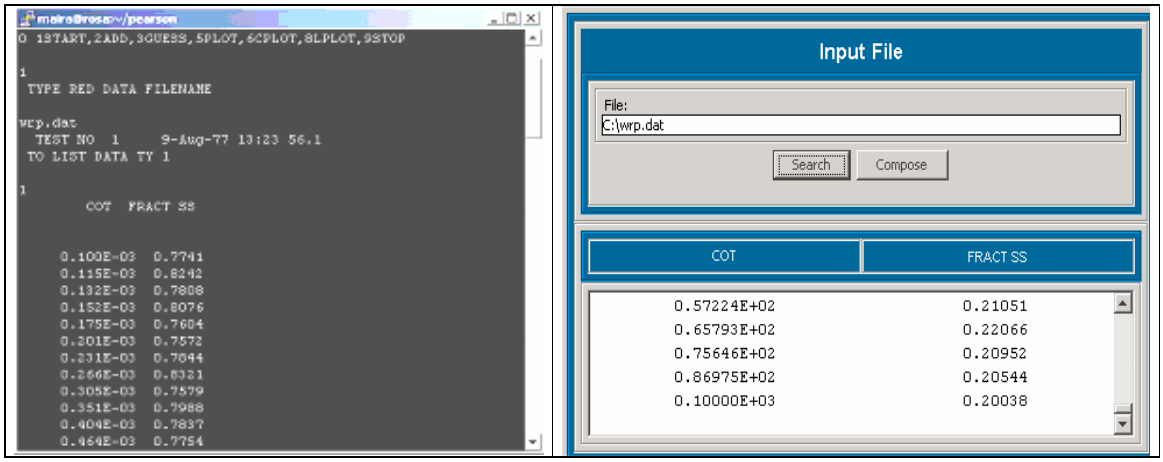

Figura 23. Comparativo entre a tela de entrada de dados entre Siscot / Pearson

A primeira comparação foi feita para a entrada do arquivo de dados. O programa de Pearson, como pode ser observados na Figura 23, além de ter suas entradas de dados pelo prompt do DOS, este não fornece a listagem com os arquivos disponíveis considerando que o usuário já conhece o caminho e o nome do arquivo de entrada a ser usado pelo programa. Além disso, este arquivo deverá estar contido na mesma pasta em que programa está instalado.

O programa novo – Siscot – possui uma interface para esta entrada de dados bem mais amigável e permite que o usuário percorra todos os diretórios contidos em sua máquina e escolha o caminho em que o arquivo se em encontra sem que o mesmo precise ser digitado.

Além disso, o *Siscot* tem a opção de criar o arquivo de entrada se o mesmo ainda não tiver sido criado e gravado. Esta opção de inserção e criação de arquivo de dados de entrada é disponibilizada por um pacote adicional do Pearson chamado *Cotfil*, porém esta inclusão dos

pares de dados (*Cot* e *FractSS*) devem ser realizadas sem que haja qualquer erro de digitação pois isto levaria ao início do processo e perda de todos os valores entrados até o momento do erro.

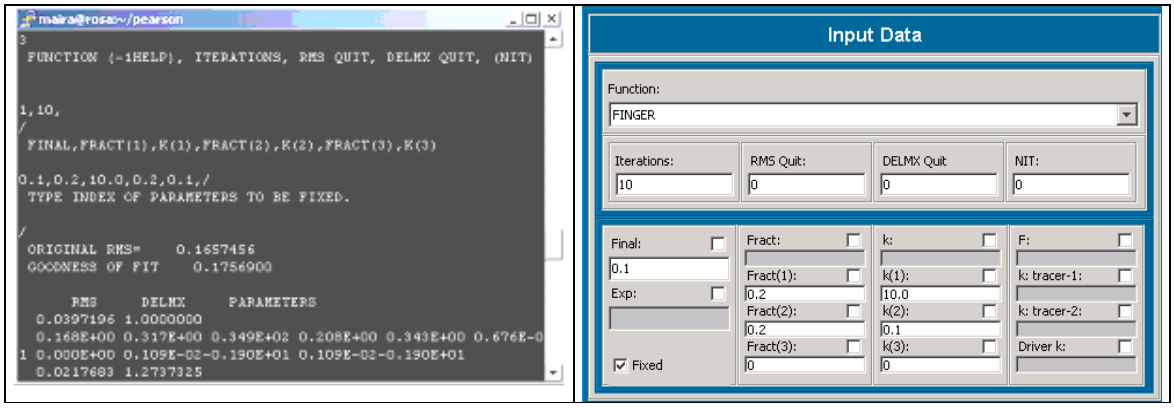

Figura 24. Comparativo entre a tela de entrada de parâmetros e escolha da função entre Siscot / Pearson

A segunda comparação exibida na Figura 24 foi feita com a opção de escolha da função usada na regressão e entrada de seus parâmetros. No programa de Pearson , a escolha da função é através da digitação do número que a referencia. Caso o usuário não saiba qual é este número, existe a opção "help" que lista as funções disponíveis no programa. Após a escolha da função, o usuário deverá entrar com os valores de parâmetros de acordo com o que é pedido após a confirmação do uso da função escolhida. Esta lista de parâmetros deve ser digitada separada por vírgulas e finalizada por uma barra.

O programa não faz a verificação do número de entradas que o usuário digitou, podendo este entrar com um número maior ou menor do solicitado. Caso este número seja inferior, o programa completa os valores em branco com o número zero.

No *Siscot* a opção de escolha das funções é dada por uma lista de funções em que o usuário apenas seleciona a função desejada. Após a escolha da função, os campos referentes a esta são habilitados. Caso todos os campos não sejam preenchidos, uma mensagem é ativada para que operação seja terminada.

O programa ainda disponibiliza uma caixa de seleção ao lado de cada entrada de parâmetro que deverá ser habilitada caso este seja fixado.

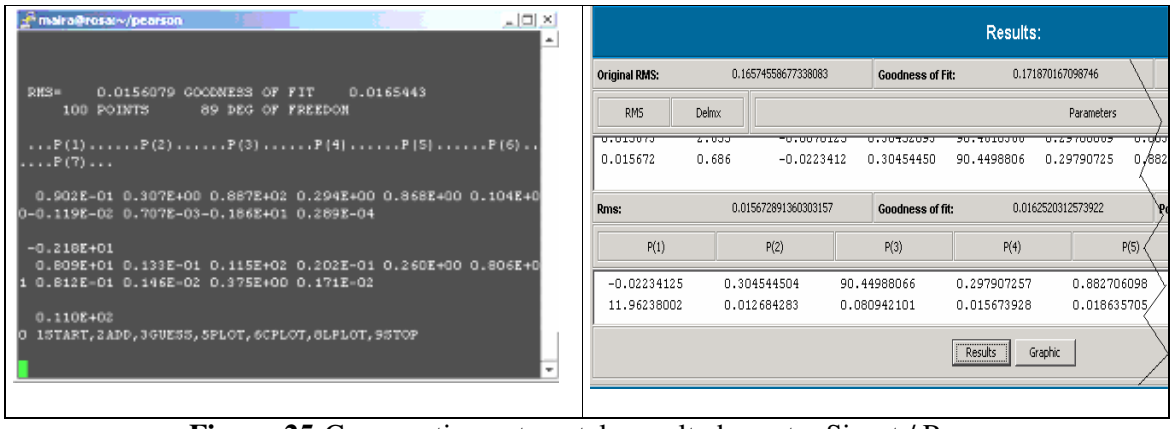

Figura 25. Comparativo entre a tela resultados entre Siscot / Pearson

A Figura 25 mostra os resultados impressos de acordo com a função escolhida. Estes resultados são disponibilizados em um quadro de forma organizada e de fácil entendimento.

Para a validação dos resultados de saída do Siscot, estes foram minuciosamente comparados aos resultados no programa do Pearson para um conjunto de exemplos de entradas e saídas de dados do mesmo.

Além da comparação entres resultados, uma das finalidades da criação do sistema novo foi a de detectar um maior número de falhas contidas no programa de Pearson, como dito anteriomente.

Este erro também foi resolvido para o Siscot que por sua vez calcula o mesmo número de parâmetros que lhe é dado como entrada, eliminando assim esta falha.

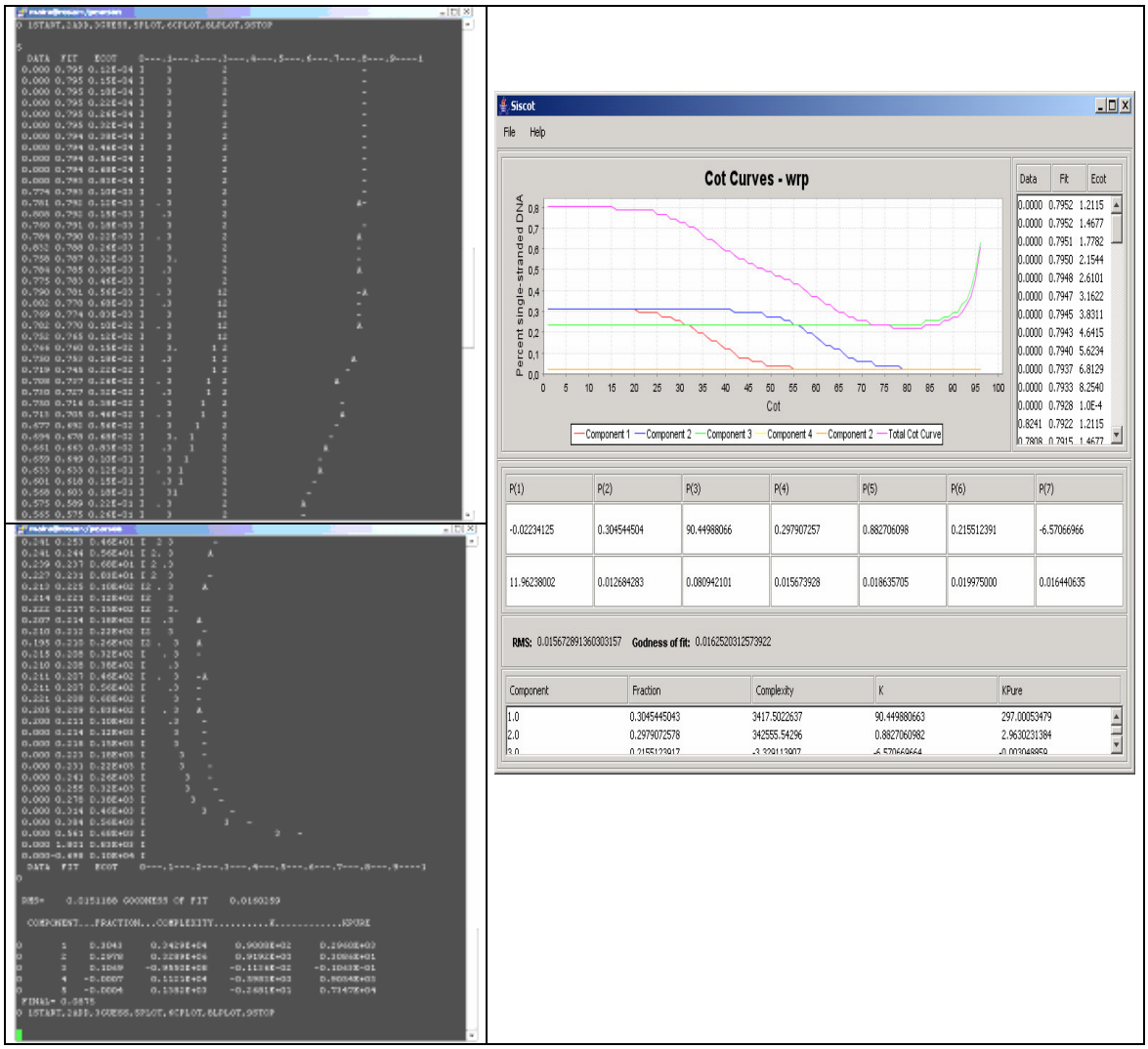

Figura 26. Comparativo entre a tela de saída da curva de Cot entre Siscot / Pearson

Como pode ser observado na Figura 25, além da opção de geração de resultados, existe uma opção para a geração de gráfico. Quando a opção de gerar gráfico é selecionada, uma janela com o resumo dos resultados e o gráfico de *Cot* resultante é exibida.

Comparando o gráfico gerado pelo Pearson como o gerado pelo *Siscot*, observa-se o quanto o sistema novo proporcionou uma melhoria na vizualização dos resultados finais,

principalmente quando se trata da impressão do gráfico. Além da melhora visual, o Siscot gera os mesmos resultados independente da plataforma ou configuração da máquina que ele foi instalado. A consistência de dados é de fundamental importância na execução de qualquer programa. Esta consistência não é verificada no programa de Pearson, pois além de seus resultados serem diferentes em compilações de máquinas diferentes, ainda existe uma diferença entre resultados gerados na mesma máquina com a mesma compilação.

Enfim, pode-se observar nos testes realizados com o Siscot que os resultados numéricos foram iguais ou bastante aproximados se comparados aos gerados pelo Pearson e que ele proporcionou ao usuário uma interface melhor, resultados mais consistente, independência de plataforma, maior funcionalidade e possibilidade de uso em ambiente de rede. Estes benefícios, objetivos do projeto, foram verificados junto ao usuário que conta agora com uma ferramenta muito mais ajustada às suas necessidades atuais.

Capítulo 6

Conclusão

6.1 Conclusão

O software produzido neste trabalho é melhor estruturado e mais adequado para a análise da cinética de reassociação de DNA, do que o programa original de Pearson, Davidson e Britten [12].

A migração e as adaptações feitas neste trabalho permitiram a utilização do programa em terminais gráficos com geração da curva de regressão e apresentação "*userfriendly*" dos resultados numéricos. A distribuição das janelas de entrada e saída de dados é bastante intuitiva e permite ao usuário com conhecimento de cinética de reassociação a utilização do programa para diferentes fins, principalmente análise do tamanho e composição de genomas candidatos a projetos de seqüenciamento.

Ainda se fez a fusão do programa contido no pacote de Pearson – *Cotfil* – responsável pela inclusão dos dados (*Cot* e *Fract SS*) um a um no prograna original. Esta inclusão agilisou o trabalho visto que antes estes este era feito separadamente e com diversas restrições como já descritas no decorrer deste trabalho.

Este programa facilitará aos pesquisadores, principalmente na área de genômica de plantas, o uso do sistema de clonagem por fracionamento de *Cot* do DNA genômico.

Ainda se preocupou com a confiabilidade dos dados que no programa de Pearson resultavam direfentes valores quando executados em máquinas com configurações distintas. Para isto, foi feita uma busca detalhada em seu código para que fossem achados o maior números de erros possíveis.

Apesar da linguagem Fortran ainda ser a linguagem mais indicada para cálculos científicos, proporcionando um melhor desempenho aos programas, o *Siscot* - desenvolvido em liguagem Java – correspondeu de forma positiva visto que este teve como principal objetivo o seu uso em aulas na própria academia. Desta forma não foi importante a análise de desempenho do sistema neste primeiro instante.

Assim, o programa permite uma interface gráfica amigável e a possibilidade de sua utilização nestas aulas para permitir que os alunos entendam, na prática, os passos envolvidos na determinação das constantes de reassociação.Esta contribuição foi sugerida pelo Professor Doutor Carlos Eduardo Winter do Instituto de Ciências Biomédicas do Departamento de Parasitologia Da Universidade de São Paulo - USP. O Professor Winter, pessoalmente ainda validou e aprovou

o novo sistema que, segundo ele vai ser muito útil no futuro pois lhe economizará muito tempo durante suas tarefas de análise e interpretação dos dados.

6.2 Trabalhos futuros

Apesar do trabalho ter atingido todos os seus objetivos e havendo tantas contribuições reais para o usuário, o sistema desenvolvido nesta monografia ainda poderá ser extendido e melhorado. Abaixo segue uma lista de sugestões e pontos para essas melhorias:

- Uma primeira sugestão é a melhoria ainda maior de interface do sistema. Poder-seá aumentar o suporte aos parâmetros do sistema aumentando ainda mais a flexibilidade do sistema e melhorando a geração dos resultados automaticamente a partir da escolha dos parâmetros sem que seja necessário clicar no botão referente á execução destes cálculos.
- Um segundo aspecto observado de melhoria, foi que a base dados ainda é armazenada em um único repositório. A sugestão seria fazer com que o sistema fosse adaptado para a rede *wide area network* (WAN) de comunicação, podendo ser acessado de qualquer terminal e possuindo uma base de dados comum.
- A última melhoria recomendada neste trabalho, seria uma integração dos dados obtidos nas análises laboratoriais com o sistema. Atualmente os dados são reportados para planilhas de entrada de dados e tranferidos para um arquivo texto num formato predeterminado para que este possa ser interpretado pelo sistema. A idéia que isto seja feito de forma automatizada com um arquivo comum entre as duas etapas de análise dos dados – laboratorial e pelos dados fornecidos pelo sistema – para otimizar o processo de análise do genoma em estudo.

Bibliografia

[1]. SILBERSCHARTZ, P. Galvin, G. Gagne, *Sistemas Operacional: conceitos e aplicações. 1*. ed. Campus, 2000. 585p

[2]. AHN I-Y. *Determinação do tamanho, complexidade e estrutura do genoma do nematóide Oscheius tipulae (linhagem CEW1)* [Tese de Doutorado]. São Paulo; Instituto de Ciências Biomédicas da Universidade de São Paulo; 2004 [90pp]

[3]. BIO Informática: Reportagens. Atualizado em 10 de agosto de2003. Disponível em: <http://www.comciencia.br/reportagens/bioinformatica>. Acesso em: 10 de agosto 2004.

[4]. BRITTEN, R.J., and Kohne, D.E., 1970, *Repeated Segments of DNA, Scientific American*, 222: 24-31.

[5]. BRITTEN, R.J., and Kohne, D.E., 1968, Repeated Sequences of DNA, Science, 161: 526-640.

[6]. FRANCISCO J.S. Lara – *de ácidos Nucléicos (1995)* – Sociedade Brasileira de Genética

[7]. J. D. Watson e F. H. C. Crick. *A structure for Deoxyribose Nucleic Acid*. Disponível em: <http://www.icb.ufmg.br/~lbcd/grupoa/>. Acesso em: 01 outubro 2004

[8]. Jornal da Ciência. *Técnica facilita estudo de genes de plantas.* JC e-mail 2433, de 22 de Dezembro de 2003. Disponível em: <http://www.jornaldaciencia.org.br>. Acesso em: 23 agosto 2004.

[9]. LEVENBERG, K. *"A Method for the Solution of Certain Problems in Least Squares."* Quart. Appl. Math. 2, 164-168, 1944.

[10]. MARQUARDT, D. *"An Algorithm for Least-Squares Estimation of Nonlinear Parameters."* SIAM J. Appl. Math. 11, 431-441, 1963.

[11]. NEVES, O.P., *Projeto Genoma*. Disponível em <http://www.ufv.br/dbg/BIO240/G04.htm>. Acesso em: 01 novembro2004.

[12]. PEARSON, W.R., Davidson, E.H., and Britten, R.J. 1977. *A program for least squares analysis of reassociation and hybridization data.*

[13]. R.B. Goldberg, From Cot curves to genomics. *How gene cloning established new concepts in plant biology*, Plant Physiol. 125 (2001) 4–8.

[14]. STEIN. *Genome sequence of Caenorhabditis briggsae: a platform for comparative genomics*. PLoS Biol. 1, 166-192, 2003.

[15]. SUN Microsystems. Using NROFF & TROFF. 1990

[16]. VENNERS, B. Inside the JAVA 2 Virtual Machine, 2nd. Ed., New York: McGraw-Hill, 1999. 703p.

[17]. ANDREWS, G. *Foundations of Multithreaded, Parallel, and Distributed Programming,* Boston: Addison-Wesley, 2000. 664p.

[18]. VASCONCELOS, A. M. L. & Maciel, T. M. M. *Introdução à Engenharia de Software e aos Princípios de Qualidade 1*. ed. Lavras: FAEPE, 2003. 104p.

[19]. SOUZA,G.S. Introdução aos Modelos de Regressão Linear e Não linear. Brasília, DF. Embrapa - SEA, 1998.

[20]. R.J. Britten, D.E. Graham, B.R. Neufeld, *Analysis of repeating DNA sequences by reassociation, Meth. Enzymol*. 29 (1974) 363– 405.

[21]. D.G. Peterson, W.R. Pearson, S.M. Stack, *Characterization of tomato genome using in vitro and in situ DNA reassociation,* Genome 41 (1998) 346–356.

[22]. R.B. Goldberg, From Cot curves to genomics. *How gene cloning established new concepts in plant biology*, Plant Physiol. 125 (2001) 4–8.

[23]. E. Schae.eler, T. Lang, U.M. Zanger, M. Eichelbaum, M. Schwab, *High-throughput genotyping of thiopurine S-methyltransferase by denaturing HPLC*, Clin. Chem. 47 (2001) 548–555.

[24]. GARDINER, in: Haken (Ed.), *Hand Book of Stochastic Methods Springer*, Berlin, 1983, pp. 1–11.

[25]. J.G. Sutcli.e, Cold Spring Harb. *Symp*. Quant. Biol. 43 (1979) 77– 90.

[26]. Wirth, N., *Programação Sistemática em Pascal*, Editora Campus, Rio de Janeiro, 1978.

[27]. FORBELLONE, A. L. V., e Eberspächer, H.F., *Lógica de Programação - A Construção de Algoritmos e Estrutura de Dados*, Makron Books, 1993.

[28]. FAYYAD, Usama M. et al. *Advances in knoledge discovery and datamining. Menlo Park*, Califórnia EUA: AAAI Press, 1996.

[29]. PRESSMAN, Roger S. - *Engenharia de Software*, Makron Books, 1995.

[30]. CHAPMAN, Stephen J. *FORTRAN 90/95 for Scientists and Engineer*s

[31]. CEREDA, Ronaldo L.D. e José Carlos Maldonado. *Introdução ao FORTRAN 77 para microcomputadores Maldonado*

[32]. DECYK, Victor K., Charles D. Norton, Boleslaw K. Szymanski - Introduction to Object-Oriented Concepts using Fortran90 Disponível em: <http://www.cs.rpi.edu/~szymansk/OOF90/F90_Objects.html> Acesso em: 31 julho 2005.

[33]. Visual Fortran – HP Invent,Disponível em: <http://h18009.www1.hp.com/fortran/fortran-redirect.htm> Acesso em: 31 julho 2005.

Apêndice A

Parecer Carlos E. Winter

Instituto de Ciências Biomédicas
Departamento de Parasitologia
Av. Prof. Lineu Prestes, 1374 - 05508-900 São Paulo - BRASIL
fone: (55)(11)3091-7269 FAX: (55)(11)3091-7417 ICB2/SALA 14
e-mail: cewinter@icb.usp.br http://icb.usp.br/~cewinter/

PARECER

O programa SISCOT desenvolvido em JAVA por MAIRA PASCHOALINO FERNANDES, sob orientação do DR FERNANDO BUARQUE DE LIMA NETO, foi baseado no programa original de Pearson, Davidson e Britten. Este último programa foi desenvolvido no fim da década de 1970 em linguagem FORTRAN e tinha como objetivo a análise de cinéticas de reassociação de ácidos nucléicos para os primeiros estudos de organização genômica.

O programa original só podia ser utilizado em terminais não gráficos, tendo sido originalmente desenvolvido para computadores da série PDP. Posteriormente foi adaptado para uso em máquinas com sistema operacional UNIX (Linux, Solaris, etc.), mas ainda em terminal não gráfico.

As adapatações feitas por Maira P. Fernandes irão permitir a utilização do programa em terminais gráficos com geração da curva de regressão e apresentação "user-friendly" dos resultados numéricos. A distribuição das janelas de entrada e saida de dados é bastante intuitiva e permite ao usuário com conhecimento de cinética de reassociação a utilização do programa para diferentes fins, inclusive análise do tamanho de genomas candidatos a programas de seqüenciamento.

Creio que a disponibilização deste programa facilitará aos pesquisadores, principalmente na área de genômica de plantas, o uso do sistema de clonagem por fracionamento de Cot do DNA genômico.

Um um outro aspecto do programa com uma interface gráfica amigável é a possibilidade de sua utilização em aula para permitir que os alunos entendam, na prática, os passos envolvidos na determinação das constantes de reassociação.

O esforço bem sucedido da Maira em compreender as bases bioquímicas e moleculares que embasam o programa original, tornam o resultado final obtido por ela, ainda mais importante.

São Paulo, 04 de maio de 2005

Dr Carlos Eduardo Winter