IT 39

UNIVERSIDADE EDUARDO MONDLANE
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

TÍTULO: **PLANO ÓPTIMO DE MANUTENÇÃO DO SOFTWARE**

AUTOR: Elsa da Graça Zandamela Liçai

Maputo, Junho de 2000

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE LICENCIATURA

PLANO ÓPTIMO DE MANUTENÇÃO DO SOFTWARE

Supervisores : **dr. Mário Getimane**
eng.º José Grachane

Autor: **Elsa da Graça Zandamela Liçai**

DEDICATÓRIA

Dedico este trabalho aos meus pais que me trouxeram ao mundo, ao meu esposo e aos meus filhos pela compreensão durante longas horas de estudo.

Elsa G. Z. Liçai

AGRADECIMENTOS

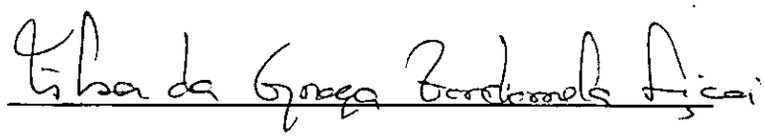
Ao dr. Mário Getimane e ao engº José Grachane, pela disposição e dedicação na supervisão desta tese.

Os meus agradecimentos estendem-se para os funcionários da biblioteca do DMI e para aquelas pessoas que directa ou indirectamente contribuíram para o efectivação do trabalho.

Elsa Z. G. Liçai

DECLARAÇÃO DE HONRA

Declaro por minha honra que este trabalho é fruto da minha própria investigação, e que o mesmo foi realizado para ser submetido como trabalho de **licenciatura em informática** na Universidade Eduardo Mondlane.



Elsa da Graça Zandamela Liçai

Maputo, Junho de 2000

RESUMO

O presente trabalho apresenta um plano óptimo de manutenção do software, onde o número de erros é uma variável aleatória que obedece a distribuição de poisson com média λ conhecida.
No modelo em estudo, supõe-se que o software é desenvolvido por uma equipa dentro de uma organização (In-House Software). Quando o usuário reporta a ocorrência de anomalias do software, o ideal seria identificar o erro, corrigi-lo imediatamente e definitivamente. Na perspectiva do gestor da organização, este não pode ser um procedimento apropriado em termos de análise de custos e benefícios. O gestor afirma que os custos de manutenção preventiva, passam a ser maiores que os custos de manutenção correctiva.

Pelo facto, o procedimento adequado é planificar as actividades de manutenção para o grupo de desenvolvimento do software.

Como solução do problema, o trabalho apresenta procedimentos para determinar o intervalo óptimo de manutenção do software e o número óptimo de manutenções de forma a minimizar os custos de manutenção.
Para efectivação do trabalho, emprega-se uma técnica denominada Programação Dinâmica (PD) e considera-se o modelo de controlo com um espaço finito de disturbâncias independentes e identicamente distribuídas onde o espaço de estados e acções é o conjunto IR^+.

Na primeira parte do trabalho descreve-se os conceitos básicos de PD, os modelos existentes em PD, a propriedade estrutural das soluções (monotonia, continuidade, etc.), bem como os conceitos sobre o modelo bayesiano que é empregue no presente estudo.

A segunda parte ocupa-se da simulação do modelo em estudo e verifica-se que a distribuição probabilística dos custos finais depende da média de erros numa fase de manutenção. Do estudo feito conclui-se que aplicando o modelo de controle bayesiano, ao fim de N períodos, a escolha do número óptimo de manutenções e os tempos óptimos depende do comportamento dos custo finais, pois, este é um dos principais parâmetros do modelo em estudo. Na prática é possível obter um custo mínimo desejável.

ÍNDICE

Capítulo I

1 Introdução

Definição 1.1

Programação Dinâmica é um método matemático orientado fundamentalmente para aumentar a eficácia dos cálculos em problemas de programação matemática *(Taha,1987).*

A ideia básica da programação dinâmica (PD) é a resolução do problema em etapas. Supõe-se que está-se perante um problema P, e em vez de resolvê-lo directamente, este é subdivido em n+1 subproblemas (p_0, p_1, p_2,...,p_n) fáceis de resolver. É de notar que a resolução dos subproblemas deve conduzir à solução do problema P. Supõe-se que a solução do subproblema p_0 é conhecida. O primeiro subproblema é resolvido na base da solução de p_0, de seguida o segundo subproblema p_2 é resolvido na base da solução de p_1 e assim sucessivamente até ao final do problema.

Observação:

Para resolver um problema P procede-se da seguinte maneira:

Assume-se que a solução do subproblema p_0 é conhecido.

Resolve-se o subproblema p_i dado as soluções dos subproblemas $p_0,p_1,p_2,\ldots,p_{n-i}$.

Obtida a solução do subproblema p_n, determina-se a solução do problema P.

Muitos processos de decisão multiestágios apresentam retornos (custos ou benefícios) associados a cada período de decisão. A PD adapta-se melhor à solução de problemas que exigem decisões que devem ser tomadas em sequência com a particularidade de que uma decisão tomada afectará as decisões futuras (Cox e Miller,1967).

A PD, baseia-se nos conceitos desenvolvidos por *Richard Bellman* e evoluiu como resultado do estudo de problemas de programação nos quais as decisões são tomadas ao longo do tempo, razão pela qual denominou-se *Programação Dinâmica*. O conceito é simples mas é difícil de aplicar, uma das dificuldades na aplicação é a falta de uma formulação mecânica na modelação

do problema. Como consequência cada problema exige uma estratégia própria de formulação. A formulação do problema assemelha-se à tentativa de solucionar um problema de probabilidade *(Shamblin,1988).*

A formulação de modelos em PD exige uma vasta experiência que se adquire com o tempo. Alguns autores afirmam que a PD ***é uma arte de encontrar um modelo apropriado*** *(Hinderer,1979).*

Apesar deste constragimento a PD pode ser usada em problemas lineares, não lineares, problema estocásticos, problemas que contém funções contínuas ou problemas de natureza discreta.

O ponto crucial ao se iniciar a resolução de um problema em PD, reside na definição dos estados. Uma definição conveniente diminui o tempo envolvido nos cálculos *(Ehrlich,1985).*

1.2 Objectivos do Trabalho

a) **Objectivos Gerais**:

- Conceber um plano para as actividades de manutenção do software.

b) **Objectivos Específicos:**

Desenvolver um algorítmo para:

- Calcular o intervalo de tempo óptimo para manutenção do software;
- Determinar o custo total mínimo esperado ao longo de todo processo da fase de manutenção;
- Verificar o valor óptimo de manutenções por período.

Capítulo II

2 Materiais e Métodos

Neste trabalho, usa-se a metodologia descritiva e comparativa que é baseada nos princípios de programação dinâmica. PD é uma forma de resolução de problemas que envolve processos de decisão por meio de uma estratégia óptima. Diversos problemas que envolvem custos (neste caso o objectivo é minimizar), ganhos (neste caso o objectivo é maximizar) e outros são resolvidos usando a Programação Dinâmica que envolve a subdivisão do problema em subproblemas mais simples de manipular *(Bertsekas,1987)*.

De forma a obter dados necessários, recorreu-se as fontes bibliográficas listadas no fim do trabalho, observações participativas e a realização de entrevistas semi-estruturadas com gestores das organizações de modo a obter-se informação mais detalhada do modelo em estudo. De referir que os conceitos iniciais sobre o estudo, foram colhidos por meio de palestras com o supervisor do trabalho.

Das entrevistas interactivas com vários supervisores de aplicações informáticas, constatou-se que a maior parte das organizações realizam a manutenção do software sem usar um plano e desenvolvem o software sem documentação. A prior parece não existir nenhum problema, pois, se ocorrer uma anomalia o essencial é corrigi-la de imediato e repor o funcionamento do sistema. O tempo que a equipe leva a fazer a correção de falhas é um tempo produtivo em relação as novas tarefas previamente orçamentadas (ex: a produção de uma nova versão do software). Uma vez que a equipe não produz as tarefas úteis e planificadas em termos de custos o resultado tem sido negativo.

Por outro lado, foi feita a observação participativa na empresa Dataserv vocacionada na manutenção de software e hardware, onde foi possivel verificar o grau de insatisfação do usuário no concernente a software específico, motivado pela falta de interação entre o cliente e o fornecedor acrescido do tempo de resposta que é muito elevado. Do ponto de vista do utilizador, logo que surge uma anomalia, esta deve ser imediatamente eliminada enquanto que na

prespectiva do gestor da organização, este não pode ser um procedimento apropriado em termos de análise de custos e benefícios. Pelo facto, o procedimento adequado é planificar as actividades de manutenção para o grupo de desenvolvimento do software *(Wee, 1990)*.

Para o efeito, o trabalho foi concebido seguindo uma estrutura do modelo bayesiano que se baseia no uso de probabilidade a prior e permite simular dados de sistemas reais. Este modelo, é muito usado nas teorias de decisão, em problemas com uma certa ambiguidade, problemas grandes e complexos. O mesmo é vastamente aplicado na medicina, engenharia genética, climatologia, informática e em mais outras áreas. Quando se emprega o modelo bayesiano, considera-se a prior certos valores e usa-se a probabilidade para quantificar os valores desconhecidos, neste caso os dados conhecidos são fixos e os desconhecidos são aleatórios. O modelo de Bayes requer uma informção a prior quantitativa para se obter a informação a posterior *(IEEE Computer Society, 1997)*.

O desenvolvimento do modelo para a concepcão do plano óptimo de manutenção do software, comporta duas fases. A primeira é o estudo das propriedades estruturais da solução (continuidade, monotomia, suposições do modelo) baseiadas no modelo bayesiano. A segunda parte é composta pelas suposições do modelo, definição dos parámetros do modelo com base na definição matemática do modelo bayesiano. O número de erros é distribuido segundo a distribuição de Poisson com média λ o espaço de acções é o conjunto IR^+ *(Wee ,1990)*.

O algorítimo é concebido com base nos teoremas e colorários definidos na propriedade estrutural das soluções e os resultados numéricos são obtidos usando a linguangem de programação Pascal versão 6.0.

Capítulo III

3 Resultados

O objectivo da PD é a determinação de uma sequência óptima de acções. Em PD existem dois modelos, ***Modelo Determinístico,*** e ***Modelo Estocástico*** que podem ser estacionários ou não estacionários.

3.1 Modelo Determinístico Estacionário em Programação Dinâmica (MEDPD)

Este tipo de modelo foi desenvolvido para a resolução de problemas em que o resultado de uma acção é completamente previsível. O objectivo principal da programação dinâmica é maximizar ou minimizar uma função de N variáveis *(Cox e Miller,1967).*
Considere os seguintes exemplos:

Exemplo 3.1:

Têm-se um determinado capital $K \in IR_+$ que deve ser usado ao longo de um determinado número de períodos (a este, dá-se o nome de ***horizonte de planeamento***) sobre os quais deve-se tomar decisões de forma a obter um ganho máximo. Assume-se que o consumo de b unidades de K ($b \in IR_+$) resulta num ganho denotado por $r(b) \in IR_+$. Devido ao facto de se trabalhar com o tempo, o factor de desconto é tomado em consideração. O problema resume-se em como investir o capital no início de cada período de forma a maximizar a soma dos ganhos por período? Qual é o ponto máximo se existir ou qual é o supremo?

Matematicamente têm-se:

$$\sum_{i=0}^{N-1} r(b_i) - - - - - - > \max \qquad \text{onde:} \qquad \text{N = Número de períodos} \qquad \textbf{(3.1)}$$

r= Ganho *(Hinderer, 1993).*

Exemplo 3.2:

Considere uma rede simples com quatro nós S_1,S_2,S_3,S_4. Os nós representam cidades e os arcos representam alternativas para a estrada projectada que liga os pontos S_n e S_{n+1}, $1<=n<=3$. Os números nos arcos (veja fig.3.1) representam os custos de construção da correspondente ligação. O problema é encontrar o caminho com o custo mínimo de construção *(Hinderer, 1993).*

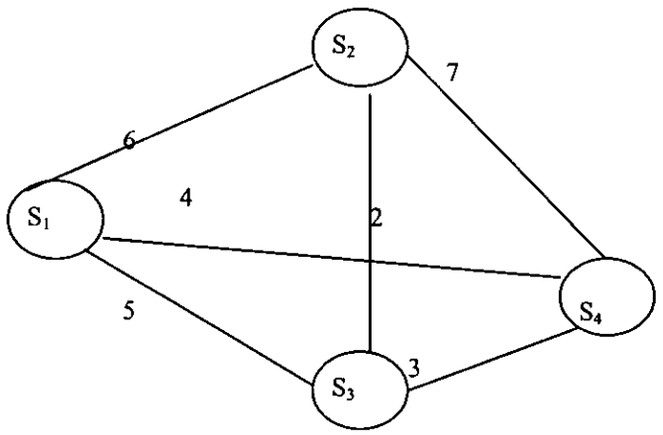


Fig. 3.1

Analizando os dois exemplos, tiram-se as seguintes observações:

1. Durante um determinado número N de períodos chamado **horizonte de planeamento**, o sistema inicia no estado $s_0 \in S$ designado **estado inicial,** é chamado **espaço de estados**. O sistema move-se por meio de uma variável de estado e sob influência de acções tomadas em cada período n=0,1...,N-1. $N \in A$ e A é designado **espaço de acções**.

   As acções determinam o movimento do sistema. Nos exemplos anteriores as acções seriam o capital investido ou a estrada escolhida *(Hinderer, 1993).*

Em geral se o sistema estiver no estado s, nem todas as acções do espaço de acções A podem ser tomadas. Somente as acções pertencentes a um subconjunto D(s) podem ser tomadas. D(s), é designado **conjunto de acções admissíveis no estado s**, e D:é o **conjunto restrição** . Matematicamente têm –se $D := \{(s,a) \in S \times A : a \in D(s)\}$ *(Hinderer, 1993).*

3. A transição de um estado para o outro é descrita por uma funçãoT:D->S chamada **função de transição** e tem o seguinte significado:
Se no período n o sistema estiver no estado s_n e a acção $a_n \in D(s_n)$ for tomada, então o sistema move-se para um outro estado $s_{n+1} := T(s_n, a_n)$ *(Hinderer, 1993).*

4 No fim do processo de tomada de decisão, obtém-se o ganho terminal ou perda terminal que depende do estado terminal *(Hinderer, 1993).*

Para invetigação do tipo de problemas anteriormente ilustrados nos dois exemplos, torna-se necessário introduzir modelos matemáticos adequados. A seguinte definição introduz o modelo estacionário determinístico em PD.

Definição 3.1

O modelo estacionário deteminístico em programação dinâmica *(MEDPD) é uma sequência* (S,A,D,T,r,V_0,β) *onde:*

S - é conjunto de estados, (usa-se s para designação do estado genérico).

A - conjunto de todas acções possíveis.

D – conjunto restrição, $D \subset SxA$.

D(s) - conjunto de acções admissíveis no estado s.

T - função de transição, $T: D \rightarrow S$.

r - função do ganho, $T: D \rightarrow S$.

V_0 *- ganho terminal,* $V : S \rightarrow IR$.

β *- factor de desconto. Na maioria das aplicações, este é considerado menor do que 1 (Hinderer, 1993).*

Na definição formal não se incluiu o estado inicial e o horizonte, pois no geral o problema é considerado simultaneamente para todos os estados iniciais $s_0 \in S$ e todos horizontes N $\in$ IN.

Portanto dado um problema estacionário prático, o primeiro passo para a solução é modelá-lo por meio de uma escolha apropriada de S,A,T,r,V_0,β. A escolha de S é muito importante por vezes podem existir várias alternativas e a escolha da melhor depende da habilidade do modelador. O procedimento de modelação deve ser acompanhado de reflexões acerca das propriedades do modelo.

Existem duas formulações de problemas de optimização de um MEDPD. A primeira formulação é relativa as acções e a segunda é em termos de políticas *(Hinderer, 1993).*

3.1.1 Formulação do Problema de Optimização de MEDPD em Termos de Acções

Dado um estado inicial s_0 arbitrário, diz-se que a sequência de acções $(a_1,a_2,\ldots,a_{N-1}) \in A^N$ é admissível para s_0 se:

$a_0 \in D(s_0)$

$a_1 \in D(s_1)$, *onde* $s_1 := T(s_0,a_0)$

$a_2 \in D(s_2)$, *onde* $s_2 := T(s_1,a_1)$

……

$a_{N-1} \in D(s_{N-1})$ *onde* $s_{N-1} := T(s_{N-2},a_{N-2})$

Definição 3.2

A sequência $(s_1,s_2,\ldots,s_N)$ *é chamada* ***sequência de estados gerados pelo estado inicial*** s_0 *e sequência de acções admissíveis* $y := (a_n)_0^{N-1}$ *(Hinderer, 1993).*

Quando se pretende enfatizar a sequência de acções de y escreve-se (s_{ny}) no lugar de (s_n).

O conjunto de acções admissíveis para o estado s_0 é denotada por $A^N(s_0)$. A equação

$s_{n+1} := \mathrm{T}(s_n, a_n)$ $\quad 0 \le n \le N-1$ que para um dado s_0 determina sequência de acções $(a_n)_0^{N-1} \in A^N(s_0)$ é chamada por **equação do sistema**.

Para cada estado $s_0 \in S$ e cada sequência de acções, $y := (a_n)_0^{N-1} \in A^N(s_0)$ o ganho total no período é definido pela equação:

$$V_{Ny}(s_0) := \sum_{i:0}^{N-1} \beta^n r(s_n, a_n) + \beta^N V_0(s_N) \qquad \textbf{(3.2)}$$

Onde $(s_n)_1^N = (s_{ny}(s_0))_1^N$ é a sequência de estados gerados por s_0 e y.

A expressão 3.2 deve ser maximizada, pois constitui o ganho. Assim o problema de optimização de um MEDPD em termos de acções tem a seguinte formulação:

1) Achar Sup $\{V_{Ny}(s_0) : y \in A^N(s_o)\}$.

2) Achar o ponto de máximo de $y \to V_{NY}(s_0)$.

3.1.2 Formulação do Problema de Optimização de MEDPD em Termos de Políticas

A numeração do período é feita no sentido da esquerda para direita. Na resolução do problema, parte-se do último período para o período inicial. A esta numeração chama-se **estágio.** O conceito de estágio está relacionado com as acções óptimas (a^*_3, a^*_2, a^*_1), tomadas em cada período . Portanto, a^*_2 é a decisão tomada no estágio 2, quando falta um período para o processo terminar.

As políticas são na prática acções, só que nas acções trabalha-se com períodos de decisão enquanto que nas políticas trabalha-se com estágios.

A formulação do problema de optimização de MEDPD em termos de acções está directamente ligada ao estado inicial do sistema. Se um novo estado inicial for tomado em consideração, será

necessário repetir todo o processo de cálculo. O ideal é ter uma formulação independente do estado inicial. Pelo facto surge o que é designado por **regras de decisão** *(Hinderer, 1993).*

Defenição 3.3

1) A função $\Phi : S \to A$ *tal que* $\Phi_n(s) \in D(s)$ *chama-se* ***regra de eecisão (rd)*** *para todo* $s \in S$.

2) A sequência $\pi = (\Phi_0, \Phi_1, ..., \Phi_{N-1})$ *de regras de decisão é designado por* ***política de N estágios*** *(Hinderer, 1993).*

O conjunto de todas regras de decisão é denotado por ***F; F^N*** denota o **conjunto de todas políticas para N estágios** e não depende do estado inicial.

A condição $\Phi(s) \in D(s)$ garante que ao se usar uma política, para cada período e para cada estado seleciona-se apenas acções admissíveis para o estado em consideração.

Se o sistema estiver no estado inicial s_o e a politica $\pi = (\Phi_n)_0^{N-1} \in F^N$ for usada então o sistema move-se sucessivamente pelos seguintes estados:

$$s_1 := T(s_0, \Phi_0(s_0))$$
$$s_2 := T(s_1, \Phi_1(s_1))$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$s_N := T(s_{N-1}, \Phi_{N-1}(s_{N-1}))$$

Observação:

Sabe-se que s_n, $1 \leq n \leq N$ depende de s_0 e de $\pi := s_n = s_{n\pi}(s_0)$. Deste modo a equação do estado assume a seguinte forma:

$$s_{n+1} := T(s_n, \Phi_n(s_n))$$

(3.3)

$(s_n)_1^N$ é a sequência de estados gerado por s_0 e pela politica π denotado por $\left(\Phi_n(S_n)\right)_n^{N-1}$.

(Hinderer1993).

Associado ao desenvolvimento do sistema introduz-se uma função que dá o ganho total de N estágios definida da seguinte forma:

$$V_{N\pi}(s_o) := \sum_{i=0}^{N-1} \beta^n r\left(s_n, \Phi_n(s_n)\right) + \beta^N V_0(s_N) \qquad \textbf{(3.4)}$$

A expressão 3.4 deve ser maximizada, pois constitui o **ganho total de N estágios sob a política** π **e estado inicial** s_0 V_N é designada **função de valor de N períodos** . O problema de optimização de um MEDPD em termos de políticas tem a seguinte formulação:

1) Achar o Sup $\{ V_{N\pi}(s_0) : \pi \in F^N \}$.
2) Achar se possível o ponto de máximo da função $\pi \rightarrow V_{N\pi}(s_0)$.
3) Achar a *política óptima*, isto é, a política que é óptima para cada estado inicial s_0.

De facto existe uma relação entre as duas formulações, segundo a seguinte proposição:

Proposição 3.1

Seja s_0 um estado inicial arbitrário.

1) Tem-se:

$$Sup\{V_{Ny}(s_0) : y \in A^N(s_o)\} = Sup\{V_{N\pi}(s_0) : F \in F^N\} := V_N(s_0) \qquad \textbf{(3.5)}$$

2) Se π^* é uma política de N estágios e se y^* é uma sequência de acções geradas por s_0 e π^*, então π^* é s_0-optimal se e somente se y^* é s_0-optimal. (Hinderer, 1993)

Observações:

O conceito de sequência de acções está directamente ligado com um estado inicial específico e uma sequência de estados, ao passo que o conceito de política não depende do estado inicial. O método fundamental de solução requer que todas as vezes excepto o período 0, todos os estados estejam incluidos na pesquisa da acção óptima. Como consequência, os resultados teóricos são frequentemente fáceis de formular em termos de políticas do que em termos de sequência de acções. Por outro lado a solução de um problema prático é usualmente formulado em termos de sequência de acções.

> Maximizar sobre um conjunto de sequência de acções pode ser à primeira vista mais fácil do que maximizar sobre um conjunto de políticas. O primeiro comporta-se como uma sequência de pontos e o segundo comporta-se como uma sequência finita de funções $\Phi : S \rightarrow A$. Isto é reforçado pelo facto de o conjunto de todas as regras F^N ter uma cardinalidade enorme devido a estrutura que é um produto cartesiano.

Está-se livre de usar a formulação que mais se adeque ao problema em estudo.

3.2 Modelo Estocástico em Programação Dinâmica (MEPD)

Nesta secção detalha-se apenas o modelo de controlo com disturbância aleatória independente e identicamente distribuída (i.i.d). A transição do estado s_n para s_{n+1} é especificada por uma função de transição T que é disturbada por uma variável aleatório X_{n+1} que toma valores num conjunto M finito designado **conjunto de disturbância.** *(Hinderer, 1993).*

Pode-se assumir que os $X_{i's}$ estão definidos no espaço de probabilidade P finito (Ω, P). A regra de transição para o modelo estocástico é a seguinte:

$$T: DxM \rightarrow S \qquad \textbf{(3.6)}$$

Esta função pode ser interpretada do seguinte modo:
Se no período de tempo n o sistema estiver no estado s_n , uma acção a_n fôr tomada e se a pertubância X_{n+1} assumir o valor x_{n+1} então o sistema move-se para um novo estado aleatório.

$$s_{n+1} := T(s_n, a_n, x_{n+1}). \qquad 0 \leq n \leq N-1 \qquad \textbf{(3.7)}$$

No modelo estocástico o resultado de cada decisão é desconhecido, pois este é influenciado por um processo aleatório.

Dada uma politíca de N estágios $\pi = (\Phi_n)_0^{N-1} \in F^N$ na óptica do modelo estocástico o estado do sistema no período n governado pela política π e iniciado em s_0, é uma variável aleatória $\varsigma_{n\pi}$ sobre (Ω, P) definida recursivamente pela equação diferencial estocástica.

$$s_{n+1,\pi} := T_{\Phi n}(\varsigma_{\pi n}, x_{n+1}) \qquad \text{com a condição de } \varsigma_{0\pi} \equiv s, \tag{3.8}$$

Onde:

$$T_{\Phi}(s,x) := T(s,\Phi(s),x) \qquad \Phi \in F \qquad \textit{(Hinderer, 1993).} \tag{3.9}$$

Observação:

1) $\varsigma_{n\pi}$ é uma variável aleatória e não depende de $X_{n+1},\ldots,X_N$. Consequentemente, $\varsigma_{n\pi}$ e X_{n+1} são estocásticamente independentes.
2) Para s e π fixos $\varsigma_{n\pi}$ assume valores finitos, pois, M é finito.

Exemplo 3.3:

Uma empresa D vocacionada na venda de computadores deseja saber a probabilidade de vender X computadores num determinado período.

Para a resolução do problema, os seguintes factores devem ser tomados em consideração:

Informação sobre habitantes existentes.

.Informaçao sobre o poder de compra.

Cada um destes factores tem uma contribuição na solução do problema. Considerando que existe uma informaçao estatística, os factores em causa que constituem parámetros do problema, podem ser obtidos.

De seguida introduz-se a definição do modelo aplicável a situações tais como a descrita no exemlpo 3.3.

Defenição 3.4

*O **modelo de controlo (CM)** com disturbância independente e identicamente distribuida (i.i.d) e com um espaço de disturbância finito, é uma sequência (S,A,D,M,Q, T ,r,V_0, β) onde :*

1. S,A,D,r,V_0, β têm o mesmo significado que no modelo determinístico;

2. M é um conjunto finito de valores de disturbância;

3. Q é a funçao de probabilidade das disturbâncias i.i.d;

4. T *é uma função de transição estocástica,* T: *DxM* $\rightarrow$ *S (Hinderer, 1993).*

Se no período de tempo n, o sistema estiver no estado s_n, uma acção a_n for tomada e ocorrer uma disturbância x_{n+1} então obtém-se um ganho $r(s_n, a_n, x_{n+1})$. O ganho de um período é o valor da esperança matemática do ganho $r(s_n, a_n, X)$, isto é:

$$r(s,a) := Er(s,a,X) \text{ (Hinderer, 1993).} \qquad \textbf{(3.10)}$$

Para cada estado inicial s_o e para cada acção da política $\pi = (\Phi_n)_0^{N-1}$, têm-se um **ganho aleatório de N-estágios** :

$$R_{N\pi}(s,Y) := \sum_{i=0}^{N-1} \beta^n r(\varsigma_{n\pi}, \Phi_n(\varsigma_{n\pi})) + \beta^N V_0(\varsigma_{N\pi}) \qquad \textbf{(3.11)}$$

onde, $Y := (X_i)_1^N$. A esperança matemática do ganho de n estágios é um número real tal que:

$$V_{N\pi}(s) := ER_{N\pi}(s,Y) := \sum_{y \in M^N} R_{N\pi}(s,y) P(Y = y) \qquad \textbf{(3.12)}$$

Analogamente ao modelo determinístico, o valor da função de N estágios $V_N : S \rightarrow (-\infty, +\infty]$ é definido por:

$$S \quad V_N(s) := Sup\{V_{N\pi}(s) : \pi \quad F^N\} \qquad N \in IN$$

(3.13)

A noção de política de N estágios e as regras de decisão F e F^N são análogas ao modelo determinístico *(Hinderer, 1993).*

3.2.2 Teoria Básica para Minimização de Custos para MEPD

A função de custos mínimos C_n satisfaz o valor de interação na forma:

$$C_n(s) = \inf[-r(s,a) + \beta C_{n-1}(T(s,a))] =$$
$$= \inf[-r_f(s) + \beta C_{n-1}(T_f(s)] \qquad n \in IN, s \in S \qquad (3.14)$$

Se para $0 \le n \le N$, f_n é o minimizador no estágio n, então $(f_n)_1^N$ é a política óptima e é critério de optimalidade.

Os custos de um estágio são denotados por –r(s,a), os custos terminais por $C_0(s):=-V_0(s)$. O custo total mínimo para um problema de n-estágio é obtido pela formula na expressão 3.14. Em qualquer problema de minimização de custos, o primeiro passo para resolução do problema é a identificação dos custos.

3.3 Modelo de Controlo com Segurança

No modelo de controlo anteriormente visto, o parámetro na lei de distribuição Q é conhecido. kVeja-se o seguinte exemplo :

Exemplo 3.3

Uma empresa C vocacionada na produção de peças, tem no início de cada período c_1 peças no máximo e pode armazenar c_2 peças. A procura de peças $x_n \le c_1$ é aleatória e deve ser satisfeita. O custo para encomendar a peça a:a>0 é de e_1+e_2*a e o custo de armazenar s peças ao fim de um determinado período é e_3*s unidades monetárias. Um stock final de s_n peças tem um valor de e_4*s_n unidades monetárias. O probolema reside em encontrar um plano de encomedas que minimize a soma dos custos de encomenda de armazenagem e do valor de liquidação.

Modelando o problema tem-se:

$S=\{0,1,2,\ldots,c_2\}$

$A=\{0,1,2,\ldots,c_1\}$

Uma vez que a procura deve ser satisfeita, o novo stock será:

$s_{n+1} = T(s_n, a_n, x_n) := s_n + a_n - x_n$ deve satisfazer a restrição $0 \le s_{n+1} \le c_2$

Onde:

s_n stock existente no início do período

a_n quantidade encomendada

x_n procura de mercadoria. *(Hinderer, 1993)*

x_n é uma realização de uma variável aleatória X_n e que é caracterizada por uma lei de distribuição $p(x_n, \theta)$, onde θ tem as seguinte variantes:

1 θ conhecido, segundo visto no modelo já introduzido.

2 θ não é conhecido.

Para o o segundo caso, introduz-se o seguinte modelo:

3.3.1 Interpretação dos Modelos de Controlo Com Segurança

Seja $\theta \in \Theta$ no instante $n \in \mathbb{N}$. Estando-se no estado $s_n \in S$ e tomando-se a acção $a_n \in D \subset S_n$ tem-se um rendimento $r(\theta, s_n, a_n)$ e uma disturbância Z_n que é governada por $Q(\theta, s_n, a_n)$.

No geral tem-se:

$$s_n = T(s_{n-1}, a_{n-1}, z_{n-1}) \qquad \textbf{(3.15)}$$

Observação: a probabilidade de Z_n depende de θ.

Analogamente define-se o modelo de controlo com insegurança.

Definição 3.5

Considere o modelo de controlo $(S,A,D,T,Z,Q,r,V_O,\beta)$, onde, q - função de densidade, M - espaço de disturbância. Um ***modelo de controlo com insegurança*** *é uma sequência $(S,A,D,T,Z,\theta,Q,r,V_o,\beta)$, onde:*

1 Q é lei de distribuição de disturbâncias, que neste caso é influenciado pelo parâmetro $\theta \in \Theta$.

2. Z espaço de disturbância (Hinderer, 1993).

Para resoluçao de problemas que envolvem modelos de controlo com insegurança, torna-se necessário a descrição da história do sistema.

Definição 3.6

A história do sistema no instante 0 é o próprio S, isto é o seguinte:

$H_o := S$

$H_1 := SxAxZxH_o$

.....

$H_n := SxAxZxH_{n-1}$ *; Z é o espaço das disturbâncias (Hinderer, 1993).*

Em suma para $n \in IN$, $h_n=(s_0,a_0,z_0,s_1,a_1,z_1,\ldots,s_n) \in H_n$ chama-se ***história do sistema no instante n*** *(Hinderer, 1993).*

À semelhança dos outros modelos, introduz-se a seguinte definição:

Definição 3.7

*Uma **regra de decisão** é uma função $f_n: h \rightarrow A$ com $f_n(h_n) \in D(s_n)$, onde f_n é o conjunto de todas as regras de decisão no instante n (Hinderer, 1993).*

De igual modo, define-se política da seguinte forma:

Definição 3.8

*Uma **Política** π é uma sequência de regras de decisão ($\pi = (f_n)$). $\Delta := F_0xF_1xF_2x\ldots$ é o **conjunto de todas as políticas** π (Hinderer, 1993).*

Uma vez que é deconhecido o parâmetro θ, a lei de distribuiçaõ torna-se inadequada. Para contornar esta situação, introduz-se o modelo de bayes.

3.4 Princípio do Modelo Bayesiano

Assume-se que o parámetro θ é uma grandeza aleatória que obdece uma distribuição que não é conhecida. Para iniciar-se o estudo assume-se que θ obdece uma distribuiçaõ μ_0, distribuição a prior. À semelhança do modelo de controlo com insegurança introduz-se a seguinte definição:

Definição 3.9

Modelo de controlo bayesiano *é uma sequência* $(S,A,D,Z,\ \Theta\ ,\mu_0,T,Q,r,V_0,\beta)$, *onde:*

1. μ_0 *é distribuição a prior;*
2. *Os restantes parámetros têm o mesmo significado que no modelo de controlo com insegurança (Hartigan,1983).*

Analogamente ao modelo de controlo com insegurança, define-se a seguinte política.

Definição 3.10

Uma política $\pi^* \in \Delta$ *chama-se* ***política de Bayes optimal*** *em relação a* μ_0 *caso ela seja o ponto de máximo da função:*

$$\pi \to \int V_{\infty\pi}^{\theta}(s)\mu_0(d\theta)$$

E para todos os $s \in S$, assume-se que as probabilidades tem função de densidade.

3.4.1 Modelo de Controlo de Bayes

Definição 3.11

$V_{\infty\pi}(s) := \int V^{\theta}_{\infty\pi}(s)\mu_0(d\theta)$ ***é o ganho total de bayes com horizonte infinito.***

Usando a política π *e começando em S, têm-se* $V_{\infty}(s) := \sup_{\pi\in A} V_{\infty\pi}(s)$ *que é o* ***ganho maximal de Bayes*** *(Hartigan,1983).*

Definição 3.12

Uma política $\pi \in \Delta$ *chama-se política optimal Bayesiana em relação a* μ_0 *se* $V_{\infty\pi} = V\infty(s)$ *(Hartigan,1983).*

Definição 3.13

Uma função $\Phi : SxP(\theta)xAxZ \to P(\theta)$: $\Phi(s,\rho,a,z)(B) = \begin{cases} \dfrac{\int_{\beta} q(z/\theta,s,a)\rho(d(\theta))}{\int_{\theta} q(z/\theta;s,a)\rho(d\theta^{\cdot\cdot})} \\ \rho(B) \end{cases}$ *chama-se* ***operador de bayes ou de actualização*** *onde:*

$P(\theta)$ *é o conjunto de todas as distribuições a prior que podemos escolher.*

ρ *é uma distribuição a prior.*

Q está relacionado com densidade e é a função das disturbâncias.

$B : (\mathrm{B} \subset \Theta)$.

A sequência das distribuições no instante n é dado por:

$\mu_0(h_0;.) := \mu_0$.

$\mu_{n+1}(h_n, a_n, z_n, s_{n+1};.) := \Phi(s_n, \mu_n(h_n;.); a_n, z_n)$ *é a distribuição à posterior (Hartigan,1983).*

Para h_n fixo, $\mu_n(h_n;.)$ é uma probabilidade sobre θ. Quando se escreve $\mu_n(h_n;B)$ pressupõe-se que $B \subset \Theta$. $\mu_n(h_n;B)$ é a probabilidade de que o prámetro θ no instante n está em B na condição de ter-se uma história h_n.

Observações:

1. $\mu_n(h_n;.)$ não depende de $s_n \in S$
2. $\mu_n(h_n;.)$ só depende das disturbâncias $z_0, z_1, \ldots, z_{n-1}$.

O lemma seguinte permite calcular os μ_n com base nos dados do modelo.

Lemma 3.1

Seja $B \subset \theta$, $n \in IN$, então temos a seguinte expressão:

$$\mu_n(h_n;B) := \frac{\int_B \prod_{\gamma}^{n-1} q(z_\gamma/\theta, s_\gamma, a_\gamma)\mu_0(d\theta)}{\int_\theta \prod_{\gamma=0}^{n-1} q(z_\gamma/\theta, s_\gamma, a_\gamma)\mu_0(d\theta)}$$

onde:

$q(z_\gamma/\theta, s_\gamma, a_\gamma) \rangle 0$

q-função de distribuição

$q(z_\gamma/\theta, s_\gamma, a_\gamma)$ *é a história do processo.*

μ_0 *é a distribuição a prior (Hartigan,1983).*

3.5 Suposições do Modelo

Um dos objectivos do modelo em estudo é a minimização dos custos de manutenção, para tal, é necessário identificá-los. Em cada actividade de manutenção tem-se o custo por correção de falha denotado por C e o custo médio por actividade de manutenção denotado por K. As suposições do modelo são as seguintes:

- O número desconhecido de erros no software, N, é aleatório e segue a distribuição de Poisson com média λ conhecida.
- Dado N, cada erro é susceptível de provocar falhas independentemente dos outros erros com taxa desconhecida θ=1.
- O custo por cada falha é C.
- A média de custo por actividade de manutenção é K.
- O factor de desconto é α, o valor actual por unidade de custo que ocorre no tempo T é $e^{\alpha T}$.
- Assume-se que λ é a variável de estado, pois é a média de erros e sendo assim, pode-se exercer um certo controlo
- Assume-se que todas as falhas que ocorrem num determinado período são eliminadas quando sujeitas a manutenção.
- Para construção do modelo as seguintes funções são definidas:

$V(\lambda)$ - Total mínimo de custos esperados;

$T(\lambda)$ - Próximo período óptimo para realizar a actividade de manutenção;

$M(\lambda)$ - Número total óptimo de manutenção.

3.5.1 Propriedade Estrutural das Soluções

O teorema que se segue, ilustra a fórmula para o cálculo do custo total mínimo em toda fase de manutenção

Teorema 3.1

$$V(\lambda) = \min_{0 \le t \le \infty} \left\{ \frac{C\lambda}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V\left(\lambda e^{-t}\right)\right] \right\}$$

Onde:

C - custo por falha

λ - média de erros

K - custo por actividade de manutenção

α ; $\left(1 - e^{-\alpha t}\right)$ - *factor de desconto*

t - tempo ao longo da fase de manutenção

$V\left(\lambda e^{-t}\right)$ - *o custo mínimo da correcção anterior (Wee, 1988).*

Demonstração:

Aplicando limite quando $t \rightarrow \infty$ tem-se:

$\lim_{t \to \infty} V(\lambda) = \frac{C\lambda}{\alpha}$, pois, quanto mais tempo durar uma actividade de manutenção, C λ que é o custo médio para corrigir todos erros numa actividade de manutenção torna-se maior que o custo médio K. Portanto o resultado do limite é o custo efectivo da actividade.

c.q.d.

Observações: Na fórmula do teorema, t deverá ser apenas positivo e nunca igual a zero (t>0), pois se t for igual a 0, estar-se-ia no tempo inicial e neste período apenas existe o custo médio por actividade de manutenção.

O teorema que se segue é importante do ponto de vista de análise de custos e benefícios, pois faz uma análise comparativa de médias de erros em dois softwares similares.

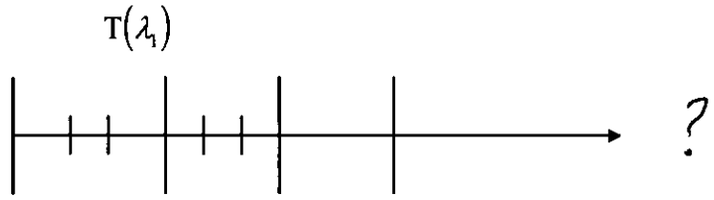
Teorema 3.2

Se $\lambda_1 < \lambda_2$, *então* $\lambda_1 e^{-T(\lambda_1)} \leq \lambda_2 e^{-T(\lambda_2)}$ *(Wee, 1988).*

Demonstração:

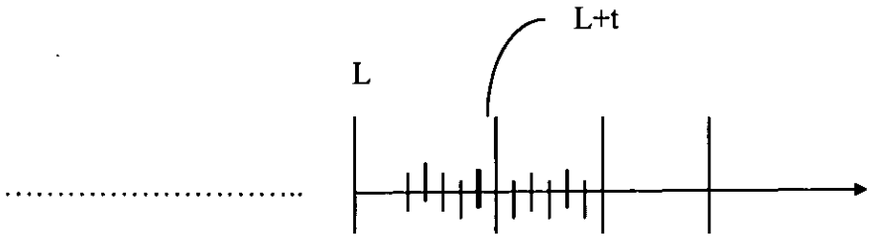
Supõe-se que se tem dois softwares com diferentes médias de erros λ_1 e λ_2 . Seja $\lambda_1^* = \lambda_1 e^{-T(\lambda_1)} \lambda_1^*$ é a média dos erros restantes após λ_1, erros terem sido corrigidos. Para o segundo software supõe-se a manutenção do mesmo é feita no peródo de tempo L (após manutenção do 1° software) tal que $\lambda_1^* = \lambda_2 e^{-L}$.

Considerando os custos totais mínimos no 1° software e 2° software repectivamente fig.3.1 e fig.3.2, têm-se:



(Fig.3.1)

média de erros λ_1 média de erros $\lambda_1^* = \lambda_1 e^{-T(\lambda_1)}$



(Fig. 3.2)

média de erros λ_2 média de erros $\lambda_1^* = \lambda_2 e^{-L}$

$$V(\lambda_1) = \min_{0<t\leq\infty} \left\{ \frac{C\lambda_1}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V\left(\lambda_1^* e^{-t}\right)\right] \right\}$$

$$V(\lambda_2) = \min_{0<t\leq\infty} \left\{ \frac{C\lambda_2}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V\left(\lambda_1^* e^{-t}\right)\right] \right\}$$

Uma vez assumido que $\lambda_1 < \lambda_2$, $C\lambda_1 < C\lambda_2$, pois a média de erros será no 2º software maior logo:

$$\left\{ \frac{C\lambda_1}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V\left(\lambda_1^* e^{-t}\right)\right] \right\} < \left\{ \frac{C\lambda_2}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V\left(\lambda_1^* e^{-t}\right)\right] \right\}$$

Considere os custos para λ_1^* nos 1º e 2º software respectivamente:

$$V(\lambda_1^*) = \min_{0<t\leq\infty} \left\{ \frac{C\lambda_1^*}{\alpha}\left(1 - e^{-\alpha t}\right) + e^{-\alpha t}[K + B] \right\}$$ onde B = custo mínimo do erro anterior.

$$V(\lambda_1^*) = \min_{0<S\leq\infty} \left\{ \frac{C\lambda_1^*}{\alpha}\left(1 - e^{-L}\right) + e^{-L}[K + D] \right\}$$ onde D = custo mínimo do erro actual.

Recorde-se que foi assumido que $\lambda_1^* = \lambda_1 e^{-T(\lambda_1)}$ e $\lambda_1^* = \lambda_2 e^{-L}$, mas supõe-se que enquanto a correcção no 1º software termina, no segundo software existe pelo menos um período L+t em que ocorre a correcção de um erro, logo :

$$\lambda_1 e^{-T(\lambda_1)} \leq \lambda_2 e^{-T(\lambda_2)}$$

c.q.d.

Considera-se apenas um número de manutenções finito designado por n. Nestas condições, seja, $V_n(\lambda), T_n(\lambda)$, $M_n(\lambda)$, respectivamente o total de custos esperados, o próximo período de manutenção, o número óptimo de manutenções em toda fase de manutenção. Note que $M_n(\lambda) = n$, pois, o n representa o número de manutenções.

As fórmulas apresentadas nos dois seguintes lemmas sem demonstração, ilustram a interação dos custos nos diferentes estados e as formulas explícitas para o modelo em estudo.

Lemma 3.2

$$V_n(\lambda) = \min_{0\ t} \frac{C\lambda}{(1 \quad e^{\alpha t}) + e^{\alpha t}\left[K + V_{n-1}(\lambda e^{t})\right]} \quad para\ n \geq 1,$$

$$V(\lambda) = \lim_{n \to \infty} V_n(\lambda) \ (Wee,\ 1988).$$

Para n muito grande a expressão do teorema tende a crescer formando uma sucessão daí que aplicado o o limite. Este limite .existe e pode ser infinito

c.q.d

No lemma que se segue, o custo inicial no sistema é efectivamente o custo por falha, pois, no período inicial o custo médio por actividade de manutenção não é usado. Estando-se no período inicial, o período óptimo para a próxima manutenção é imprevisível, pois este parámetro é perturbado pelo número de erros definidos aleatoriamente.

Lemma 3.3

$$V_0(\lambda) = \frac{C\lambda}{\alpha} \quad , \quad T_0(\lambda) = \infty \quad M_0(\lambda)=0 \text{ (Wee, 1988).}$$

No lemma 3.3, há uma necessidade de avaliar dois casos possíveis:
Considere-se a realização de 0 manutenções, no estado inicial o custo efectivo é $C\lambda$ aplicada o respectivo factor de desconto consequentemente o próximo período óptimo para realizar a manutenção é imprevisível.

O outro caso é a realização de pelo menos uma manutenção. Então o próximo período óptimo de manutenção deixa de ser imprevisível e os custos envolvem também o custo médio por actividade de manutenção (veja o seguinte lemma).

Lemma 3.4

As fórmulas explícitas de $V_1(\lambda)$, $M_1(\lambda)$, $T_1(\lambda)$ *são as seguintes (Wee,1988):*

$$M_1(\lambda) = \begin{cases} 0 \\ 1 \end{cases} \quad , \; T_1(\lambda) = \begin{cases} \infty, \; se \;\; 0 < \lambda \le \dfrac{\alpha K}{C} \\ \ln\left(\dfrac{C\lambda}{C\lambda - \alpha K}\dfrac{\alpha+1}{\alpha}\right), \; se \;\; \dfrac{\alpha K}{C} < \lambda \end{cases}$$

$$V_1(\lambda) = \begin{cases} \dfrac{C\lambda}{\alpha}, \; se \;\; 0 < \lambda \le \dfrac{\alpha K}{C} \\ \dfrac{C\lambda}{\alpha} - \dfrac{\alpha^{\alpha-1}}{(\alpha+1)^{\alpha+1}}\dfrac{(C\lambda - \alpha K)}{C\lambda^{\alpha}}, \; se \;\; \dfrac{\alpha K}{C} < \lambda \end{cases}$$

(Wee, 1988).

Demonstração:
Partindo do lemma 3.2, tem-se:

$$V_1(\lambda) = \min_{0 \le t \le \infty} \left\{ \frac{C\lambda}{\alpha} \left(1 - e^{-\alpha t}\right) + e^{-\alpha t}\left[K + V_0\left(\lambda e^{-t}\right)\right] \right\}$$

$$V_1(\lambda) = \min_{0 \le t \le \infty} \left\{ \frac{C\lambda}{\alpha} + e^{-\alpha t}\,\frac{C\lambda}{\alpha} + e^{-\alpha t}K + e^{-\alpha t}\,\frac{C\lambda}{\alpha}\,e^{-t} \right\}$$

$$V_1(\lambda) = \min_{0 \le t \le \infty} \left\{ \frac{C\lambda}{\alpha} + e^{-\alpha t}\left(K - \frac{C\lambda}{\alpha} + \frac{C\lambda}{\alpha}\,e^{-t}\right) \right\}$$

Considere as condições apresentadas no lemma 3.3.

Se o número total de falhas for superior ao número médio de falhas por actividade manutenção ou $0 \langle \lambda \le \frac{\alpha K}{C}$ tem-se:

$K - \frac{C\lambda}{\alpha} \ge 0$ e deduz-se que $e^{-\alpha t}\left(K - \frac{C\lambda}{\alpha} + \frac{C\lambda}{\alpha}\,e^{-t}\right) \ge 0$, esta equação é estritamente nula se $t = \infty$.

No entanto $V_1(\lambda) = \frac{C\lambda}{\alpha}$, $T_1(\lambda) = \infty$, $M_1(\lambda) = 0$

Se o número total de falhas for inferior ao número médio de falhas por actividade manutenção ou $\lambda \rangle \frac{\alpha K}{C}$ tem-se:

$K - \frac{C\lambda}{\alpha} \langle 0$ e seja $f(t) = \frac{C\lambda}{\alpha} + e^{-\alpha t}\left(K - \frac{C\lambda}{\alpha} + \frac{C\lambda}{\alpha}\,e^{-t}\right)$

aplicando o diferencial:

$$\frac{df}{dt} = \left(Ke^{-\alpha t} - e^{-\alpha t}\,\frac{C\lambda}{\alpha} + e^{-\alpha t}\,\frac{C\lambda}{\alpha}\,e^{-t}\right)' = e^{-\alpha t}\left(-\alpha K + C\lambda - C\lambda e^{-t} - \frac{C\lambda}{\alpha}\,e^{-t}\right)$$

Seja $h(t) = \left[-\alpha K + C\lambda - C\lambda e^{-t} - \frac{C\lambda}{\alpha}\,e^{-t}\right]$

Achando:

$h(0) = -\alpha K - \frac{C\lambda}{\alpha} < 0 \rightarrow h(0) < 0$

$h(\infty) = -\alpha K + C\lambda > 0 \rightarrow h(\infty) > 0$, pois sendo α: $\alpha \in [0,1]$, a parcela $-\alpha$K tende a ser menor, então $h(t)$ é crescente, consequentemente $\exists T^*$: $h(T^*) = 0$

e $V_1(\lambda) = f(t)$ tem um mínimo global no ponto T^*. Logo, $T_1(\lambda)^*$ é um valor diferente de infinito pois existe pelo menos uma manutenção, isto é pelo facto de $\lambda > \frac{\alpha K}{C}$, num período curto possível, será necessário realizar outra manutenção daí que,

$$T_1(\lambda) = T^* = T_1(\lambda) = \ln\left(\frac{C\lambda}{C\lambda - \alpha K}\frac{\alpha + 1}{\alpha}\right)$$

$$V_1(\lambda) = f(T^*) = \frac{C\lambda}{\alpha}\ \frac{\alpha^{\alpha-1}}{(\alpha+1)^{\alpha+1}}\frac{(C\lambda\ \alpha K)}{C\lambda^{\alpha}}$$

$M_1(\lambda) = 1.$

c.q.d.

Com o aumento de falhas no sistema há necessidade de uma actividades de manutenção frenquente, isto é, ocorrerão situações em que a média de erros λ cresce e o número total óptimo de manutenções também tenderá a crescer como se pode constatar no teorema 3.3.

Teorema 3.3

$M(\lambda)$ *é crescente como função de* λ *(Wee, 1988).*

Demonstração:

$M(\lambda)$ é crescente se $\forall\ \lambda_1, \lambda_2 : \lambda_1 < \lambda_2\ M(\lambda_1) < M(\lambda_2)$

Considerando que $M(\lambda_2) = n$ número de manutenções, e aplicando a indução matemática, tem-se:

1. Para $M(\lambda_2) = 0$, pelo lemma 3.3 cumpre-se a condição $\lambda_2 \leq \frac{\alpha K}{C}$ e por transitividade $\lambda_1 \leq \frac{\alpha K}{C}$ e $M(\lambda_1) = 0$.

2. Supõe-se que $M(\lambda_2) \leq n - 1$ é verdadeiro por hipótese onde n é um número inteiro.

3. Para $M(\lambda_2) = n$.

Uma vez que $\lambda_1 < \lambda_2$ então pelo teorema 3.2 $\lambda_1^* = \lambda_1 e^{-T(\lambda_1)} \leq \lambda_1^* = \lambda_2 e^{-T(\lambda_2)}$. Supõe-se que $M(\lambda_2^*)$ é a penúltima manutenção então $M(\lambda_2^*) = M(\lambda_2) - 1$. Foi suposto que $M(\lambda_2) \leq n - 1$ é verdadeiro então,

$M(\lambda_2^*) = M(\lambda_2) - 1 \leq n - 1$ logo $M(\lambda_1^*) \leq M(\lambda_2^*)$. Analogamente por dedução tem-se :

$M(\lambda_2) = M(\lambda_2^*) + 1 \geq M(\lambda_1^*) + 1 = M(\lambda_1) \rightarrow M(\lambda_2) \geq M(\lambda_1)$

c.q.d

O teorema que se segue ilustra a fórmula explícita para o cálculo do próximo período óptimo de manutenção dado uma média de erros λ. O número restante de erros obedece a distribuição de poisson com a, média $\lambda\ e^{-T}$ conhecida.

Teorema 3.4

Para um determinado λ, seja $\lambda^* = \lambda e^{-T(\lambda)}$ então

$$T(\lambda_i) = \ln\left(\frac{C\lambda_i}{C\lambda_i - \alpha K}\,\frac{\alpha + 1 - e^{-\alpha T(\lambda_i^*)}}{\alpha}\right)$$ *(Wee, 1988).*

Demonstração:

Assume-se que o número óptimo de manutenções $M(\lambda) = 1$, então o próximo período óptimo de manutenção é dado pela formula $T(\lambda) = \ln\left(\frac{C\lambda}{C\lambda - \alpha K}\frac{\alpha + 1}{\alpha}\right)$, segundo o lemma 3.4.

Por outro lado se o número óptimo de manutenções para os erros restantes for $M(\lambda^*) = 0$ então período óptimo de manutenção é imprevisível, isto é $T(\lambda^*) = \infty$,

pelo lemma 3.3.

Assume-se que o número óptimo de manutenções é um valor tal que $1\langle M(\lambda)\langle\infty$, considere-se $M(\lambda) = n$. Seja T_i o comprimento óptimo entre duas manutenções consecutivas, isto é, i e i-ésima manutenção para i=1,...,n. Seja U_k comprimento da fase de manutenção onde k=1,2...,n e U_0=0, representando U_k

$U_k = \sum_{i=1}^{K} T_i$ então, partindo da formula $V(\lambda) = \frac{C\lambda}{\alpha} + e^{-\alpha t}\left(K - \frac{C\lambda}{\alpha} + \frac{C\lambda}{\alpha}e^{-t}\right)$ representando $V(\lambda)$ usando U_k tem-se

$$V(\lambda) = \sum_{i=0}^{n}\frac{C\lambda}{\alpha}e^{-(1+\alpha)U_i} - \sum_{i=0}^{n-1}\frac{C\lambda}{\alpha}e^{-(U_i+\alpha U_{i+1})} + \sum_{i=0}^{n}Ke^{-\alpha U_i}$$

Aplicando a optimalidade de períodos óptimos de manutenção, $(T_1,T_2,...T_n)$, tem-se:

$$\frac{dV}{dT_1} = \frac{dV}{dT_2} = ... = \frac{dV}{dT_n} = 0, \text{ pois, } \frac{dV}{dT_1} - \frac{dV}{dT_2} =$$

$$= \frac{(1+\alpha)C\lambda}{\alpha}e^{-(U_1+\alpha U_2)} + C\lambda e^{-\alpha U_1} - C\lambda e^{-(U_1+\alpha U_2)} - \alpha K e^{-\alpha U_1} = 0$$

Se $T_1 = T(K)$ que é primeiro período óptimo de manutenção e $T_2 = T(\lambda^*)$ que é o segundo período óptimo de manutenção, então os valores $T_1, T_2,...,T_n$ são verdadeiros e representam os tempos óptimos de manutenção.

c.q.d

Com o decrescer de número de falhas no sistema, a necessidade de manutenções frenquentes diminui, isto é o próximo período óptimo de manutenção é mais longo, além disso, para a média de erros λ: $\lambda \rangle (\alpha K)/C$, $T(\lambda)$ estritamente decrescente O teorema que se segue ilustra tal situação.

Teorema 3.5

$T(\lambda)$ é monótonamente decrescente em λ e para $\lambda \rangle (\alpha K)/C$, $T(\lambda)$ é estritamente monótona e decrescente (Wee, 1988).

Demonstração:

Demonstra-se directamente a situação em que para $\lambda \rangle (\alpha K)/C$, $T(\lambda)$ estritamente decrescente.

$T(\lambda)$ é decrescente se para valores λ_1, $\lambda_2 : \lambda_1 \langle \lambda_2$ $T(\lambda_1) \rangle T(\lambda_2)$

Supõe-se que $\frac{C\lambda}{\alpha} \langle \lambda_1 \langle \lambda_2$ usando indução em $M(\lambda_2)$

Seja $M(\lambda_2) = 1$ pelo teorema 3.3 $M(\lambda_1) \leq 1$ pois $\lambda_1 \langle \lambda_2$ e $\lambda_1^* = \lambda_1 e^{-T(\lambda_1)} \leq$

$\lambda_1^* = \lambda_2 e^{-T(\lambda_2)}$. $M(\lambda_2^*) = n - 1$, pois é a última manutenção.

Tendo o resultado da indução $T(\lambda_1^*) \rangle T(\lambda_2^*)$ e pelo teorema 3.4

$$T(\lambda_i) = \ln\left(\frac{C\lambda_i}{C\lambda_i - \alpha K} \frac{\alpha + 1 - e^{-\alpha T(\lambda_i^*)}}{\alpha}\right) \text{ desde que}$$

$$\frac{C\lambda_i}{C\lambda_i - \alpha K} \rangle \frac{C\lambda_i}{C\lambda_i - \alpha K} \rangle 0 \; ; \; \frac{\alpha + 1 - e^{-\alpha T(\lambda_i^*)}}{\alpha} \rangle \frac{\alpha + 1 - e^{-\alpha T(\lambda_i^*)}}{\alpha} \rangle 0$$

assim $T(\lambda_1) \rangle T(\lambda_2)$

c.q.d.

$T(\lambda)$ é uma função de tempo. A característica de função é enunciada pelo teorema que se segue e ilustra que a actividade de manutenção deve decorrer sem interrupção.

Teorema 3.6

$T(\lambda)$ *é contínua e portanto único para qualquer* λ *(Wee, 1988).*

Demonstração :

Por definição $T(\lambda)$ contínua se $\lim_{\lambda \to \lambda_1} T(\lambda) = T(\lambda_1)$ e $T(\lambda_1)$ existe e é definido.

Pelo lemma 3.3 $T(\lambda) = \ln\left(\frac{C\lambda}{C\lambda - \alpha K} \frac{\alpha + 1}{\alpha}\right)$. Calculando: $1.T(\lambda_1) = \frac{C_1\lambda}{} \frac{\alpha + 1}{\alpha}$

desde que: $\frac{C_1\lambda}{C\lambda_1 - \alpha K} \frac{\alpha + 1}{\alpha} \rangle 0 \; ; C\lambda_1 \neq \alpha K$ e $\alpha \neq 0$

2. $\lim_{\lambda \to \lambda_1} T(\lambda) = T(\lambda_1)$

c.q.d.

O lemma que se segue descreve que se um sistema tem falhas, independentemente da média de erros, sempre existirá um número óptimo de manutenções que deve ser um valor inteiro não negativo.

Lemma 3.5

$\forall\ n : n>0, \exists\ \lambda : 0 \langle \lambda \leq \infty : M(\lambda) = n$ *(Wee, 1988).*

Demonstração:

Demontração por contradição.

Supõe-se que existe um número inteiro L:L>0 e $M(\lambda) \neq L$, se $M(\lambda) \leq L$ para $0 \leq \lambda \leq \infty$, tm-se

$\lim_{\lambda \to \infty} T(\lambda) = \lim_{\lambda \to \infty} \ln\left(\frac{C\lambda}{C\lambda - \alpha K} \frac{\alpha + 1}{\alpha}\right) = \infty$, o que contradiz o lemma 3.3

c.q.d.

O teorema que se segue é consequência do teorema 3.6 e é apresentado sem demonstração.

Teorema 3.7

$\forall\ n : n>0$, *seja* I_n *o conjunto de* λ *'s:* $\mathrm{M}(\lambda) = n$.

1. *Define-se D:* $I_n \rightarrow I_{n-1} : D(\lambda) = \lambda e^{-T(\lambda)}$. *Então D é estritamente monótona*

2 *Seja* $\lambda_n = \inf I_n$ *e* $\mu_n = \sup I_n$. *Então,* $\lambda_{n+1} = \mu_n$ *e* $I_n = (\lambda_n, \lambda_{n+1})$ *também têm-se* $D(\lambda_{n+1}) = \lambda_n$.

O teorema que se segue ilustra fórmulas explícitas para o cálculo de médias de erros e é apresentado sem demonstração . No período inicial é obvio que se tem uma média nula. Também é desejável que a média de erros não ultrapasse os custos inicialmente previstos. As restantes médias são calculadas recursivamente.

Teorema 3.8

Seja λ_n *definido como no teorema 3.7, então:*

$$\lambda_0 = 0, \qquad \lambda_1 = \frac{\alpha K}{C}$$

$$\lambda_{n+1} = \frac{\alpha K}{C} + \frac{1+\alpha}{\alpha}\lambda_n - \frac{1}{\alpha}\frac{\lambda_{n-1}^{\alpha}}{\lambda_n^{\alpha-1}} \qquad \textit{se } n>=1 \textit{ (Wee, 1988).}$$

Colorário 3.1

$\lim_{n\to\infty} \lambda_n = \infty$ *(Wee, 1988).*

Demonstração:

Usando Indução matemática têm-se:

1 Para k=0 λ_0 pelo teorema3.8 cumpre-se a condição $\lim_{n} \lambda_n = 0$

2.Supõe-se que para k=n-1 a condição é verdadeira,

3 Para k=n+1 têm-se: $\lim_{K\to N+1}\lambda_{N+1} = \lim_{k\to n+1}\left(\frac{\alpha K}{C}+\frac{1+\alpha}{\alpha}\lambda_n - \frac{1}{\alpha}\frac{\lambda_{n-1n}^{\alpha}}{\lambda_n^{\alpha-1}}\right)=\infty$

Pela condição do teorema 3.8 a condição para k=n+1 cumpre-se.

c.q.d.

O corolário que se segue é apresentado sem demonstração e menciona o caso em que a manutenção não é feita no momento planeado.

Teorema 3.9

Supõe-se que $\mathrm{M}(\lambda) = n$. *Define-se a função* f: $(\lambda_0,\ \lambda_1] \to (\lambda_n,\ \lambda_{n+1}]$: $f(x) = xe\sum_{i=1}^{n} \mathrm{T}_i$,

onde $\{\mathrm{T}_i\}_{i=1}^{n}$ *são definidos recursivamente como se segue:*

1. $Cxe^{S_{i+1}}\left(\alpha+1-e^{-\alpha \mathrm{T}_{i+1}}\right)=\propto \left(Cxe^{(S_{i+1}+\tau_i)}-\alpha K\right)$ *para i=n,...,1*

2. $\mathrm{T}_{n+1} = \infty$

3 $S_{n+1} = 0$

4 $S_i = \sum_j^n \mathrm{T}_j$, *para j=1,...,n. Então D é estritamente monótona*

$\mathrm{T}_i(i = 1, \ldots, n)$, *é o comprimento óptimo entre i e i-ésima manutenção para f(x) (Wee, 1988).*

Demonstração:

Pelo teorema 3.4, para um dado $x\in(\lambda_0,\lambda_1]$ existe um y: $y\in(\lambda_n,\lambda_{n-1}]$ de forma que após n manutenções óptimas, y decresce até x.

Considere que $\mathrm{T}(x)=\infty$ seja $\mathrm{T}_{n+1}=\infty$, então, pode-se determinar o intervalo óptimo de manutenção para o y usando teorema 3.4. Pelo teorema 3.7 e 3.2 a função f é monótona.

c.q.d

A actividade de manutenção é contínua e consiste na inspecção dos dados de saída, identificação de erros e a respectiva correcção. Sendo assim, a abordagem sequencial de Bayes é tomada em consideração. Para o modelo bayesiano segue-se que depois de uma manutenção no tempo T, o número restante de erros é distribuido segundo a distribuição de poisson com a média λ conhecida *(Wee, 1988).*

Um plano de manutenção consiste de várias actividades de manutenção. De seguida descreve-se as parâmetros do modelo:

3.5.2 Parâmetros do modelo

De seguida apresenta-se a definição dos parámetros do modelo em estudo:

Variável de estado - $\lambda : \lambda = [0,\infty[$

Conjunto de acções e conjunto de acções admissíveis = {a_n : espaço de tempo até a próxima manutenção}, isto é $A = d(S) = [0,\infty[$

Função de transição - $\lambda_{n+1} = \mathrm{T}(\lambda_n, a_n, z_n)$

Espaço de Disturbâncias Z = IN, z_n : Quantidade de anomalias que persistem após n-ésima Manutenção.

Factor de desconto – Não é constante $\beta(\lambda, a) = e^{-\alpha a}$

Lei de Distribuição - $Q(\vartheta, \lambda, a) = \begin{cases} b(\vartheta, e^{-a}), & \lambda \neq 1 \\ \lambda_0, & \lambda = 0 \end{cases}$

Função de Ganho num estágio - $r(\vartheta, \lambda, a) = \begin{cases} \dfrac{d(\vartheta)}{\alpha}\left(1 - e^{-\alpha a}\right) - Ke^{-\alpha a} & \lambda \neq 1 \\ 0 & \lambda = 0 \end{cases}$

Ganho terminal - $V_0(\vartheta,\lambda,a)=\begin{cases}\dfrac{d\vartheta}{\alpha} & \lambda\neq 1\\ 0 & \lambda=0\end{cases}$

Distribuição a prior $\mu_0=\pi(\lambda)$

Do programa computacional em anexo (Anexo A), obteve-se a seguinte tabela de valores de $V_n(\lambda)$ custos finais para n manutenções considerando diferentes médias de erros λ

s	*n*	$V_n(s)$	*Tempos Óptimos* T_i																		
			T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	18	T19	T20
0	0	0	∞																			
3	5	2.2	1.8	2.0	3.0	1.4	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	**8**	**2.9**	**0.5**	**0.3**	**0.5**	**0.5**	**1.2**	**1.5**	**2.0**	**2.5**	0	0	0	0	0	0	0	0	0	0	0	0
5	9	3.9	0.8	0.9	0.5	1.5	1.5	2.6	0.7	0.2	0.3	0	0	0	0	0	0	0	0	0	0	0
8	12	4.9	1.2	1.5	0.5	0.5	0.6	1.2	0.3	0.9	0.1	0.8	0.5	0	0	0	0	0	0	0	0	0
8	13	5.3	0.5	0.4	0.8	0.8	0.8	0.8	0.5	1.0	0.5	0.4	1.0	1.0	0.5	0	0	0	0	0	0	0
10	15	6.3	1.0	0.6	0.2	0.3	0.5	0.4	0.4	0.5	1.4	0.5	0.5	1.5	0.4	0.5	0.3	0	0	0	0	0
10	17	7.9	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.5	0.5	0.6	0.7	0.7	0.8	1.0	1.2	1.5	0	0	0
15	18	8.5	0.5	0.5	0.5	0.5	1.0	0.2	0.3	0.5	0.3	0.8	0.2	0.5	0.2	0.5	0.6	0.8	0.6	0.5	0	0
15	20	9.5	0.6	0.5	0.3	0.2	0.5	0.5	0.5	0.4	0.1	0.4	0.3	0.3	0.9	0.9	0.8	0.4	0.5	0.5	0.3	0.1

Tabela 3.1:Valores de $V_n(\lambda)$ custos finais para n manutenções considerando diferentes médias de erros λ com β=0.97 λ=s.

3.5.3 Algorítmos de Solução

De seguida apresentam-se os algorítmos para computarizar os valores de $V(\lambda)$, $T(\lambda)$, $M(\lambda)$, dado um determinado λ.

3.5.3.1 Computarização do número total óptimo de manutenções $M(\lambda)$

Achar $M(\lambda)$ é precisamente encontrar n: $\lambda_n<\lambda\leq\lambda_{n+1}$ computarizando λ_i recursivamente segundo **teorema 3.8**

$$\lambda_1 = \frac{\alpha K}{C}$$

Begin

n = número de manuteções permitidas

Input K Custo médio por actividade de manutenção

Input α Factor de desconto

Input C Custo médio por cada anomalia

λ Média de erros a considerar.

Main loop

For I=1 to n

$$\lambda_{i+1} = \frac{\alpha K}{C} + \frac{1+\alpha}{\alpha}\lambda_i - \frac{1}{\alpha}\frac{\lambda_{i-1}^{\alpha}}{\lambda_i^{\alpha-1}}$$ *{Fórmula da função de transição do sistema}*

End loop

3.5.3.2 Computarização de $V_n(\lambda)$

Begin

$M(\lambda) = n$ Número óptimo de manutenções do software.

T_i= Intervalo óptimo entre a i manutenção e a i-ésima manutenção sendo i=1 … n

U_0=0 Variável para inicializar a soma dos tempos óptimos anteriores

$U_k = \sum_{i=1}^{k} T_i$ k=1,..,n {Acumulação dos tempos óptimos anteriores

Main loop

For k=1 to n

$$V_i(\lambda) = \sum_{i=0}^{n} \frac{C\lambda}{\alpha} e^{-(1+\alpha)U_i} - \sum_{i=0}^{n-1} \frac{C\lambda}{\alpha} e^{-(U_i+\alpha U_{i+1})} + \sum_{i=0}^{n} K e^{-\alpha U_i}$$

End For

End Main Loop

End

3.5.3.3 Computarização de $T_n(\lambda)$

Para um dado λ (média de erros num software) T_i é o intervalo entre i-i ésima manutenção. Usando as suposições do teorema 3.10, têm-se o seguinte algorítmo:

Begin

Input l comprimento predeterminado {distancia entre λ_i e λ_{i-1} que é usada para achar o valor de x método de bisecção}

$a_1 = \lambda_0$

$b_1 = \lambda_1$

i=1

$M(\lambda) = n$

$S_{n+1}=0$

$T_{n+1}=\infty$

Bloco principal

1 $z = b_1 - a_1$ {Cálculo do ponto médio para achar x}

2 If z>l

Go to step 1 {Volta ao cálculo do ponto médio}

Else

$x = \frac{a_i + b_1}{2}$ {Acha o ponto x e calcula os Ti no passo 5}

Go to step 5

Endif

3 Find $f(x)$ {Esta função calcula-se com base nas condições do teorema 3.10}

4 If $f(x) < \lambda$

$a_{i+1} = \frac{a_i + b_1}{2}$ {Passagem de valores para os arrays auxiliares}

$b_{i+1} = b_i$

go to step 1 {Cálculo do ponto médio para achar outro x}

EndIf

If $f(x) > \lambda$

$b_{i+1} = \frac{a_i + b_1}{2}$ {Passagem de valores para os arrays auxiliares}

$a_{i+1} = a_i$

go to step 1 {Cálculo do ponto médio para achar outro x}

EndIf

If $f(x) = \lambda$

$x = \frac{a_i + b_1}{2}$ {Acha o ponto x e calcula os Ti no passo 5}

go to step 5 {Acha o ponto x e calcula os Ti no passo 5}

EndIf

5 For i=n to 1

Compute T_1 by theorem 3.10 {Calculo do Ti tempos óptimos de manutenção do software}

$S_i = \sum_{j=i}^{n} T_j$

Next i

End

Capítulo IV

4 Discusão

No modelo em estudo, supõe-se que o número de erros é aleatório (disturbância) e influencia o processo de manutenção, sendo assim, dificilmente pode-se prever o nível de erros no fim da fase de manutenção. Não é ideal terminar a manutenção enquanto o nível de erro for elevado. Para controlar o processo aleatório, é necessário informação sobre a probabilidade do nível de erro pertencer a um certo intervalo.

A transição do sistema de um estado para o outro, é governada pela função de transição descrita na expressão 3.15

A variável aleatória discreta, Z, obedece a distribuição de Poisson com a média λ conhecida, se a variável aleatória tomar valores abaixo da média de erros, significa que está-se perante uma diminuição de nível de erro. Se por outro lado a variável aleatória tomar valores muito acima da média, significa que está-se perante o aumento do nível de erro. A sequência das acções óptimas, para além de ser escolhida com base no custo mínimo que esta traduz, também terá que comportar um nível de erro baixo a medida que o sistema aproxima-se do estado final.
Com base acima exposto, num problema prático, pode-se determinar o seguinte:
A função de valor $V_n(\lambda)$ que representa os custos, aumenta quanto maior forem os períodos de decisão e a média de erros (veja o gráfico comparativo no anexo B gráfico 7.2).

Com a função de valor $V_n(\lambda)$ a crescer em relação a média λ, numa dada manutenção de comprimento n, é importanate observar a sequência de acções óptimas que são os tempos óptimos para realizar as manutenções.

No estado final do sistema, verifica-se uma relação entre o nível de erro e a distribuição probabilística, isto significa que os custos finais $V_n(\lambda)$ dependem da média de erros e quanto este for menor significa que a probabilidade de reduzir os custos de manutenção é maior (veja o gráfico de função de $V_n(\lambda)$, Anexo B gráfico 7.1).

Capítulo V

5 Conclusões e Recomendações

Para o modelo em estudo, no concernente a minimização de custos chegou-se as seguintes conclusão:

Aplicando as propriedades estruturais das soluções segundo o modelo de controlo bayesiano com disturbâncias aleatórias i.i.d, verifica-se que a função de valor do custo $V_n(\lambda)$ cresce em função da média de erros.λ.

O número de manutenções cresce com a média de erros λ, pois, há necessidade de uma manutenção mais frequente, consequentemente a função de valor $V_n(\lambda)$ também cresce.

O tempo óptimo para se realizar a seguinte manutenção depende do número de erros λ, pois com o decrescer de número de anomalias no sistema, a actividade de manutenção é realizada com menor frequência e vice versa.

Os resultados deste estudo, correspondem a propriedade estrutural das soluções segundo modelo de bayes, particularmente os teoremas 3.3 e 3.5.

Recomenda-se um outro estudo seguindo a mesma distribuição e considerando variáveis aleatórias mutuamente dependentes, onde cada erro é susceptível de provocar falhas dependendo dos outros erros. Por outro lado recomenda-se o uso de outros modelos de controlo.

Capítulo VI

6. Bibliografia

Bertsekas D. P (1987) . Dynamic Programing, "Deterministic and Sthocastic Models, New Jersey

Cox, D. R. e Miller, H. D.(1967). Theory of Stochastic Processes", Imperial college, Londres.

Ehrlich, Pierre Jacques (1985) . Pesquisa Operacional, Atlas São Paulo.

Hartigan, J. A. (1983). Bayes Theory, Berlim.

Hinderer, K. (1984). On the Stucture of Sthocastic Dynamic Programs, in the proceeding of the Seventh Conference on , Brasov.

Hinderer, K. (1993). Lectures Notes on Sthocastic Dynamic Programs, in the proceeding of the Seventh Conference on , Brasov.

IEEE Computer Society (1993). Conference on Software Maintenance, Miami.

IEEE Computer Society, (1999). International Conference on Software Maintenance, Montery.

IEEE Computer Society,(1994). IEEE Standard for Software Maintenance, Miami.

Kaufmann, A- (1967). Graphic, Dynamic Programing and finite games, Academic Press, Nova York

Longstreet, H david, (1990) Software Maintenance and computers, *IEEE Computer Society.*

Low, M Averil and Kelton, W David (1991) Simulation Modeling & Analisys, McGraw – Hill Inc., New York.

Pigoskim M Thomas (1996). Practical Software Maintenance, John Wiley&Sons, New York.

Radford, E Loren andHigh, W Roger (1989). Turbo Pascal, Guanabara.

Shamblin, James E.(1988). Pesquisa Operacional., Atlas São Paulo.

Shewlost, A. walter e Wilks, Samuel S (1985). Stochastic Process, Willey publishing, Londres.

Taha, Handy A. (1987). Operation Research",Macmillan publishing company.

Wee, N (1988). A Bayesian for determining optimal testing intervals for computer Software, Ph. D. dissertation, University of Califórnia.

White, D,.J. (1985) . Operation Research, Great Britain.

Capítulo VII

7 Anexos

7.1 Anexo A - Programa Fonte

```
PROGRAM Plano_Optimo_Manutencao;
USES CRT, graph; interpol_metod, bissec_metod, cp3;

Const
        Num_Manutencao = 8;
        Max_stream_num = 32;
        ScreenBorderY = 5;
        ScreenBorderX =5;
        UseRules=11;
        Haswidth=8;
        Numstrings=4;
        Numrules = (1,2,3,4,5,6,7,8,9,10,11)
        Title="Gráfico de Custos mínimos Vn(s) com Diferentes Valores de s"
        Title1="Gráfico Comparativo do Valores da Função Vn(s ) com Diferentes n e diferentes
s "

Type
        Lambdas_c  = array[0..Num_Manutencao] of real;
        Cardinal = 1..maxint;
        Stream_num  = 1..max_stream_num;
        Num_n = array[0..num_manutencao] of real;
        Listofstrings = array[1..4] of string;
        Listofnums = array[1..4] of integer;

Var Sair            :boolean;
```

```
  opcao, H          :integer;
   K, alfa,    :Real;
        Seeds, original_seeds :array[Stream_num] of cardinal;
        w:Cardinal;
        C, Vi, Ti:Num_n;
        lambdas  :Lambdas_c;
        Maxx,Maxy,Gmode:integer;
        x:string;
        ch:char;
        title:string
{---------------------------------------- Painel de Introdução ----------------------------------------}
Procedure Intro;
Var i, y, graphdriver, graphmode, errorcode, posicaoX, posicaoY:integer;
        altura:word;
Begin
        CLRSCR;
        Graphdriver := Detect;
        InitGraph(graphdriver, graphmode, 'c:\Tp\BGI');
        ErrorCode:=GraphResult;
        If ErrorCode<>grOk then
        begin
                Writeln('Erro grafico: ' ,GraphErrorMsg(errorcode));
                Writeln('Programa Abortado');
                Readln;
         end;
        SetLineStyle(SolidLn,3,ThickWidth);
        Rectangle(0,0,getMaxX,getMaxY);
        Settextjustify(CenterText,CenterText);
        SettextStyle(SansSeriffont, Horizdir, 3);
        OutTextXY(GetMaxX div 2, GetMaxY div 10, 'PLANO OPTIMO DE MANUTENCAO
        DO SOFTWARE' );
        SettextStyle(Defaultfont, Horizdir,0);
```

```
        OutTextXY(GetMaxX div 2, GetMaxY div 2, 'O programa apresenta o numero optimo
        de manutencoes do softawre' );
        PosicaoY:= GetMaxY div 2;
        Altura:=TextHeight('O programa apresenta o numero optimo de manutencoes do
        softawre');
        OutTextXY(GetMaxX div 2, PosicaoY+Altura+6, 'e intervalo optimo entre as
        manutencoes de forma a minimizar os custos');
        OutTextXY(GetMaxX div 2, PosicaoY+Altura+18, 'totais de manutencao do software.' );
        OutTextXY(GetMaxX div 2, GetMaxY - 10, 'Pressione a tecla Enter para continuar');
        Readln;
        CloseGraph;
end;

{---------------------------------------- Menu Principal ----------------------------------------
Procedure Main_Menu(var opcao:integer);
Var i, y, graphdriver, graphmode, errorcode, posicaoX, posicaoY:integer;
altura:word;
Begin
        Clrscr;
        Graphdriver := Detect;
        InitGraph(graphdriver, graphmode, 'c:\Tp\BGI');
        ErrorCode:=GraphResult;
        If ErrorCode<>grOk then
        Begin
                Writeln('Erro grafico: ' ,GraphErrorMsg(errorcode));
                Writeln('Programa Abortado');
                Readln;
        end;
        SetLineStyle(SolidLn,3,ThickWidth);
        SetBkcolor(3);
        Rectangle(100,100,getMaxX-100,getMaxY-100);
        Settextjustify(CenterText,CenterText);
```

```
        SettextStyle(TriplexFont, Horizdir, 2);
        OutTextXY(GetMaxX div 2, 120, '********** Menu Principal **********' );
        Altura:=TextHeight('******* Menu Principal *********');
        PosicaoY:=120;
        SettextStyle(Defaultfont, Horizdir,0);
        OutTextXY(GetMaxX div 2, PosicaoY+Altura+60, '1    Introduzir Dados' );
        PosicaoY:= GetMaxY div 2;
        Altura:=TextHeight('1    Introduzir Dados');
        PosicaoY:= PosicaoY+Altura;
        OutTextXY(GetMaxX div 2, PosicaoY, '2    Resultados Resumidos');
        PosicaoY:= PosicaoY+Altura+30;
        OutTextXY(GetMaxX div 2, PosicaoY+Altura, '3    Resultados Graficos' );
        PosicaoY:= PosicaoY+Altura+30;
        OutTextXY(GetMaxX div 2, PosicaoY+Altura, '4    Sair do Sistema' );
        PosicaoY:= PosicaoY+Altura+30;
        OutTextXY(GetMaxX div 2, GetMaxY - 10, 'Escolha uma opcao');
        Readln(opcao);
        CloseGraph;
end;

{----------------------------------------Introducão de Dados----------------------------------------}
Procedure Intro_Dados;
Var i:integer;
Begin
        Clrscr;
        Write('Introduza o Horizonte de Planeamneto, em dias: ');
        Readln(H);
        While (H>180) or (H<90) do
        Begin
                Writeln(' O horizonte de planeamneto est  incorrecto');
                Write('Introduza o Horizonte de Planeamneto, entre 90 e 180  dias: ');
                Readln(H);
```

```
            Writeln;
        end;
        Write('Introduza o Factor de Desconto: ');
        Readln(alfa);
        Writeln;
        While (alfa>1) or (alfa<0) do
        Begin
            Writeln(' O Facto de Desconto est  incorrecto');
            Write('Introduza o Factor de Desconto, entre 0 e 1: ');
            Readln(alfa);
            Writeln;
        end;
        Write('Introduza o Custo por Actividade de Manutencao: ');
        Readln(K);
        Writeln;
        While (K<=0) do
        Begin
            Writeln(' O Custo por Actividade de Manutencao esta incorrecto');
            Write('Introduza o Custo por Actividade de Manutencao, valor positivo: ');
            Readln(K);
            Writeln;
        end;
end;

{---------------------------------------- Geração para Sementes ----------------------------------}
Procedure Make_streams;
Var i : stream_num;
Begin
        For i:=1 to max_stream_num do
            original_seeds[i] := i*1000+7;
        seeds := original_seeds;
end;
```

```
{-----------------------------------------Devolve um número aleatório ----------------------------}
Function Rnd(s:stream_num):real;
Const
mult = 3993;
divid = maxint;
add=1;
Var r, quotient: real;
        aux:integer;

begin
   aux:=(seeds[s]*mult+add);
   seeds[s]:=aux mod divid;
   If seeds[s]<0 then
     quotient:=r/seeds[s];
   seeds[s]:=seeds[s]+maxint+1;
   Rnd:=abs(seeds[s]/divid);
end;

{---------------------------------- Gerar variável aleatória segundo distribuição de poisson----------
--}

Function Gerar_Zi:cardinal;
Var p, f, b, u : real;
      x : cardinal;
      s:stream_num;

Begin

  p:=exp(-lambda);
  x:=1;
  b:=1;
```

```
  While b>p do
  begin
    x:=x+1;
    u:=Rnd(s);
    b:=b*u;
  end;
  gerar_Zi:=x;
end;

{--------------------------------Custo por falha em cada manutencao----------------------------------}
Function Custo_Falha:real;
Begin
  Custo_Falha:=K/Gerar_Zi;
end;
{--------------------------------Função que calcula potencia de x elevado a y real-------------------}
Function Expoente(factor:real;i:integer):real;
Var s:real;

begin
  s:=Interpolacao(factor;i);
  Expoente:=s;
end;
{-----------------------------Calcula os diferentes estados do sistema---------------------------------}
Procedure Achar_lambdas; { \lambda_{i+1} = \frac{\alpha K}{C} + \frac{1+\alpha}{\alpha}\lambda_i - \frac{1}{\alpha}\frac{\lambda_{i-1}^{\alpha}}{\lambda_i^{\alpha-1}} }
Var i:integer;

Begin
  Make_Streams;
  lambdas[0]:=lambda;
  C[0]:= K/lambda;
  lambdas[1]:=alfa*K/C[0];
```

```
  For i:=2 to Num_Manutencao do
  Begin
     C[i-2]:=Custo_falha;
     lambdas[i]:=alfa*K/C[i-2]+((1+alfa)/alfa)*lambdas[i-1]-(Expoente(alfa,i-
2)*alfa*Expoente(alfa-1,i-1));
end;
  readln;
end;

{-------------------------Cálculo do somatório dos  comprimento dos tempos óptimos anteriores---}
Function Ui(i:integer):real;
Var z:integer;
  somatorio:real;

begin
  somatorio:=0;
  If i=0 then
    Ui:=0
  else
     For z:=1 to i do
     somatorio:=somatorio+Ti[i];
     Ui:=somatorio;
end;

{-----------------------------------------------------------------------}
Function Parcela1(i:integer):real;
Var somatorio:real;

Begin
    For i:=0 to num_manutencao do
       Somatorio:=C[i]*lambdas[i]*Exp(-(1+alfa)*Ui(i));
Parcela1:=(1/alfa)*Somatorio;
```

```
end;
{-------------------------------------------------------------------------}

Function Parcela2(i:integer):real;
Var somatorio:real;

Begin
    For i:=0 to num_manutencao-1 do
       Somatorio:=C[i]*lambdas[i]*(Exp(-(Ui(i)+alfa*Ui(i+1))));
       Parcela2:=(1/alfa)*Somatorio;

end;

{-------------------------------------------------------------------------}
Function Parcela3(i:integer):real;
Var  somatorio:real;

Begin
    For i:=0 to num_manutencao do
       Somatorio:=Exp(-alfa*Ui(i));
    Parcela3:=K*Somatorio;
end;

{-----------------------------------------------------------------------------------------------------------}
Function Somar_Ti(x:real):real;

Var  i:integer;
   s:real;
   f:real;
Begin
   s:=0;
```

```
Ti[Num_manutencao]:=Ln((C[Num_manutencao]*x*(alfa+1)+sqr(alfa)*K)/alfa*C[Num_manut
encao]*x);
   s:=s+Ti[Num_manutencao];
   For i:=Num_manutencao-1 to 1 do
   Begin
      Ti[i]:=Ln((C[i]*x*exp(s)*(alfa+1-exp(-alfa*Ti[i+1]))+sqr(alfa)*K)/(alfa*C[i]*x*exp(s)));
      s:=s+Ti[i];
   end;
   Somar_Ti:=s;

end;
{----------Cálculo dos tempos óptimos Ti----------}
```

$\{Cxe^{S_{i+1}}\left(\alpha+1-e^{-\alpha T_{i+1}}\right)=\alpha\left(Cxe^{(S_{i+1}+T_i)}-\alpha K\right)\}$

```
Procedure Achar_Ti; {}

var z,i:integer;
   a,b: lambdas_c;
    lambda_medio,x,m,funcaox,s:real;
    achado:boolean;

begin
  z:=0;
  Achado:=false;
  lambdas[z]:=a[z];
  lambdas[z+1]:=b[z];
  l:=Bisseccao(lambdas[z],lambdas[z+1]);
  Repeat
   lambda_medio:=(lambdas[z+1]+lambdas[z])/2;
   m:=b[z]-a[z];
   If m>l then
   begin
```

```
      funcaox:=x*exp(1)*Somar_Ti(x);
      If funcaox<lambda_medio then
      begin
         a[z+1]:= (a[z]+b[z])/2;
         b[z+1]:=b[z];
         z:=z+1;
       end;

       If funcaox>lambda_medio then
       begin
         a[z+1]:=a[z];
         b[z+1]:= (a[z]+b[z])/2;
         z:=z+1;
       end;

       If funcaox=lambda_medio then
       begin
         x:= (a[z]+b[z])/2;
         s:=Somar_Ti(x);
         achado:=true;
       end;
    end;
    If lambda_medio<=l then
    begin
      x:=(a[z]+b[z+1])/2;
      s:=Somar_Ti(x);
      achado:=true;
    end;
  until achado=True;

end;
```

{----------Cálculo os custos finais ----------}

$$V_i(\lambda) = \sum_{i=0}^{n} \frac{C\lambda}{\alpha} e^{-(1+\alpha)U_i} - \sum_{i=0}^{n-1} \frac{C\lambda}{\alpha} e^{-(U_i+\alpha U_{i+1})} + \sum_{i=0}^{n} K e^{-\alpha U_i}$$

```
Procedure Achar_Vi;
var i:integer;

begin
   For i:=0 to num_manutencao do
      Vi[i]:=Parcela1(i)-parcela2(i)+Parcela3(i);
end;
{----------Visualização dos resultados----------}
Procedure Visualizar_Resultado;
Var i, altura,y, graphdriver, graphmode, errorcode, posicao_l, posicaoY:integer;
  Z, vi_minimo:real;
  x:string;

Begin
   clrscr;
   posicao_l:=50;
   z:=0.3;
   Graphdriver := Detect;
   InitGraph(graphdriver, graphmode, 'c:\Tp\BGI');
   ErrorCode:=GraphResult;
   If ErrorCode<>grOk then
   begin
      writeln('Erro grafico: ' ,GraphErrorMsg(errorcode));
      Writeln('Programa Abortado');
      Readln;
   end;
   SetLineStyle(SolidLn,3,ThickWidth);
   Settextjustify(CenterText,CenterText);
   SettextStyle(TriplexFont, Horizdir, 2);
```

```
    OutTextXY(getMaxX div 2, 50, '************ Resultados Resumidos ***********' );
    Altura:=TextHeight('************ Resultados Resumidos ***********');
    PosicaoY:=70;
    Settextjustify(LeftText,LeftText);
    SettextStyle(Defaultfont, Horizdir,0);
    PosicaoY:= PosicaoY+Altura+40;
    STR(Num_Manutencao,x);
    OutTextXY(50, PosicaoY, Concat('Numero Optimo de Manutencoes M(lambda) ', x ));
    Altura:=TextHeight('Numero Optimo de Manutencoes M(lambda)');
    PosicaoY:= PosicaoY+Altura+40;
    STR(lambda,x);
    OutTextXY(50,PosicaoY, Concat('Media de Erros Lambda :',x));
    Altura:=TextHeight('Media de Erros Lambda : ');
    PosicaoY:= PosicaoY+Altura+40;
    OutTextXY(50, PosicaoY, 'Tempos Optimos :' );
    Altura:=TextHeight('Tempos Optimos :');
    PosicaoY:= PosicaoY+Altura+30;
    For i:=1 to Num_manutencao do
    begin
       z:=T[i];
       STR(z:2:2,x);
       OutTextXY(posicao_l, PosicaoY, x);
       Posicao_l:=Posicao_l+TextWidth(x)+20;
    end;
    Altura:=TextHeight('x');
Minimo_Vi(Vi_minimo);
    STR(Viminimo:4:2,x);
    PosicaoY:= PosicaoY+Altura+altura+40;
    OutTextXY(50, PosicaoY, Concat('Custo Total Minimo V(Lambda) :' ,x));
    Readln;
    CloseGraph;
end;
```

```
{----------Processamento dos parametros Ti e Vi e λi----------}
Procedure Produzir_Dados;

begin
   Clrscr;
   Achar_Lambdas;
   Achar_Ti;
   Achar_Vi;

end;
{-------------------------------------------------------------------------------------}
Function IntTostr(n:Logint);
Var s:string;

begin
   Str(n,s);
   IntToStr:=s;
end;

{-------------------------------------------------------------------------------------}

Procedure ScaleText(Left, Top, Right, Bottom:integer; TheString:string);
Var height : integer;

begin
  SetTextJustify(CenterText, TopText);
  SetUserCharSize(1,1,1,1);
  SetTextStile(TriplexFont, HGorizDir, Usercharsize);
  Height:=TextHeight(TheString)*5 div 4
  If Heigth>Bottom-top then
```

```
  Begin
    SetUserCharsize(Bottom-top,Height,Bottom,Height);
    SetTextStile(TriplexFont, HorizDir, Usercharsize);
    If textWidth(TheString)<=Right-Left then
      OutTextXY((Rigt+Left) div 2, Top, TheString);
    else
    begin
      SetUserCharSize(1,1,1,1);
      SetTextStyle(TriplexFont, HGorizDir, Usercharsize);
      SetUsercharSize(right-Left,TextWidth(TheString),Bottom-Top, Height);
      SetTextStyle(TriplexFont, HorizDir, Usercharsize);
      OutTextXY((Rigt+Left) div 2, Top, TheString)
    end;
  end
else if TextWidth(TheString) >Right-Left then
begin
  SetUsercharSize(right-Left,TextWidth(TheString),right-Left, TextWidth(TheString));
  SetTextStyle(TriplexFont, HorizDir, UserCharsize);
  If textWidth(TheString)<=bottom-top then
    OutTextXY((Rigt+Left) div 2, Top, TheString);
  else
  begin
    SetUserCharSize(1,1,1,1);
    SetTextStyle(TriplexFont, HGorizDir, Usercharsize);
    SetUsercharSize(right-Left,TextWidth(TheString),Bottom-Top, Height);
    SetTextStyle(TriplexFont, HorizDir, Usercharsize);
    OutTextXY((Rigt+Left) div 2, Top, TheString);
   End
Else
Begin
    SetUserCharSize(1,1,1,1);
    SetTextStyle(TriplexFont, HGorizDir, Usercharsize);
```

```
        SetUsercharSize(right-Left,TextWidth(TheString),Bottom-Top, Height);
        SetTextStyle(TriplexFont, HorizDir, Usercharsize);
        OutTextXY((Rigt+Left) div 2, Top, TheString);
 end
end


else
        OuttextXY((Right+Left) div 22, Top , thestring);
end;
{-------------------------------------------------------------------------------------------------------}
Function GetMax(values:Listofnums):integer;
Var I, largest:integer)
Begin
        I:=0;
Largest:=1;
For I:=1 to rules
If Value[I]>largest the
        Largest := nvalues[I];
Getmax:=Largest;
End;
{-------------------------------------------------------------------------------------------------------}
Procedure Line Graph
Var I:integer;
Begin
        Cpinit(17);
        Rcmask(3);
        RcRange(13,-3smax,smax+1);
        RcTitle(Title);
        Rcxax(6,num,"s");
        Rcyax((8,num,"Vn(s))
For I:0 to num_manutencao do
Begin
```

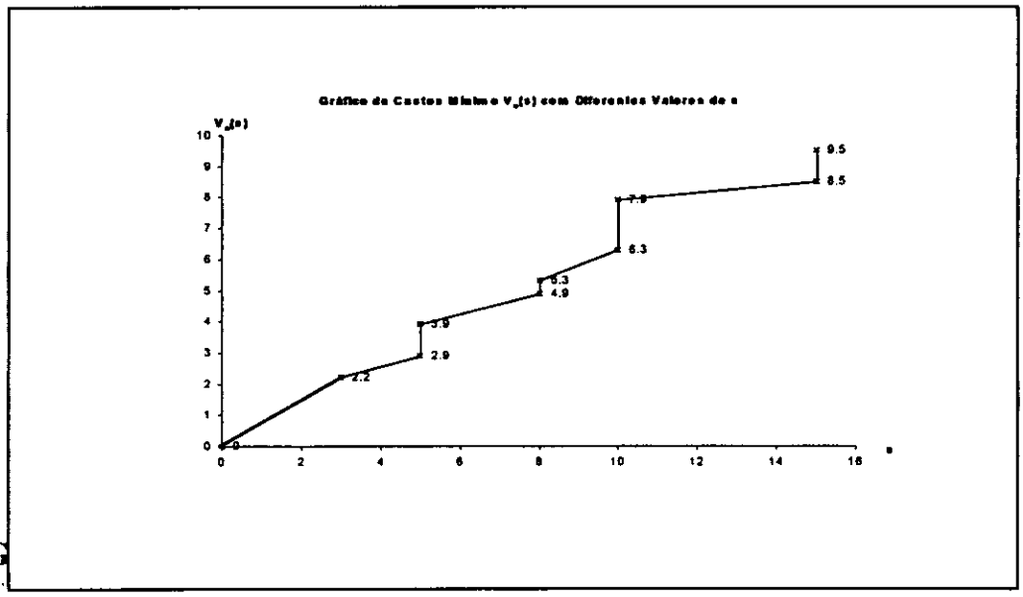
```
        Rcmarks(s,Vi[I], ord("*"));
Rcdrw(s,V[I],sV[I,s+1];
end;
End;
readln
cpexit
End;

{-------------------------------------------------------------------------------------------------------}
Procedure DrawChart(Left,Top,Bottom,Numrules:integer;Maxyvalue:integer;Xstrings:
ListofStrings;Yvalues:Listofnums);
Var
Scale:real;
I, height, incr:integer;
Buffer:string[10];
Offset:integer;
Begin
        SetLineStyle(SolidLn,0,Thin Width);
        SetFillStyle(closeDoFill,blue);
        Bar3D(Left, top, right, bottom, 0, false);
        Offset:=(bottom-Top) div Numrules;
        SetJustify(rightText, center Text);
        SetTextStyle(DefaultFont,HorizDir,1);
        For I:= 1to numrules do
        Begin
        SetLineStyle(SolidLn,0,Normwidth);
        Line(Left, top+(I-1)*offset, right, top+(I-1)*offset)
        SetLineStyle(SolidLn,0,ThinWidth);
        Line(Left-hashwidth, top+(I-1)*offset, left, top+(I-1)*offset);
        Buffer:=intostr(Incr*(NumRules –1 +1));
        OuttextXY(Left-hashwidth-TextWidth(buffer), top+(I-1)*offset, buffer);
        End;
```

```
        Line(Left-hashwidth, bottom. Left, bottom);
        OuttextXY(Left-hashwidth-TextWidth(buffer), bottom, 0);
        Scale:=(bottom - top)/ maxYvalue;
        SetFillStyle(HatchFill,blue);
        Offset:=(right-left) div (numstring + 1);
        SetTextJustify(CenterText,TopText);
        SetTextStyle(DefaultFont, Horizdir, 1);
        For I:=1 to Numstrings do
Begin
        SetLineStyle(SolidLn, 0, Thickwidth);
        Line(Left+I*offset, bottom+hashwidth+2, xstring[I]);
        Height:=round(Yvalues[I]*scale);
        Bar3D(Left+I*offset-offset div 4, bottom-height, left+I*offset+offset div 4, bottom, 0,
        false);
      End;
End;
Readln;
LineGraph;
End;
{-------------------------------------------------------------------------------------------------------}
Procedure DisplayChart(Title:string;Xstring:ListofStrings;
Yvalues:ListofNums; NumRules:integer);

Var
  MaxYvalue, i, left, top, right, bottom:integer;

Begin
   If NumRules < 0 tghen
     NumRules := 1;
     Left:=ScreenBorder+MaxX div 8;
     Top := ScreenBorderY;
     right:=MaxX-ScreenBorderX;
```

```
    bottom:=MaxY-ScreenBorderY;
    ScaleText(Left,0,Right, Top,title);
    MaxYValue:=getMax(Yvalue);
    While (MaxYValue mod NumRules) <> 0 do
        Inc(MaxYValue);
    DrawChart(Left, top, right, bottom, NumRules, maxYValue, Xstrings, Yvalues);
end;

end;
{-------------------------------------------------------------------------------------------------------------}

Procedure Produzir_Graficos;

Begin
   Graphdriver := Detect;
   InitGraph(graphdriver, graphmode, 'c:\Tp\BGI');
   ErrorCode:=GraphResult;
   If ErrorCode<>grOk then
   begin
      writeln('Erro grafico: ' ,GraphErrorMsg(errorcode));
      Writeln('Programa Abortado');
      Readln;
   end;
   MaxX:=GetMax;
   MaxY:=GetMaxY;
   DisplayChart(Title,Strings,Yvalues,UseRules);
   Readln;
   Closegraph
end;

{ ---------------------------------------- Programa Principal ----------------------------------------}
Begin
```

```
    Sair := false;
    Intro;
    Repeat
        Main_Menu(Opcao);
        Case Opcao of
            1 :begin
                    Intro_Dados;
                    Produzir_Dados;
               end;
            2 : Visualizar_Resultado;
            3 : Produzir_Graficos;
            4 : Sair:= true;
        end;
    until sair=true;
end.
```

7.2 Anexo B - Resultados Gráfico



G

Gráfico 7.2 – Gráfico Comparativo do valor de $V_n(s)$ com diferentes n e diferentes $s=\lambda$